

# STC8H 系列单片机 技术参考手册

STC MCU

技术支持网站: [www.STCAI.com](http://www.STCAI.com)

官方技术论坛: [www.STCAIMCU.com](http://www.STCAIMCU.com)

资料更新日期: 2023/3/1

# 目录

|       |  |    |
|-------|--|----|
| 1     | 概述.....  | 1  |
| 2     | 特性、价格及管脚.....                                      | 3  |
| 2.1   | STC8H1K08-36I-TSSOP20/QFN20 系列.....                | 3  |
| 2.1.1 | 特性及价格.....   | 3  |
| 2.1.2 | 管脚图, 最小系统.....                                     | 6  |
| 2.1.3 | 管脚说明.....  | 8  |
| 2.2   | STC8H1K28-36I-LQFP32/QFN32,TSSOP20 系列.....         | 11 |
| 2.2.1 | 特性及价格.....   | 11 |
| 2.2.2 | 管脚图, 最小系统.....                                     | 14 |
| 2.2.3 | 管脚说明.....  | 16 |
| 2.3   | STC8H3K64S2-45I-LQFP48/32,QFN48/32,TSSOP20 系列..... | 20 |
| 2.3.1 | 特性及价格 (有 16 位硬件乘除法器 MDU16, 准 16 位单片机).....         | 20 |
| 2.3.2 | 管脚图, 最小系统.....                                     | 23 |
| 2.3.3 | 管脚说明.....  | 26 |
| 2.4   | STC8H3K64S4-45I-LQFP48/32,QFN48/32,TSSOP20 系列..... | 32 |
| 2.4.1 | 特性及价格 (有 16 位硬件乘除法器 MDU16, 准 16 位单片机).....         | 32 |
| 2.4.2 | 管脚图, 最小系统.....                                     | 35 |
| 2.4.3 | 管脚说明.....  | 38 |
| 2.5   | STC8H8K64U-45I-LQFP/QFN-64/48/32,TSSOP20.....      | 44 |
| 2.5.1 | 特性及价格 (有 16 位硬件乘除法器 MDU16, 准 16 位单片机).....         | 44 |
| 2.5.2 | 管脚图, 最小系统.....                                     | 47 |
| 2.5.3 | 管脚说明.....  | 52 |
| 2.6   | STC8H4K64TL-45I-LQFP48/32、QFN48、TSSOP20.....       | 58 |
| 2.6.1 | 特性及价格 (有 16 位硬件乘除法器 MDU16, 准 16 位单片机).....         | 58 |
| 2.6.2 | 管脚图, 最小系统.....                                     | 61 |
| 2.6.3 | RTC 实战线路图.....                                     | 64 |
| 2.6.4 | 管脚说明.....  | 65 |
| 2.7   | STC8H4K64TLCD-45I-LQFP64/48、QFN64/48.....          | 72 |
| 2.7.1 | 特性及价格 (有 16 位硬件乘除法器 MDU16, 准 16 位单片机).....         | 72 |
| 2.7.2 | 管脚图, 最小系统.....                                     | 75 |
| 2.7.3 | RTC 实战线路图.....                                     | 77 |
| 2.7.4 | 管脚说明.....  | 78 |
| 2.8   | STC8H1K08T-36I-TSSOP20/QFN20.....                  | 87 |
| 2.8.1 | 特性及价格 (有 16 位硬件乘除法器 MDU16, 准 16 位单片机).....         | 87 |
| 2.8.2 | 管脚图, 最小系统.....                                     | 90 |
| 2.8.3 | 管脚说明.....  | 91 |
| 3     | 功能脚切换.....   | 95 |
| 3.1   | 功能脚切换相关寄存器.....                                    | 95 |
| 3.1.1 | 外设端口切换控制寄存器 1 (P_SW1), 串口 1、SPI 切换.....            | 95 |
| 3.1.2 | 外设端口切换控制寄存器 2 (P_SW2), 串口 2/3/4、I2C、比较器输出切换.....   | 96 |

|        |   |            |
|--------|---|------------|
| 3.1.3  | 时钟选择寄存器 (MCLKOCR) .....                           | 96         |
| 3.1.4  | T3/T4 选择寄存器 (T3T4PIN) .....                       | 96         |
| 3.1.5  | 高级 PWM 选择寄存器 (PWMx_PS) .....                      | 98         |
| 3.1.6  | 高级 PWM 功能脚选择寄存器 (PWMx_ETRPS) .....                | 100        |
| 3.2    | 范例程序 .....  | 101        |
| 3.2.1  | 串口 1 切换 .....                                     | 101        |
| 3.2.2  | 串口 2 切换 .....                                     | 102        |
| 3.2.3  | 串口 3 切换 .....                                     | 103        |
| 3.2.4  | 串口 4 切换 .....                                     | 105        |
| 3.2.5  | SPI 切换 .....                                      | 106        |
| 3.2.6  | I2C 切换 .....                                      | 108        |
| 3.2.7  | 比较器输出切换 .....                                     | 109        |
| 3.2.8  | 主时钟输出切换 .....                                     | 110        |
| 4      | <b>封装尺寸图 .....</b>                                | <b>113</b> |
| 4.1    | SOP8 封装尺寸图 .....                                  | 113        |
| 4.2    | DFN8 封装尺寸图 (3mm*3mm) .....                        | 114        |
| 4.3    | SOP16 封装尺寸图 .....                                 | 115        |
| 4.4    | TSSOP20 封装尺寸图 .....                               | 116        |
| 4.5    | QFN20 封装尺寸图 (3mm*3mm) .....                       | 117        |
| 4.6    | LQFP32 封装尺寸图 (9mm*9mm) .....                      | 118        |
| 4.7    | QFN32 封装尺寸图 (4mm*4mm) .....                       | 119        |
| 4.8    | LQFP48 封装尺寸图 (9mm*9mm) .....                      | 120        |
| 4.9    | QFN48 封装尺寸图 (6mm*6mm) .....                       | 121        |
| 4.10   | LQFP64S 封装尺寸图 (12mm*12mm) .....                   | 122        |
| 4.11   | QFN64 封装尺寸图 (8mm*8mm) .....                       | 123        |
| 4.12   | STC8H 系列单片机命名规则 .....                             | 124        |
| 5      | <b>ISP 下载及典型应用线路图 .....</b>                       | <b>125</b> |
| 5.1    | STC8H 系列 ISP 下载应用线路图 .....                        | 125        |
| 5.1.1  | 硬件 USB 直接 ISP 下载 (5V 系统) .....                    | 125        |
| 5.1.2  | 硬件 USB 直接 ISP 下载 (3.3V 系统) .....                  | 128        |
| 5.1.3  | 软件模拟硬件 USB 直接 ISP 下载, 建议尝试, 不支持仿真 (5V 系统) .....   | 130        |
| 5.1.4  | 软件模拟硬件 USB 直接 ISP 下载, 建议尝试, 不支持仿真 (3.3V 系统) ..... | 132        |
| 5.1.5  | 使用 STC-USB Link1D 工具下载, 支持在线和脱机下载 .....           | 134        |
| 5.1.6  | 使用一箭双雕之 USB 转串口工具下载 .....                         | 137        |
| 5.1.7  | 使用 U8-Mini 工具下载, 支持 ISP 在线和脱机下载, 也可支持仿真 .....     | 139        |
| 5.1.8  | 使用 U8W 工具下载, 支持 ISP 在线和脱机下载, 也可支持仿真 .....         | 140        |
| 5.1.9  | 使用 RS-232 转换器下载, 也可支持仿真 .....                     | 141        |
| 5.1.10 | 使用 PL2303-GL 下载, 也可支持仿真 .....                     | 142        |
| 5.1.11 | 单机电源控制参考电路 .....                                  | 144        |
| 5.2    | 用 STC 一箭双雕之 USB 转双串口仿真 STC8 系列 MCU .....          | 145        |
| 5.3    | STC-ISP 下载软件高级应用 .....                            | 154        |
| 5.3.1  | 发布项目程序 .....                                      | 154        |
| 5.3.2  | 程序加密后传输 (防烧录时串口分析出程序) .....                       | 158        |
| 5.3.3  | 发布项目程序+程序加密后传输结合使用 .....                          | 162        |

|          |  |            |
|----------|--|------------|
| 5.3.4    | 用户自定义下载（实现不停电下载） .....                       | 163        |
| <b>6</b> | <b>时钟、复位、看门狗与电源管理 .....</b>                  | <b>167</b> |
| 6.1      | 系统时钟控制 .....                                 | 167        |
| 6.1.1    | 系统时钟选择寄存器（CLKSEL） .....                      | 169        |
| 6.1.2    | 时钟分频寄存器（CLKDIV） .....                        | 169        |
| 6.1.3    | 内部高速高精度 IRC 控制寄存器（HIRCCR） .....              | 170        |
| 6.1.4    | 外部振荡器控制寄存器（XOSCCR） .....                     | 170        |
| 6.1.5    | 外部 32K 振荡器控制寄存器（X32KCR） .....                | 171        |
| 6.1.6    | 内部 32KHz 低速 IRC 控制寄存器（IRC32KCR） .....        | 172        |
| 6.1.7    | 主时钟输出控制寄存器（MCLKOCR） .....                    | 173        |
| 6.1.8    | USB 时钟控制寄存器（USBCLK） .....                    | 173        |
| 6.1.9    | PLL 时钟控制寄存器（PLLCR） .....                     | 173        |
| 6.2      | STC8H 系列内部 IRC 频率调整 .....                    | 175        |
| 6.2.1    | IRC 频段选择寄存器（IRCBAND） .....                   | 175        |
| 6.2.2    | 内部 IRC 频率调整寄存器（IRTRIM） .....                 | 176        |
| 6.2.3    | 内部 IRC 频率微调寄存器（LIRTRIM） .....                | 177        |
| 6.2.4    | 时钟分频寄存器（CLKDIV） .....                        | 177        |
| 6.2.5    | 分频出 3MHz 用户工作频率，并用用户动态改变频率追频示例 .....         | 178        |
| 6.3      | 系统复位 .....                                   | 181        |
| 6.3.1    | 看门狗复位（WDT_CONTR） .....                       | 182        |
| 6.3.2    | 软件复位（IAP_CONTR） .....                        | 184        |
| 6.3.3    | 低压复位（RSTCFG） .....                           | 185        |
| 6.3.4    | 复位标志寄存器（RSTFLAG） .....                       | 185        |
| 6.3.5    | 低电平上电复位参考电路（一般不需要） .....                     | 187        |
| 6.3.6    | 低电平按键复位参考电路 .....                            | 187        |
| 6.3.7    | 传统 8051 高电平上电复位参考电路 .....                    | 188        |
| 6.4      | 外部晶振及外部时钟电路 .....                            | 189        |
| 6.4.1    | 外部晶振输入电路 .....                               | 189        |
| 6.4.2    | 外部时钟输入电路（P1.6 不可用作普通 I/O） .....              | 189        |
| 6.5      | 时钟停振/省电模式与系统电源管理 .....                       | 190        |
| 6.5.1    | 电源控制寄存器（PCON） .....                          | 190        |
| 6.6      | 掉电唤醒定时器 .....                                | 191        |
| 6.6.1    | 掉电唤醒定时器计数寄存器（WKTCL, WKTCH） .....             | 191        |
| 6.7      | 范例程序 .....                                   | 192        |
| 6.7.1    | 选择系统时钟源 .....                                | 192        |
| 6.7.2    | 主时钟分频输出 .....                                | 194        |
| 6.7.3    | 看门狗定时器应用 .....                               | 195        |
| 6.7.4    | 软复位实现自定义下载 .....                             | 197        |
| 6.7.5    | 低压检测 .....                                   | 199        |
| 6.7.6    | 省电模式 .....                                   | 200        |
| 6.7.7    | 使用 INT0/INT1/INT2/INT3/INT4 管脚中断唤醒省电模式 ..... | 202        |
| 6.7.8    | 使用 T0/T1/T2/T3/T4 管脚中断唤醒省电模式 .....           | 205        |
| 6.7.9    | 使用 RxD/RxD2/RxD3/RxD4 管脚中断唤醒省电模式 .....       | 209        |
| 6.7.10   | 使用 I2C 的 SDA 脚唤醒 MCU 省电模式 .....              | 212        |



|          |  |            |
|----------|--|------------|
| 6.7.11   | 使用掉电唤醒定时器唤醒省电模式 .....                                  | 214        |
| 6.7.12   | LVD 中断唤醒省电模式, 建议配合使用掉电唤醒定时器 .....                      | 216        |
| 6.7.13   | 比较器中断唤醒省电模式, 建议配合使用掉电唤醒定时器 .....                       | 219        |
| 6.7.14   | 使用 LVD 功能检测工作电压 (电池电压) .....                           | 221        |
| <b>7</b> | <b>自动频率校准, 自动追频 (CRE) .....</b>                        | <b>226</b> |
| 7.1      | 相关寄存器 .....  | 226        |
| 7.1.1    | CRE 控制寄存器 (CRECR) .....                                | 226        |
| 7.1.2    | CRE 校准计数值寄存器 (CRECNT) .....                            | 227        |
| 7.1.3    | CRE 校准误差值寄存器 (CRERES) .....                            | 227        |
| 7.2      | 范例程序 .....   | 228        |
| 7.2.1    | 自动校准内部高速 IRC (HIRC) .....                              | 228        |
| <b>8</b> | <b>存储器 .....</b>                                       | <b>230</b> |
| 8.1      | 程序存储器 .....  | 230        |
| 8.2      | 数据存储器 .....  | 231        |
| 8.2.1    | 内部 RAM .....   | 231        |
| 8.2.2    | 程序状态寄存器 (PSW) .....                                    | 232        |
| 8.2.3    | 内部扩展 RAM, XRAM, XDATA .....                            | 233        |
| 8.2.4    | 辅助寄存器 (AUXR) .....                                     | 233        |
| 8.2.5    | 外部扩展 RAM, XRAM, XDATA .....                            | 234        |
| 8.2.6    | 总线速度控制寄存器 (BUS_SPEED) .....                            | 234        |
| 8.2.7    | 8051 中可位寻址的数据存储器 .....                                 | 235        |
| 8.2.8    | 扩展 SFR 使能寄存器 EAXFR 的使用说明 .....                         | 237        |
| 8.3      | 存储器中的特殊参数, 在 ISP 下载时可烧录进程序 FLASH .....                 | 238        |
| 8.4      | 只读特殊功能寄存器中存储的唯一 ID 号和重要参数 (CHIPID) .....               | 243        |
| 8.4.1    | CHIP 之全球唯一 ID 号解读 .....                                | 245        |
| 8.4.2    | CHIP 之内部参考信号源解读 .....                                  | 245        |
| 8.4.3    | CHIP 之内部 32K 的 IRC 振荡频率解读 .....                        | 246        |
| 8.4.4    | CHIP 之高精度 IRC 参数解读 .....                               | 247        |
| 8.4.5    | CHIP 之测试时间参数解读 .....                                   | 248        |
| 8.4.6    | CHIP 之芯片封装形式编号解读 .....                                 | 248        |
| 8.5      | 范例程序 .....   | 249        |
| 8.5.1    | 读取内部 1.19V 参考信号源-BGV 值 (从 CHIPID 中读取) .....            | 249        |
| 8.5.2    | 读取内部 1.19V 参考信号源-BGV 值 (从 Flash 程序存储器 (ROM) 中读取) ..... | 252        |
| 8.5.3    | 读取内部 1.19V 参考信号源-BGV 值 (从 RAM 中读取) .....               | 254        |
| 8.5.4    | 读取全球唯一 ID 号 (从 CHIPID 中读取) .....                       | 257        |
| 8.5.5    | 读取全球唯一 ID 号 (从 Flash 程序存储器 (ROM) 中读取) .....            | 260        |
| 8.5.6    | 读取全球唯一 ID 号 (从 RAM 中读取) .....                          | 263        |
| 8.5.7    | 读取 32K 掉电唤醒定时器的频率 (从 CHIPID 中读取) .....                 | 266        |
| 8.5.8    | 读取 32K 掉电唤醒定时器的频率 (从 Flash 程序存储器 (ROM) 中读取) .....      | 269        |
| 8.5.9    | 读取 32K 掉电唤醒定时器的频率 (从 RAM 中读取) .....                    | 272        |
| 8.5.10   | 用户自定义内部 IRC 频率 (从 CHIPID 中读取) .....                    | 275        |
| 8.5.11   | 用户自定义内部 IRC 频率-两频段 (从 Flash 程序存储器 (ROM) 中读取) .....     | 282        |
| 8.5.12   | 用户自定义内部 IRC 频率-四频段 (从 Flash 程序存储器 (ROM) 中读取) .....     | 287        |
| 8.5.13   | 用户自定义内部 IRC 频率 (从 RAM 中读取) .....                       | 293        |

|           |  |            |
|-----------|--|------------|
| <b>9</b>  | <b>特殊功能寄存器.....</b>                      | <b>296</b> |
| 9.1       | STC8H1K08 系列.....                        | 296        |
| 9.2       | STC8H1K28 系列.....                        | 298        |
| 9.3       | STC8H3K64S4 系列.....                      | 299        |
| 9.4       | STC8H3K64S2 系列.....                      | 301        |
| 9.5       | STC8H8K64U-64Pin/48Pin USB 系列.....       | 303        |
| 9.6       | STC8H4K64TL 系列.....                      | 305        |
| 9.7       | STC8H4K64TLCD 系列.....                    | 307        |
| 9.8       | STC8H1K08T 系列.....                       | 309        |
| 9.9       | 特殊功能寄存器列表.....                           | 311        |
| <b>10</b> | <b>I/O 口.....</b>                        | <b>324</b> |
| 10.1      | I/O 口相关寄存器.....                          | 324        |
| 10.1.1    | 端口数据寄存器 (Px).....                        | 327        |
| 10.1.2    | 端口模式配置寄存器 (PxM0, PxM1).....              | 327        |
| 10.1.3    | 端口上拉电阻控制寄存器 (PxPU).....                  | 328        |
| 10.1.4    | 端口施密特触发控制寄存器 (PxNCS).....                | 328        |
| 10.1.5    | 端口电平转换速度控制寄存器 (PxSR).....                | 329        |
| 10.1.6    | 端口驱动电流控制寄存器 (PxDR).....                  | 329        |
| 10.1.7    | 端口数字信号输入使能控制寄存器 (PxIE).....              | 329        |
| 10.2      | 配置 I/O 口.....                            | 331        |
| 10.3      | I/O 的结构图.....                            | 332        |
| 10.3.1    | 准双向口 (弱上拉).....                          | 332        |
| 10.3.2    | 推挽输出.....                                | 332        |
| 10.3.3    | 高阻输入.....                                | 333        |
| 10.3.4    | 开漏输出.....                                | 333        |
| 10.3.5    | 新增 4.1K 上拉电阻.....                        | 334        |
| 10.3.6    | 如何设置 I/O 口对外输出速度.....                    | 334        |
| 10.3.7    | 如何设置 I/O 口电流驱动能力.....                    | 335        |
| 10.3.8    | 如何降低 I/O 口对外辐射.....                      | 335        |
| 10.4      | 范例程序.....                                | 336        |
| 10.4.1    | 端口模式设置.....                              | 336        |
| 10.4.2    | 双向口读写操作.....                             | 337        |
| 10.5      | 一种典型三极管控制电路.....                         | 339        |
| 10.6      | 典型发光二极管控制电路.....                         | 340        |
| 10.7      | 混合电压供电系统 3V/5V 器件 I/O 口互连.....           | 340        |
| 10.8      | 如何让 I/O 口上电复位时为低电平.....                  | 341        |
| 10.9      | 利用 74HC595 驱动 8 个数码管(串行扩展,3 根线)的线路图..... | 342        |
| 10.10     | I/O 口直接驱动 LED 数码管应用线路图.....              | 343        |
| 10.11     | 用 STC 系列 MCU 的 I/O 口直接驱动段码 LCD.....      | 344        |
| 10.12     | 使用 I/O 和 R-2R 电阻分压实现 DAC 的经典线路图.....     | 363        |
| <b>11</b> | <b>指令系统.....</b>                         | <b>364</b> |
| <b>12</b> | <b>中断系统.....</b>                         | <b>368</b> |
| 12.1      | STC8H 系列中断源.....                         | 368        |
| 12.2      | STC8H 中断结构图.....                         | 371        |

|         |  |            |
|---------|--|------------|
| 12.3    | STC8H 系列中断列表 .....                           | 372        |
| 12.4    | 中断相关寄存器 .....                                | 375        |
| 12.4.1  | 中断使能寄存器 (中断允许位) .....                        | 377        |
| 12.4.2  | 中断请求寄存器 (中断标志位) .....                        | 382        |
| 12.4.3  | 中断优先级寄存器 .....                               | 386        |
| 12.5    | 范例程序 .....                                   | 393        |
| 12.5.1  | INT0 中断 (上升沿和下降沿), 可同时支持上升沿和下降沿 .....        | 393        |
| 12.5.2  | INT0 中断 (下降沿) .....                          | 394        |
| 12.5.3  | INT1 中断 (上升沿和下降沿), 可同时支持上升沿和下降沿 .....        | 396        |
| 12.5.4  | INT1 中断 (下降沿) .....                          | 398        |
| 12.5.5  | INT2 中断 (下降沿), 只支持下降沿中断 .....                | 399        |
| 12.5.6  | INT3 中断 (下降沿), 只支持下降沿中断 .....                | 401        |
| 12.5.7  | INT4 中断 (下降沿), 只支持下降沿中断 .....                | 403        |
| 12.5.8  | 定时器 0 中断 .....                               | 404        |
| 12.5.9  | 定时器 1 中断 .....                               | 406        |
| 12.5.10 | 定时器 2 中断 .....                               | 408        |
| 12.5.11 | 定时器 3 中断 .....                               | 410        |
| 12.5.12 | 定时器 4 中断 .....                               | 411        |
| 12.5.13 | UART1 中断 .....                               | 413        |
| 12.5.14 | UART2 中断 .....                               | 415        |
| 12.5.15 | UART3 中断 .....                               | 418        |
| 12.5.16 | UART4 中断 .....                               | 420        |
| 12.5.17 | ADC 中断 .....                                 | 422        |
| 12.5.18 | LVD 中断 .....                                 | 424        |
| 12.5.19 | 比较器中断 .....                                  | 426        |
| 12.5.20 | SPI 中断 .....                                 | 428        |
| 12.5.21 | I2C 中断 .....                                 | 429        |
| 13      | <b>所有的 I/O 口均可中断 (4 种模式), 不是传统外部中断 .....</b> | <b>432</b> |
| 13.1    | I/O 口中断相关寄存器 .....                           | 432        |
| 13.1.1  | 端口中断使能寄存器 (PxINTE) .....                     | 433        |
| 13.1.2  | 端口中断标志寄存器 (PxINTF) .....                     | 434        |
| 13.1.3  | 端口中断模式配置寄存器 (PxIM0, PxIM1) .....             | 434        |
| 13.1.4  | 端口中断优先级控制寄存器 (PINIPL, PINIPH) .....          | 435        |
| 13.1.5  | 端口中断掉电唤醒使能寄存器 (PxWKUE) .....                 | 435        |
| 13.2    | 范例程序 .....                                   | 436        |
| 13.2.1  | P0 口下降沿中断 .....                              | 436        |
| 13.2.2  | P1 口上升沿中断 .....                              | 439        |
| 13.2.3  | P2 口低电平中断 .....                              | 442        |
| 13.2.4  | P3 口高电平中断 .....                              | 446        |
| 13.2.5  | 使用拓展 Keil 中断号方案的 I/O 口中断范例 .....             | 449        |
| 14      | <b>定时器/计数器 .....</b>                         | <b>452</b> |
| 14.1    | 定时器 3/4 功能脚切换 .....                          | 452        |
| 14.2    | 定时器的相关寄存器 .....                              | 453        |
| 14.3    | 定时器 0/1 .....                                | 454        |

|         |  |     |
|---------|--|-----|
| 14.3.1  | 定时器 0/1 控制寄存器 (TCON)                     | 454 |
| 14.3.2  | 定时器 0/1 模式寄存器 (TMOD)                     | 454 |
| 14.3.3  | 定时器 0 模式 0 (16 位自动重装载模式)                 | 455 |
| 14.3.4  | 定时器 0 模式 1 (16 位不可重装载模式)                 | 456 |
| 14.3.5  | 定时器 0 模式 2 (8 位自动重装载模式)                  | 457 |
| 14.3.6  | 定时器 0 模式 3 (不可屏蔽中断 16 位自动重装载, 实时操作系统节拍器) | 457 |
| 14.3.7  | 定时器 1 模式 0 (16 位自动重装载模式)                 | 458 |
| 14.3.8  | 定时器 1 模式 1 (16 位不可重装载模式)                 | 459 |
| 14.3.9  | 定时器 1 模式 2 (8 位自动重装载模式)                  | 460 |
| 14.3.10 | 定时器 0 计数寄存器 (TL0, TH0)                   | 460 |
| 14.3.11 | 定时器 1 计数寄存器 (TL1, TH1)                   | 460 |
| 14.3.12 | 辅助寄存器 1 (AUXR)                           | 460 |
| 14.3.13 | 中断与时钟输出控制寄存器 (INTCLKO)                   | 461 |
| 14.3.14 | 定时器 0 计算公式                               | 461 |
| 14.3.15 | 定时器 1 计算公式                               | 462 |
| 14.4    | 定时器 2 (24 位定时器, 8 位预分频+16 位定时)           | 463 |
| 14.4.1  | 辅助寄存器 1 (AUXR)                           | 463 |
| 14.4.2  | 中断与时钟输出控制寄存器 (INTCLKO)                   | 463 |
| 14.4.3  | 定时器 2 计数寄存器 (T2L, T2H)                   | 463 |
| 14.4.4  | 定时器 2 的 8 位预分频寄存器 (TM2PS)                | 463 |
| 14.4.5  | 定时器 2 工作模式                               | 464 |
| 14.4.6  | 定时器 2 计算公式                               | 464 |
| 14.5    | 定时器 3/4 (24 位定时器, 8 位预分频+16 位定时)         | 465 |
| 14.5.1  | 定时器 4/3 控制寄存器 (T4T3M)                    | 465 |
| 14.5.2  | 定时器 3 计数寄存器 (T3L, T3H)                   | 465 |
| 14.5.3  | 定时器 4 计数寄存器 (T4L, T4H)                   | 465 |
| 14.5.4  | 定时器 3 的 8 位预分频寄存器 (TM3PS)                | 466 |
| 14.5.5  | 定时器 4 的 8 位预分频寄存器 (TM4PS)                | 466 |
| 14.5.6  | 定时器 3 工作模式                               | 466 |
| 14.5.7  | 定时器 4 工作模式                               | 468 |
| 14.5.8  | 定时器 3 计算公式                               | 468 |
| 14.5.9  | 定时器 4 计算公式                               | 469 |
| 14.6    | 范例程序                                     | 470 |
| 14.6.1  | 定时器 0 (模式 0—16 位自动重载), 用作定时              | 470 |
| 14.6.2  | 定时器 0 (模式 1—16 位不自动重载), 用作定时             | 471 |
| 14.6.3  | 定时器 0 (模式 2—8 位自动重载), 用作定时               | 473 |
| 14.6.4  | 定时器 0 (模式 3—16 位自动重载不可屏蔽中断), 用作定时        | 475 |
| 14.6.5  | 定时器 0 (外部计数—扩展 T0 为外部下降沿中断)              | 477 |
| 14.6.6  | 定时器 0 (测量脉宽—INT0 高电平宽度)                  | 478 |
| 14.6.7  | 定时器 0 (模式 0), 时钟分频输出                     | 480 |
| 14.6.8  | 定时器 1 (模式 0—16 位自动重载), 用作定时              | 482 |
| 14.6.9  | 定时器 1 (模式 1—16 位不自动重载), 用作定时             | 483 |
| 14.6.10 | 定时器 1 (模式 2—8 位自动重载), 用作定时               | 485 |
| 14.6.11 | 定时器 1 (外部计数—扩展 T1 为外部下降沿中断)              | 487 |

|           |                                 |            |
|-----------|---------------------------------|------------|
| 14.6.12   | 定时器 1 (测量脉宽—INT1 高电平宽度)         | 489        |
| 14.6.13   | 定时器 1 (模式 0), 时钟分频输出            | 491        |
| 14.6.14   | 定时器 1 (模式 0) 做串口 1 波特率发生器       | 492        |
| 14.6.15   | 定时器 1 (模式 2) 做串口 1 波特率发生器       | 496        |
| 14.6.16   | 定时器 2 (16 位自动重载), 用作定时          | 500        |
| 14.6.17   | 定时器 2 (外部计数—扩展 T2 为外部下降沿中断)     | 501        |
| 14.6.18   | 定时器 2, 时钟分频输出                   | 503        |
| 14.6.19   | 定时器 2 做串口 1 波特率发生器              | 505        |
| 14.6.20   | 定时器 2 做串口 2 波特率发生器              | 508        |
| 14.6.21   | 定时器 2 做串口 3 波特率发生器              | 512        |
| 14.6.22   | 定时器 2 做串口 4 波特率发生器              | 516        |
| 14.6.23   | 定时器 3 (16 位自动重载), 用作定时          | 520        |
| 14.6.24   | 定时器 3 (外部计数—扩展 T3 为外部下降沿中断)     | 522        |
| 14.6.25   | 定时器 3, 时钟分频输出                   | 524        |
| 14.6.26   | 定时器 3 做串口 3 波特率发生器              | 525        |
| 14.6.27   | 定时器 4 (16 位自动重载), 用作定时          | 529        |
| 14.6.28   | 定时器 4 (外部计数—扩展 T4 为外部下降沿中断)     | 531        |
| 14.6.29   | 定时器 4, 时钟分频输出                   | 533        |
| 14.6.30   | 定时器 4 做串口 4 波特率发生器              | 535        |
| <b>15</b> | <b>超级简单的 STC USB-CDC 虚拟串口应用</b> | <b>539</b> |
| 15.1      | USB-CDC 虚拟串口概述                  | 539        |
| 15.2      | 新建 Keil 项目并加入 CDC 模块            | 540        |
| 15.3      | USB-CDC 虚拟串口与电脑进行数据传输           | 547        |
| 15.4      | STC USB-CDC 虚拟串口实现不停电自动 ISP 下载  | 548        |
| <b>16</b> | <b>串口通信</b>                     | <b>549</b> |
| 16.1      | 串口功能脚切换                         | 549        |
| 16.2      | 串口相关寄存器                         | 550        |
| 16.3      | 串口 1                            | 551        |
| 16.3.1    | 串口 1 控制寄存器 (SCON)               | 551        |
| 16.3.2    | 串口 1 数据寄存器 (SBUF)               | 551        |
| 16.3.3    | 电源管理寄存器 (PCON)                  | 552        |
| 16.3.4    | 辅助寄存器 1 (AUXR)                  | 552        |
| 16.3.5    | 串口 1 模式 0, 模式 0 波特率计算公式         | 552        |
| 16.3.6    | 串口 1 模式 1, 模式 1 波特率计算公式         | 553        |
| 16.3.7    | 串口 1 模式 2, 模式 2 波特率计算公式         | 556        |
| 16.3.8    | 串口 1 模式 3, 模式 3 波特率计算公式         | 557        |
| 16.3.9    | 自动地址识别                          | 557        |
| 16.3.10   | 串口 1 从机地址控制寄存器 (SADDR, SADEN)   | 557        |
| 16.4      | 串口 2                            | 559        |
| 16.4.1    | 串口 2 控制寄存器 (S2CON)              | 559        |
| 16.4.2    | 串口 2 数据寄存器 (S2BUF)              | 559        |
| 16.4.3    | 串口 2 模式 0, 模式 0 波特率计算公式         | 559        |
| 16.4.4    | 串口 2 模式 1, 模式 1 波特率计算公式         | 560        |
| 16.5      | 串口 3                            | 562        |

|           |  |            |
|-----------|--|------------|
| 16.5.1    | 串口 3 控制寄存器 (S3CON) .....                     | 562        |
| 16.5.2    | 串口 3 数据寄存器 (S3BUF) .....                     | 562        |
| 16.5.3    | 串口 3 模式 0, 模式 0 波特率计算公式 .....                | 562        |
| 16.5.4    | 串口 3 模式 1, 模式 1 波特率计算公式 .....                | 563        |
| 16.6      | 串口 4 .....                                   | 565        |
| 16.6.1    | 串口 4 控制寄存器 (S4CON) .....                     | 565        |
| 16.6.2    | 串口 4 数据寄存器 (S4BUF) .....                     | 565        |
| 16.6.3    | 串口 4 模式 0, 模式 0 波特率计算公式 .....                | 565        |
| 16.6.4    | 串口 4 模式 1, 模式 1 波特率计算公式 .....                | 566        |
| 16.7      | 串口注意事项 .....                                 | 568        |
| 16.8      | 范例程序 .....                                   | 569        |
| 16.8.1    | 串口 1 使用定时器 2 做波特率发生器 .....                   | 569        |
| 16.8.2    | 串口 1 使用定时器 1 (模式 0) 做波特率发生器 .....            | 572        |
| 16.8.3    | 串口 1 使用定时器 1 (模式 2) 做波特率发生器 .....            | 576        |
| 16.8.4    | 串口 2 使用定时器 2 做波特率发生器 .....                   | 580        |
| 16.8.5    | 串口 3 使用定时器 2 做波特率发生器 .....                   | 584        |
| 16.8.6    | 串口 3 使用定时器 3 做波特率发生器 .....                   | 587        |
| 16.8.7    | 串口 4 使用定时器 2 做波特率发生器 .....                   | 591        |
| 16.8.8    | 串口 4 使用定时器 4 做波特率发生器 .....                   | 595        |
| 16.8.9    | 串口多机通讯 .....                                 | 600        |
| 16.8.10   | 串口中断收发—MODBUS 协议 .....                       | 601        |
| 16.8.11   | 串口转 LIN 总线 .....                             | 612        |
| <b>17</b> | <b>比较器, 掉电检测, 内部 1.19V 参考信号源 (BGV) .....</b> | <b>620</b> |
| 17.1      | 比较器内部结构图 .....                               | 621        |
| 17.2      | 比较器输出功能脚切换 .....                             | 621        |
| 17.3      | 比较器相关的寄存器 .....                              | 622        |
| 17.3.1    | 比较器控制寄存器 1 (CMPCR1) .....                    | 622        |
| 17.3.2    | 比较器控制寄存器 2 (CMPCR2) .....                    | 623        |
| 17.3.3    | 比较器扩展配置寄存器 (CMPEXCFG) .....                  | 623        |
| 17.4      | 范例程序 .....                                   | 625        |
| 17.4.1    | 旧版比较器的使用 (中断方式) .....                        | 625        |
| 17.4.2    | 旧版比较器的使用 (查询方式) .....                        | 627        |
| 17.4.3    | 新版比较器的使用 (中断方式) .....                        | 629        |
| 17.4.4    | 新版比较器的使用 (查询方式) .....                        | 632        |
| 17.4.5    | 旧版比较器的多路复用应用 (比较器+ADC 输入通道) .....            | 635        |
| 17.4.6    | 新版比较器的多路复用应用 (比较器+ADC 输入通道) .....            | 636        |
| 17.4.7    | 比较器作外部掉电检测 (掉电过程中应及时保存用户数据到 EEPROM 中) .....  | 639        |
| 17.4.8    | 比较器检测工作电压 (电池电压) .....                       | 640        |
| <b>18</b> | <b>IAP/EEPROM/DATA-FLASH .....</b>           | <b>644</b> |
| 18.1      | EEPROM 操作时间 .....                            | 644        |
| 18.2      | EEPROM 相关的寄存器 .....                          | 644        |
| 18.2.1    | EEPROM 数据寄存器 (IAP_DATA) .....                | 644        |
| 18.2.2    | EEPROM 地址寄存器 (IAP_ADDR) .....                | 645        |
| 18.2.3    | EEPROM 命令寄存器 (IAP_CMD) .....                 | 645        |



|           |   |            |
|-----------|---|------------|
| 18.2.4    | EEPROM 触发寄存器 (IAP_TRIG) .....               | 645        |
| 18.2.5    | EEPROM 控制寄存器 (IAP_CONTR) .....              | 645        |
| 18.2.6    | EEPROM 等待时间控制寄存器 (IAP_TPS) .....            | 646        |
| 18.3      | EEPROM 大小及地址 .....                          | 647        |
| 18.4      | 范例程序 .....                                  | 650        |
| 18.4.1    | EEPROM 基本操作 .....                           | 650        |
| 18.4.2    | 使用 MOVC 读取 EEPROM .....                     | 653        |
| 18.4.3    | 使用串口送出 EEPROM 数据 .....                      | 656        |
| 18.4.4    | 串口 1 读写 EEPROM-带 MOVC 读 .....               | 660        |
| 18.4.5    | 口令擦除写入-多扇区备份-串口 1 操作 .....                  | 667        |
| <b>19</b> | <b>ADC 模数转换, 内部 1.19V 参考信号源 (BGV) .....</b> | <b>677</b> |
| 19.1      | ADC 相关的寄存器 .....                            | 677        |
| 19.1.1    | ADC 控制寄存器 (ADC_CONTR), PWM 触发 ADC 控制 .....  | 678        |
| 19.1.2    | ADC 配置寄存器 (ADCCFG) .....                    | 679        |
| 19.1.3    | ADC 转换结果寄存器 (ADC_RES, ADC_RES1) .....       | 680        |
| 19.1.4    | ADC 时序控制寄存器 (ADCTIM) .....                  | 681        |
| 19.1.5    | ADC 扩展配置寄存器 (ADCEXCFG) .....                | 682        |
| 19.2      | ADC 相关计算公式 .....                            | 683        |
| 19.2.1    | ADC 速度计算公式 .....                            | 683        |
| 19.2.2    | ADC 转换结果计算公式 .....                          | 683        |
| 19.2.3    | 反推 ADC 输入电压计算公式 .....                       | 684        |
| 19.2.4    | 反推工作电压计算公式 .....                            | 685        |
| 19.3      | 10 位 ADC 静态特性 .....                         | 685        |
| 19.4      | 12 位 ADC 静态特性 .....                         | 685        |
| 19.5      | ADC 应用参考线路图 .....                           | 686        |
| 19.5.1    | 一般精度 ADC 参考线路图 .....                        | 686        |
| 19.5.2    | 高精度 ADC 参考线路图 .....                         | 687        |
| 19.5.3    | 高精度 ADC 参考线路图 (有独立 AVcc 和 AGnd) .....       | 688        |
| 19.6      | 范例程序 .....                                  | 689        |
| 19.6.1    | ADC 基本操作 (查询方式) .....                       | 689        |
| 19.6.2    | ADC 基本操作 (中断方式) .....                       | 691        |
| 19.6.3    | 格式化 ADC 转换结果 .....                          | 693        |
| 19.6.4    | 利用 ADC 第 15 通道测量外部电压或电池电压 .....             | 695        |
| 19.6.5    | ADC 做电容感应触摸按键 .....                         | 698        |
| 19.6.6    | ADC 作按键扫描应用线路图 .....                        | 710        |
| 19.6.7    | 检测负电压参考线路图 .....                            | 711        |
| 19.6.8    | 常用加法电路在 ADC 中的应用 .....                      | 712        |
| <b>20</b> | <b>同步串行外设接口 SPI .....</b>                   | <b>713</b> |
| 20.1      | SPI 功能脚切换 .....                             | 713        |
| 20.2      | SPI 相关的寄存器 .....                            | 713        |
| 20.2.1    | SPI 状态寄存器 (SPSTAT) .....                    | 713        |
| 20.2.2    | SPI 控制寄存器 (SPCTL), SPI 速度控制 .....           | 714        |
| 20.2.3    | SPI 数据寄存器 (SPDAT) .....                     | 715        |
| 20.3      | SPI 通信方式 .....                              | 716        |

|           |  |            |
|-----------|--|------------|
| 20.3.1    | 单主单从 .....                                   | 716        |
| 20.3.2    | 互为主从 .....                                   | 716        |
| 20.3.3    | 单主多从 .....                                   | 717        |
| 20.4      | 配置 SPI .....                                 | 718        |
| 20.5      | 数据模式 .....                                   | 720        |
| 20.6      | 范例程序 .....                                   | 721        |
| 20.6.1    | SPI 单主单从系统主机程序（中断方式） .....                   | 721        |
| 20.6.2    | SPI 单主单从系统从机程序（中断方式） .....                   | 723        |
| 20.6.3    | SPI 单主单从系统主机程序（查询方式） .....                   | 725        |
| 20.6.4    | SPI 单主单从系统从机程序（查询方式） .....                   | 727        |
| 20.6.5    | SPI 互为主从系统程序（中断方式） .....                     | 729        |
| 20.6.6    | SPI 互为主从系统程序（查询方式） .....                     | 731        |
| <b>21</b> | <b>I<sup>2</sup>C 总线 .....</b>               | <b>735</b> |
| 21.1      | I <sup>2</sup> C 功能脚切换 .....                 | 735        |
| 21.2      | I <sup>2</sup> C 相关的寄存器 .....                | 735        |
| 21.3      | I <sup>2</sup> C 主机模式 .....                  | 737        |
| 21.3.1    | I <sup>2</sup> C 配置寄存器（I2CCFG），总线速度控制 .....  | 737        |
| 21.3.2    | I <sup>2</sup> C 主机控制寄存器（I2CMSCR） .....      | 738        |
| 21.3.3    | I <sup>2</sup> C 主机辅助控制寄存器（I2CMSAUX） .....   | 739        |
| 21.3.4    | I <sup>2</sup> C 主机状态寄存器（I2CMSST） .....      | 739        |
| 21.4      | I <sup>2</sup> C 从机模式 .....                  | 741        |
| 21.4.1    | I <sup>2</sup> C 从机控制寄存器（I2CSLCR） .....      | 741        |
| 21.4.2    | I <sup>2</sup> C 从机状态寄存器（I2CSLST） .....      | 741        |
| 21.4.3    | I <sup>2</sup> C 从机地址寄存器（I2CSLADR） .....     | 743        |
| 21.4.4    | I <sup>2</sup> C 数据寄存器（I2CTXD, I2CRXD） ..... | 744        |
| 21.5      | 范例程序 .....                                   | 745        |
| 21.5.1    | I <sup>2</sup> C 主机模式访问 AT24C256（中断方式） ..... | 745        |
| 21.5.2    | I <sup>2</sup> C 主机模式访问 AT24C256（查询方式） ..... | 750        |
| 21.5.3    | I <sup>2</sup> C 主机模式访问 PCF8563 .....        | 756        |
| 21.5.4    | I <sup>2</sup> C 从机模式（中断方式） .....            | 761        |
| 21.5.5    | I <sup>2</sup> C 从机模式（查询方式） .....            | 765        |
| 21.5.6    | 测试 I <sup>2</sup> C 从机模式代码的主机代码 .....        | 769        |
| <b>22</b> | <b>16 位高级 PWM 定时器，支持正交编码器 .....</b>          | <b>775</b> |
| 22.1      | 简介 .....                                     | 778        |
| 22.2      | 主要特性 .....                                   | 778        |
| 22.3      | 时基单元 .....                                   | 779        |
| 22.3.1    | 读写 16 位计数器 .....                             | 780        |
| 22.3.2    | 16 位 PWMA_ARR 寄存器的写操作 .....                  | 780        |
| 22.3.3    | 预分频器 .....                                   | 780        |
| 22.3.4    | 向上计数模式 .....                                 | 780        |
| 22.3.5    | 向下计数模式 .....                                 | 782        |
| 22.3.6    | 中间对齐模式（向上/向下计数） .....                        | 783        |
| 22.3.7    | 重复计数器 .....                                  | 784        |
| 22.4      | 时钟/触发控制器 .....                               | 785        |



|         |  |     |
|---------|--|-----|
| 22.4.1  | 预分频时钟 (CK_PSC) .....                   | 785 |
| 22.4.2  | 内部时钟源 (f <sub>MASTER</sub> ) .....     | 785 |
| 22.4.3  | 外部时钟源模式 1 .....                        | 786 |
| 22.4.4  | 外部时钟源模式 2 .....                        | 787 |
| 22.4.5  | 触发同步 .....                             | 788 |
| 22.4.6  | 与 PWMB 同步 .....                        | 790 |
| 22.5    | 捕获/比较通道 .....                          | 793 |
| 22.5.1  | 16 位 PWMA_CCRi 寄存器的写流程 .....           | 794 |
| 22.5.2  | 输入模块 .....                             | 794 |
| 22.5.3  | 输入捕获模式 .....                           | 794 |
| 22.5.4  | 输出模块 .....                             | 796 |
| 22.5.5  | 强制输出模式 .....                           | 797 |
| 22.5.6  | 输出比较模式 .....                           | 797 |
| 22.5.7  | PWM 模式 .....                           | 798 |
| 22.5.8  | 使用刹车功能 (PWMFLT) .....                  | 803 |
| 22.5.9  | 在外部事件发生时清除 OCiREF 信号 .....             | 804 |
| 22.5.10 | 编码器接口模式 .....                          | 805 |
| 22.6    | 中断 .....                               | 806 |
| 22.7    | PWMA/PWMB 寄存器描述 .....                  | 808 |
| 22.7.1  | 高级 PWM 功能脚切换 .....                     | 808 |
| 22.7.2  | 输出使能寄存器 (PWMx_ENO) .....               | 809 |
| 22.7.3  | 输出附加使能寄存器 (PWMx_IOAUX) .....           | 810 |
| 22.7.4  | 控制寄存器 1 (PWMx_CR1) .....               | 811 |
| 22.7.5  | 控制寄存器 2 (PWMx_CR2), 及实时触发 ADC .....    | 812 |
| 22.7.6  | 从模式控制寄存器 (PWMx_SMCR) .....             | 814 |
| 22.7.7  | 外部触发寄存器 (PWMx_ETR) .....               | 816 |
| 22.7.8  | 中断使能寄存器 (PWMx_IER) .....               | 816 |
| 22.7.9  | 状态寄存器 1 (PWMx_SR1) .....               | 817 |
| 22.7.10 | 状态寄存器 2 (PWMx_SR2) .....               | 818 |
| 22.7.11 | 事件产生寄存器 (PWMx_EGR) .....               | 818 |
| 22.7.12 | 捕获/比较模式寄存器 1 (PWMx_CCMR1) .....        | 819 |
| 22.7.13 | 捕获/比较模式寄存器 2 (PWMx_CCMR2) .....        | 823 |
| 22.7.14 | 捕获/比较模式寄存器 3 (PWMx_CCMR3) .....        | 824 |
| 22.7.15 | 捕获/比较模式寄存器 4 (PWMx_CCMR4) .....        | 825 |
| 22.7.16 | 捕获/比较使能寄存器 1 (PWMx_CCER1) .....        | 826 |
| 22.7.17 | 捕获/比较使能寄存器 2 (PWMx_CCER2) .....        | 828 |
| 22.7.18 | 计数器高 8 位 (PWMx_CNTRH) .....            | 829 |
| 22.7.19 | 计数器低 8 位 (PWMx_CNTRL) .....            | 829 |
| 22.7.20 | 预分频器高 8 位 (PWMx_PSCRH), 输出频率计算公式 ..... | 829 |
| 22.7.21 | 预分频器低 8 位 (PWMx_PSCRL) .....           | 829 |
| 22.7.22 | 自动重装载寄存器高 8 位 (PWMx_ARRH) .....        | 830 |
| 22.7.23 | 自动重装载寄存器低 8 位 (PWMx_ARRL) .....        | 830 |
| 22.7.24 | 重复计数器寄存器 (PWMx_RCR) .....              | 830 |
| 22.7.25 | 捕获/比较寄存器 1/5 高 8 位 (PWMx_CCR1H) .....  | 830 |

|         |  |            |
|---------|--|------------|
| 22.7.26 | 捕获/比较寄存器 1/5 低 8 位 (PWMx_CCR1L) .....          | 830        |
| 22.7.27 | 捕获/比较寄存器 2/6 高 8 位 (PWMx_CCR2H) .....          | 831        |
| 22.7.28 | 捕获/比较寄存器 2/6 低 8 位 (PWMx_CCR2L) .....          | 831        |
| 22.7.29 | 捕获/比较寄存器 3/7 高 8 位 (PWMx_CCR3H) .....          | 831        |
| 22.7.30 | 捕获/比较寄存器 3/7 低 8 位 (PWMx_CCR3L) .....          | 831        |
| 22.7.31 | 捕获/比较寄存器 4/8 高 8 位 (PWMx_CCR4H) .....          | 831        |
| 22.7.32 | 捕获/比较寄存器 4/8 低 8 位 (PWMx_CCR4L) .....          | 831        |
| 22.7.33 | 刹车寄存器 (PWMx_BKR) .....                         | 832        |
| 22.7.34 | 死区寄存器 (PWMx_DTR) .....                         | 833        |
| 22.7.35 | 输出空闲状态寄存器 (PWMx_OISR) .....                    | 833        |
| 22.8    | 范例程序 .....                                     | 834        |
| 22.8.1  | BLDC 无刷直流电机驱动(带 HALL) .....                    | 834        |
| 22.8.2  | BLDC 无刷直流电机驱动(无 HALL), 一个比较器当 3 个比较器分时复用.....  | 843        |
| 22.8.3  | 使用高级 PWM 实现编码器.....                            | 850        |
| 22.8.4  | 正交编码器模式 .....                                  | 853        |
| 22.8.5  | 单脉冲模式 (触发控制脉冲输出) .....                         | 854        |
| 22.8.6  | 门控模式 (输入电平使能计数器) .....                         | 855        |
| 22.8.7  | 外部时钟模式 .....                                   | 857        |
| 22.8.8  | 输入捕获模式测量脉冲周期 (捕获上升沿到上升沿或者下降沿到下降沿) .....        | 859        |
| 22.8.9  | 输入捕获模式测量脉冲高电平宽度 (捕获上升沿到下降沿) .....              | 861        |
| 22.8.10 | 输入捕获模式测量脉冲低电平宽度 (捕获下降沿到上升沿) .....              | 863        |
| 22.8.11 | 输入捕获模式同时测量脉冲周期和占空比 .....                       | 865        |
| 22.8.12 | 同时捕获 4 路输入信号的周期和占空比 .....                      | 867        |
| 22.8.13 | 输出占空比为 100%和 0%的 PWM 波形的的方法 (以 PWM1P 为例) ..... | 873        |
| 22.8.14 | 带死区控制的 PWM 互补输出.....                           | 874        |
| 22.8.15 | PWM 端口做外部中断 (下降沿中断或者上升沿中断) .....               | 875        |
| 22.8.16 | 输出任意周期和任意占空比的波形 .....                          | 876        |
| 22.8.17 | 使用 PWM 的 CEN 启动 PWMA 定时器, 实时触发 ADC.....        | 876        |
| 22.8.18 | PWM 周期重复触发 ADC .....                           | 877        |
| 22.8.19 | 利用 PWM 实现 16 位 DAC 的参考线路图 .....                | 878        |
| 22.8.20 | 利用 PWM 实现互补 SPWM.....                          | 878        |
| 22.8.21 | 高级 PWM 输出-频率可调-脉冲计数 (软件方式) .....               | 882        |
| 22.8.22 | 高级 PWM 输出-频率可调-脉冲计数 (硬件方式) .....               | 885        |
| 22.8.23 | 产生 3 路相位差 120 度的互补 PWM 波形 (网友提供) .....         | 889        |
| 22.8.24 | 高级 PWM 时钟输出应用 (系统时钟 2 分频输出) .....              | 890        |
| 23      | <b>高速高级 PWM (HSPWM) .....</b>                  | <b>892</b> |
| 23.1    | 相关寄存器 .....                                    | 892        |
| 23.1.1  | HSPWM 配置寄存器 (HSPWMn_CFG) .....                 | 892        |
| 23.1.2  | HSPWM 地址寄存器 (HSPWMn_ADR) .....                 | 893        |
| 23.1.3  | HSPWM 数据寄存器 (HSPWMn_DAT) .....                 | 893        |
| 23.2    | 范例程序 .....                                     | 894        |
| 23.2.1  | 使能高级 PWM 的高速模式 (异步模式) .....                    | 894        |
| 24      | <b>USB 通用串行总线 .....</b>                        | <b>897</b> |
| 24.1    | USB 相关的寄存器 .....                               | 897        |

|         |   |            |
|---------|---|------------|
| 24.1.1  | USB 控制寄存器 (USBCON) .....                        | 898        |
| 24.1.2  | USB 时钟控制寄存器 (USBCLK) .....                      | 898        |
| 24.1.3  | USB 间址地址寄存器 (USBADR) .....                      | 899        |
| 24.1.4  | USB 间址数据寄存器 (USBDAT) .....                      | 899        |
| 24.2    | USB 控制器寄存器 (SIE) .....                          | 900        |
| 24.2.1  | USB 功能地址寄存器 (FADDR) .....                       | 901        |
| 24.2.2  | USB 电源控制寄存器 (POWER) .....                       | 901        |
| 24.2.3  | USB 端点 IN 中断标志位 (INTRIN1) .....                 | 901        |
| 24.2.4  | USB 端点 OUT 中断标志位 (INTRROUT1) .....              | 902        |
| 24.2.5  | USB 电源中断标志 (INTRUSB) .....                      | 902        |
| 24.2.6  | USB 端点 IN 中断允许寄存器 (INTRIN1E) .....              | 903        |
| 24.2.7  | USB 端点 OUT 中断允许寄存器 (INTRROUT1E) .....           | 903        |
| 24.2.8  | USB 电源中断允许寄存器 (INTRUSBE) .....                  | 903        |
| 24.2.9  | USB 数据帧号寄存器 (FRAME <sub>n</sub> ) .....         | 904        |
| 24.2.10 | USB 端点索引寄存器 (INDEX) .....                       | 904        |
| 24.2.11 | IN 端点的最大数据包大小 (INMAXP) .....                    | 904        |
| 24.2.12 | USB 端点 0 控制状态寄存器 (CSR0) .....                   | 904        |
| 24.2.13 | IN 端点控制状态寄存器 1 (INCSR1) .....                   | 905        |
| 24.2.14 | IN 端点控制状态寄存器 2 (INCSR2) .....                   | 906        |
| 24.2.15 | OUT 端点的最大数据包大小 (OUTMAXP) .....                  | 906        |
| 24.2.16 | OUT 端点控制状态寄存器 1 (OUTCSR1) .....                 | 906        |
| 24.2.17 | OUT 端点控制状态寄存器 2 (OUTCSR2) .....                 | 907        |
| 24.2.18 | USB 端点 0 的 OUT 长度 (COUNT0) .....                | 908        |
| 24.2.19 | USB 端点的 OUT 长度 (OUTCOUNT <sub>n</sub> ) .....   | 908        |
| 24.2.20 | USB 端点的 FIFO 数据访问寄存器 (FIFO <sub>n</sub> ) ..... | 908        |
| 24.3    | USB 产品开发注意事项 .....                              | 908        |
| 24.4    | 范例程序 .....                                      | 909        |
| 24.4.1  | HID 人机接口设备范例 .....                              | 909        |
| 24.4.2  | HID(Human Interface Device)协议范例 .....           | 920        |
| 24.4.3  | CDC(Communication Device Class)协议范例 .....       | 920        |
| 24.4.4  | 基于 HID 协议的 USB 键盘范例 .....                       | 920        |
| 24.4.5  | 基于 HID 协议的 USB 鼠标范例 .....                       | 920        |
| 24.4.6  | 基于 WINUSB 协议的范例 .....                           | 920        |
| 24.4.7  | MSC(Mass Storage Class)协议范例 .....               | 921        |
| 25      | <b>触摸按键控制器, 先工程样片, 可少量送样 .....</b>              | <b>922</b> |
| 25.1    | 触摸按键控制器内部框架图 .....                              | 923        |
| 25.2    | 低功耗触摸唤醒 .....                                   | 924        |
| 25.3    | 仅做触摸按键功能时的操作步骤 .....                            | 924        |
| 25.4    | 低功耗触摸按键唤醒功能的操作步骤 .....                          | 924        |
| 25.5    | 触摸按键相关的寄存器 .....                                | 925        |
| 25.5.1  | 触摸按键使能寄存器 (TSCHE <sub>Nn</sub> ) .....          | 926        |
| 25.5.2  | 触摸按键配置寄存器 (TSCFG <sub>n</sub> ) .....           | 926        |
| 25.5.3  | 触摸按键低电模式唤醒时间控制寄存器 (TSWUTC) .....                | 927        |
| 25.5.4  | 触摸按键控制寄存器 (TSCTRL) .....                        | 927        |

|           |                                |            |
|-----------|--------------------------------|------------|
| 25.5.5    | 触摸按键状态寄存器 1 (TSSTA1)           | 928        |
| 25.5.6    | 触摸按键状态寄存器 2 (TSSTA2)           | 929        |
| 25.5.7    | 触摸按键时间控制寄存器 (TSRT)             | 930        |
| 25.5.8    | 触摸按键数据寄存器 (TSDAT)              | 930        |
| 25.5.9    | 触摸按键阈值寄存器 (TSTH)               | 930        |
| 25.6      | 基本参考电路图以及注意事项                  | 931        |
| 25.7      | 范例程序                           | 932        |
| 25.7.1    | 触摸按键配置软件介绍                     | 932        |
| <b>26</b> | <b>LED 驱动器</b>                 | <b>935</b> |
| 26.1      | LED 驱动器内部框架图                   | 936        |
| 26.2      | LED 驱动相关的寄存器                   | 936        |
| 26.2.1    | COM 使能寄存器 (COMEN)              | 937        |
| 26.2.2    | SEG 使能寄存器 (SEGEN)              | 937        |
| 26.2.3    | LED 控制寄存器 (LEDCTRL)            | 937        |
| 26.2.4    | LED 时钟分频寄存器 (LEDCKS)           | 938        |
| 26.2.5    | LED 共阳模式数据寄存器 (COMn_DA)        | 939        |
| 26.2.6    | LED 共阴模式数据寄存器 (COMn_DC)        | 939        |
| 26.3      | LED 共阴模式 (LEDMODE = 00)        | 940        |
| 26.4      | LED 共阳模式 (LEDMODE = 01)        | 941        |
| 26.5      | LED 共阴/共阳模式 (LEDMODE = 10)     | 942        |
| 26.6      | 触摸按键与 LED 分时复用 I/O             | 943        |
| 26.7      | 共阴模式参考电路图                      | 945        |
| 26.8      | 共阳模式参考电路图                      | 945        |
| 26.9      | 共阴/共阳模式参考电路图 1                 | 946        |
| 26.10     | 共阴/共阳模式参考电路图 2                 | 947        |
| 26.11     | 范例程序                           | 948        |
| 26.11.1   | 共阴/共阳模式驱动 16 个 7 段数码管          | 948        |
| <b>27</b> | <b>RTC 实时时钟</b>                | <b>950</b> |
| 27.1      | RTC 相关的寄存器                     | 950        |
| 27.1.1    | RTC 控制寄存器 (RTCCR)              | 952        |
| 27.1.2    | RTC 配置寄存器 (RTCCFG)             | 952        |
| 27.1.3    | RTC 中断使能寄存器 (RTCIEN)           | 952        |
| 27.1.4    | RTC 中断请求寄存器 (RTCIF)            | 953        |
| 27.1.5    | RTC 闹钟设置寄存器                    | 953        |
| 27.1.6    | RTC 实时时钟初始值设置寄存器               | 953        |
| 27.1.7    | RTC 实时时钟计数寄存器                  | 954        |
| 27.2      | RTC 实战线路图                      | 955        |
| 27.3      | 范例程序                           | 956        |
| 27.3.1    | 串口打印 RTC 时钟范例                  | 956        |
| 27.3.2    | 利用 ISP 软件的用户接口实现不停电下载保持 RTC 参数 | 959        |
| 27.3.3    | 内部 RTC 时钟低功耗休眠唤醒-比较器检测电压程序     | 965        |
| <b>28</b> | <b>LCD 液晶驱动, 即传统的段式/笔画式屏驱动</b> | <b>970</b> |
| 28.1      | LCD 功能脚切换                      | 970        |
| 28.2      | LCD 相关的寄存器                     | 971        |

|           |  |            |
|-----------|--|------------|
| 28.2.1    | LCD 配置寄存器 (LCDCFG) .....                           | 973        |
| 28.2.2    | LCD 配置寄存器 2 (LCDCFG2) .....                        | 973        |
| 28.2.3    | 死区时间长度配置寄存器 (DBLEN) .....                          | 974        |
| 28.2.4    | COM 时间长度配置寄存器 (COMLEN <sub>x</sub> ) .....         | 974        |
| 28.2.5    | 闪烁率配置寄存器 (BLANKRATE) .....                         | 974        |
| 28.2.6    | LCD 控制寄存器 (LCDCR) .....                            | 974        |
| 28.2.7    | COM 线使能寄存器 (COMON) .....                           | 974        |
| 28.2.8    | SEG 线使能寄存器 (SEGON <sub>x</sub> ) .....             | 975        |
| 28.2.9    | LCD 数据寄存器 (CxSEGV <sub>x</sub> ) .....             | 975        |
| 28.3      | LCD 显示相关配置 .....                                   | 975        |
| 28.3.1    | 配置 LCD 刷新率 (帧速率) .....                             | 975        |
| 28.3.2    | 配置 LCD 闪烁率 .....                                   | 976        |
| 28.4      | 范例程序 .....   | 977        |
| 28.4.1    | LCD 显示 RTC 时钟范例 .....                              | 977        |
| <b>29</b> | <b>LCM 接口 (8/16 位彩屏模块 I8080/M6800 接口) .....</b>    | <b>978</b> |
| 29.1      | LCM 接口功能脚切换 .....                                  | 979        |
| 29.2      | LCM 相关的寄存器 .....                                   | 979        |
| 29.2.1    | LCM 接口配置寄存器 (LCMIFCFG) .....                       | 979        |
| 29.2.2    | LCM 接口配置寄存器 2 (LCMIFCFG2) .....                    | 980        |
| 29.2.3    | LCM 接口控制寄存器 (LCMIFCR) .....                        | 980        |
| 29.2.4    | LCM 接口状态寄存器 (LCMIFSTA) .....                       | 981        |
| 29.2.5    | LCM 接口数据寄存器 (LCMIFDATL, LCMIFDATH) .....           | 981        |
| 29.3      | I8080/M6800 模式 LCM 接口时序图 .....                     | 982        |
| 29.3.1    | I8080 模式 .....                                     | 982        |
| 29.3.2    | M6800 模式 .....                                     | 983        |
| <b>30</b> | <b>DMA (批量数据传输) .....</b>                          | <b>984</b> |
| 30.1      | DMA 相关的寄存器 .....                                   | 984        |
| 30.2      | 存储器与存储器之间的数据读写 (M2M_DMA) .....                     | 987        |
| 30.2.1    | M2M_DMA 配置寄存器 (DMA_M2M_CFG) .....                  | 987        |
| 30.2.2    | M2M_DMA 控制寄存器 (DMA_M2M_CR) .....                   | 987        |
| 30.2.3    | M2M_DMA 状态寄存器 (DMA_M2M_STA) .....                  | 987        |
| 30.2.4    | M2M_DMA 传输总字节寄存器 (DMA_M2M_AMT) .....               | 988        |
| 30.2.5    | M2M_DMA 传输完成字节寄存器 (DMA_M2M_DONE) .....             | 988        |
| 30.2.6    | M2M_DMA 发送地址寄存器 (DMA_M2M_TXA <sub>x</sub> ) .....  | 988        |
| 30.2.7    | M2M_DMA 接收地址寄存器 (DMA_M2M_RXA <sub>x</sub> ) .....  | 988        |
| 30.3      | ADC 数据自动存储 (ADC_DMA) .....                         | 989        |
| 30.3.1    | ADC_DMA 配置寄存器 (DMA_ADC_CFG) .....                  | 989        |
| 30.3.2    | ADC_DMA 控制寄存器 (DMA_ADC_CR) .....                   | 989        |
| 30.3.3    | ADC_DMA 状态寄存器 (DMA_ADC_STA) .....                  | 989        |
| 30.3.4    | ADC_DMA 接收地址寄存器 (DMA_ADC_RXA <sub>x</sub> ) .....  | 989        |
| 30.3.5    | ADC_DMA 配置寄存器 2 (DMA_ADC_CFG2) .....               | 990        |
| 30.3.6    | ADC_DMA 通道使能寄存器 (DMA_ADC_CHSW <sub>x</sub> ) ..... | 990        |
| 30.3.7    | ADC_DMA 的数据存储格式 .....                              | 991        |
| 30.4      | SPI 与存储器之间的数据交换 (SPI_DMA) .....                    | 993        |



|         |   |      |
|---------|---|------|
| 30.4.1  | SPI_DMA 配置寄存器 (DMA_SPI_CFG) .....           | 993  |
| 30.4.2  | SPI_DMA 控制寄存器 (DMA_SPI_CR) .....            | 993  |
| 30.4.3  | SPI_DMA 状态寄存器 (DMA_SPI_STA) .....           | 994  |
| 30.4.4  | SPI_DMA 传输总字节寄存器 (DMA_SPI_AMT) .....        | 994  |
| 30.4.5  | SPI_DMA 传输完成字节寄存器 (DMA_SPI_DONE) .....      | 994  |
| 30.4.6  | SPI_DMA 发送地址寄存器 (DMA_SPI_TXAx) .....        | 994  |
| 30.4.7  | SPI_DMA 接收地址寄存器 (DMA_SPI_RXAx) .....        | 994  |
| 30.4.8  | SPI_DMA 配置寄存 2 器 (DMA_SPI_CFG2) .....       | 995  |
| 30.5    | 串口 1 与存储器之间的数据交换 (UR1T_DMA, UR1R_DMA) ..... | 996  |
| 30.5.1  | UR1T_DMA 配置寄存器 (DMA_UR1T_CFG) .....         | 996  |
| 30.5.2  | UR1T_DMA 控制寄存器 (DMA_UR1T_CR) .....          | 996  |
| 30.5.3  | UR1T_DMA 状态寄存器 (DMA_UR1T_STA) .....         | 996  |
| 30.5.4  | UR1T_DMA 传输总字节寄存器 (DMA_UR1T_AMT) .....      | 997  |
| 30.5.5  | UR1T_DMA 传输完成字节寄存器 (DMA_UR1T_DONE) .....    | 997  |
| 30.5.6  | UR1T_DMA 发送地址寄存器 (DMA_UR1T_TXAx) .....      | 997  |
| 30.5.7  | UR1R_DMA 配置寄存器 (DMA_UR1R_CFG) .....         | 997  |
| 30.5.8  | UR1R_DMA 控制寄存器 (DMA_UR1R_CR) .....          | 997  |
| 30.5.9  | UR1R_DMA 状态寄存器 (DMA_UR1R_STA) .....         | 998  |
| 30.5.10 | UR1R_DMA 传输总字节寄存器 (DMA_UR1R_AMT) .....      | 998  |
| 30.5.11 | UR1R_DMA 传输完成字节寄存器 (DMA_UR1R_DONE) .....    | 998  |
| 30.5.12 | UR1R_DMA 接收地址寄存器 (DMA_UR1R_RXAx) .....      | 998  |
| 30.6    | 串口 2 与存储器之间的数据交换 (UR2T_DMA, UR2R_DMA) ..... | 999  |
| 30.6.1  | UR2T_DMA 配置寄存器 (DMA_UR2T_CFG) .....         | 999  |
| 30.6.2  | UR2T_DMA 控制寄存器 (DMA_UR2T_CR) .....          | 999  |
| 30.6.3  | UR2T_DMA 状态寄存器 (DMA_UR2T_STA) .....         | 999  |
| 30.6.4  | UR2T_DMA 传输总字节寄存器 (DMA_UR2T_AMT) .....      | 1000 |
| 30.6.5  | UR2T_DMA 传输完成字节寄存器 (DMA_UR2T_DONE) .....    | 1000 |
| 30.6.6  | UR2T_DMA 发送地址寄存器 (DMA_UR2T_TXAx) .....      | 1000 |
| 30.6.7  | UR2R_DMA 配置寄存器 (DMA_UR2R_CFG) .....         | 1000 |
| 30.6.8  | UR2R_DMA 控制寄存器 (DMA_UR2R_CR) .....          | 1000 |
| 30.6.9  | UR2R_DMA 状态寄存器 (DMA_UR2R_STA) .....         | 1001 |
| 30.6.10 | UR2R_DMA 传输总字节寄存器 (DMA_UR2R_AMT) .....      | 1001 |
| 30.6.11 | UR2R_DMA 传输完成字节寄存器 (DMA_UR2R_DONE) .....    | 1001 |
| 30.6.12 | UR2R_DMA 接收地址寄存器 (DMA_UR2R_RXAx) .....      | 1001 |
| 30.7    | 串口 3 与存储器之间的数据交换 (UR3T_DMA, UR3R_DMA) ..... | 1002 |
| 30.7.1  | UR3T_DMA 配置寄存器 (DMA_UR3T_CFG) .....         | 1002 |
| 30.7.2  | UR3T_DMA 控制寄存器 (DMA_UR3T_CR) .....          | 1002 |
| 30.7.3  | UR3T_DMA 状态寄存器 (DMA_UR3T_STA) .....         | 1002 |
| 30.7.4  | UR3T_DMA 传输总字节寄存器 (DMA_UR3T_AMT) .....      | 1003 |
| 30.7.5  | UR3T_DMA 传输完成字节寄存器 (DMA_UR3T_DONE) .....    | 1003 |
| 30.7.6  | UR3T_DMA 发送地址寄存器 (DMA_UR3T_TXAx) .....      | 1003 |
| 30.7.7  | UR3R_DMA 配置寄存器 (DMA_UR3R_CFG) .....         | 1003 |
| 30.7.8  | UR3R_DMA 控制寄存器 (DMA_UR3R_CR) .....          | 1003 |
| 30.7.9  | UR3R_DMA 状态寄存器 (DMA_UR3R_STA) .....         | 1004 |

|             |   |             |
|-------------|---|-------------|
| 30.7.10     | UR3R_DMA 传输总字节寄存器 (DMA_UR3R_AMT) .....      | 1004        |
| 30.7.11     | UR3R_DMA 传输完成字节寄存器 (DMA_UR3R_DONE) .....    | 1004        |
| 30.7.12     | UR3R_DMA 接收地址寄存器 (DMA_UR3R_RXAx) .....      | 1004        |
| 30.8        | 串口 4 与存储器之间的数据交换 (UR4T_DMA, UR4R_DMA) ..... | 1005        |
| 30.8.1      | UR4T_DMA 配置寄存器 (DMA_UR4T_CFG) .....         | 1005        |
| 30.8.2      | UR4T_DMA 控制寄存器 (DMA_UR4T_CR) .....          | 1005        |
| 30.8.3      | UR4T_DMA 状态寄存器 (DMA_UR4T_STA) .....         | 1005        |
| 30.8.4      | UR4T_DMA 传输总字节寄存器 (DMA_UR4T_AMT) .....      | 1006        |
| 30.8.5      | UR4T_DMA 传输完成字节寄存器 (DMA_UR4T_DONE) .....    | 1006        |
| 30.8.6      | UR4T_DMA 发送地址寄存器 (DMA_UR4T_TXAx) .....      | 1006        |
| 30.8.7      | UR4R_DMA 配置寄存器 (DMA_UR4R_CFG) .....         | 1006        |
| 30.8.8      | UR4R_DMA 控制寄存器 (DMA_UR4R_CR) .....          | 1006        |
| 30.8.9      | UR4R_DMA 状态寄存器 (DMA_UR4R_STA) .....         | 1007        |
| 30.8.10     | UR4R_DMA 传输总字节寄存器 (DMA_UR4R_AMT) .....      | 1007        |
| 30.8.11     | UR4R_DMA 传输完成字节寄存器 (DMA_UR4R_DONE) .....    | 1007        |
| 30.8.12     | UR4R_DMA 接收地址寄存器 (DMA_UR4R_RXAx) .....      | 1007        |
| 30.9        | LCM 与存储器之间的数据读写 (LCM_DMA) .....             | 1008        |
| 30.9.1      | LCM_DMA 配置寄存器 (DMA_LCM_CFG) .....           | 1008        |
| 30.9.2      | LCM_DMA 控制寄存器 (DMA_LCM_CR) .....            | 1008        |
| 30.9.3      | LCM_DMA 状态寄存器 (DMA_LCM_STA) .....           | 1009        |
| 30.9.4      | LCM_DMA 传输总字节寄存器 (DMA_LCM_AMT) .....        | 1009        |
| 30.9.5      | LCM_DMA 传输完成字节寄存器 (DMA_LCM_DONE) .....      | 1009        |
| 30.9.6      | LCM_DMA 发送地址寄存器 (DMA_LCM_TXAx) .....        | 1009        |
| 30.9.7      | LCM_DMA 接收地址寄存器 (DMA_LCM_RXAx) .....        | 1009        |
| 30.10       | 范例程序 .....                                  | 1010        |
| 30.10.1     | 串口 1 中断模式与电脑收发测试 - DMA 接收超时中断 .....         | 1010        |
| 30.10.2     | 串口 1 中断模式与电脑收发测试 - DMA 数据校验 .....           | 1015        |
| <b>31</b>   | <b>增强型双数据指针 .....</b>                       | <b>1022</b> |
| 31.1        | 相关的特殊功能寄存器 .....                            | 1022        |
| 31.1.1      | 第 1 组 16 位数据指针寄存器 (DPTR0) .....             | 1022        |
| 31.1.2      | 第 2 组 16 位数据指针寄存器 (DPTR1) .....             | 1022        |
| 31.1.3      | 数据指针控制寄存器 (DPS) .....                       | 1022        |
| 31.1.4      | 数据指针控制寄存器 (TA) .....                        | 1023        |
| 31.2        | 范例程序 .....                                  | 1025        |
| 31.2.1      | 示例代码 1 .....                                | 1025        |
| 31.2.2      | 示例代码 2 .....                                | 1026        |
| <b>32</b>   | <b>MDU16 硬件 16 位乘法器 .....</b>               | <b>1028</b> |
| 32.1        | 相关的特殊功能寄存器 .....                            | 1028        |
| 32.1.1      | 操作数 1 数据寄存器 (MD0~MD3) .....                 | 1028        |
| 32.1.2      | 操作数 2 数据寄存器 (MD4~MD5) .....                 | 1029        |
| 32.1.3      | MDU 模式控制寄存器 (ARCON), 运算所需时钟数 .....          | 1029        |
| 32.1.4      | MDU 操作控制寄存器 (OPCON) .....                   | 1030        |
| 32.2        | 范例程序 .....                                  | 1031        |
| <b>附录 A</b> | <b>编译器 (汇编器) / 仿真器 / 头文件使用指南 .....</b>      | <b>1032</b> |

|        |   |      |
|--------|---|------|
| 附录 B   | STC-ISP 下载软件高级应用.....                       | 1039 |
| B.1    | 发布项目程序.....                                 | 1039 |
| B.2    | 程序加密后传输（防烧录时串口分析出程序）.....                   | 1043 |
| B.3    | 发布项目程序+程序加密后传输结合使用.....                     | 1047 |
| B.4    | 用户自定义下载（实现不停电下载）.....                       | 1048 |
| 附录 C   | 如何测试 I/O 口.....                             | 1052 |
| 附录 D   | 如何让传统的 8051 单片机学习板可仿真.....                  | 1053 |
| 附录 E   | STC 软件模拟硬件 USB 设备驱动程序安装说明.....              | 1055 |
| 附录 F   | USB 下载步骤演示.....                             | 1119 |
| 附录 G   | USB 仿真步骤演示.....                             | 1123 |
| 附录 H   | RS485 自动控制或 I/O 口控制线路图.....                 | 1128 |
| 附录 I   | STC 工具使用说明书.....                            | 1129 |
| I.1    | 概述.....                                     | 1129 |
| I.2    | 系统可编程（ISP）流程说明.....                         | 1129 |
| I.3    | USB 型联机/脱机下载工具 U8W/U8W-Mini.....            | 1130 |
| I.3.1  | 安装 U8W/U8W-Mini 驱动程序.....                   | 1132 |
| I.3.2  | U8W 的功能介绍.....                              | 1135 |
| I.3.3  | U8W 的在线联机下载使用说明.....                        | 1136 |
| I.3.4  | U8W 的脱机下载使用说明.....                          | 1139 |
| I.3.5  | U8W-Mini 的功能介绍.....                         | 1147 |
| I.3.6  | U8W-Mini 的在线联机下载使用说明.....                   | 1148 |
| I.3.7  | U8W-Mini 的脱机下载使用说明.....                     | 1149 |
| I.3.8  | 制作/更新 U8W/U8W-Mini.....                     | 1155 |
| I.3.9  | U8W/U8W-Mini 设置直通模式（可用于仿真）.....             | 1157 |
| I.3.10 | U8W/U8W-Mini 的参考电路.....                     | 1157 |
| I.4    | STC 通用 USB 转串口工具.....                       | 1159 |
| I.4.1  | STC 通用 USB 转串口工具外观图.....                    | 1159 |
| I.4.2  | STC 通用 USB 转串口工具布局图.....                    | 1160 |
| I.4.3  | STC 通用 USB 转串口工具驱动安装.....                   | 1161 |
| I.4.4  | 使用 STC 通用 USB 转串口工具下载程序到 MCU.....           | 1162 |
| I.4.5  | 使用 STC 通用 USB 转串口工具仿真用户代码.....              | 1164 |
| I.5    | 应用线路图.....                                  | 1171 |
| I.5.1  | U8W 工具应用参考线路图.....                          | 1171 |
| I.5.2  | STC 通用 USB 转串口工具应用参考线路图.....                | 1171 |
| 附录 J   | STC 仿真使用说明书.....                            | 1173 |
| J.1    | 概述.....                                     | 1173 |
| J.2    | 安装 Keil 软件.....                             | 1174 |
| J.3    | 安装仿真驱动.....                                 | 1175 |
| J.4    | 串口直接仿真.....                                 | 1178 |
| J.4.1  | 制作串口仿真芯片.....                               | 1178 |
| J.4.2  | 在 Keil 软件中进行串口仿真设置.....                     | 1181 |
| J.4.3  | 在 Keil 软件中使用串口进行仿真.....                     | 1183 |
| J.5    | USB 直接仿真（目前只有 STC8H8K64U-B/C/D 版本芯片支持）..... | 1185 |
| J.5.1  | 制作 USB 仿真芯片.....                            | 1185 |



|        |  |      |
|--------|--|------|
| J.5.2  | 在 Keil 软件中进行 USB 仿真设置 .....                | 1190 |
| J.5.3  | 在 Keil 软件中使用 USB 进行仿真 .....                | 1192 |
| 附录 K   | U8W 下载工具中 RS485 部分线路图 .....                | 1194 |
| 附录 L   | 运行用户程序时收到用户命令后自动启动 ISP 下载(不停电) .....       | 1195 |
| 附录 M   | 使用 STC 的 IAP 系列单片机开发自己的 ISP 程序 .....       | 1197 |
| 附录 N   | 用户程序复位到系统区进行 ISP 下载的方法 (不停电) .....         | 1209 |
| 附录 O   | 单片机电源系统最简易自我保护电路 .....                     | 1215 |
| 附录 P   | 使用第三方 MCU 对 STC8H 系列单片机进行 ISP 下载范例程序 ..... | 1216 |
| P.1    | 电源控制参考电路 .....                             | 1216 |
| P.2    | 通信协议流程图 .....                              | 1217 |
| P.3    | 参考代码 (C 语言) .....                          | 1220 |
| 附录 Q   | 使用第三方应用程序调用 STC 发布项目程序对单片机进行 ISP 下载 .....  | 1228 |
| 附录 R   | STC8H 系列正交解码示例 (成都逐飞科技友情提供) .....          | 1232 |
| 附录 S   | 在 Keil 中建立多文件项目的方法 .....                   | 1236 |
| 附录 T   | 关于中断号大于 31 在 Keil 中编译出错的处理 .....           | 1240 |
| T.1    | 使用网上流行的中断号拓展工具 .....                       | 1240 |
| T.2    | 使用保留中断号进行中转 .....                          | 1242 |
| 附录 U   | 电气特性 .....                                 | 1252 |
| U.1    | 绝对最大额定值 .....                              | 1252 |
| U.2    | 直流特性 (3.3V) .....                          | 1253 |
| U.3    | 直流特性 (5.0V) .....                          | 1255 |
| U.4    | 内部 IRC 温漂特性 (参考温度 25℃) .....               | 1256 |
| U.5    | 低压复位门槛电压 (测试温度 25℃) .....                  | 1256 |
| 附录 V   | 应用注意事项 .....                               | 1257 |
| V.1    | 关于 STC8H 系列 IO 口的注意事项 .....                | 1257 |
| V.2    | STC8H8K64U 系列 .....                        | 1257 |
| V.3    | STC8H3K64S2/S4 系列 .....                    | 1257 |
| V.4    | STC8H4K64TLR 系列 .....                      | 1257 |
| V.5    | STC8H2K64T 系列 .....                        | 1258 |
| 附录 W   | 触摸按键的 PCB 设计指导 .....                       | 1259 |
| 附录 X   | QFN/DFN 封装元器件焊接方法 .....                    | 1261 |
| 附录 Y   | 关于回流焊前是否要烘烤 .....                          | 1264 |
| 附录 Z   | 如何使用万用表检测芯片 I/O 口好坏 .....                  | 1265 |
| 附录 AA  | 大批量生产, 如何省去专门的烧录人员, 如何无烧录环节 .....          | 1266 |
| 附录 BB  | 关于 Keil 软件中 0xFD 问题的说明 .....               | 1267 |
| 附录 CC  | 如何使用 STC-ISP 下载软件制作和编辑 EEPROM 文件 .....     | 1268 |
| 附录 DD  | STC8H 系列头文件定义 .....                        | 1270 |
| 附录 EE  | 单片机是否可以提供裸芯 .....                          | 1289 |
| 附录 FF  | STC8H 系列单片机取代 STC15 系列的注意事项 .....          | 1290 |
| 附录 GG  | STC8H 系列单片机取代 STC8A/8F 系列的注意事项 .....       | 1292 |
| 附录 HH  | 内部测试型号 .....                               | 1293 |
| HH.1   | STC8H2K64T-351-LQFP48/QFN48 .....          | 1293 |
| HH.1.1 | 特性及价格 .....                                | 1293 |
| HH.1.2 | 管脚图, 最小系统 .....                            | 1296 |

|        |  |      |
|--------|--|------|
| HH.2   | STC8H4K64LCD-45I-LQFP64/QFN64/LQFP48/QFN48 .....               | 1297 |
| HH.2.1 | 特性及价格（有 16 位硬件乘除法器 MDU16，准 16 位单片机） .....                      | 1297 |
| HH.2.2 | 管脚图，最小系统 .....   | 1300 |
| HH.3   | STC8H4K64TLR-45I-LQFP48/32/QFN48/TSSOP20(触摸按键/LED/RTC 系列)..... | 1302 |
| HH.3.1 | 特性及价格（有 16 位硬件乘除法器 MDU16，准 16 位单片机） .....                      | 1302 |
| HH.3.2 | 管脚图，最小系统 .....   | 1306 |
| 附录 II  | 更新记录 .....   | 1310 |
| 附录 JJ  | STC8 系列命名花絮 .....  | 1324 |

# 1 概述

STC8H 系列单片机是不需要外部晶振和外部复位的单片机, 是以超强抗干扰/超低价/高速/低功耗为目标的 8051 单片机, 在相同的工作频率下, STC8H 系列单片机比传统的 8051 约快 12 倍(速度快 11.2~13.2 倍), 依次按顺序执行完全部的 111 条指令, STC8H 系列单片机仅需 147 个时钟, 而传统 8051 则需要 1944 个时钟。STC8H 系列单片机是 STC 生产的单时钟/机器周期(1T)的单片机, 是宽电压/高速/高可靠/低功耗/强抗静电/较强抗干扰的新一代 8051 单片机, 超级加密。指令代码完全兼容传统 8051。

MCU 内部集成高精度 R/C 时钟( $\pm 0.3\%$ , 常温下 $+25^{\circ}\text{C}$ ),  $-1.38\% \sim +1.42\%$  温飘( $-40^{\circ}\text{C} \sim +85^{\circ}\text{C}$ ),  $-0.88\% \sim +1.05\%$  温飘( $-20^{\circ}\text{C} \sim +65^{\circ}\text{C}$ )。ISP 编程时 4MHz~35MHz 宽范围可设置(注意: 温度范围为 $-40^{\circ}\text{C} \sim +85^{\circ}\text{C}$ 时, 最高频率须控制在 35MHz 以下), 可彻底省掉外部昂贵的晶振和外部复位电路(内部已集成高可靠复位电路, ISP 编程时 4 级复位门槛电压可选)。

MCU 内部有 3 个可选时钟源: 内部高精度 IRC 时钟(可适当调高或调低)、内部 32KHz 的低速 IRC、外部 4M~33M 晶振或外部时钟信号。用户代码中可自由选择时钟源, 时钟源选定后可再经过 8-bit 的分频器分频后再将时钟信号提供给 CPU 和各个外设(如定时器、串口、SPI 等)。

MCU 提供两种低功耗模式: IDLE 模式和 STOP 模式。IDLE 模式下, MCU 停止给 CPU 提供时钟, CPU 无时钟, CPU 停止执行指令, 但所有的外设仍处于工作状态, 此时功耗约为 1.3mA (6MHz 工作频率)。STOP 模式即为主时钟停振模式, 即传统的掉电模式/停电模式/停机模式, 此时 CPU 和全部外设都停止工作, 功耗可降低到  $0.6\mu\text{A}@V_{\text{CC}}=5.0\text{V}$ ,  $0.4\mu\text{A}@V_{\text{CC}}=3.3\text{V}$ 。

掉电模式可以使用 INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、T3(P0.4)、T4(P0.6)、RXD(P3.0/P3.6/P1.6/P4.3)、RXD2(P1.0/P4.6)、RXD3(P0.0/P5.0)、RXD4(P0.2/P5.2)、I2C\_SDA(P1.4/P2.4/P3.3)以及比较器中断、低压检测中断、掉电唤醒定时器唤醒。

MCU 提供了丰富的数字外设(串口、定时器、高级 PWM 以及 I<sup>2</sup>C、SPI、USB)接口与模拟外设(超高速 ADC、比较器), 可满足广大用户的设计需求。

STC8H 系列单片机内部集成了增强型的双数据指针。通过程序控制, 可实现数据指针自动递增或递减功能以及两组数据指针的自动切换功能。

| 产品线                 | I/O | UART | 定时器 | ADC                               | 高级 PWM | CMP | SPI I2C | USB | MDU16 | LED 驱动 | Touch Key | RTC | I/O 中断 | 彩屏 LCM | LCD 驱动 | DMA |
|---------------------|-----|------|-----|-----------------------------------|--------|-----|---------|-----|-------|--------|-----------|-----|--------|--------|--------|-----|
| STC8H1K08 系列        | 17  | 2    | 3   | 9 <sub>CH</sub> *10 <sub>B</sub>  | ●      | ●   | ●       |     |       |        |           |     |        |        |        |     |
| STC8H1K28 系列        | 29  | 2    | 5   | 12 <sub>CH</sub> *10 <sub>B</sub> | ●      | ●   | ●       |     |       |        |           |     |        |        |        |     |
| STC8H3K64S4 系列      | 45  | 4    | 5   | 12 <sub>CH</sub> *12 <sub>B</sub> | ●      | ●   | ●       |     | ●     |        |           |     | ●      |        |        |     |
| STC8H3K64S2 系列      | 45  | 2    | 5   | 12 <sub>CH</sub> *12 <sub>B</sub> | ●      | ●   | ●       |     | ●     |        |           |     | ●      |        |        |     |
| STC8H8K64U-A 版本     | 60  | 4    | 5   | 15 <sub>CH</sub> *12 <sub>B</sub> | ●      | ●   | ●       | ●   | ●     |        |           |     |        |        |        |     |
| STC8H8K64U-B/C/D 版本 | 60  | 4    | 5   | 15 <sub>CH</sub> *12 <sub>B</sub> | ●      | ●   | ●       | ●   | ●     |        |           | ●   | ●      | ●      |        | ●   |
| STC8H4K64TL 系列      | 44  | 4    | 5   | 15 <sub>CH</sub> *12 <sub>B</sub> | ●      | ●   | ●       |     | ●     | ●      | ●         | ●   | ●      | ●      |        | ●   |
| STC8H4K64TLCD 系列    | 60  | 4    | 5   | 15 <sub>CH</sub> *12 <sub>B</sub> | ●      | ●   | ●       |     | ●     |        | ●         | ●   | ●      | ●      | ●      | ●   |
| STC8H1K08T 系列       | 16  | 2    | 3   | 15 <sub>CH</sub> *12 <sub>B</sub> | ●      | ●   | ●       |     | ●     |        | ●         | ●   | ●      |        |        | ●   |

## 2 特性、价格及管脚

### 2.1 STC8H1K08-36I-TSSOP20/QFN20 系列

#### 2.1.1 特性及价格

➤ 选型价格（不需要外部晶振、不需要外部复位，10 位 ADC，9 通道）

| 单片机型号     | 工作电压 (V) | Flash 程序存储器 10 万次 字节 | idata <sup>1</sup> 内部传统 8051 RAM 字节 | xdata <sup>2</sup> 内部大容量扩展 SRAM 字节 | 强大的双 DTR 可增可减 | EEPROM 10 万次 字节 | I/O 口最多数量 | 传统 I/O 中断 (INT0/INT1/INT2/INT3/INT4) 并可掉电唤醒 | 串口并可掉电唤醒 | SPI | I <sup>2</sup> C 并可掉电唤醒 | 定时器计数器 (T0/T1/T2 管脚也可掉电唤醒) | 16 位高级 PWM 定时器 互补对称死区控制 | 掉电唤醒专用定时器 | 9 路高速 ADC (8 路 PWM 可当 8 路 D/A 使用) | 比较器 (可当 1 路 A/D <sup>3</sup> 可作外部掉电检测) | 内部低压检测中断并可掉电唤醒 | 看门狗 复位定时器 | 内部高可靠复位 (可选复位门檻电压) | 内部高精度时钟 (36MHz 以下可调) 追频 | 可对外输出时钟及复位 | 程序加密后传输 (防拦截) | 可设置下次更新程序需口令 | 支持 255 下载 | 支持软件模拟硬件 I2S 直接下载 | 本身就可在线仿真 | 价格及封装           |                       | 主力产品供货信息 |
|-----------|----------|----------------------|-------------------------------------|------------------------------------|---------------|-----------------|-----------|---|----------|-----|-------------------------|----------------------------|-------------------------|-----------|-----------------------------------|--|----------------|-----------|--------------------|-------------------------|------------|---------------|--------------|-----------|-------------------|----------|-----------------|-----------------------|----------|
|           |          |                      |                                     |                                    |               |                 |           |   |          |     |                         |                            |                         |           |                                   |  |                |           |                    |                         |            |               |              |           |                   |          | QFN20 <3mm*3mm> | TSSOP20 <6.5mm*6.5mm> |          |
| STC8H1K08 | 1.9-5.5  | 8K                   | 256                                 | 1K                                 | 2             | 4K              | 17        | 有   | 2        | 有   | 有                       | 3                          | 8                       | 有         | 10 位                              | 有                                      | 有              | 有         | 4 级                | 有                       | 是          | 有             | 是            | 是         | 是                 | 是        | ¥1.3            | ¥1.3                  | 现货       |
| STC8H1K17 | 1.9-5.5  | 17K                  | 256                                 | 1K                                 | 2             | IAP             | 17        | 有   | 2        | 有   | 有                       | 3                          | 8                       | 有         | 10 位                              | 有                                      | 有              | 有         | 4 级                | 有                       | 是          | 有             | 是            | 是         | -                 | -        | ¥1.45           | ¥1.45                 |          |

- 内核
  - ✓ 超高速 8051 内核 (1T)，比传统 8051 约快 12 倍以上
  - ✓ 指令代码完全兼容传统 8051
  - ✓ 17 个中断源，4 级中断优先级
  - ✓ 支持在线仿真
- 工作电压
  - ✓ 1.9V~5.5V
- 工作温度
  - ✓ -40℃~85℃（芯片为-40℃~125℃制程，超温度范围应用请参考电气特性章节说明）
- Flash 存储器
  - ✓ 最大 17K 字节 FLASH 程序存储器（ROM），用于存储用户代码
  - ✓ 支持用户配置 EEPROM 大小，512 字节单页擦除，擦写次数可达 10 万次以上
  - ✓ 支持在系统编程方式（ISP）更新用户应用程序，无需专用编程器
  - ✓ 支持单芯片仿真，无需专用仿真器，理论断点个数无限制
- SRAM
  - ✓ 128 字节内部直接访问 RAM（DATA，C 语言程序中使用 data 关键字进行声明）
  - ✓ 128 字节内部间接访问 RAM（IDATA，C 语言程序中使用 idata 关键字进行声明）
  - ✓ 1024 字节内部扩展 RAM（内部 XDATA，C 语言程序中使用 xdata 关键字进行声明）

## ➤ 时钟控制

- ✓ 内部高精度 IRC (4MHz~36MHz, ISP 编程时选择或手动输入, 还可以用户软件分频到较低的频率工作, 如 100KHz)
  - ⊕ 误差±0.3% (常温下 25℃)
  - ⊕ -1.35%~+1.30%温漂 (全温度范围, -40℃~85℃)
  - ⊕ -0.76%~+0.98%温漂 (温度范围, -20℃~65℃)
- ✓ 内部 32KHz 低速 IRC (误差较大)
- ✓ 外部晶振 (4MHz~36MHz) 和外部时钟  
用户可自由选择上面的 3 种时钟源

## ➤ 复位

- ✓ 硬件复位
  - ⊕ 上电复位。(在芯片未使能低压复位功能时有效)
  - ⊕ 复位脚复位。出厂时 P5.4 默认为 I/O 口, ISP 下载时可将 P5.4 管脚设置为复位脚 (注意: 当设置 P5.4 管脚为复位脚时, 复位电平为低电平)
  - ⊕ 看门狗溢出复位
  - ⊕ 低压检测复位, 提供 4 级低压检测电压: 2.0V、2.4V、2.7V、3.0V。
- ✓ 软件复位
  - ⊕ 软件方式写复位触发寄存器

## ➤ 中断

- ✓ 提供 17 个中断源: INT0 (支持上升沿和下降沿中断)、INT1 (支持上升沿和下降沿中断)、INT2 (只支持下降沿中断)、INT3 (只支持下降沿中断)、INT4 (只支持下降沿中断)、定时器 0、定时器 1、定时器 2、串口 1、串口 2、ADC 模数转换、LVD 低压检测、SPI、I<sup>2</sup>C、比较器、PWMA、PWMB
- ✓ 提供 4 级中断优先级
- ✓ 时钟停振模式下可以唤醒的中断: INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、RXD(P3.0/P3.6/P1.6)、RXD2(P1.0)、I2C\_SDA(P1.4/P3.3) 以及比较器中断、低压检测中断、掉电唤醒定时器唤醒。

## ➤ 数字外设

- ✓ 3 个 16 位定时器: 定时器 0、定时器 1、定时器 2, 其中定时器 0 的模式 3 具有 NMI (不可屏蔽中断) 功能, 定时器 0 和定时器 1 的模式 0 为 16 位自动重载模式
- ✓ 2 个高速串口: 串口 1、串口 2, 波特率时钟源最快可为 FOSC/4
- ✓ 8 路/2 组高级 PWM, 可实现带死区的控制信号, 并支持外部异常检测功能, 另外还支持 16 位定时器、8 个外部中断、8 路外部捕获测量脉宽等功能
- ✓ SPI: 支持主机模式和从机模式以及主机/从机自动切换
- ✓ I<sup>2</sup>C: 支持主机模式和从机模式

## ➤ 模拟外设

- ✓ 超高速 ADC, 支持 10 位高精度 9 通道 (通道 0~通道 1、通道 8~通道 14) 的模数转换, 速度最快能达到 500K (每秒进行 50 万次 ADC 转换)
- ✓ ADC 的通道 15 用于测试内部 1.19V 参考信号源 (芯片在出厂时, 内部参考信号源已调整为 1.19V)
- ✓ 比较器, 一组比较器 (比较器的正端可选择 CMP+ 端口和所有的 ADC 输入端口, 所以比较器可当作多路比较器进行分时复用)
- ✓ DAC: 8 路高级 PWM 定时器可当 8 路 DAC 使用

## ➤ GPIO

- ✓ 最多可达 17 个 GPIO: P1.0~P1.7、P3.0~P3.7、P5.4

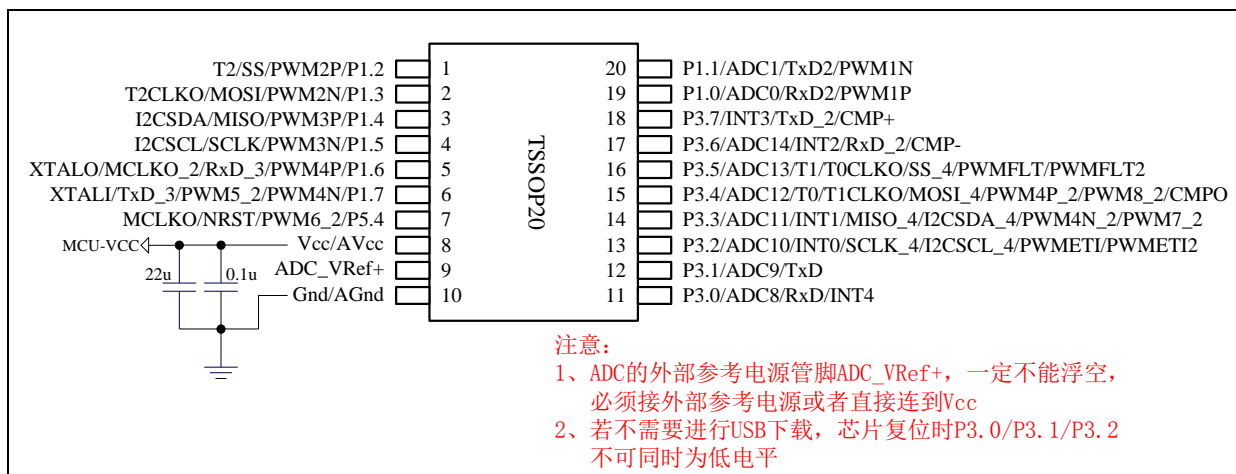
- ✓ 所有的 GPIO 均支持如下 4 种模式: 准双向口模式、强推挽输出模式、开漏输出模式、高阻输入模式
- ✓ 除 P3.0 和 P3.1 外, 其余所有 IO 口上电后的状态均为高阻输入状态, 用户在使用 IO 口时必须先设置 IO 口模式。另外每个 I/O 均可独立使能内部 4K 上拉电阻

➤ 封装

- ✓ TSSOP20 <6.5mm\*6.5mm>、QFN20 <3mm\*3mm>

STC MCU

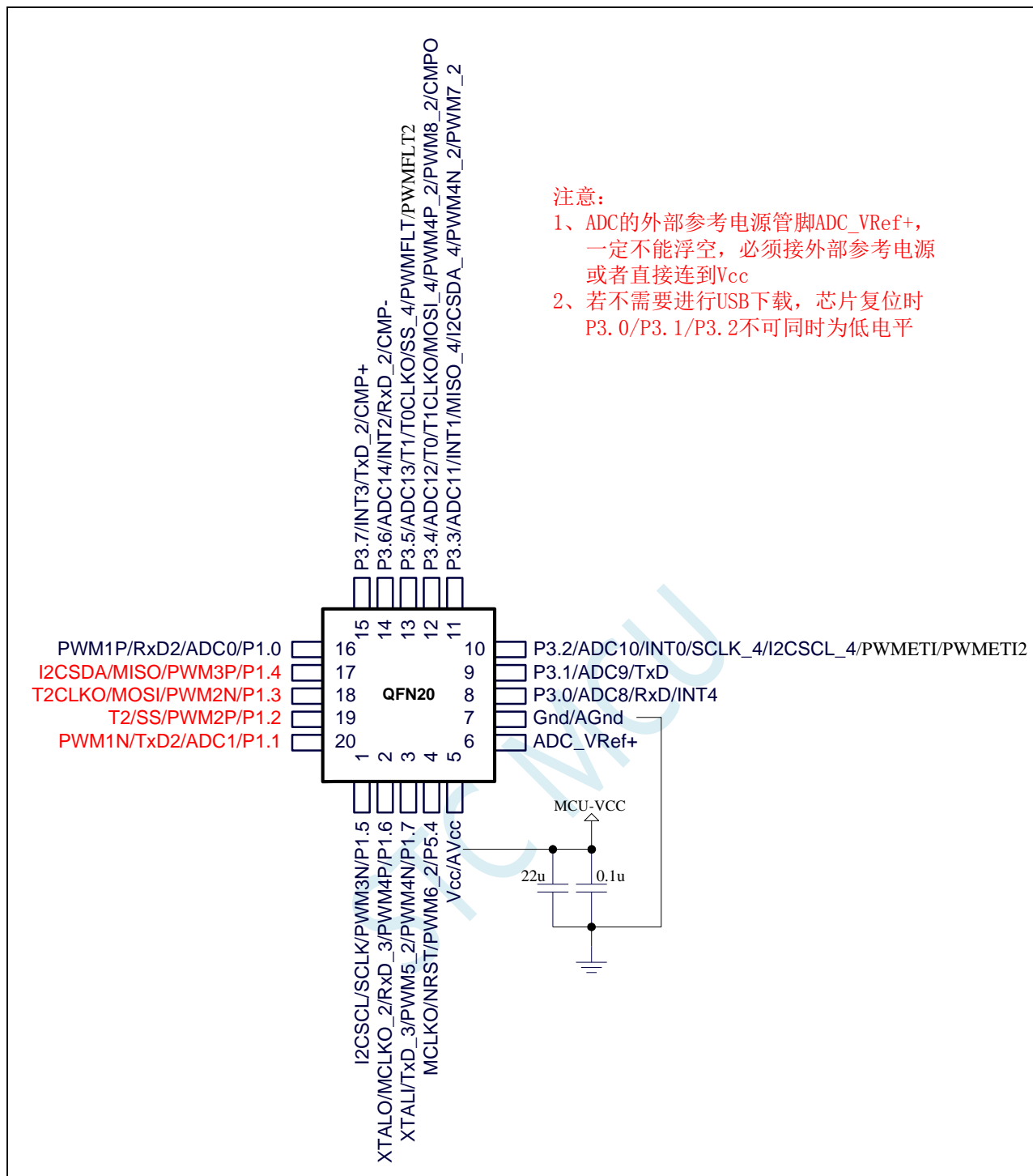
## 2.1.2 管脚图，最小系统



- 注意:
- 1、除 P3.0 和 P3.1 外, 其余所有 I/O 口上电后的状态均为高阻输入状态, 用户在使用 I/O 口时必须先设置 I/O 口模式
  - 2、所有的 I/O 口均可以设置为准双向口模式、强推挽输出模式、开漏输出模式或者高阻输入模式, 另外每个 I/O 均可独立使能内部 4K 上拉电阻
  - 3、当使能 P5.4 口为复位脚时, 复位电平为低电平

建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF, 可去除电源线噪声, 提高抗干扰能力





## 注意:

- 1、ADC的外部参考电源管脚ADC\_VRef+, 一定不能浮空, 必须接外部参考电源或者直接连到Vcc
- 2、若不需要进行USB下载, 芯片复位时 P3.0/P3.1/P3.2不可同时为低电平

正看芯片丝印左下方小圆点处为第一脚

正看芯片丝印最下面一行最后一个字母为芯片版本号

建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF, 可去除电源线噪声, 提高抗干扰能力

## 2.1.3 管脚说明

| 编号      |       | 名称        | 类型  | 说明                                 |
|---------|-------|-----------|-----|------------------------------------|
| TSSOP20 | QFN20 |           |     |                                    |
| 1       | 19    | P1.2      | I/O | 标准 IO 口                            |
|         |       | SS        | I/O | SPI 从机选择                           |
|         |       | T2        | I   | 定时器 2 外部时钟输入                       |
|         |       | PWM2P     | I/O | PWM2 的捕获输入和脉冲输出正极                  |
| 2       | 18    | P1.3      | I/O | 标准 IO 口                            |
|         |       | MOSI      | I/O | SPI 主机输出从机输入                       |
|         |       | T2CLKO    | O   | 定时器 2 时钟分频输出                       |
|         |       | PWM2N     | I/O | PWM2 的捕获输入和脉冲输出负极                  |
| 3       | 17    | P1.4      | I/O | 标准 IO 口                            |
|         |       | MISO      | I/O | SPI 主机输入从机输出                       |
|         |       | SDA       | I/O | I2C 接口的数据线                         |
|         |       | PWM3P     | I/O | PWM3 捕获输入和脉冲输出正极                   |
| 4       | 1     | P1.5      | I/O | 标准 IO 口                            |
|         |       | SCLK      | I/O | SPI 的时钟脚                           |
|         |       | SCL       | I/O | I2C 的时钟线                           |
|         |       | PWM3N     | I/O | PWM3 的捕获输入和脉冲输出负极                  |
| 5       | 2     | P1.6      | I/O | 标准 IO 口                            |
|         |       | RxD_3     | I   | 串口 1 的接收脚                          |
|         |       | PWM4P     | I/O | PWM4 捕获输入和脉冲输出正极                   |
|         |       | MCLKO_2   | O   | 主时钟分频输出                            |
|         |       | XTALO     | O   | 外部晶振的输出脚                           |
| 6       | 3     | P1.7      | I/O | 标准 IO 口                            |
|         |       | TxD_3     | O   | 串口 1 的发送脚                          |
|         |       | PWM4N     | I/O | PWM4 的捕获输入和脉冲输出负极                  |
|         |       | PWM5_2    | I/O | PWM5 捕获输入和脉冲输出正极                   |
|         |       | XTALI     | I   | 外部晶振/外部时钟的输入脚                      |
| 7       | 4     | P5.4      | I/O | 标准 IO 口                            |
|         |       | NRST      | I   | 复位引脚（低电平复位）                        |
|         |       | MCLKO     | O   | 主时钟分频输出                            |
|         |       | PWM6_2    | I/O | PWM6 捕获输入和脉冲输出正极                   |
| 8       | 5     | Vcc       | Vcc | 电源脚                                |
|         |       | AVcc      | Vcc | ADC 电源                             |
| 9       | 6     | ADC_VRef+ | I   | ADC 外部参考电压源输入脚，要求不高时可直接接 MCU 的 VCC |
| 10      | 7     | Gnd       | Gnd | 地线                                 |
|         |       | AGnd      | Gnd | ADC 地线                             |

| 编号      |       | 名称      | 类型  | 说明                |
|---------|-------|---------|-----|-------------------|
| TSSOP20 | QFN20 |         |     |                   |
| 11      | 8     | P3.0    | I/O | 标准 IO 口           |
|         |       | ADC8    | I   | ADC 模拟输入通道 8      |
|         |       | RxD     | I   | 串口 1 的接收脚         |
|         |       | INT4    | I   | 外部中断 4            |
| 12      | 9     | P3.1    | I/O | 标准 IO 口           |
|         |       | ADC9    | I   | ADC 模拟输入通道 9      |
|         |       | TxD     | O   | 串口 1 的发送脚         |
| 13      | 10    | P3.2    | I/O | 标准 IO 口           |
|         |       | ADC10   | I   | ADC 模拟输入通道 10     |
|         |       | INT0    | I   | 外部中断 0            |
|         |       | SCLK_4  | I/O | SPI 的时钟脚          |
|         |       | SCL_4   | I/O | I2C 的时钟线          |
|         |       | PWMETI  | I   | PWM 外部触发输入脚       |
|         |       | PWMETI2 | I   | PWM 外部触发输入脚 2     |
| 14      | 11    | P3.3    | I/O | 标准 IO 口           |
|         |       | ADC11   | I   | ADC 模拟输入通道 11     |
|         |       | INT1    | I   | 外部中断 1            |
|         |       | MISO_4  | I/O | SPI 主机输入从机输出      |
|         |       | SDA_4   | I/O | I2C 的数据线          |
|         |       | PWM4N_4 | I/O | PWM4 的捕获输入和脉冲输出负极 |
|         |       | PWM7_2  | I/O | PWM7 捕获输入和脉冲输出正极  |
| 15      | 12    | P3.4    | I/O | 标准 IO 口           |
|         |       | ADC12   | I   | ADC 模拟输入通道 12     |
|         |       | T0      | I   | 定时器 0 外部时钟输入      |
|         |       | T1CLKO  | O   | 定时器 1 时钟分频输出      |
|         |       | MOSI_4  | I/O | SPI 主机输出从机输入      |
|         |       | PWM4P_4 | I/O | PWM4 捕获输入和脉冲输出正极  |
|         |       | PWM8_2  | I/O | PWM8 捕获输入和脉冲输出正极  |
|         |       | CMPO    | O   | 比较器输出             |
| 16      | 13    | P3.5    | I/O | 标准 IO 口           |
|         |       | ADC13   | I   | ADC 模拟输入通道 13     |
|         |       | T1      | I   | 定时器 1 外部时钟输入      |
|         |       | T0CLKO  | O   | 定时器 0 时钟分频输出      |
|         |       | SS_4    | I/O | SPI 从机选择          |
|         |       | PWMFLT  | I   | PWMA 的外部异常检测脚     |
|         |       | PWMFLT2 | I   | PWMB 的外部异常检测脚     |

| 编号      |       | 名称    | 类型  | 说明                |
|---------|-------|-------|-----|-------------------|
| TSSOP20 | QFN20 |       |     |                   |
| 17      | 14    | P3.6  | I/O | 标准 IO 口           |
|         |       | ADC14 | I   | ADC 模拟输入通道 14     |
|         |       | INT2  | I   | 外部中断 2            |
|         |       | RxD_2 | I   | 串口 1 的接收脚         |
|         |       | CMP-  | I   | 比较器负极输入           |
| 18      | 15    | P3.7  | I/O | 标准 IO 口           |
|         |       | INT3  | I   | 外部中断 3            |
|         |       | TxD_2 | O   | 串口 1 的发送脚         |
|         |       | CMP+  | I   | 比较器正极输入           |
| 19      | 16    | P1.0  | I/O | 标准 IO 口           |
|         |       | ADC0  | I   | ADC 模拟输入通道 0      |
|         |       | RxD2  | O   | 串口 2 的接收脚         |
|         |       | PWM1P | I/O | PWM1 的捕获输入和脉冲输出正极 |
| 20      | 20    | P1.1  | I/O | 标准 IO 口           |
|         |       | ADC1  | I   | ADC 模拟输入通道 1      |
|         |       | TxD2  | O   | 串口 2 的发送脚         |
|         |       | PWM1N | I/O | PWM1 的捕获输入和脉冲输出负极 |

## 2.2 STC8H1K28-36I-LQFP32/QFN32,TSSOP20 系列

### 2.2.1 特性及价格

- 选型价格（不需要外部晶振、不需要外部复位，10 位 ADC，12 通道）

| 价格及封装 | TSSOP20 <6.5mm*6.5mm> |  | QFN32<4mm*4mm> |  | LQFP32<9mm*9mm> |  | 主力产品供货信息          |  |  |  |  |  |            |  |  |  |  |  |                       |  |  |  |  |  |                   |  |  |  |  |  |           |  |  |  |  |  |                |  |  |  |  |  |                          |  |  |  |  |  |                                   |  |  |  |  |  |           |  |  |  |  |  |                         |  |  |  |  |  |                                   |  |  |  |  |  |                         |  |  |  |  |  |     |  |  |  |  |  |          |  |  |  |  |  |   |  |  |  |  |  |           |  |  |  |  |  |                 |  |  |  |  |  |                            |  |  |  |  |  |                                    |  |  |  |  |  |                                     |  |  |  |  |  |                      |  |  |  |  |  |         |  |  |  |  |  |       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|-------|-----------------------|--|----------------|--|-----------------|--|-------------------|--|--|--|--|--|------------|--|--|--|--|--|-----------------------|--|--|--|--|--|-------------------|--|--|--|--|--|-----------|--|--|--|--|--|----------------|--|--|--|--|--|--------------------------|--|--|--|--|--|-----------------------------------|--|--|--|--|--|-----------|--|--|--|--|--|-------------------------|--|--|--|--|--|-----------------------------------|--|--|--|--|--|-------------------------|--|--|--|--|--|-----|--|--|--|--|--|----------|--|--|--|--|--|---|--|--|--|--|--|-----------|--|--|--|--|--|-----------------|--|--|--|--|--|----------------------------|--|--|--|--|--|------------------------------------|--|--|--|--|--|-------------------------------------|--|--|--|--|--|----------------------|--|--|--|--|--|---------|--|--|--|--|--|-------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
|       | 支持 RS485 下载           |  |                |  |                 |  | 支持软件模拟硬件 USB 直接下载 |  |  |  |  |  | 本身就可在线仿真   |  |  |  |  |  |                       |  |  |  |  |  |                   |  |  |  |  |  |           |  |  |  |  |  |                |  |  |  |  |  |                          |  |  |  |  |  |                                   |  |  |  |  |  |           |  |  |  |  |  |                         |  |  |  |  |  |                                   |  |  |  |  |  |                         |  |  |  |  |  |     |  |  |  |  |  |          |  |  |  |  |  |   |  |  |  |  |  |           |  |  |  |  |  |                 |  |  |  |  |  |                            |  |  |  |  |  |                                    |  |  |  |  |  |                                     |  |  |  |  |  |                      |  |  |  |  |  |         |  |  |  |  |  |       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|       | 可设置下次更新程序需口令          |  |                |  |                 |  | 程序加密后传输（防拦截）      |  |  |  |  |  | 可对外输出时钟及复位 |  |  |  |  |  | 内部高精度时钟（36MHz 以下可调）追频 |  |  |  |  |  | 内部高可靠复位（可选复位门檻电压） |  |  |  |  |  | 看门狗 复位定时器 |  |  |  |  |  | 内部低压检测中断并可掉电唤醒 |  |  |  |  |  | 比较器（可当 1 路 A/D，可作外部掉电检测） |  |  |  |  |  | 12 路高速 ADC（8 路 PWM 可当 8 路 D/A 使用） |  |  |  |  |  | 掉电唤醒专用定时器 |  |  |  |  |  | 16 位高级 PWM 定时器 互补对称死区控制 |  |  |  |  |  | 定时器计数器（T0/T1/T2/T3/T4 外部管脚也可掉电唤醒） |  |  |  |  |  | I <sup>2</sup> C 并可掉电唤醒 |  |  |  |  |  | SPI |  |  |  |  |  | 串口并可掉电唤醒 |  |  |  |  |  | 传统 I/O 中断（INT0/INT1/INT2/INT3/INT4）并可掉电唤醒 |  |  |  |  |  | I/O 口最多数量 |  |  |  |  |  | EEPROM 10 万次 字节 |  |  |  |  |  | 强大的双 DPT <sup>3</sup> 可增可减 |  |  |  |  |  | xdata <sup>2</sup> 内部大容量扩展 SRAM 字节 |  |  |  |  |  | idata <sup>1</sup> 内部传统 8051 RAM 字节 |  |  |  |  |  | Flash 程序存储器 10 万次 字节 |  |  |  |  |  | 工作电压（V） |  |  |  |  |  | 单片机型号 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|       |                       |  |                |  |                 |  |                   |  |  |  |  |  |            |  |  |  |  |  |                       |  |  |  |  |  |                   |  |  |  |  |  |           |  |  |  |  |  |                |  |  |  |  |  |                          |  |  |  |  |  |                                   |  |  |  |  |  |           |  |  |  |  |  |                         |  |  |  |  |  |                                   |  |  |  |  |  |                         |  |  |  |  |  |     |  |  |  |  |  |          |  |  |  |  |  |   |  |  |  |  |  |           |  |  |  |  |  |                 |  |  |  |  |  |                            |  |  |  |  |  |                                    |  |  |  |  |  |                                     |  |  |  |  |  |                      |  |  |  |  |  |         |  |  |  |  |  |       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

➤ 内核

- ✓ 超高速 8051 内核 (1T)，比传统 8051 约快 12 倍以上
- ✓ 指令代码完全兼容传统 8051
- ✓ 19 个中断源，4 级中断优先级
- ✓ 支持在线仿真

➤ 工作电压

- ✓ 1.9V~5.5V

➤ 工作温度

- ✓ -40℃~85℃ (芯片为-40℃~125℃制程，超温度范围应用请参考电气特性章节说明)

➤ Flash 存储器

- ✓ 最大 33K 字节 FLASH 程序存储器 (ROM)，用于存储用户代码
- ✓ 支持用户配置 EEPROM 大小，512 字节单页擦除，擦写次数可达 10 万次以上
- ✓ 支持在系统编程方式 (ISP) 更新用户应用程序，无需专用编程器
- ✓ 支持单芯片仿真，无需专用仿真器，理论断点个数无限制

➤ SRAM

- ✓ 128 字节内部直接访问 RAM (DATA，C 语言程序中使用 data 关键字进行声明)
- ✓ 128 字节内部间接访问 RAM (IDATA，C 语言程序中使用 idata 关键字进行声明)
- ✓ 1024 字节内部扩展 RAM (内部 XDATA，C 语言程序中使用 xdata 关键字进行声明)

## ➤ 时钟控制

- ✓ 内部高精度 IRC (4MHz~36MHz, ISP 编程时选择或手动输入, 还可以用户软件分频到较低的频率工作, 如 100KHz)
  - ⊕ 误差±0.3% (常温下 25℃)
  - ⊕ -1.35%~+1.30%温漂 (全温度范围, -40℃~85℃)
  - ⊕ -0.76%~+0.98%温漂 (温度范围, -20℃~65℃)
- ✓ 内部 32KHz 低速 IRC (误差较大)
- ✓ 外部晶振 (4MHz~36MHz) 和外部时钟  
用户可自由选择上面的 3 种时钟源

## ➤ 复位

- ✓ 硬件复位
  - ⊕ 上电复位。(在芯片未使能低压复位功能时有效)
  - ⊕ 复位脚复位。出厂时 P5.4 默认为 I/O 口, ISP 下载时可将 P5.4 管脚设置为复位脚 (注意: 当设置 P5.4 管脚为复位脚时, 复位电平为低电平)
  - ⊕ 看门狗溢出复位
  - ⊕ 低压检测复位, 提供 4 级低压检测电压: 2.0V、2.4V、2.7V、3.0V。
- ✓ 软件复位
  - ⊕ 软件方式写复位触发寄存器

## ➤ 中断

- ✓ 提供 19 个中断源: INT0 (支持上升沿和下降沿中断)、INT1 (支持上升沿和下降沿中断)、INT2 (只支持下降沿中断)、INT3 (只支持下降沿中断)、INT4 (只支持下降沿中断)、定时器 0、定时器 1、定时器 2、定时器 3、定时器 4、串口 1、串口 2、ADC 模数转换、LVD 低压检测、SPI、I<sup>2</sup>C、比较器、PWMA、PWMB
- ✓ 提供 4 级中断优先级
- ✓ 时钟停振模式下可以唤醒的中断: INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、T3(P0.4)、T4(P0.6)、RXD(P3.0/P3.6/P1.6)、RXD2(P1.0)、I2C\_SDA(P1.4/P2.4/P3.3) 以及比较器中断、低压检测中断、掉电唤醒定时器唤醒。

## ➤ 数字外设

- ✓ 5 个 16 位定时器: 定时器 0、定时器 1、定时器 2、定时器 3、定时器 4, 其中定时器 0 的模式 3 具有 NMI (不可屏蔽中断) 功能, 定时器 0 和定时器 1 的模式 0 为 16 位自动重载模式
- ✓ 2 个高速串口: 串口 1、串口 2, 波特率时钟源最快可为 FOSC/4
- ✓ 8 路/2 组高级 PWM, 可实现带死区的控制信号, 并支持外部异常检测功能, 另外还支持 16 位定时器、8 个外部中断、8 路外部捕获测量脉宽等功能
- ✓ SPI: 支持主机模式和从机模式以及主机/从机自动切换
- ✓ I<sup>2</sup>C: 支持主机模式和从机模式

## ➤ 模拟外设

- ✓ 超高速 ADC, 支持 10 位高精度 12 通道 (通道 0~通道 11) 的模数转换, 速度最快能达到 500K (每秒进行 50 万次 ADC 转换)
- ✓ ADC 的通道 15 用于测试内部 1.19V 参考信号源 (芯片在出厂时, 内部参考信号源已调整为 1.19V)
- ✓ 比较器, 一组比较器 (比较器的正端可选择 CMP+端口和所有的 ADC 输入端口, 所以比较器可当作多路比较器进行分时复用)
- ✓ DAC: 8 路高级 PWM 定时器可当 8 路 DAC 使用

## ➤ GPIO

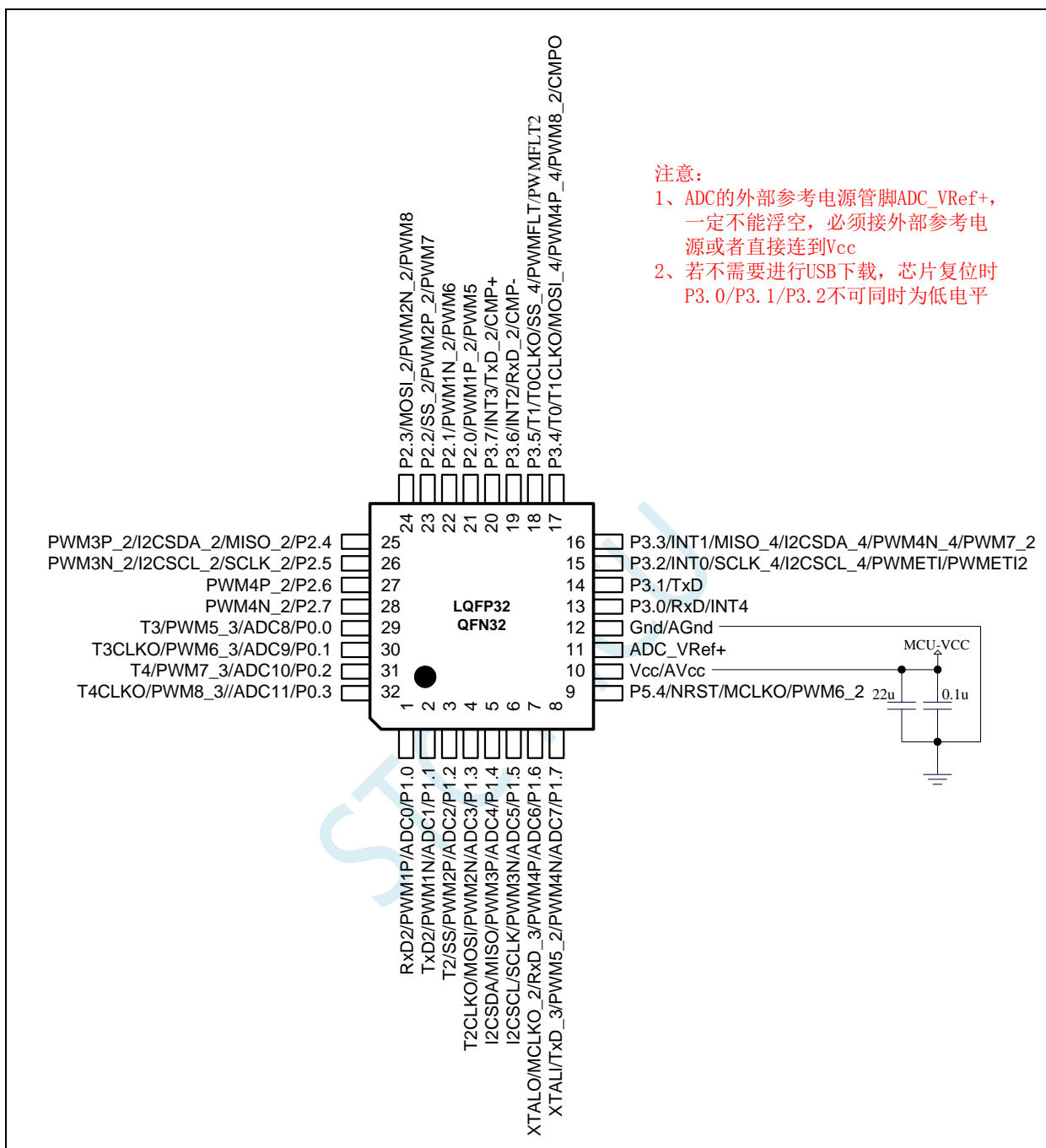
- ✓ 最多可达 29 个 GPIO: P0.0~P0.3、P1.0~P1.7、P2.0~P2.7、P3.0~P3.7、P5.4
- ✓ 所有的 GPIO 均支持如下 4 种模式: 准双向口模式、强推挽输出模式、开漏输出模式、高阻输入模式
- ✓ 除 P3.0 和 P3.1 外, 其余所有 IO 口上电后的状态均为高阻输入状态, 用户在使用 IO 口时必须先设置 IO 口模式。另外每个 I/O 均可独立使能内部 4K 上拉电阻

➤ 封装

- ✓ LQFP32 <9mm\*9mm>、QFN32 <4mm\*4mm>

STC MCU

## 2.2.2 管脚图，最小系统



正看芯片丝印左下方小圆点处为第一脚

正看芯片丝印最下面一行最后一个字母为芯片版本号

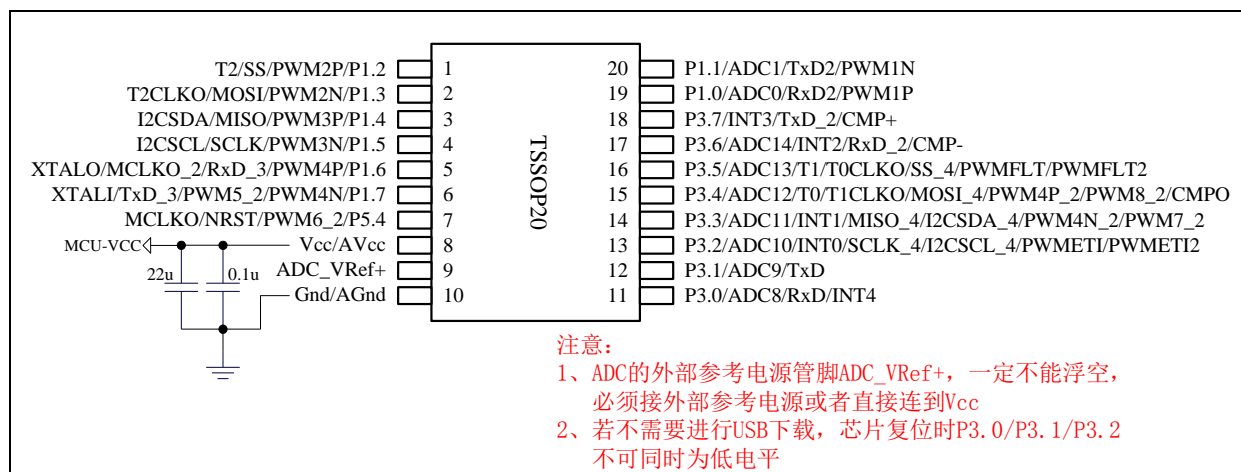
建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF, 可去除电源线噪声, 提高抗干扰能力

注意: 1、除 P3.0 和 P3.1 外, 其余所有 I/O 口上电后的状态均为高阻输入状态, 用户在使用 I/O 口时必须先设置 I/O 口模式

2、所有的 I/O 口均可以设置为准双向口模式、强推挽输出模式、开漏输出模式或者高阻输入模式, 另外每个 I/O 均可独立使能内部 4K 上拉电阻

3、当使能 P5.4 口为复位脚时, 复位电平为低电平





建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF, 可去除电源线噪声, 提高抗干扰能力

## 2.2.3 管脚说明

| 编号           |         | 名称     | 类型  | 说明                |
|--------------|---------|--------|-----|-------------------|
| LQFP32/QFN32 | TSSOP20 |        |     |                   |
| 1            | 19      | P1.0   | I/O | 标准 IO 口           |
|              |         | RxD2   | O   | 串口 2 的接收脚         |
|              |         | ADC0   | I   | ADC 模拟输入通道 0      |
|              |         | PWM1P  | I/O | PWM1 的捕获输入和脉冲输出正极 |
| 2            | 20      | P1.1   | I/O | 标准 IO 口           |
|              |         | TxD2   | O   | 串口 2 的发送脚         |
|              |         | ADC1   | I   | ADC 模拟输入通道 1      |
|              |         | PWM1N  | I/O | PWM1 的捕获输入和脉冲输出负极 |
| 3            | 1       | P1.2   | I/O | 标准 IO 口           |
|              |         | ADC2   | I   | ADC 模拟输入通道 2      |
|              |         | SS     | I/O | SPI 从机选择          |
|              |         | T2     | I   | 定时器 2 外部时钟输入      |
|              |         | PWM2P  | I/O | PWMB 捕获输入和脉冲输出正极  |
| 4            | 2       | P1.3   | I/O | 标准 IO 口           |
|              |         | ADC3   | I   | ADC 模拟输入通道 3      |
|              |         | MOSI   | I/O | SPI 主机输出从机输入      |
|              |         | T2CLKO | O   | 定时器 2 时钟分频输出      |
|              |         | PWM2N  | I/O | PWM2 的捕获输入和脉冲输出负极 |
| 5            | 3       | P1.4   | I/O | 标准 IO 口           |
|              |         | ADC4   | I   | ADC 模拟输入通道 4      |
|              |         | MISO   | I/O | SPI 主机输入从机输出      |
|              |         | SDA    | I/O | I2C 接口的数据线        |
|              |         | PWM3P  | I/O | PWM3 捕获输入和脉冲输出正极  |
| 6            | 4       | P1.5   | I/O | 标准 IO 口           |
|              |         | ADC5   | I   | ADC 模拟输入通道 5      |
|              |         | SCLK   | I/O | SPI 的时钟脚          |
|              |         | SCL    | I/O | I2C 的时钟线          |
|              |         | PWM3N  | I/O | PWM3 的捕获输入和脉冲输出负极 |

| 编号           |         | 名称        | 类型  | 说明                                  |
|--------------|---------|-----------|-----|-------------------------------------|
| LQFP32/QFN32 | TSSOP20 |           |     |                                     |
| 7            | 5       | P1.6      | I/O | 标准 IO 口                             |
|              |         | ADC6      | I   | ADC 模拟输入通道 6                        |
|              |         | RxD_3     | I   | 串口 1 的接收脚                           |
|              |         | PWM4P     | I/O | PWM4 捕获输入和脉冲输出正极                    |
|              |         | MCLKO_2   | O   | 主时钟分频输出                             |
|              |         | XTALO     | O   | 外部晶振的输出脚                            |
| 8            | 6       | P1.7      | I/O | 标准 IO 口                             |
|              |         | ADC7      | I   | ADC 模拟输入通道 7                        |
|              |         | TxD_3     | O   | 串口 1 的发送脚                           |
|              |         | PWM4N     | I/O | PWM4 的捕获输入和脉冲输出负极                   |
|              |         | PWM5_2    | I/O | PWM5 捕获输入和脉冲输出正极                    |
|              |         | XTALI     | I   | 外部晶振/外部时钟的输入脚                       |
| 9            | 7       | P5.4      | I/O | 标准 IO 口                             |
|              |         | NRST      | I   | 复位引脚 (低电平复位)                        |
|              |         | MCLKO     | O   | 主时钟分频输出                             |
|              |         | PWM6_2    | I/O | PWM6 捕获输入和脉冲输出正极                    |
| 10           | 8       | Vcc       | Vcc | 电源脚                                 |
|              |         | AVcc      | Vcc | ADC 电源                              |
| 11           | 9       | ADC_VRef+ | I   | ADC 外部参考电压源输入脚, 要求不高时可直接接 MCU 的 VCC |
| 12           | 10      | Gnd       | Gnd | 地线                                  |
|              |         | AGnd      | Gnd | ADC 地线                              |
| 13           | 11      | P3.0      | I/O | 标准 IO 口                             |
|              |         | RxD       | I   | 串口 1 的接收脚                           |
|              |         | INT4      | I   | 外部中断 4                              |
| 14           | 12      | P3.1      | I/O | 标准 IO 口                             |
|              |         | TxD       | O   | 串口 1 的发送脚                           |
| 15           | 13      | P3.2      | I/O | 标准 IO 口                             |
|              |         | INT0      | I   | 外部中断 0                              |
|              |         | SCLK_4    | I/O | SPI 的时钟脚                            |
|              |         | SCL_4     | I/O | I2C 的时钟线                            |
|              |         | PWMETI    | I   | PWM 外部触发输入脚                         |
|              |         | PWMETI2   | I   | PWM 外部触发输入脚 2                       |
| 16           | 14      | P3.3      | I/O | 标准 IO 口                             |
|              |         | INT1      | I   | 外部中断 1                              |
|              |         | MISO_4    | I/O | SPI 主机输入从机输出                        |
|              |         | SDA_4     | I/O | I2C 的数据线                            |
|              |         | PWM4N_4   | I/O | PWM4 的捕获输入和脉冲输出负极                   |
|              |         | PWM7_2    | I/O | PWM7 捕获输入和脉冲输出正极                    |

| 编号           |         | 名称      | 类型  | 说明                |
|--------------|---------|---------|-----|-------------------|
| LQFP32/QFN32 | TSSOP20 |         |     |                   |
| 17           | 15      | P3.4    | I/O | 标准 IO 口           |
|              |         | T0      | I   | 定时器 0 外部时钟输入      |
|              |         | T1CLKO  | O   | 定时器 1 时钟分频输出      |
|              |         | MOSI_4  | I/O | SPI 主机输出从机输入      |
|              |         | PWM4P_4 | I/O | PWM4 捕获输入和脉冲输出正极  |
|              |         | PWM8_2  | I/O | PWM8 捕获输入和脉冲输出正极  |
|              |         | CMPO    | O   | 比较器输出             |
| 18           | 16      | P3.5    | I/O | 标准 IO 口           |
|              |         | T1      | I   | 定时器 1 外部时钟输入      |
|              |         | T0CLKO  | O   | 定时器 0 时钟分频输出      |
|              |         | SS_4    | I/O | SPI 从机选择          |
|              |         | PWMFLT  | I   | PWMA 的外部异常检测脚     |
|              |         | PWMFLT2 | I   | PWMB 的外部异常检测脚     |
| 19           | 17      | P3.6    | I/O | 标准 IO 口           |
|              |         | INT2    | I   | 外部中断 2            |
|              |         | RxD_2   | I   | 串口 1 的接收脚         |
|              |         | CMP-    | I   | 比较器负极输入           |
| 20           | 18      | P3.7    | I/O | 标准 IO 口           |
|              |         | INT3    | I   | 外部中断 3            |
|              |         | TxD_2   | O   | 串口 1 的发送脚         |
|              |         | CMP+    | I   | 比较器正极输入           |
| 21           |         | P2.0    | I/O | 标准 IO 口           |
|              |         | PWM1P_2 | I/O | PWM1 的捕获输入和脉冲输出正极 |
|              |         | PWM5    | I/O | PWM5 捕获输入和脉冲输出正极  |
| 22           |         | P2.1    | I/O | 标准 IO 口           |
|              |         | PWM1N_2 | I/O | PWM1 的捕获输入和脉冲输出负极 |
|              |         | PWM6    | I/O | PWM6 捕获输入和脉冲输出正极  |
| 23           |         | P2.2    | I/O | 标准 IO 口           |
|              |         | PWM2P_2 | I/O | PWM2 的捕获输入和脉冲输出正极 |
|              |         | PWM7    | I/O | PWM7 捕获输入和脉冲输出正极  |
|              |         | SS_2    | I/O | SPI 从机选择          |
| 24           |         | P2.3    | I/O | 标准 IO 口           |
|              |         | PWM2N_2 | I/O | PWM2 的捕获输入和脉冲输出负极 |
|              |         | PWM8    | I/O | PWM8 捕获输入和脉冲输出正极  |
|              |         | MOSI_2  | I/O | SPI 主机输出从机输入      |

| 编号           |  | 名称      | 类型  | 说明                |
|--------------|--|---------|-----|-------------------|
| LQFP32/QFN32 |  |         |     |                   |
| 25           |  | P2.4    | I/O | 标准 IO 口           |
|              |  | PWM3P_2 | I/O | PWM3 的捕获输入和脉冲输出正极 |
|              |  | MISO_2  | I/O | SPI 主机输入从机输出      |
|              |  | SDA_2   | I/O | I2C 的数据线          |
| 26           |  | P2.5    | I/O | 标准 IO 口           |
|              |  | PWM3N_2 | I/O | PWM3 的捕获输入和脉冲输出负极 |
|              |  | SCLK_2  | I/O | SPI 的时钟脚          |
|              |  | SCL_2   | I/O | I2C 的时钟线          |
| 27           |  | P2.6    | I/O | 标准 IO 口           |
|              |  | PWM4P_2 | I/O | PWM4 的捕获输入和脉冲输出正极 |
| 28           |  | P2.7    | I/O | 标准 IO 口           |
|              |  | PWM4N_2 | I/O | PWM4 的捕获输入和脉冲输出负极 |
| 29           |  | P0.0    | I/O | 标准 IO 口           |
|              |  | ADC8    | I   | ADC 模拟输入通道 8      |
|              |  | T3      | I   | 定时器 3 外部时钟输入      |
|              |  | PWM5_3  | I/O | PWM5 捕获输入和脉冲输出正极  |
| 30           |  | P0.1    | I/O | 标准 IO 口           |
|              |  | ADC9    | I   | ADC 模拟输入通道 9      |
|              |  | T3CLKO  | O   | 定时器 3 时钟分频输出      |
|              |  | PWM6_3  | I/O | PWM6 捕获输入和脉冲输出正极  |
| 31           |  | P0.2    | I/O | 标准 IO 口           |
|              |  | ADC10   | I   | ADC 模拟输入通道 10     |
|              |  | T4      | I   | 定时器 4 外部时钟输入      |
|              |  | PWM7_3  | I/O | PWM7 捕获输入和脉冲输出正极  |
| 32           |  | P0.3    | I/O | 标准 IO 口           |
|              |  | ADC11   | I   | ADC 模拟输入通道 11     |
|              |  | T4CLKO  | O   | 定时器 4 时钟分频输出      |
|              |  | PWM8_3  | I/O | PWM8 捕获输入和脉冲输出正极  |

## 2.3 STC8H3K64S2-45I-LQFP48/32,QFN48/32,TSSOP20 系列

### 2.3.1 特性及价格(有 16 位硬件乘除法器 MDU16, 准 16 位单片机)

➤ 选型价格(不需要外部晶振、不需要外部复位, 12 位 ADC, 12 通道)

| 单片机型号       | 工作电压 (V) | Flash 程序存储器 10 万次 字节 | idata <sup>*</sup> 内部传统 8051 RAM 字节 | xdata <sup>*</sup> 内部大容量扩展 SRAM 字节 | EEPROM 10 万次 字节 | 强大的双 DPT <sup>*</sup> 可增可减 | I/O 口最多数量 | 传统 I/O 中断 (INT0/INT1/INT2/INT3/INT4) 并可掉电唤醒 | 所有的 I/O 口均支持中断并可掉电唤醒 | 串口并可掉电唤醒 | SPI 并可掉电唤醒 | I <sup>2</sup> C 并可掉电唤醒 | MDU16 硬件 16 位乘除法器 | 定时器/计数器 (T0/T1/T2/T3/T4 外部管脚也可掉电唤醒) | 16 位高级 PWM 定时器 互补对称死区控制 | 掉电唤醒专用定时器 | 12 路高速 ADC (8 路 PWM 可当 8 路 D/A 使用) | 比较器 (可当 1 路 A/D <sup>*</sup> 可作外部掉电检测) | 内部低压检测中断并可掉电唤醒 | 看门狗 复位定时器 | 内部高精度时钟 (45MHz 以下可调) 追频 | 内部高可靠复位 (可选复位门电压) | 可对外输出时钟及复位 | 程序加密后传输 (防拦截) | 可设置下次更新程序需口令 | 支持 RS485 下载 | 支持软件模拟硬件 USB 直接下载 | 本身就可在线仿真 | 价格及封装 |      |       |       | 供货信息 |
|-------------|----------|----------------------|-------------------------------------|------------------------------------|-----------------|----------------------------|-----------|---|----------------------|----------|------------|-------------------------|-------------------|-------------------------------------|-------------------------|-----------|------------------------------------|--|----------------|-----------|-------------------------|-------------------|------------|---------------|--------------|-------------|-------------------|----------|-------|------|-------|-------|------|
|             |          |                      |                                     |                                    |                 |                            |           |   |                      |          |            |                         |                   |                                     |                         |           |                                    |  |                |           |                         |                   |            |               |              |             |                   |          |       |      |       |       |      |
| STC8H3K32S2 | 1.9-5.5  | 32K                  | 256                                 | 3K                                 | 2               | 32K                        | 43        | 有   | 有                    | 2        | 有          | 有                       | 有                 | 5                                   | 8                       | 有         | 12 位                               | 有                                      | 有              | 有         | 4 级                     | 有                 | 是          | 有             | 是            | 是           | 是                 | 是        | ¥2.4  | ¥2.4 | ¥1.85 |       | 现 货  |
| STC8H3K48S2 | 1.9-5.5  | 48K                  | 256                                 | 3K                                 | 2               | 16K                        | 43        | 有   | 有                    | 2        | 有          | 有                       | 有                 | 5                                   | 8                       | 有         | 12 位                               | 有                                      | 有              | 有         | 4 级                     | 有                 | 是          | 有             | 是            | 是           | 是                 | 是        | ¥2.4  | ¥2.4 | √     | ¥1.85 |      |
| STC8H3K64S2 | 1.9-5.5  | 64K                  | 256                                 | 3K                                 | 2               | IAP                        | 43        | 有   | 有                    | 2        | 有          | 有                       | 有                 | 5                                   | 8                       | 有         | 12 位                               | 有                                      | 有              | 有         | 4 级                     | 有                 | 是          | 有             | 是            | 是           | 是                 | 是        | ¥2.4  | ¥2.4 | √     | ¥1.85 |      |

#### ➤ 内核

- ✓ 超高速 8051 内核 (1T), 比传统 8051 约快 12 倍以上
- ✓ 指令代码完全兼容传统 8051
- ✓ 25 个中断源, 4 级中断优先级
- ✓ 支持在线仿真

#### ➤ 工作电压

- ✓ 1.9V~5.5V

#### ➤ 工作温度

- ✓ -40℃~85℃ (芯片为-40℃~125℃制程, 超温度范围应用请参考电气特性章节说明)

#### ➤ Flash 存储器

- ✓ 最大 64K 字节 FLASH 程序存储器 (ROM), 用于存储用户代码
- ✓ 支持用户配置 EEPROM 大小, 512 字节单页擦除, 擦写次数可达 10 万次以上
- ✓ 支持在系统编程方式 (ISP) 更新用户应用程序, 无需专用编程器
- ✓ 支持单芯片仿真, 无需专用仿真器, 理论断点个数无限制

#### ➤ SRAM

- ✓ 128 字节内部直接访问 RAM (DATA, C 语言程序中使用 data 关键字进行声明)
- ✓ 128 字节内部间接访问 RAM (IDATA, C 语言程序中使用 idata 关键字进行声明)
- ✓ 3072 字节内部扩展 RAM (内部 XDATA, C 语言程序中使用 xdata 关键字进行声明)

## ➤ 时钟控制

- ✓ 内部高精度 IRC (4MHz~45MHz, ISP 编程时选择或手动输入, 还可以用户软件分频到较低的频率工作, 如 100KHz)
  - ✦ 误差±0.3% (常温下 25℃)
  - ✦ -1.35%~+1.30%温漂 (全温度范围, -40℃~85℃)
  - ✦ -0.76%~+0.98%温漂 (温度范围, -20℃~65℃)
  - ✦ **注意: 本系列的内部 IRC, 32M~37M 可能是盲区, 强烈建议选择 30MHz 及以下的频率, 或 40MHz 及以上的频率, 不可高于 45MHz**
- ✓ 内部 32KHz 低速 IRC (误差较大)
- ✓ 外部晶振 (4MHz~45MHz) 和外部时钟  
用户可自由选择上面的 3 种时钟源
- ✓ 关于 STC8H3K64S2 系列 B 版产品内部高速 IRC 的重要说明
  - ✦ 由于制造原因, 部分芯片的内部高速 IRC 在 34MHz~36MHz 可能存在盲区, 建议不要将工作频率设定在此区域
  - ✦ 内部高速 IRC 低温的温漂较高温时要大一些, 低频率段的温漂比高频率段要大一些。一般的, 20MHz~40MHz 的工作频率, 在高温 85℃的温漂可控制在 0.8% 以内

## ➤ 复位

- ✓ 硬件复位
  - ✦ 上电复位。(在芯片未使能低压复位功能时有效)
  - ✦ 复位脚复位。出厂时 P5.4 默认为 I/O 口, ISP 下载时可将 P5.4 管脚设置为复位脚 (注意: 当设置 P5.4 管脚为复位脚时, 复位电平为低电平)
  - ✦ 看门狗溢出复位
  - ✦ 低压检测复位, 提供 4 级低压检测电压: 2.0V、2.4V、2.7V、3.0V。
- ✓ 软件复位
  - ✦ 软件方式写复位触发寄存器

## ➤ 中断

- ✓ 提供 25 个中断源: INT0 (支持上升沿和下降沿中断)、INT1 (支持上升沿和下降沿中断)、INT2 (只支持下降沿中断)、INT3 (只支持下降沿中断)、INT4 (只支持下降沿中断)、定时器 0、定时器 1、定时器 2、定时器 3、定时器 4、串口 1、串口 2、ADC 模数转换、LVD 低压检测、SPI、I<sup>2</sup>C、比较器、PWMA、PWMB、P0 口中断、P1 口中断、P2 口中断、P3 口中断、P4 口中断、P5 口中断
- ✓ 提供 4 级中断优先级
- ✓ 时钟停振模式下可以唤醒的中断: INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、T3(P0.4)、T4(P0.6)、RXD(P3.0/P3.6/P1.6/P4.3)、RXD2(P1.0/P4.6)、I2C\_SDA(P1.4/P2.4/P3.3)以及比较器中断、低压检测中断、掉电唤醒定时器唤醒。

## ➤ 数字外设

- ✓ 5 个 16 位定时器: 定时器 0、定时器 1、定时器 2、定时器 3、定时器 4, 其中定时器 0 的模式 3 具有 NMI (不可屏蔽中断) 功能, 定时器 0 和定时器 1 的模式 0 为 16 位自动重载模式
- ✓ 2 个高速串口: 串口 1、串口 2, 波特率时钟源最快可为 FOSC/4
- ✓ 8 路/2 组高级 PWM, 可实现带死区的控制信号, 并支持外部异常检测功能, 另外还支持 16 位定时器、8 个外部中断、8 路外部捕获测量脉宽等功能
- ✓ SPI: 支持主机模式和从机模式以及主机/从机自动切换
- ✓ I<sup>2</sup>C: 支持主机模式和从机模式
- ✓ MDU16: 硬件 16 位乘除法器 (支持 32 位除以 16 位、16 位除以 16 位、16 位乘 16 位、数据移位以及数据规格化等运算)



- ✓ I/O 口中断: 所有的 I/O 均支持中断, 每组 I/O 中断有独立的中断入口地址, 所有的 I/O 中断可支持 4 种中断模式: 高电平中断、低电平中断、上升沿中断、下降沿中断

(注意: STC8H3K64S2 系列 A 版芯片的 I/O 口中断不能进行掉电唤醒, **B 版芯片的 I/O 口中断可以进行掉电唤醒, 但只有一级中断优先级。B 版芯片的 I/O 口中断经测试发现问题, 暂请不要使用**)

#### ➤ 模拟外设

- ✓ 超高速 ADC, 支持 12 位高精度 12 通道 (通道 0~通道 2、通道 6~通道 14, 无 P1.3/P1.4/P1.5 端口, 所以少了通道 3~5) 的模数转换, 速度最快能达到 800K (每秒进行 80 万次 ADC 转换)
- ✓ ADC 的通道 15 用于测试内部 1.19V 参考信号源 (芯片在出厂时, 内部参考信号源已调整为 1.19V)
- ✓ 比较器, 一组比较器 (比较器的正端可选择 CMP+端口和所有的 ADC 输入端口, 所以比较器可当作多路比较器进行分时复用)
- ✓ DAC: 8 路高级 PWM 定时器可当 8 路 DAC 使用

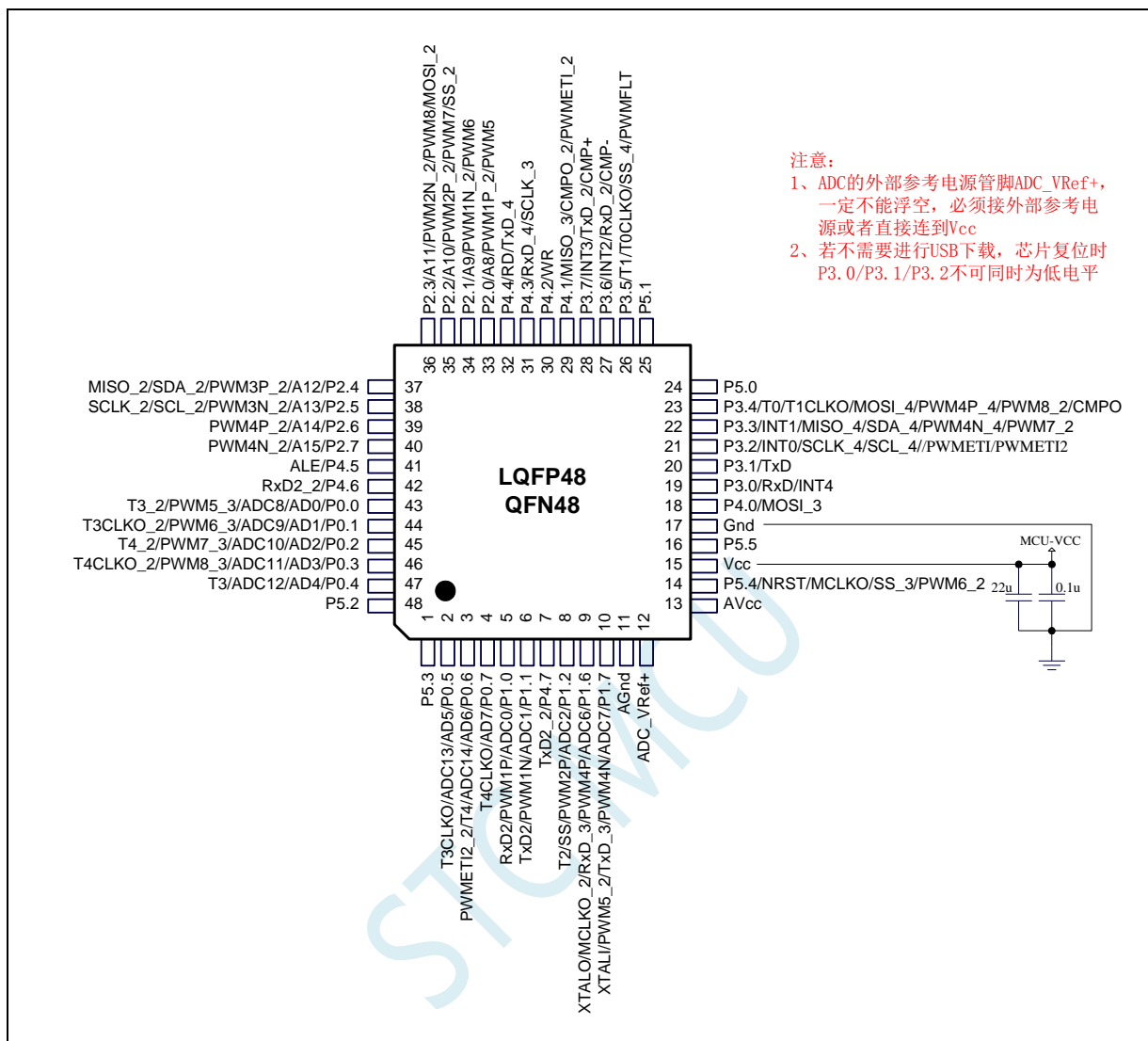
#### ➤ GPIO

- ✓ 最多可达 43 个 GPIO: P0.0~P0.7、P1.0~P1.2、P1.6~P1.7、P2.0~P2.7、P3.0~P3.7、P4.0~P4.7、P5.0~P5.5
- ✓ 所有的 GPIO 均支持如下 4 种模式: 准双向口模式、强推挽输出模式、开漏输出模式、高阻输入模式
- ✓ 除 P3.0 和 P3.1 外, 其余所有 IO 口上电后的状态均为高阻输入状态, 用户在使用 IO 口时必须先设置 IO 口模式。另外每个 I/O 均可独立使能内部 4K 上拉电阻

#### ➤ 封装

- ✓ LQFP48 <9mm\*9mm>、QFN48 <6mm\*6mm>、LQFP32 <9mm\*9mm>、QFN32 <4mm\*4mm>、TSSOP20 <6.5mm\*6.5mm> (**LQFP32、QFN32、TSSOP2 暂无样品, 后续会有, 若有需要请提前订货**)

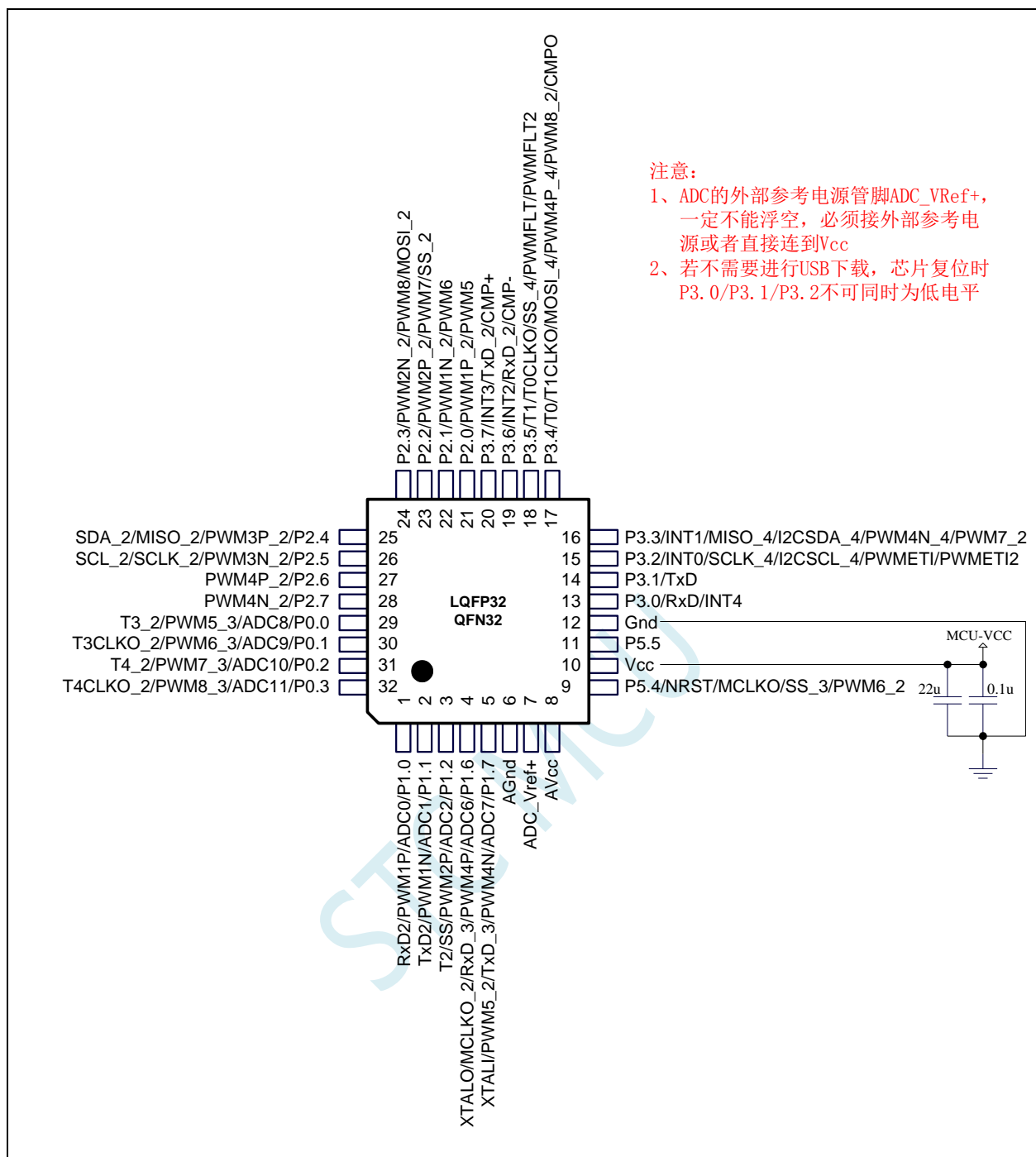
## 2.3.2 管脚图，最小系统



正看芯片丝印左下方小圆点处为第一脚

正看芯片丝印最下面一行最后一个字母为芯片版本号

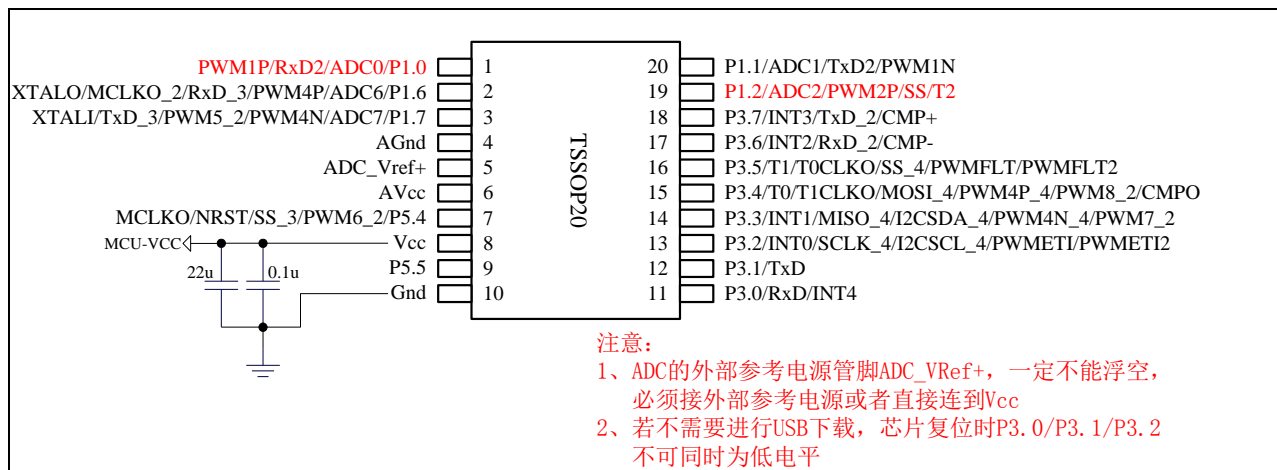
建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF，可去除电源线噪声，提高抗干扰能力



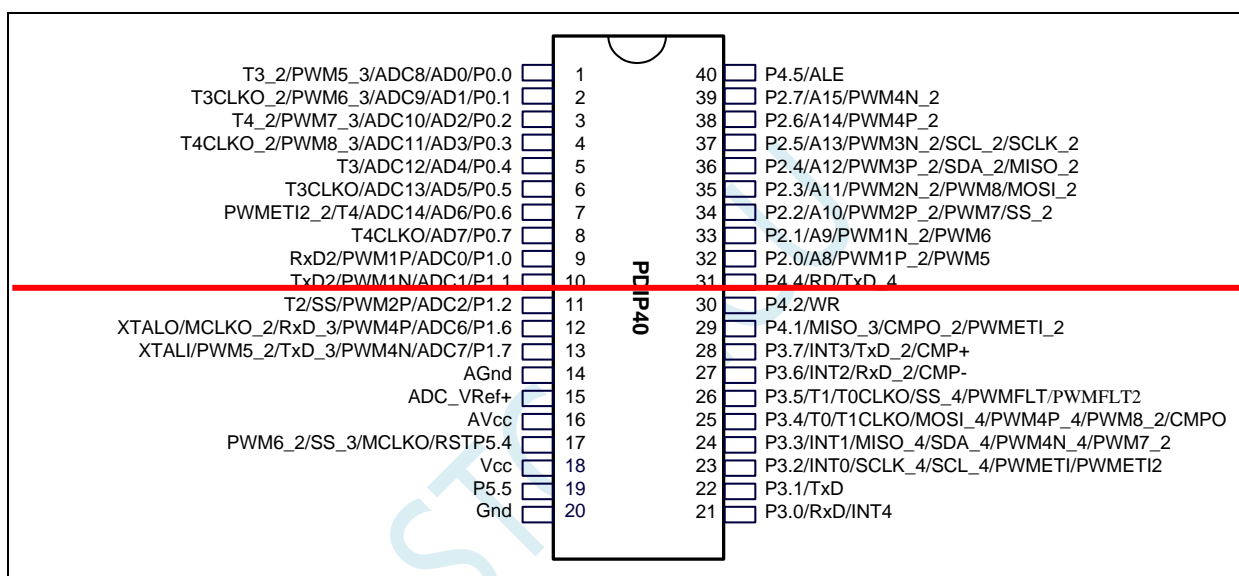
正看芯片丝印左下方小圆点处为第一脚

正看芯片丝印最下面一行最后一个字母为芯片版本号

建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF, 可去除电源线噪声, 提高抗干扰能力



建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF, 可去除电源线噪声, 提高抗干扰能力



### 2.3.3 管脚说明

| 编号              |                 |         |  | 名称        | 类型  | 说明                |
|-----------------|-----------------|---------|--|-----------|-----|-------------------|
| LQFP48<br>QFN48 | LQFP32<br>QFN32 | TSSOP20 |  |           |     |                   |
| 1               |                 |         |  | P5.3      | I/O | 标准 IO 口           |
| 2               |                 |         |  | P0.5      | I/O | 标准 IO 口           |
|                 |                 |         |  | AD5       | I/O | 地址/数据总线           |
|                 |                 |         |  | ADC13     | I   | ADC 模拟输入通道 13     |
|                 |                 |         |  | T3CLKO    | O   | 定时器 3 时钟分频输出      |
| 3               |                 |         |  | P0.6      | I/O | 标准 IO 口           |
|                 |                 |         |  | AD6       | I/O | 地址/数据总线           |
|                 |                 |         |  | ADC14     | I   | ADC 模拟输入通道 14     |
|                 |                 |         |  | T4        | I   | 定时器 4 外部时钟输入      |
|                 |                 |         |  | PWMETI2_2 | I   | PWM 外部触发输入脚 2     |
| 4               |                 |         |  | P0.7      | I/O | 标准 IO 口           |
|                 |                 |         |  | AD7       | I/O | 地址/数据总线           |
|                 |                 |         |  | T4CLKO    | O   | 定时器 4 时钟分频输出      |
| 5               | 1               | 1       |  | P1.0      | I/O | 标准 IO 口           |
|                 |                 |         |  | ADC0      | I   | ADC 模拟输入通道 0      |
|                 |                 |         |  | PWM1P     | I/O | PWM1 的捕获输入和脉冲输出正极 |
|                 |                 |         |  | RxD2      | I   | 串口 2 的接收脚         |
| 6               | 2               | 20      |  | P1.1      | I/O | 标准 IO 口           |
|                 |                 |         |  | ADC1      | I   | ADC 模拟输入通道 1      |
|                 |                 |         |  | PWM1N     | I/O | PWM1 的捕获输入和脉冲输出负极 |
|                 |                 |         |  | TxD2      | I   | 串口 2 的发送脚         |
| 7               |                 |         |  | P4.7      | I/O | 标准 IO 口           |
|                 |                 |         |  | TxD2_2    | I   | 串口 2 的发送脚         |
| 8               | 3               | 19      |  | P1.2      | I/O | 标准 IO 口           |
|                 |                 |         |  | ADC2      | I   | ADC 模拟输入通道        |
|                 |                 |         |  | PWM2P     | I/O | PWM2 的捕获输入和脉冲输出正极 |
|                 |                 |         |  | SS        | I   | SPI 的从机选择脚（主机为输出） |
|                 |                 |         |  | T2        | I   | 定时器 2 外部时钟输入      |

| 编号              |                 |         |  | 名称        | 类型  | 说明                                  |
|-----------------|-----------------|---------|--|-----------|-----|-------------------------------------|
| LQFP48<br>QFN48 | LQFP32<br>QFN32 | TSSOP20 |  |           |     |                                     |
| 9               | 4               | 2       |  | P1.6      | I/O | 标准 IO 口                             |
|                 |                 |         |  | ADC6      | I   | ADC 模拟输入通道 6                        |
|                 |                 |         |  | RxD_3     | I   | 串口 1 的接收脚                           |
|                 |                 |         |  | PWM4P     | I/O | PWM4 的捕获输入和脉冲输出正极                   |
|                 |                 |         |  | MCLKO_2   | O   | 主时钟分频输出                             |
|                 |                 |         |  | XTALO     | O   | 外部晶振的输出脚                            |
| 10              | 5               | 3       |  | P1.7      | I/O | 标准 IO 口                             |
|                 |                 |         |  | ADC7      | I   | ADC 模拟输入通道 7                        |
|                 |                 |         |  | TxD_3     | O   | 串口 1 的发送脚                           |
|                 |                 |         |  | PWM4N     | I/O | PWM4 的捕获输入和脉冲输出负极                   |
|                 |                 |         |  | PWM5_2    | I/O | PWM5 的捕获输入和脉冲输出                     |
|                 |                 |         |  | XTALI     | I   | 外部晶振/外部时钟的输入脚                       |
| 11              | 6               | 4       |  | AGnd      | Gnd | ADC 地线                              |
| 12              | 7               | 5       |  | ADC_VRef+ | I   | ADC 外部参考电压源输入脚, 要求不高时可直接接 MCU 的 VCC |
| 13              | 8               | 6       |  | AVcc      | Vcc | ADC 电源脚                             |
| 14              | 9               | 7       |  | P5.4      | I/O | 标准 IO 口                             |
|                 |                 |         |  | NRST      | I   | 复位引脚 (低电平复位)                        |
|                 |                 |         |  | MCLKO     | O   | 主时钟分频输出                             |
|                 |                 |         |  | SS_3      | I   | SPI 的从机选择脚 (主机为输出)                  |
|                 |                 |         |  | PWM6_2    | I/O | PWM6 的捕获输入和脉冲输出                     |
| 15              | 10              | 8       |  | Vcc       | Vcc | 电源脚                                 |
| 16              | 11              | 9       |  | P5.5      | I/O | 标准 IO 口                             |
| 17              | 12              | 10      |  | Gnd       | Gnd | 地线                                  |

| 编号              |                 |         |  | 名称      | 类型  | 说明                |
|-----------------|-----------------|---------|--|---------|-----|-------------------|
| LQFP48<br>QFN48 | LQFP32<br>QFN32 | TSSOP20 |  |         |     |                   |
| 18              |                 |         |  | P4.0    | I/O | 标准 IO 口           |
|                 |                 |         |  | MOSI_3  | I/O | SPI 主机输出从机输入      |
| 19              | 13              | 11      |  | P3.0    | I/O | 标准 IO 口           |
|                 |                 |         |  | RxD     | I   | 串口 1 的接收脚         |
|                 |                 |         |  | INT4    | I   | 外部中断 4            |
| 20              | 14              | 12      |  | P3.1    | I/O | 标准 IO 口           |
|                 |                 |         |  | TxD     | O   | 串口 1 的发送脚         |
| 21              | 15              | 13      |  | P3.2    | I/O | 标准 IO 口           |
|                 |                 |         |  | INT0    | I   | 外部中断 0            |
|                 |                 |         |  | SCLK_4  | I/O | SPI 的时钟脚          |
|                 |                 |         |  | SCL_4   | I/O | I2C 的时钟线          |
|                 |                 |         |  | PWMETI  | I   | PWM 外部触发输入脚       |
|                 |                 |         |  | PWMETI2 | I   | PWM 外部触发输入脚 2     |
| 22              | 16              | 14      |  | P3.3    | I/O | 标准 IO 口           |
|                 |                 |         |  | INT1    | I   | 外部中断 1            |
|                 |                 |         |  | MISO_4  | I/O | SPI 主机输入从机输出      |
|                 |                 |         |  | SDA_4   | I/O | I2C 接口的数据线        |
|                 |                 |         |  | PWM4N_4 | I/O | PWM4 的捕获输入和脉冲输出负极 |
|                 |                 |         |  | PWM7_2  | I/O | PWM7 的捕获输入和脉冲输出   |
| 23              | 17              | 15      |  | P3.4    | I/O | 标准 IO 口           |
|                 |                 |         |  | T0      | I   | 定时器 0 外部时钟输入      |
|                 |                 |         |  | T1CLKO  | O   | 定时器 1 时钟分频输出      |
|                 |                 |         |  | MOSI_4  | I/O | SPI 主机输出从机输入      |
|                 |                 |         |  | PWM4P_4 | I/O | PWM4 的捕获输入和脉冲输出正极 |
|                 |                 |         |  | PWM8_2  | I/O | PWM8 的捕获输入和脉冲输出   |
|                 |                 |         |  | CMPO    | O   | 比较器输出             |
| 24              |                 |         |  | P5.0    | I/O | 标准 IO 口           |
| 25              |                 |         |  | P5.1    | I/O | 标准 IO 口           |



| 编号              |                 |         |  | 名称       | 类型  | 说明                |
|-----------------|-----------------|---------|--|----------|-----|-------------------|
| LQFP48<br>QFN48 | LQFP32<br>QFN32 | TSSOP20 |  |          |     |                   |
| 26              | 18              | 16      |  | P3.5     | I/O | 标准 IO 口           |
|                 |                 |         |  | T1       | I   | 定时器 1 外部时钟输入      |
|                 |                 |         |  | T0CLKO   | O   | 定时器 0 时钟分频输出      |
|                 |                 |         |  | SS_4     | I   | SPI 的从机选择脚（主机为输出） |
|                 |                 |         |  | PWMFLT   | I   | 增强 PWM 的外部异常检测脚   |
| 27              | 19              | 17      |  | P3.6     | I/O | 标准 IO 口           |
|                 |                 |         |  | INT2     | I   | 外部中断 2            |
|                 |                 |         |  | RxD_2    | I   | 串口 1 的接收脚         |
|                 |                 |         |  | CMP-     | I   | 比较器负极输入           |
| 28              | 20              | 18      |  | P3.7     | I/O | 标准 IO 口           |
|                 |                 |         |  | INT3     | I   | 外部中断 3            |
|                 |                 |         |  | TxD_2    | O   | 串口 1 的发送脚         |
|                 |                 |         |  | CMP+     | I   | 比较器正极输入           |
| 29              |                 |         |  | P4.1     | I/O | 标准 IO 口           |
|                 |                 |         |  | MISO_3   | I/O | SPI 主机输入从机输出      |
|                 |                 |         |  | CMPO_2   | O   | 比较器输出             |
|                 |                 |         |  | PWMETI_2 | I   | PWM 外部触发输入脚       |
| 30              |                 |         |  | P4.2     | I/O | 标准 IO 口           |
|                 |                 |         |  | WR       | O   | 外部总线的写信号线         |
| 31              |                 |         |  | P4.3     | I/O | 标准 IO 口           |
|                 |                 |         |  | RxD_4    | I   | 串口 1 的接收脚         |
|                 |                 |         |  | SCLK_3   | I/O | SPI 的时钟脚          |
| 32              |                 |         |  | P4.4     | I/O | 标准 IO 口           |
|                 |                 |         |  | RD       | O   | 外部总线的读信号线         |
|                 |                 |         |  | TxD_4    | O   | 串口 1 的发送脚         |
| 33              | 21              |         |  | P2.0     | I/O | 标准 IO 口           |
|                 |                 |         |  | A8       | O   | 地址总线              |
|                 |                 |         |  | PWM1P_2  | I/O | PWM1 的捕获输入和脉冲输出正极 |
|                 |                 |         |  | PWM5     | I/O | PWM5 的捕获输入和脉冲输出   |

| 编号              |                 |         |  | 名称      | 类型  | 说明                |
|-----------------|-----------------|---------|--|---------|-----|-------------------|
| LQFP48<br>QFN48 | LQFP32<br>QFN32 | TSSOP20 |  |         |     |                   |
| 34              | 22              |         |  | P2.1    | I/O | 标准 IO 口           |
|                 |                 |         |  | A9      | O   | 地址总线              |
|                 |                 |         |  | PWM1N_2 | I/O | PWM1 的捕获输入和脉冲输出负极 |
|                 |                 |         |  | PWM6    | I/O | PWM6 的捕获输入和脉冲输出   |
| 35              | 23              |         |  | P2.2    | I/O | 标准 IO 口           |
|                 |                 |         |  | A10     | O   | 地址总线              |
|                 |                 |         |  | SS_2    | I   | SPI 的从机选择脚（主机为输出） |
|                 |                 |         |  | PWM2P_2 | I/O | PWM2 的捕获输入和脉冲输出正极 |
|                 |                 |         |  | PWM7    | I/O | PWM7 的捕获输入和脉冲输出   |
| 36              | 24              |         |  | P2.3    | I/O | 标准 IO 口           |
|                 |                 |         |  | A11     | O   | 地址总线              |
|                 |                 |         |  | MOSI_2  | I/O | SPI 主机输出从机输入      |
|                 |                 |         |  | PWM2N_2 | I/O | PWM2 的捕获输入和脉冲输出负极 |
|                 |                 |         |  | PWM8    | I/O | PWM8 的捕获输入和脉冲输出   |
| 37              | 25              |         |  | P2.4    | I/O | 标准 IO 口           |
|                 |                 |         |  | A12     | O   | 地址总线              |
|                 |                 |         |  | MISO_2  | I/O | SPI 主机输入从机输出      |
|                 |                 |         |  | SDA_2   | I/O | I2C 接口的数据线        |
|                 |                 |         |  | PWM3P_2 | I/O | PWM3 的捕获输入和脉冲输出正极 |
| 38              | 26              |         |  | P2.5    | I/O | 标准 IO 口           |
|                 |                 |         |  | A13     | O   | 地址总线              |
|                 |                 |         |  | SCLK_2  | I/O | SPI 的时钟脚          |
|                 |                 |         |  | SCL_2   | I/O | I2C 的时钟线          |
|                 |                 |         |  | PWM3N_2 | I/O | PWM3 的捕获输入和脉冲输出负极 |
| 39              | 27              |         |  | P2.6    | I/O | 标准 IO 口           |
|                 |                 |         |  | A14     | O   | 地址总线              |
|                 |                 |         |  | PWM4P_2 | I/O | PWM4 的捕获输入和脉冲输出正极 |
| 40              | 28              |         |  | P2.7    | I/O | 标准 IO 口           |
|                 |                 |         |  | A15     | O   | 地址总线              |
|                 |                 |         |  | PWM4N_2 | I/O | PWM4 的捕获输入和脉冲输出负极 |
| 41              |                 |         |  | P4.5    | I/O | 标准 IO 口           |
|                 |                 |         |  | ALE     | O   | 地址锁存信号            |

| 编号              |                 |         |  | 名称       | 类型  | 说明              |
|-----------------|-----------------|---------|--|----------|-----|-----------------|
| LQFP48<br>QFN48 | LQFP32<br>QFN32 | TSSOP20 |  |          |     |                 |
| 42              |                 |         |  | P4.6     | I/O | 标准 IO 口         |
|                 |                 |         |  | RxD2_2   | I   | 串口 2 的接收脚       |
| 43              | 29              |         |  | P0.0     | I/O | 标准 IO 口         |
|                 |                 |         |  | AD0      | I/O | 地址/数据总线         |
|                 |                 |         |  | ADC8     | I   | ADC 模拟输入通道 8    |
|                 |                 |         |  | PWM5_3   | I/O | PWM5 的捕获输入和脉冲输出 |
|                 |                 |         |  | T3_2     | I   | 定时器 3 外部时钟输入    |
| 44              | 30              |         |  | P0.1     | I/O | 标准 IO 口         |
|                 |                 |         |  | AD1      | I/O | 地址/数据总线         |
|                 |                 |         |  | ADC9     | I   | ADC 模拟输入通道 9    |
|                 |                 |         |  | PWM6_3   | I/O | PWM6 的捕获输入和脉冲输出 |
|                 |                 |         |  | T3CLKO_2 | O   | 定时器 3 时钟分频输出    |
| 45              | 31              |         |  | P0.2     | I/O | 标准 IO 口         |
|                 |                 |         |  | AD2      | I/O | 地址/数据总线         |
|                 |                 |         |  | ADC10    | I   | ADC 模拟输入通道 10   |
|                 |                 |         |  | PWM7_3   | I/O | PWM7 的捕获输入和脉冲输出 |
|                 |                 |         |  | T4_2     | I   | 定时器 4 外部时钟输入    |
| 46              | 32              |         |  | P0.3     | I/O | 标准 IO 口         |
|                 |                 |         |  | AD3      | I/O | 地址/数据总线         |
|                 |                 |         |  | ADC11    | I   | ADC 模拟输入通道 11   |
|                 |                 |         |  | PWM8_3   | I/O | PWM8 的捕获输入和脉冲输出 |
|                 |                 |         |  | T4CLKO_2 | O   | 定时器 4 时钟分频输出    |
| 47              |                 |         |  | P0.4     | I/O | 标准 IO 口         |
|                 |                 |         |  | AD4      | I/O | 地址/数据总线         |
|                 |                 |         |  | ADC12    | I   | ADC 模拟输入通道 12   |
|                 |                 |         |  | T3       | I   | 定时器 3 外部时钟输入    |
| 48              |                 |         |  | P5.2     | I/O | 标准 IO 口         |

## 2.4 STC8H3K64S4-45I-LQFP48/32,QFN48/32,TSSOP20 系列

### 2.4.1 特性及价格(有 16 位硬件乘法器 MDU16, 准 16 位单片机)

- 选型价格(不需要外部晶振、不需要外部复位, 12 位 ADC, 12 通道)

| 单片机型号       | 工作电压 (V) | Flash 程序存储器 10 万次 字节 | idata <sup>*</sup> 内部传统 8051 RAM 字节 | xdata <sup>*</sup> 内部大容量扩展 SRAM 字节 | EEPROM 10 万次 字节 | I/O 最多数量 | 传统 I/O 中断 (INT0/INT1/INT2/INT3/INT4) 并可掉电唤醒 | 所有的 I/O 口均支持中断并可掉电唤醒 | 串口并可掉电唤醒 | SPI 并可掉电唤醒 | I <sup>2</sup> C 并可掉电唤醒 | MDU16 硬件 16 位乘法器 | 定时器计数器 (T0/T1/T2/T3/T4 外部管脚也可掉电唤醒) | 16 位高级 PWM 定时器 互补对称死区控制 | 掉电唤醒专用定时器 | 12 路高速 ADC (8 路 PWM 占空比 8 路 D/A 使用) | 比较器 (可当 1 路 A/D, 可作外部掉电检测) | 内部低压检测中断并可掉电唤醒 | 看门狗 复位定时器 | 内部高可靠复位 (可选复位门檻电压) | 内部高精度时钟 (45MHz 以下可调) 追频 | 可对外输出时钟及复位 | 程序加密后传输 (防拦截) | 可设置下次更新程序需口令 | 支持 RS485 下载 | 支持软件模拟硬件 (USB 直接下载) | 本身就可在线仿真 | 价格及封装          |                 |                |                 | 供货信息 |
|-------------|----------|----------------------|-------------------------------------|------------------------------------|-----------------|----------|---|----------------------|----------|------------|-------------------------|------------------|------------------------------------|-------------------------|-----------|-------------------------------------|----------------------------|----------------|-----------|--------------------|-------------------------|------------|---------------|--------------|-------------|---------------------|----------|----------------|-----------------|----------------|-----------------|------|
|             |          |                      |                                     |                                    |                 |          |   |                      |          |            |                         |                  |                                    |                         |           |                                     |                            |                |           |                    |                         |            |               |              |             |                     |          | QFN32<4mm*4mm> | LQFP32<9mm*9mm> | QFN48<6mm*6mm> | LQFP48<9mm*9mm> |      |
| STC8H3K32S4 | 1.9-5.5  | 32K                  | 256                                 | 3K                                 | 2               | 32K      | 43  | 有                    | 有        | 4          | 有                       | 有                | 5                                  | 8                       | 有         | 12 位                                | 有                          | 有              | 有         | 4 级                | 有                       | 是          | 有             | 是            | 是           | 是                   | 是        | +              | +               | +              | +               | 现货   |
| STC8H3K48S4 | 1.9-5.5  | 48K                  | 256                                 | 3K                                 | 2               | 16K      | 43  | 有                    | 有        | 4          | 有                       | 有                | 5                                  | 8                       | 有         | 12 位                                | 有                          | 有              | 有         | 4 级                | 有                       | 是          | 有             | 是            | 是           | 是                   | 是        | √              | √               | √              | √               |      |
| STC8H3K60S4 | 1.9-5.5  | 60K                  | 256                                 | 3K                                 | 2               | 4K       | 43  | 有                    | 有        | 4          | 有                       | 有                | 5                                  | 8                       | 有         | 12 位                                | 有                          | 有              | 有         | 4 级                | 有                       | 是          | 有             | 是            | 是           | 是                   | 是        | √              | √               | √              | √               |      |
| STC8H3K64S4 | 1.9-5.5  | 64K                  | 256                                 | 3K                                 | 2               | 1AP      | 43  | 有                    | 有        | 4          | 有                       | 有                | 5                                  | 8                       | 有         | 12 位                                | 有                          | 有              | 有         | 4 级                | 有                       | 是          | 有             | 是            | 是           | 是                   | 是        | √3.5           | √3.5            | √3.5           | √3.5            |      |

➤ 内核

- ✓ 超高速 8051 内核 (1T), 比传统 8051 约快 12 倍以上
- ✓ 指令代码完全兼容传统 8051
- ✓ 27 个中断源, 4 级中断优先级
- ✓ 支持在线仿真

➤ 工作电压

- ✓ 1.9V~5.5V

➤ 工作温度

- ✓ -40℃~85℃ (如果需要工作在更宽的温度范围, 请使用外部时钟或者使用较低的工作频率)

➤ Flash 存储器

- ✓ 最大 64K 字节 FLASH 程序存储器 (ROM), 用于存储用户代码
- ✓ 支持用户配置 EEPROM 大小, 512 字节单页擦除, 擦写次数可达 10 万次以上
- ✓ 支持在系统编程方式 (ISP) 更新用户应用程序, 无需专用编程器
- ✓ 支持单芯片仿真, 无需专用仿真器, 理论断点个数无限制

➤ SRAM

- ✓ 128 字节内部直接访问 RAM (DATA, C 语言程序中使用 data 关键字进行声明)
- ✓ 128 字节内部间接访问 RAM (IDATA, C 语言程序中使用 idata 关键字进行声明)
- ✓ 3072 字节内部扩展 RAM (内部 XDATA, C 语言程序中使用 xdata 关键字进行声明)

## ➤ 时钟控制

- ✓ 内部高精度 IRC (4MHz~45MHz, ISP 编程时选择或手动输入, 还可以用户软件分频到较低的频率工作, 如 100KHz)
  - ✦ 误差±0.3% (常温下 25℃)
  - ✦ -1.35%~+1.30%温漂 (全温度范围, -40℃~85℃)
  - ✦ -0.76%~+0.98%温漂 (温度范围, -20℃~65℃)
  - ✦ **注意: 本系列的内部 IRC, 32M~37M 可能是盲区, 强烈建议选择 30MHz 及以下的频率, 或 40MHz 及以上的频率, 不可高于 45MHz**
- ✓ 内部 32KHz 低速 IRC (误差较大)
- ✓ 外部晶振 (4MHz~45MHz) 和外部时钟  
用户可自由选择上面的 3 种时钟源
- ✓ 关于 STC8H3K64S4 系列 B 版产品内部高速 IRC 的重要说明
  - ✦ 由于制造原因, 部分芯片的内部高速 IRC 在 34MHz~36MHz 可能存在盲区, 建议不要将工作频率设定在此区域
  - ✦ 内部高速 IRC 低温的温漂较高温时要大一些, 低频率段的温漂比高频段要大一些。一般的, 20MHz~40MHz 的工作频率, 在高温 85℃的温漂可控制在 0.8% 以内

## ➤ 复位

- ✓ 硬件复位
  - ✦ 上电复位。(在芯片未使能低压复位功能时有效)
  - ✦ 复位脚复位。出厂时 P5.4 默认为 I/O 口, ISP 下载时可将 P5.4 管脚设置为复位脚 (注意: 当设置 P5.4 管脚为复位脚时, 复位电平为低电平)
  - ✦ 看门狗溢出复位
  - ✦ 低压检测复位, 提供 4 级低压检测电压: 2.0V、2.4V、2.7V、3.0V。
- ✓ 软件复位
  - ✦ 软件方式写复位触发寄存器

## ➤ 中断

- ✓ 提供 27 个中断源: INT0 (支持上升沿和下降沿中断)、INT1 (支持上升沿和下降沿中断)、INT2 (只支持下降沿中断)、INT3 (只支持下降沿中断)、INT4 (只支持下降沿中断)、定时器 0、定时器 1、定时器 2、定时器 3、定时器 4、串口 1、串口 2、串口 3、串口 4、ADC 模数转换、LVD 低压检测、SPI、I<sup>2</sup>C、比较器、PWMA、PWMB、P0 口中断、P1 口中断、P2 口中断、P3 口中断、P4 口中断、P5 口中断
- ✓ 提供 4 级中断优先级
- ✓ 时钟停振模式下可以唤醒的中断: INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、T3(P0.4)、T4(P0.6)、RXD(P3.0/P3.6/P1.6/P4.3)、RXD2(P1.0/P4.6)、RXD3(P0.0/P5.0)、RXD4(P0.2/P5.2)、I2C\_SDA(P1.4/P2.4/P3.3)以及比较器中断、低压检测中断、掉电唤醒定时器唤醒。

## ➤ 数字外设

- ✓ 5 个 16 位定时器: 定时器 0、定时器 1、定时器 2、定时器 3、定时器 4, 其中定时器 0 的模式 3 具有 NMI (不可屏蔽中断) 功能, 定时器 0 和定时器 1 的模式 0 为 16 位自动重载模式
- ✓ 4 个高速串口: 串口 1、串口 2、串口 3、串口 4, 波特率时钟源最快可为 FOSC/4
- ✓ 8 路/2 组高级 PWM, 可实现带死区的控制信号, 并支持外部异常检测功能, 另外还支持 16 位定时器、8 个外部中断、8 路外部捕获测量脉宽等功能
- ✓ SPI: 支持主机模式和从机模式以及主机/从机自动切换
- ✓ I<sup>2</sup>C: 支持主机模式和从机模式
- ✓ MDU16: 硬件 16 位乘除法器 (支持 32 位除以 16 位、16 位除以 16 位、16 位乘 16 位、数据移位以及数据规格化等运算)

- ✓ I/O 口中断: 所有的 I/O 均支持中断, 每组 I/O 中断有独立的中断入口地址, 所有的 I/O 中断可支持 4 种中断模式: 高电平中断、低电平中断、上升沿中断、下降沿中断

(注意: STC8H3K64S4 系列 A 版芯片的 I/O 口中断不能进行掉电唤醒, **B 版芯片的 I/O 口中断可以进行掉电唤醒, 但只有一级中断优先级。B 版芯片的 I/O 口中断经测试发现问题, 暂请不要使用**)

#### ➤ 模拟外设

- ✓ 超高速 ADC, 支持 12 位高精度 12 通道 (通道 0~通道 2、通道 6~通道 14, 无 P1.3/P1.4/P1.5 端口, 所以少了通道 3~5) 的模数转换, 速度最快能达到 800K (每秒进行 80 万次 ADC 转换)
- ✓ ADC 的通道 15 用于测试内部 1.19V 参考信号源 (芯片在出厂时, 内部参考信号源已调整为 1.19V)
- ✓ 比较器, 一组比较器 (比较器的正端可选择 CMP+端口和所有的 ADC 输入端口, 所以比较器可当作多路比较器进行分时复用)
- ✓ DAC: 8 路高级 PWM 定时器可当 8 路 DAC 使用

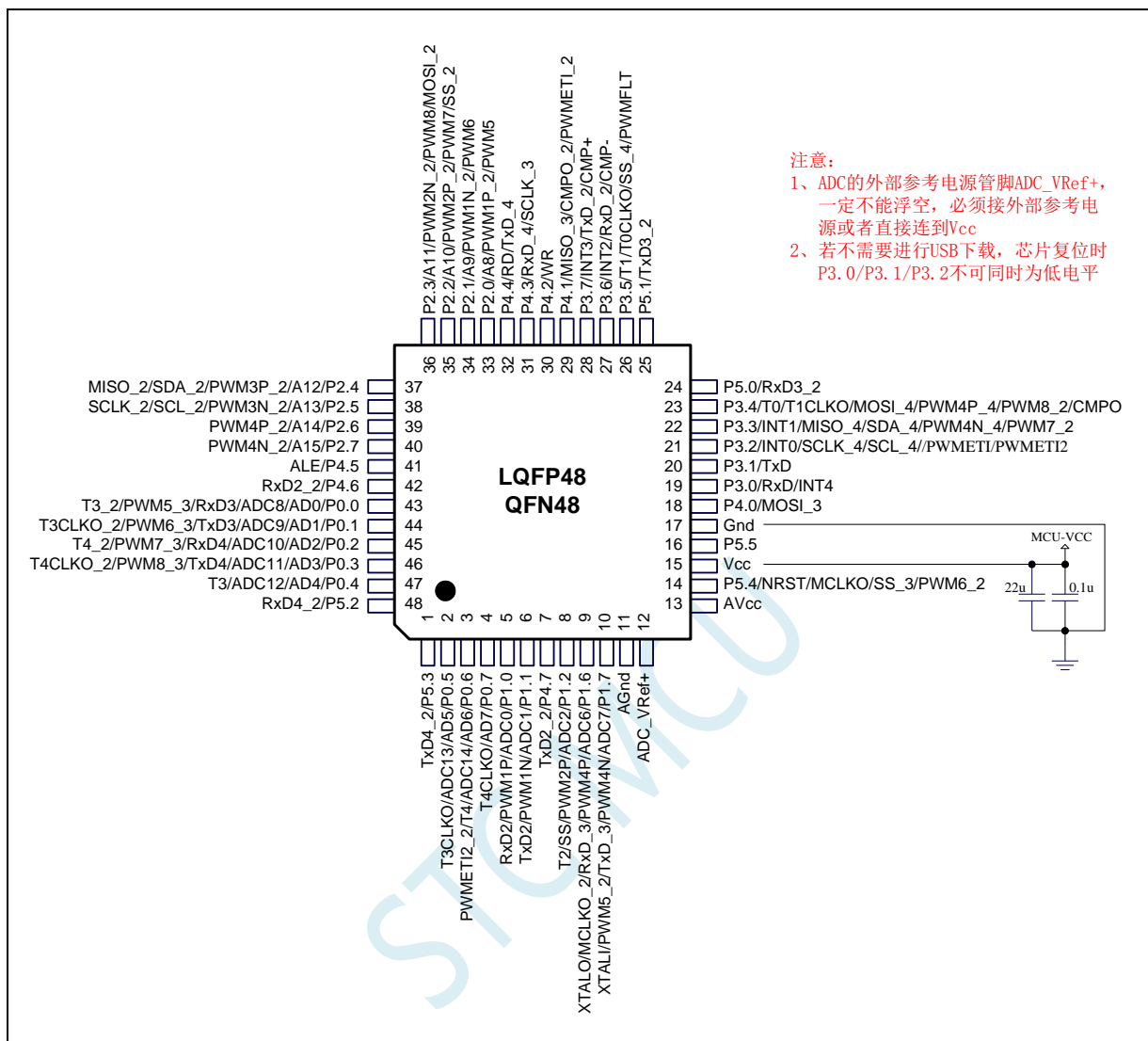
#### ➤ GPIO

- ✓ 最多可达 43 个 GPIO: P0.0~P0.7、P1.0~P1.2、P1.6~P1.7、P2.0~P2.7、P3.0~P3.7、P4.0~P4.7、P5.0~P5.5
- ✓ 所有的 GPIO 均支持如下 4 种模式: 准双向口模式、强推挽输出模式、开漏输出模式、高阻输入模式
- ✓ 除 P3.0 和 P3.1 外, 其余所有 IO 口上电后的状态均为高阻输入状态, 用户在使用 IO 口时必须先设置 IO 口模式。另外每个 I/O 均可独立使能内部 4K 上拉电阻

#### ➤ 封装

- ✓ LQFP48 <9mm\*9mm>、QFN48 <6mm\*6mm>、LQFP32 <9mm\*9mm>、QFN32 <4mm\*4mm> (**LQFP32、QFN32、暂无样品, 后续会有, 若有需要请提前订货**)

## 2.4.2 管脚图，最小系统

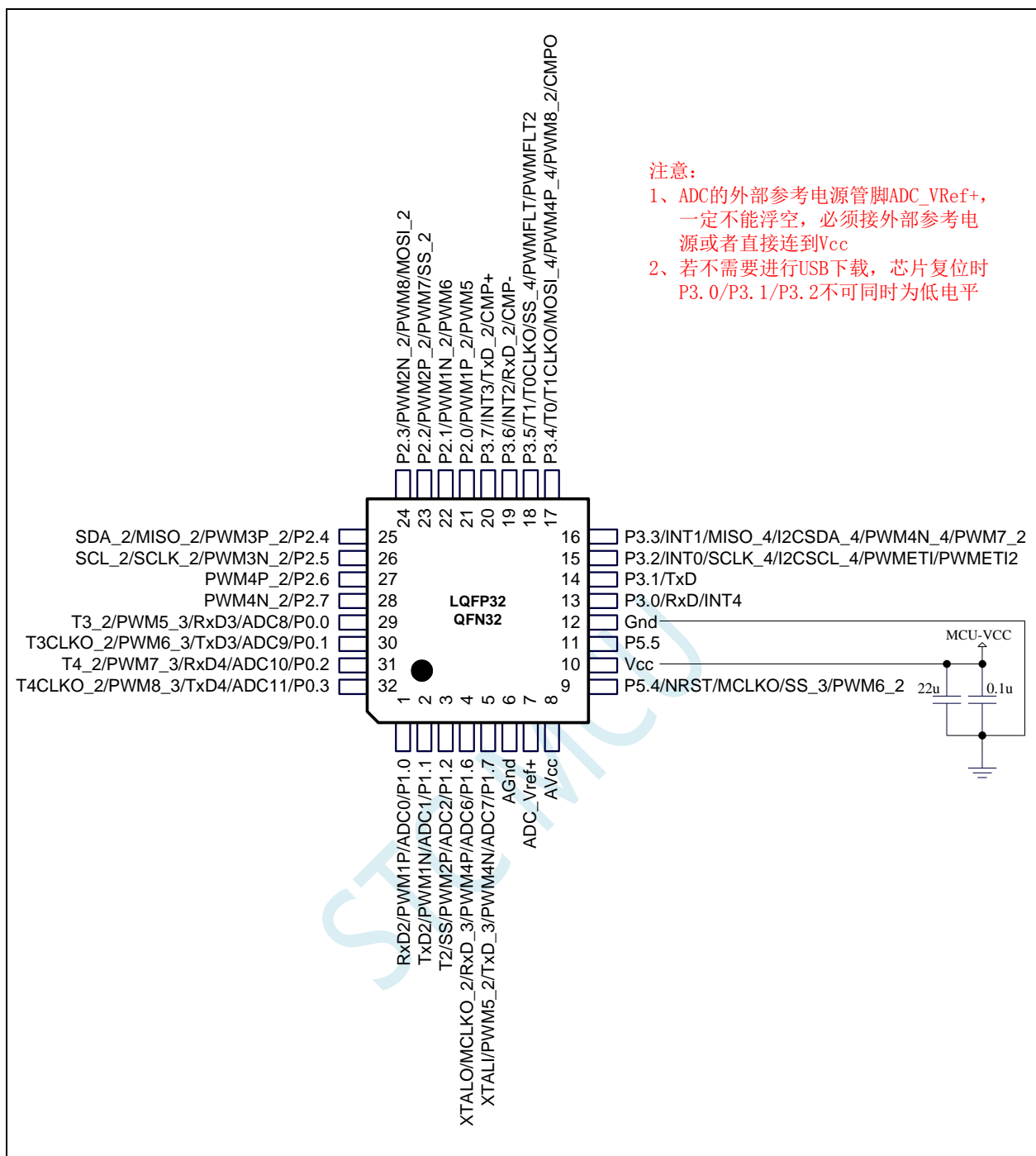


正看芯片丝印左下方小圆点处为第一脚

正看芯片丝印最下面一行最后一个字母为芯片版本号

建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF，可去除电源线噪声，提高抗干扰能力

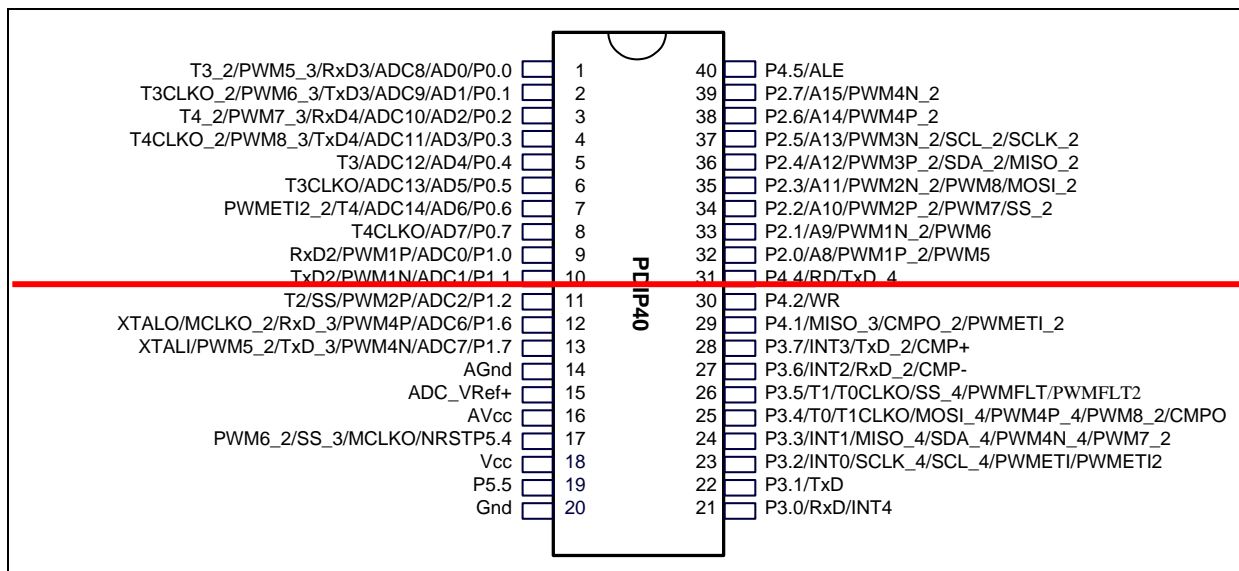




正看芯片丝印左下方小圆点处为第一脚

正看芯片丝印最下面一行最后一个字母为芯片版本号

建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF, 可去除电源线噪声, 提高抗干扰能力



## 2.4.3 管脚说明

| 编号              |                 | 名称        | 类型  | 说明                |
|-----------------|-----------------|-----------|-----|-------------------|
| LQFP48<br>QFN48 | LQFP32<br>QFN32 |           |     |                   |
| 1               |                 | P5.3      | I/O | 标准 IO 口           |
|                 |                 | TxD4_2    | O   | 串口 4 的发送脚         |
| 2               |                 | P0.5      | I/O | 标准 IO 口           |
|                 |                 | AD5       | I/O | 地址/数据总线           |
|                 |                 | ADC13     | I   | ADC 模拟输入通道 13     |
|                 |                 | T3CLKO    | O   | 定时器 3 时钟分频输出      |
| 3               |                 | P0.6      | I/O | 标准 IO 口           |
|                 |                 | AD6       | I/O | 地址/数据总线           |
|                 |                 | ADC14     | I   | ADC 模拟输入通道 14     |
|                 |                 | T4        | I   | 定时器 4 外部时钟输入      |
|                 |                 | PWMETI2_2 | I   | PWM 外部触发输入脚 2     |
| 4               |                 | P0.7      | I/O | 标准 IO 口           |
|                 |                 | AD7       | I/O | 地址/数据总线           |
|                 |                 | T4CLKO    | O   | 定时器 4 时钟分频输出      |
| 5               | 1               | P1.0      | I/O | 标准 IO 口           |
|                 |                 | ADC0      | I   | ADC 模拟输入通道 0      |
|                 |                 | PWM1P     | I/O | PWM1 的捕获输入和脉冲输出正极 |
|                 |                 | RxD2      | I   | 串口 2 的接收脚         |
| 6               | 2               | P1.1      | I/O | 标准 IO 口           |
|                 |                 | ADC1      | I   | ADC 模拟输入通道 1      |
|                 |                 | PWM1N     | I/O | PWM1 的捕获输入和脉冲输出负极 |
|                 |                 | TxD2      | I   | 串口 2 的发送脚         |
| 7               |                 | P4.7      | I/O | 标准 IO 口           |
|                 |                 | TxD2_2    | I   | 串口 2 的发送脚         |
| 8               | 3               | P1.2      | I/O | 标准 IO 口           |
|                 |                 | ADC2      | I   | ADC 模拟输入通道        |
|                 |                 | PWM2P     | I/O | PWM2 的捕获输入和脉冲输出正极 |
|                 |                 | SS        | I   | SPI 的从机选择脚（主机为输出） |
|                 |                 | T2        | I   | 定时器 2 外部时钟输入      |

| 编号              |                 | 名称        | 类型  | 说明                                  |
|-----------------|-----------------|-----------|-----|-------------------------------------|
| LQFP48<br>QFN48 | LQFP32<br>QFN32 |           |     |                                     |
| 9               | 4               | P1.6      | I/O | 标准 IO 口                             |
|                 |                 | ADC6      | I   | ADC 模拟输入通道 6                        |
|                 |                 | RxD_3     | I   | 串口 1 的接收脚                           |
|                 |                 | PWM4P     | I/O | PWM4 的捕获输入和脉冲输出正极                   |
|                 |                 | MCLKO_2   | O   | 主时钟分频输出                             |
|                 |                 | XTALO     | O   | 外部晶振的输出脚                            |
| 10              | 5               | P1.7      | I/O | 标准 IO 口                             |
|                 |                 | ADC7      | I   | ADC 模拟输入通道 7                        |
|                 |                 | TxD_3     | O   | 串口 1 的发送脚                           |
|                 |                 | PWM4N     | I/O | PWM4 的捕获输入和脉冲输出负极                   |
|                 |                 | PWM5_2    | I/O | PWM5 的捕获输入和脉冲输出                     |
|                 |                 | XTALI     | I   | 外部晶振/外部时钟的输入脚                       |
| 11              | 6               | AGnd      | Gnd | ADC 地线                              |
| 12              | 7               | ADC_VRef+ | I   | ADC 外部参考电压源输入脚, 要求不高时可直接接 MCU 的 VCC |
| 13              | 8               | AVcc      | Vcc | ADC 电源脚                             |
| 14              | 9               | P5.4      | I/O | 标准 IO 口                             |
|                 |                 | NRST      | I   | 复位引脚 (低电平复位)                        |
|                 |                 | MCLKO     | O   | 主时钟分频输出                             |
|                 |                 | SS_3      | I   | SPI 的从机选择脚 (主机为输出)                  |
|                 |                 | PWM6_2    | I/O | PWM6 的捕获输入和脉冲输出                     |
| 15              | 10              | Vcc       | Vcc | 电源脚                                 |
| 16              | 11              | P5.5      | I/O | 标准 IO 口                             |
| 17              | 12              | Gnd       | Gnd | 地线                                  |

| 编号              |                 | 名称      | 类型  | 说明                |
|-----------------|-----------------|---------|-----|-------------------|
| LQFP48<br>QFN48 | LQFP32<br>QFN32 |         |     |                   |
| 18              |                 | P4.0    | I/O | 标准 IO 口           |
|                 |                 | MOSI_3  | I/O | SPI 主机输出从机输入      |
| 19              | 13              | P3.0    | I/O | 标准 IO 口           |
|                 |                 | RxD     | I   | 串口 1 的接收脚         |
|                 |                 | INT4    | I   | 外部中断 4            |
| 20              | 14              | P3.1    | I/O | 标准 IO 口           |
|                 |                 | TxD     | O   | 串口 1 的发送脚         |
| 21              | 15              | P3.2    | I/O | 标准 IO 口           |
|                 |                 | INT0    | I   | 外部中断 0            |
|                 |                 | SCLK_4  | I/O | SPI 的时钟脚          |
|                 |                 | SCL_4   | I/O | I2C 的时钟线          |
|                 |                 | PWMETI  | I   | PWM 外部触发输入脚       |
|                 |                 | PWMETI2 | I   | PWM 外部触发输入脚 2     |
| 22              | 16              | P3.3    | I/O | 标准 IO 口           |
|                 |                 | INT1    | I   | 外部中断 1            |
|                 |                 | MISO_4  | I/O | SPI 主机输入从机输出      |
|                 |                 | SDA_4   | I/O | I2C 接口的数据线        |
|                 |                 | PWM4N_4 | I/O | PWM4 的捕获输入和脉冲输出负极 |
|                 |                 | PWM7_2  | I/O | PWM7 的捕获输入和脉冲输出   |
| 23              | 17              | P3.4    | I/O | 标准 IO 口           |
|                 |                 | T0      | I   | 定时器 0 外部时钟输入      |
|                 |                 | T1CLKO  | O   | 定时器 1 时钟分频输出      |
|                 |                 | MOSI_4  | I/O | SPI 主机输出从机输入      |
|                 |                 | PWM4P_4 | I/O | PWM4 的捕获输入和脉冲输出正极 |
|                 |                 | PWM8_2  | I/O | PWM8 的捕获输入和脉冲输出   |
|                 |                 | CMPO    | O   | 比较器输出             |
| 24              |                 | P5.0    | I/O | 标准 IO 口           |
|                 |                 | RxD3_2  | I   | 串口 3 的接收脚         |
| 25              |                 | P5.1    | I/O | 标准 IO 口           |
|                 |                 | TxD3_2  | O   | 串口 3 的发送脚         |

| 编号              |                 | 名称       | 类型  | 说明                |
|-----------------|-----------------|----------|-----|-------------------|
| LQFP48<br>QFN48 | LQFP32<br>QFN32 |          |     |                   |
| 26              | 18              | P3.5     | I/O | 标准 IO 口           |
|                 |                 | T1       | I   | 定时器 1 外部时钟输入      |
|                 |                 | T0CLKO   | O   | 定时器 0 时钟分频输出      |
|                 |                 | SS_4     | I   | SPI 的从机选择脚（主机为输出） |
|                 |                 | PWMFLT   | I   | 增强 PWM 的外部异常检测脚   |
| 27              | 19              | P3.6     | I/O | 标准 IO 口           |
|                 |                 | INT2     | I   | 外部中断 2            |
|                 |                 | RxD_2    | I   | 串口 1 的接收脚         |
|                 |                 | CMP-     | I   | 比较器负极输入           |
| 28              | 20              | P3.7     | I/O | 标准 IO 口           |
|                 |                 | INT3     | I   | 外部中断 3            |
|                 |                 | TxD_2    | O   | 串口 1 的发送脚         |
|                 |                 | CMP+     | I   | 比较器正极输入           |
| 29              |                 | P4.1     | I/O | 标准 IO 口           |
|                 |                 | MISO_3   | I/O | SPI 主机输入从机输出      |
|                 |                 | CMPO_2   | O   | 比较器输出             |
|                 |                 | PWMETI_2 | I   | PWM 外部触发输入脚       |
| 30              |                 | P4.2     | I/O | 标准 IO 口           |
|                 |                 | WR       | O   | 外部总线的写信号线         |
| 31              |                 | P4.3     | I/O | 标准 IO 口           |
|                 |                 | RxD_4    | I   | 串口 1 的接收脚         |
|                 |                 | SCLK_3   | I/O | SPI 的时钟脚          |
| 32              |                 | P4.4     | I/O | 标准 IO 口           |
|                 |                 | RD       | O   | 外部总线的读信号线         |
|                 |                 | TxD_4    | O   | 串口 1 的发送脚         |
| 33              | 21              | P2.0     | I/O | 标准 IO 口           |
|                 |                 | A8       | O   | 地址总线              |
|                 |                 | PWM1P_2  | I/O | PWM1 的捕获输入和脉冲输出正极 |
|                 |                 | PWM5     | I/O | PWM5 的捕获输入和脉冲输出   |

| 编号              |                 | 名称      | 类型  | 说明                |
|-----------------|-----------------|---------|-----|-------------------|
| LQFP48<br>QFN48 | LQFP32<br>QFN32 |         |     |                   |
| 34              | 22              | P2.1    | I/O | 标准 IO 口           |
|                 |                 | A9      | O   | 地址总线              |
|                 |                 | PWM1N_2 | I/O | PWM1 的捕获输入和脉冲输出负极 |
|                 |                 | PWM6    | I/O | PWM6 的捕获输入和脉冲输出   |
| 35              | 23              | P2.2    | I/O | 标准 IO 口           |
|                 |                 | A10     | O   | 地址总线              |
|                 |                 | SS_2    | I   | SPI 的从机选择脚（主机为输出） |
|                 |                 | PWM2P_2 | I/O | PWM2 的捕获输入和脉冲输出正极 |
|                 |                 | PWM7    | I/O | PWM7 的捕获输入和脉冲输出   |
| 36              | 24              | P2.3    | I/O | 标准 IO 口           |
|                 |                 | A11     | O   | 地址总线              |
|                 |                 | MOSI_2  | I/O | SPI 主机输出从机输入      |
|                 |                 | PWM2N_2 | I/O | PWM2 的捕获输入和脉冲输出负极 |
|                 |                 | PWM8    | I/O | PWM8 的捕获输入和脉冲输出   |
| 37              | 25              | P2.4    | I/O | 标准 IO 口           |
|                 |                 | A12     | O   | 地址总线              |
|                 |                 | MISO_2  | I/O | SPI 主机输入从机输出      |
|                 |                 | SDA_2   | I/O | I2C 接口的数据线        |
|                 |                 | PWM3P_2 | I/O | PWM3 的捕获输入和脉冲输出正极 |
| 38              | 26              | P2.5    | I/O | 标准 IO 口           |
|                 |                 | A13     | O   | 地址总线              |
|                 |                 | SCLK_2  | I/O | SPI 的时钟脚          |
|                 |                 | SCL_2   | I/O | I2C 的时钟线          |
|                 |                 | PWM3N_2 | I/O | PWM3 的捕获输入和脉冲输出负极 |
| 39              | 27              | P2.6    | I/O | 标准 IO 口           |
|                 |                 | A14     | O   | 地址总线              |
|                 |                 | PWM4P_2 | I/O | PWM4 的捕获输入和脉冲输出正极 |
| 40              | 28              | P2.7    | I/O | 标准 IO 口           |
|                 |                 | A15     | O   | 地址总线              |
|                 |                 | PWM4N_2 | I/O | PWM4 的捕获输入和脉冲输出负极 |
| 41              |                 | P4.5    | I/O | 标准 IO 口           |
|                 |                 | ALE     | O   | 地址锁存信号            |



| 编号              |                 | 名称       | 类型  | 说明              |
|-----------------|-----------------|----------|-----|-----------------|
| LQFP48<br>QFN48 | LQFP32<br>QFN32 |          |     |                 |
| 42              |                 | P4.6     | I/O | 标准 IO 口         |
|                 |                 | RxD2_2   | I   | 串口 2 的接收脚       |
| 43              | 29              | P0.0     | I/O | 标准 IO 口         |
|                 |                 | AD0      | I/O | 地址/数据总线         |
|                 |                 | ADC8     | I   | ADC 模拟输入通道 8    |
|                 |                 | RxD3     | I   | 串口 3 的接收脚       |
|                 |                 | PWM5_3   | I/O | PWM5 的捕获输入和脉冲输出 |
|                 |                 | T3_2     | I   | 定时器 3 外部时钟输入    |
| 44              | 30              | P0.1     | I/O | 标准 IO 口         |
|                 |                 | AD1      | I/O | 地址/数据总线         |
|                 |                 | ADC9     | I   | ADC 模拟输入通道 9    |
|                 |                 | TxD3     | O   | 串口 3 的发送脚       |
|                 |                 | PWM6_3   | I/O | PWM6 的捕获输入和脉冲输出 |
|                 |                 | T3CLKO_2 | O   | 定时器 3 时钟输入      |
| 45              | 31              | P0.2     | I/O | 标准 IO 口         |
|                 |                 | AD2      | I/O | 地址/数据总线         |
|                 |                 | ADC10    | I   | ADC 模拟输入通道 10   |
|                 |                 | RxD4     | I   | 串口 4 的接收脚       |
|                 |                 | PWM7_3   | I/O | PWM7 的捕获输入和脉冲输出 |
|                 |                 | T4_2     | I   | 定时器 4 外部时钟输入    |
| 46              | 32              | P0.3     | I/O | 标准 IO 口         |
|                 |                 | AD3      | I/O | 地址/数据总线         |
|                 |                 | ADC11    | I   | ADC 模拟输入通道 11   |
|                 |                 | TxD4     | O   | 串口 4 的发送脚       |
|                 |                 | PWM8_3   | I/O | PWM8 的捕获输入和脉冲输出 |
|                 |                 | T4CLKO_2 | O   | 定时器 4 时钟输入      |
| 47              |                 | P0.4     | I/O | 标准 IO 口         |
|                 |                 | AD4      | I/O | 地址/数据总线         |
|                 |                 | ADC12    | I   | ADC 模拟输入通道 12   |
|                 |                 | T3       | I   | 定时器 3 外部时钟输入    |
| 48              |                 | P5.2     | I/O | 标准 IO 口         |
|                 |                 | RxD4_2   | I   | 串口 4 的接收脚       |

## 2.5 STC8H8K64U-45I-LQFP/QFN-64/48/32,TSSOP20

### 2.5.1 特性及价格(有 16 位硬件乘除法器 MDU16, 准 16 位单片机)

- 选型价格(不需要外部晶振、不需要外部复位, 12 位 ADC, 15 通道)

| 价格及封装  | 供货信息                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 供货中 |
|--|-----------------------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|-----|
|  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
|  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
|  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 价格及封装  | TSSOP20 <6.5mm*6.5mm> |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | √   |
|  | QFN32<4mm*4mm>        |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | √   |
|  | LQFP32<9mm*9mm>       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | √   |
|  | QFN48 <6mm*6mm>       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | √   |
|  | LQFP48 <9mm*9mm>      |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | √   |
|  | QFN64 <8mm*8mm>       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | √   |
|  | LQFP64 <12mm*12mm>    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | √   |
| 本身就可在线仿真   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 支持硬件全速 USB 直接下载并可仿真  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 支持 RS485 下载  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 可设置下次更新程序需口令   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 程序加密后传输（防拦截）   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 可对外输出时钟及复位   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 内部高精度时钟（45MHz 以下可调）追频  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 内部高可靠复位（可选复位门檻电压）  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 看门狗 复位定时器  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 内部低压检测中断并可掉电唤醒   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 比较器（可当 1 路 A/D，可作外部掉电检测）   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| DMA 15 路高速 ADC（8 路 PWM 可前 8 路 D/A 使用）  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 掉电唤醒专用定时器  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 16 位高级 PWM 定时器 互补对称死区控制  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 定时器计数器（T0/T1/T2/T3/T4 外部管脚也可掉电唤醒）  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| MDU16 硬件 16 位乘除法器  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| I <sup>2</sup> C 并可掉电唤醒（暂不 DMA）  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| DMA SPI 并可掉电唤醒   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 全速 USB   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| DMA 串口并可掉电唤醒   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| RTC 实时时钟   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| DMA 8080/6800 接口 LCM 模块驱动 8 位和 16 位  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 所有的 I/O 口均支持中断并可掉电唤醒   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 传统 I/O 中断（INT0/INT1/INT2/INT3/INT4）并可掉电唤醒  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| I/O 口最多数量  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| EEPROM 10 万次 字节  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 强大的双 DPTR 可增可减   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| xdata* 内部大容量扩展 SRAM 字节   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| idata* 内部传统 8051 RAM 字节  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| Flash 程序存储器 10 万次 字节   |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 工作电压（V）  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| 单片机型号  |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |
| STC8H8K32U1.9-5.532K2568K232K60有有有有4有有有有58有12位有有有4级有是是有是是是√√ |                       |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |     |

➤ 内核

- ✓ 超高速 8051 内核 (1T), 比传统 8051 约快 12 倍以上
- ✓ 指令代码完全兼容传统 8051
- ✓ 22 个中断源, 4 级中断优先级
- ✓ 支持在线仿真

➤ 工作电压

- ✓ 1.9V~5.5V

➤ 工作温度

- ✓ -40℃~85℃ (芯片为-40℃~125℃制程, 超温度范围应用请参考电气特性章节说明)

➤ Flash 存储器

- ✓ 最大 64K 字节 FLASH 程序存储器 (ROM), 用于存储用户代码
- ✓ 支持用户配置 EEPROM 大小, 512 字节单页擦除, 擦写次数可达 10 万次以上
- ✓ 支持在系统编程方式 (ISP) 更新用户应用程序, 无需专用编程器
- ✓ 支持单芯片仿真, 无需专用仿真器, 理论断点个数无限制

➤ SRAM

- ✓ 128 字节内部直接访问 RAM (DATA, C 语言程序中使用 data 关键字进行声明)
- ✓ 128 字节内部间接访问 RAM (IDATA, C 语言程序中使用 idata 关键字进行声明)
- ✓ 8192 字节内部扩展 RAM (内部 XDATA, C 语言程序中使用 xdata 关键字进行声明)
- ✓ 1280 字节 USB 数据 RAM

## ➤ 时钟控制

- ✓ 内部高精度 IRC (4MHz~45MHz, ISP 编程时选择或手动输入, 还可以用户软件分频到较低的频率工作, 如 100KHz)
  - ⊕ 误差±0.3% (常温下 25℃)
  - ⊕ -1.35%~+1.30%温漂 (全温度范围, -40℃~85℃)
  - ⊕ -0.76%~+0.98%温漂 (温度范围, -20℃~65℃)
- ✓ 内部 32KHz 低速 IRC (误差较大)
- ✓ 外部晶振 (4MHz~45MHz) 和外部时钟  
用户可自由选择上面的 3 种时钟源

## ➤ 复位

- ✓ 硬件复位
  - ⊕ 上电复位, 实测电压值为 1.69V~1.82V。(在芯片未使能低压复位功能时有效)  
上电复位电压由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到上电复位的下限门槛电压时, 芯片处于复位状态; 当电压从 0V 上升到上电复位的上限门槛电压时, 芯片解除复位状态。
  - ⊕ 复位脚复位, 出厂时 P5.4 默认为 I/O 口, ISP 下载时可将 P5.4 管脚设置为复位脚 (注意: 当设置 P5.4 管脚为复位脚时, 复位电平为低电平)
  - ⊕ 看门狗溢出复位
  - ⊕ 低压检测复位, 提供 4 级低压检测电压: 1.9V、2.3V、2.8V、3.7V。  
每级低压检测电压都是由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到低压检测的下限门槛电压时, 低压检测生效; 当电压从 0V 上升到低压检测的上限门槛电压时, 低压检测生效。
- ✓ 软件复位
  - ⊕ 软件方式写复位触发寄存器

## ➤ 中断

- ✓ 提供 22 个中断源: INT0 (支持上升沿和下降沿中断)、INT1 (支持上升沿和下降沿中断)、INT2 (只支持下降沿中断)、INT3 (只支持下降沿中断)、INT4 (只支持下降沿中断)、定时器 0、定时器 1、定时器 2、定时器 3、定时器 4、串口 1、串口 2、串口 3、串口 4、ADC 模数转换、LVD 低压检测、SPI、I<sup>2</sup>C、比较器、PWMA、PWMB、USB
- ✓ 提供 4 级中断优先级
- ✓ 时钟停振模式下可以唤醒的中断: INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、T3(P0.4)、T4(P0.6)、RXD(P3.0/P3.6/P1.6/P4.3)、RXD2(P1.0/P4.6)、RXD3(P0.0/P5.0)、RXD4(P0.2/P5.2)、I2C\_SDA(P1.4/P2.4/P3.3)以及比较器中断、低压检测中断、掉电唤醒定时器唤醒。

## ➤ 数字外设

- ✓ 5 个 16 位定时器: 定时器 0、定时器 1、定时器 2、定时器 3、定时器 4, 其中定时器 0 的模式 3 具有 NMI (不可屏蔽中断) 功能, 定时器 0 和定时器 1 的模式 0 为 16 位自动重载模式
- ✓ 4 个高速串口: 串口 1、串口 2、串口 3、串口 4, 波特率时钟源最快可为 FOSC/4
- ✓ 8 路/2 组高级 PWM, 可实现带死区的控制信号, 并支持外部异常检测功能, 另外还支持 16 位定时器、8 个外部中断、8 路外部捕获测量脉宽等功能
- ✓ SPI: 支持主机模式和从机模式以及主机/从机自动切换
- ✓ I<sup>2</sup>C: 支持主机模式和从机模式
- ✓ MDU16: 硬件 16 位乘除法器 (支持 32 位除以 16 位、16 位除以 16 位、16 位乘 16 位、数据移位以及数据规格化等运算)
- ✓ USB: USB2.0/USB1.1 兼容全速 USB, 6 个双向端点, 支持 4 种端点传输模式 (控制传输、中断传输、批

量传输和同步传输), 每个端点拥有 64 字节的缓冲区

- ✓ RTC: 支持年、月、日、时、分、秒、次秒 (1/128 秒), 并支持时钟中断和一组闹钟 (注: A 版芯片无此功能)
- ✓ I/O 口中断: 所有的 I/O 均支持中断, 每组 I/O 中断有独立的中断入口地址, 所有的 I/O 中断可支持 4 种中断模式: 高电平中断、低电平中断、上升沿中断、下降沿中断。提供 4 级中断优先级并支持掉电唤醒功能。(注: A 版芯片无此功能)
- ✓ DMA: 支持 Memory-To-Memory、SPI、UART1TX/UART1RX、UART2TX/UART2RX、UART3TX/UART3RX、UART4TX/UART4RX、ADC (自动计算多次 ADC 结果的平均值)、LCM (注: A 版芯片无此功能)
- ✓ LCM (TFT 彩屏) 驱动: 支持 8080 和 6800 接口, 支持 8 位和 16 位数据宽度 (注: A 版芯片无此功能)
  - ✦ 8 位 8080 总线: 8 为数据线 (TD0~TD7), 读信号 (TRD), 写信号 (TWR), RS 线 (TRS)
  - ✦ 16 为 8080 总线: 16 为数据线 (TD0~TD15), 读信号 (TRD), 写信号 (TWR), RS 线 (TRS)
  - ✦ 8 为 6800 总线: 8 为数据线 (TD0~TD7), 使能信号 (TE), 读写信号 (TRW), RS 线 (TRS)
  - ✦ 16 位 6800 总线: 16 为数据线 (TD0~TD15), 使能信号 (TE), 读写信号 (TRW), RS 线 (TRS)
  - ✦ 注意: 如果使用 8 位数据线控制 TFT 屏, 一般需要 TD0~D7, TRD/TWR/TRS, 11 根数据及控制线, 外加 2 个普通 I/O 控制片选及复位 (很多 TFT 彩屏的片选及复位厂商已进行了自动处理, 并不需要软件控制)

#### ➤ 模拟外设

- ✓ 超高速 ADC, 支持 12 位高精度 15 通道 (通道 0~通道 14) 的模数转换, 速度最快能达到 800K (每秒进行 80 万次 ADC 转换)
- ✓ ADC 的通道 15 用于测试内部 1.19V 参考信号源 (芯片在出厂时, 内部参考信号源已调整为 1.19V)
- ✓ 比较器, 一组比较器 (A 版芯片: 比较器的正端可选择 CMP+ 和所有的 ADC 输入端口, 比较器的负端可选择 CMP- 和内部 1.19V 的参考源; **B 版芯片: 比较器的正端可选择 CMP+、CMP+\_2、CMP+\_3 和所有的 ADC 输入端口, 比较器的负端可选择 CMP- 端口和内部 1.19V 的参考源。**所以比较器可当作多路比较器进行分时复用)
- ✓ DAC: 8 路高级 PWM 定时器可当 8 路 DAC 使用

#### ➤ GPIO

- ✓ 最多可达 60 个 GPIO: P0.0~P0.7、P1.0~P1.7 (无 P1.2)、P2.0~P2.7、P3.0~P3.7、P4.0~P4.7、P5.0~P5.4、P6.0~P6.7、P7.0~P7.7
- ✓ 所有的 GPIO 均支持如下 4 种模式: 准双向口模式、强推挽输出模式、开漏输出模式、高阻输入模式
- ✓ 除 P3.0 和 P3.1 外, 其余所有 IO 口上电后的状态均为高阻输入状态, 用户在使用 IO 口时必须先设置 IO 口模式。另外每个 I/O 均可独立使能内部 4K 上拉电阻

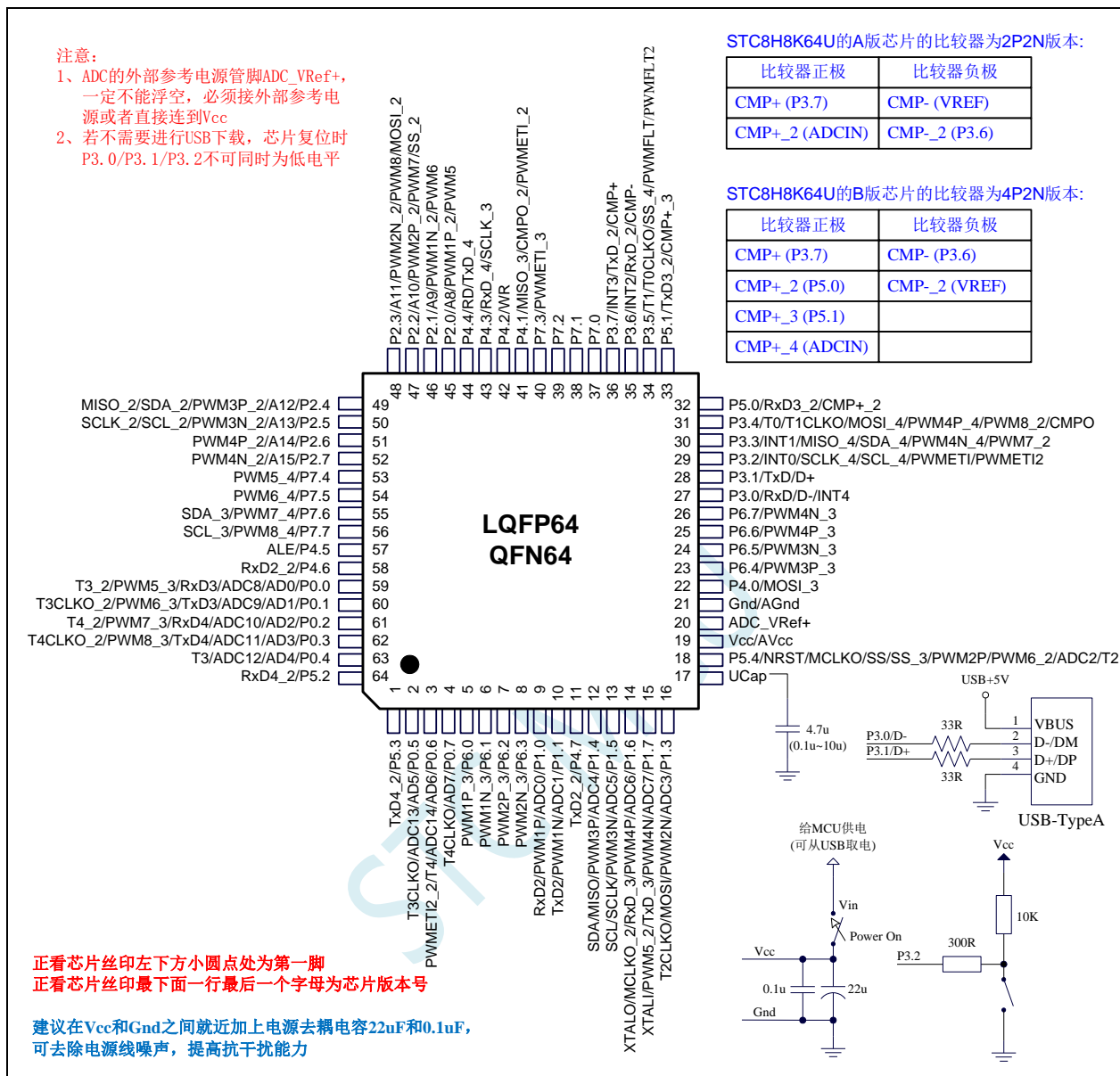
#### ➤ 封装

- ✓ LQFP64 <12mm\*12mm>、QFN64 <8mm\*8mm>、LQFP48 <9mm\*9mm>、QFN48 <6mm\*6mm>、LQFP32<9mm\*9mm>、QFN32 <4mm\*4mm>、TSSOP20

#### ✓ 特别说明

- ✓ B 版芯片有 CHIPID 功能 (A 版无 CHIPID 功能)
- ✓ B 版芯片有所有的 I/O 口中断功能, 有 4 级中断优先级, 可掉电唤醒 (A 版芯片所有的 I/O 口中断功能)
- ✓ B 版芯片的比较器为 4P+2N 版本, A 版芯片为 2P+2N 版本

## 2.5.2 管脚图，最小系统



现在 STC 的带硬件 USB 的 MCU 支持用硬件 USB 下载, 因为用的是 USB-HID 通信协议, 不需要安装任何驱动。STC 打狗棒、降龙棍、开天斧、屠龙刀核心板以及 STC 开源示波器、STC 实验箱在 D-/P3.0, D+/P3.1 与 PC-USB 端口连接好的状况下, USB-ISP 下载程序步骤:

1、按下板子上的 P3.2/INT0 按键, 就是 P3.2 接地

2、给目标芯片重新上电, 不管之前是否已通电。

===电子开关是按下停电后再松开就是上电

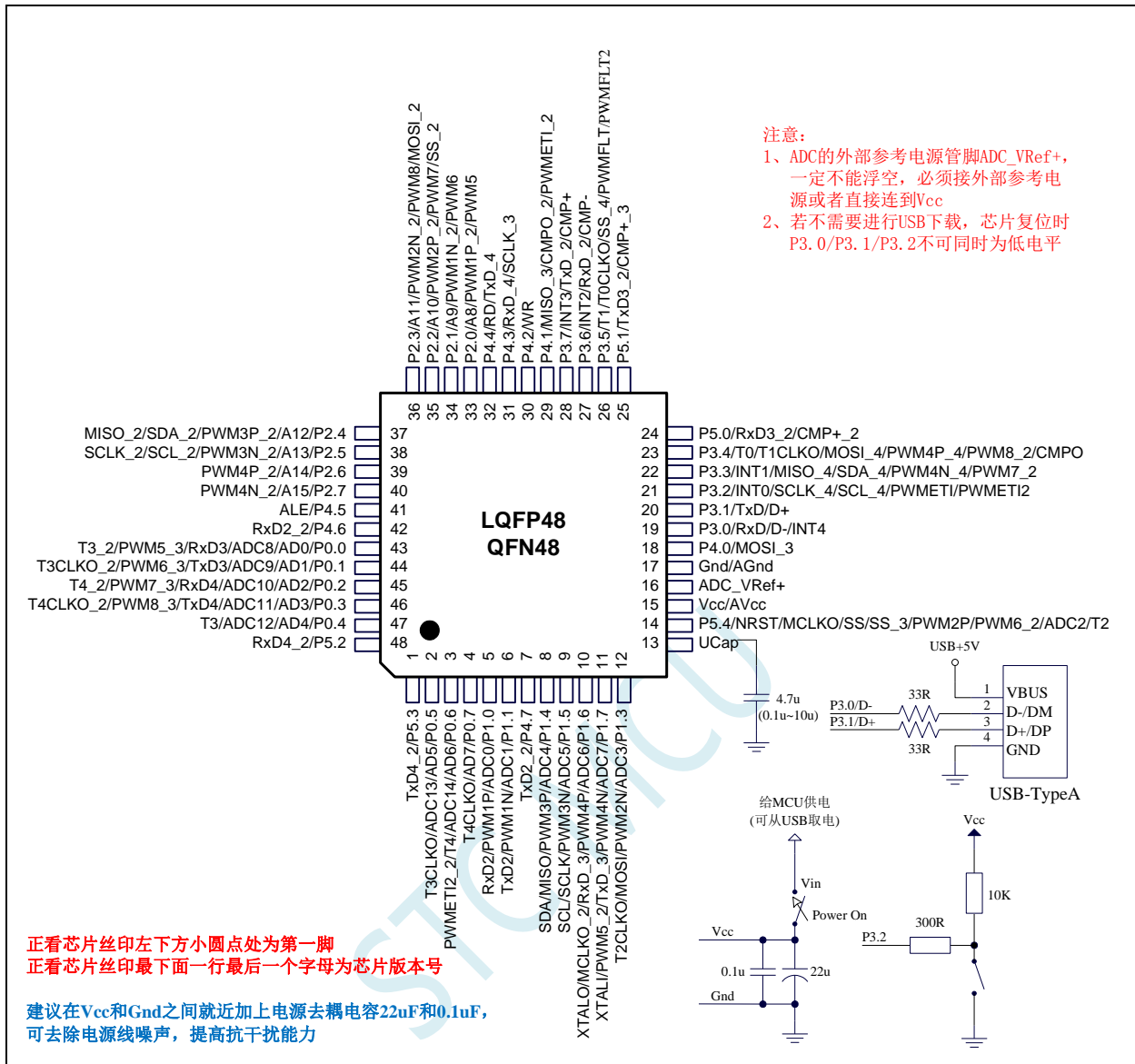
等待 STC-ISP 下载软件中自动识别出“STC USB Writer (HID1)”, 识别出来后, 就与 P3.2 状态无关了, 这时可以松开 P3.2 按键

===传统的机械自锁紧开关是按上来停电, 按下去是上电

3、点击下载软件中的“下载/编程”按钮 (注意: USB 下载与串口下载的操作顺序不同)

下载成功!

===另外从用户区软复位到系统区也是等待 USB 下载。



现在 STC 的带硬件 USB 的 MCU 支持用硬件 USB 下载, 因为用的是 USB-HID 通信协议, 不需要安装任何驱动。STC 打狗棒、降龙棍、开天斧、屠龙刀核心板以及 STC 开源示波器、STC 实验箱在 D-/P3.0, D+/P3.1 与 PC-USB 端口连接好的状况下, USB-ISP 下载程序步骤:

1、按下板子上的 P3.2/INT0 按键, 就是 P3.2 接地

2、给目标芯片重新上电, 不管之前是否已通电。

===电子开关是按下停电后再松开就是上电

等待 STC-ISP 下载软件中自动识别出“STC USB Writer (HID1)”, 识别出来后, 就与 P3.2 状态无关了, 这时可以松开 P3.2 按键

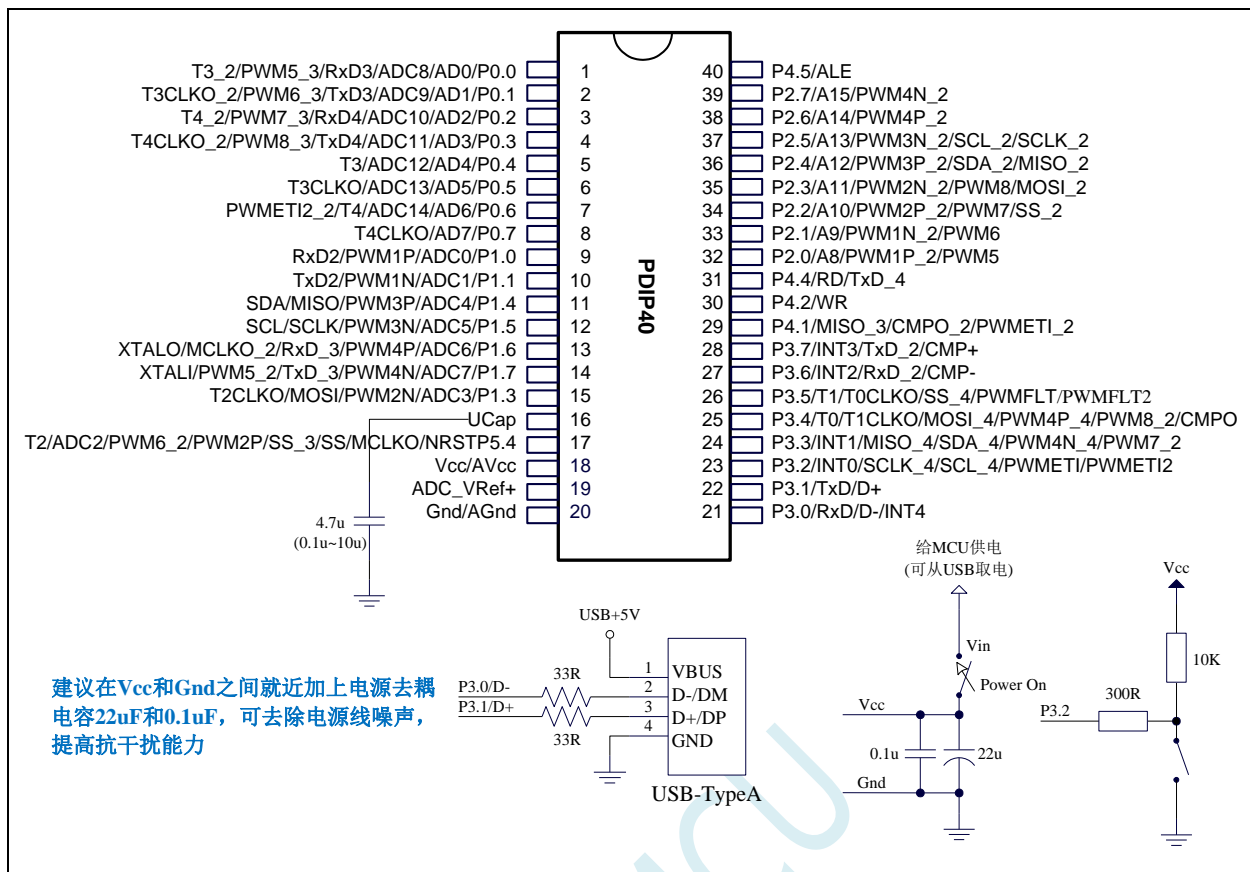
===传统的机械自锁紧开关是按上来停电, 按下去是上电

3、点击下载软件中的“下载/编程”按钮 (注意: USB 下载与串口下载的操作顺序不同)

下载成功!

===另外从用户区软复位到系统区也是等待 USB 下载。





现在 STC 的带硬件 USB 的 MCU 支持用硬件 USB 下载, 因为用的是 USB-HID 通信协议, 不需要安装任何驱动。STC 打狗棒、降龙棍、开天斧、屠龙刀核心板以及 STC 开源示波器、STC 实验箱在 D-/P3.0, D+/P3.1 与 PC-USB 端口连接好的状况下, USB-ISP 下载程序步骤:

1、按下板子上的 P3.2/INT0 按键, 就是 P3.2 接地

2、给目标芯片重新上电, 不管之前是否已通电。

===电子开关是按下停电后再松开就是上电

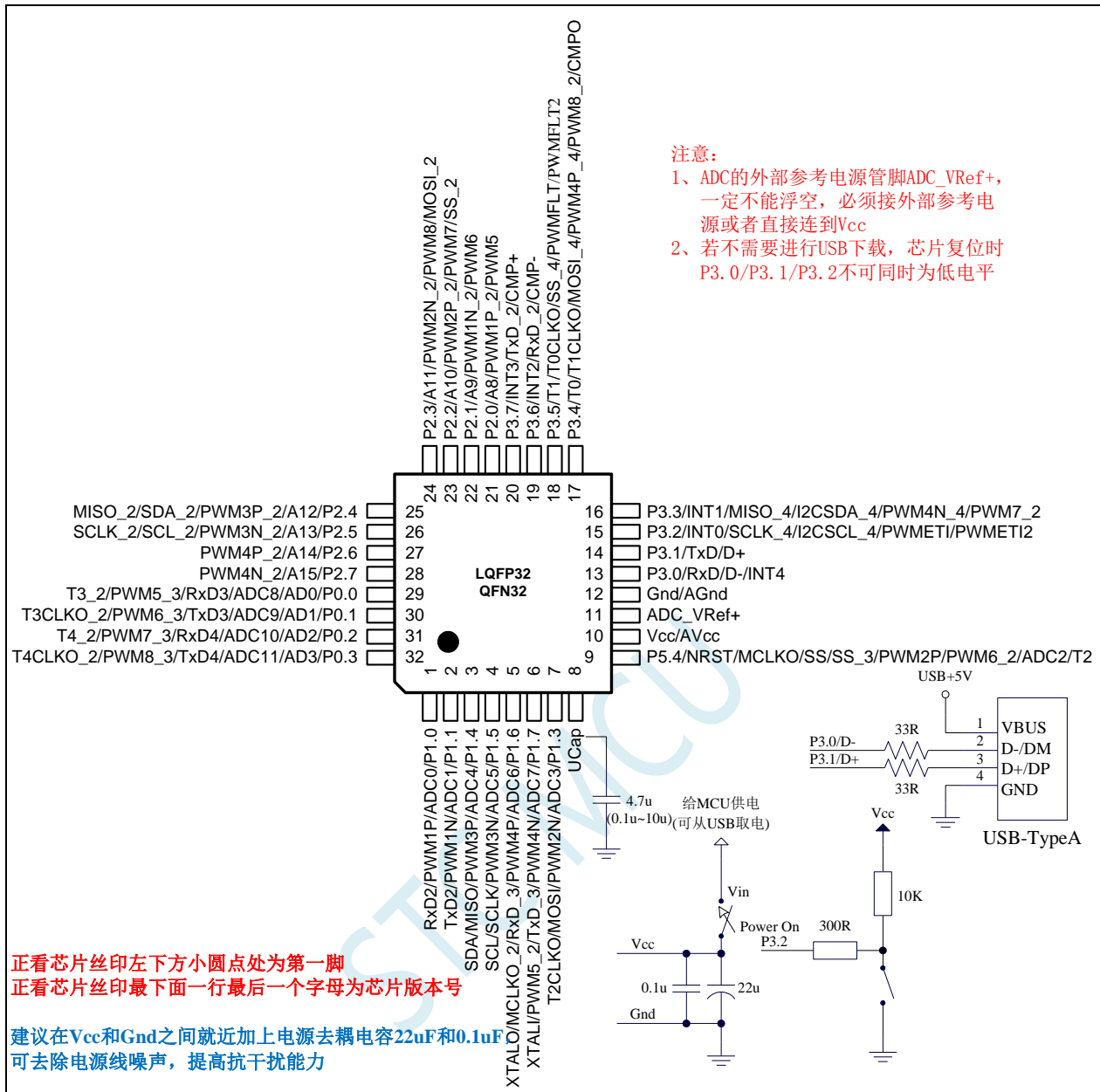
等待 STC-ISP 下载软件中自动识别出“STC USB Writer (HID1)”, 识别出来后, 就与 P3.2 状态无关了, 这时可以松开 P3.2 按键

===传统的机械自锁紧开关是按上来停电, 按下去是上电

3、点击下载软件中的“下载/编程”按钮(注意: USB 下载与串口下载的操作顺序不同)

下载成功!

===另外从用户区软复位到系统区也是等待 USB 下载。



现在 STC 的带硬件 USB 的 MCU 支持用硬件 USB 下载, 因为用的是 USB-HID 通信协议, 不需要安装任何驱动。STC 打狗棒、降龙棍、开天斧、屠龙刀核心板以及 STC 开源示波器、STC 实验箱在 D-/P3.0, D+/P3.1 与 PC-USB 端口连接好的状况下, USB-ISP 下载程序步骤:

- 1、按下板子上的 P3.2/INT0 按键, 就是 P3.2 接地
- 2、给目标芯片重新上电, 不管之前是否已通电。

===电子开关是按下停电后再松开就是上电

等待 STC-ISP 下载软件中自动识别出“STC USB Writer (HID1)”, 识别出来后, 就与 P3.2 状态无关了, 这时可以松开 P3.2 按键

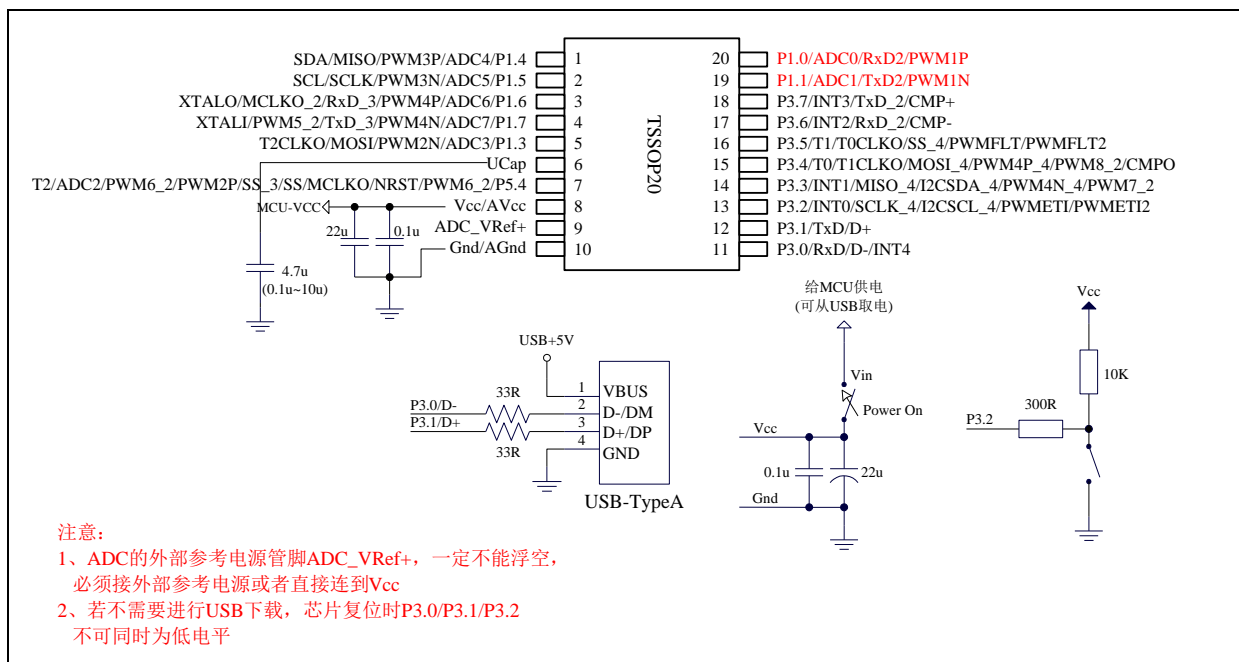
===传统的机械自锁紧开关是按上来停电, 按下去是上电

- 3、点击下载软件中的“下载/编程”按钮 (注意: USB 下载与串口下载的操作顺序不同)

下载成功!

===另外从用户区软复位到系统区也是等待 USB 下载。





建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF，可去除电源线噪声，提高抗干扰能力

现在 STC 的带硬件 USB 的 MCU 支持用硬件 USB 下载，因为用的是 USB-HID 通信协议，不需要安装任何驱动。STC 打狗棒、降龙棍、开天斧、屠龙刀核心板以及 STC 开源示波器、STC 实验箱在 D-/P3.0, D+/P3.1 与 PC-USB 端口连接好的状况下，USB-ISP 下载程序步骤：

- 1、按下板子上的 P3.2/INT0 按键，就是 P3.2 接地
- 2、给目标芯片重新上电，不管之前是否已通电。  
===电子开关是按下停电后再松开就是上电  
等待 STC-ISP 下载软件中自动识别出“STC USB Writer (HID1)”，识别出来后，就与 P3.2 状态无关了，这时可以松开 P3.2 按键  
===传统的机械自锁紧开关是按上来停电，按下去是上电
- 3、点击下载软件中的“下载/编程”按钮（注意：USB 下载与串口下载的操作顺序不同）  
下载成功！  
===另外从用户区软复位到系统区也是等待 USB 下载。

## 2.5.3 管脚说明

| 编号           |              | 名称        | 类型  | 说明                |
|--------------|--------------|-----------|-----|-------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |           |     |                   |
| 1            | 1            | P5.3      | I/O | 标准 IO 口           |
|              |              | TxD4_2    | O   | 串口 4 的发送脚         |
| 2            | 2            | P0.5      | I/O | 标准 IO 口           |
|              |              | AD5       | I/O | 地址/数据总线           |
|              |              | ADC13     | I   | ADC 模拟输入通道 13     |
|              |              | T3CLKO    | O   | 定时器 3 时钟分频输出      |
| 3            | 3            | P0.6      | I/O | 标准 IO 口           |
|              |              | AD6       | I/O | 地址/数据总线           |
|              |              | ADC14     | I   | ADC 模拟输入通道 14     |
|              |              | T4        | I   | 定时器 4 外部时钟输入      |
|              |              | PWMETI2_2 | I   | PWM 外部触发输入脚 2     |
| 4            | 4            | P0.7      | I/O | 标准 IO 口           |
|              |              | AD7       | I/O | 地址/数据总线           |
|              |              | T4CLKO    | O   | 定时器 4 时钟分频输出      |
| 5            |              | P6.0      | I/O | 标准 IO 口           |
|              |              | PWM1P_3   | I/O | PWM1 的捕获输入和脉冲输出正极 |
| 6            |              | P6.1      | I/O | 标准 IO 口           |
|              |              | PWM1N_3   | I/O | PWM1 的捕获输入和脉冲输出负极 |
| 7            |              | P6.2      | I/O | 标准 IO 口           |
|              |              | PWM2P_3   | I/O | PWM2 的捕获输入和脉冲输出正极 |
| 8            |              | P6.3      | I/O | 标准 IO 口           |
|              |              | PWM2N_3   | I/O | PWM2 的捕获输入和脉冲输出负极 |
| 9            | 5            | P1.0      | I/O | 标准 IO 口           |
|              |              | ADC0      | I   | ADC 模拟输入通道 0      |
|              |              | PWM1P     | I/O | PWM1 的捕获输入和脉冲输出正极 |
|              |              | RxD2      | I   | 串口 2 的接收脚         |
| 10           | 6            | P1.1      | I/O | 标准 IO 口           |
|              |              | ADC1      | I   | ADC 模拟输入通道 1      |
|              |              | PWM1N     | I/O | PWM1 的捕获输入和脉冲输出负极 |
|              |              | TxD2      | I   | 串口 2 的发送脚         |
| 11           | 7            | P4.7      | I/O | 标准 IO 口           |
|              |              | TxD2_2    | I   | 串口 2 的发送脚         |
| 12           | 8            | P1.4      | I/O | 标准 IO 口           |
|              |              | ADC4      | I   | ADC 模拟输入通道 4      |
|              |              | PWM3P     | I/O | PWM3 的捕获输入和脉冲输出正极 |
|              |              | MISO      | I/O | SPI 主机输入从机输出      |
|              |              | SDA       | I/O | I2C 接口的数据线        |

| 编号           |              | 名称        | 类型  | 说明                                 |
|--------------|--------------|-----------|-----|------------------------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |           |     |                                    |
| 13           | 9            | P1.5      | I/O | 标准 IO 口                            |
|              |              | ADC5      | I   | ADC 模拟输入通道 5                       |
|              |              | PWM3N     | I/O | PWM3 的捕获输入和脉冲输出负极                  |
|              |              | SCLK      | I/O | SPI 的时钟脚                           |
|              |              | SCL       | I/O | I2C 的时钟线                           |
| 14           | 10           | P1.6      | I/O | 标准 IO 口                            |
|              |              | ADC6      | I   | ADC 模拟输入通道 6                       |
|              |              | RxD_3     | I   | 串口 1 的接收脚                          |
|              |              | PWM4P     | I/O | PWM4 的捕获输入和脉冲输出正极                  |
|              |              | MCLKO_2   | O   | 主时钟分频输出                            |
|              |              | XTALO     | O   | 外部晶振的输出脚                           |
| 15           | 11           | P1.7      | I/O | 标准 IO 口                            |
|              |              | ADC7      | I   | ADC 模拟输入通道 7                       |
|              |              | TxD_3     | O   | 串口 1 的发送脚                          |
|              |              | PWM4N     | I/O | PWM4 的捕获输入和脉冲输出负极                  |
|              |              | PWM5_2    | I/O | PWM5 的捕获输入和脉冲输出                    |
|              |              | XTALI     | I   | 外部晶振/外部时钟的输入脚                      |
| 16           | 12           | P1.3      | I/O | 标准 IO 口                            |
|              |              | ADC3      | I   | ADC 模拟输入通道 3                       |
|              |              | MOSI      | I/O | SPI 主机输出从机输入                       |
|              |              | PWM2N     | I/O | PWM2 的捕获输入和脉冲输出负极                  |
|              |              | T2CLKO    | O   | 定时器 2 时钟分频输出                       |
| 17           | 13           | UCAP      | I   | USB 内核电源稳压脚                        |
| 18           | 14           | P5.4      | I/O | 标准 IO 口                            |
|              |              | NRST      | I   | 复位引脚（低电平复位）                        |
|              |              | MCLKO     | O   | 主时钟分频输出                            |
|              |              | SS_3      | I   | SPI 的从机选择脚（主机为输出）                  |
|              |              | SS        | I   | SPI 的从机选择脚（主机为输出）                  |
|              |              | PWM2P     | I/O | PWM2 的捕获输入和脉冲输出正极                  |
|              |              | PWM6_2    | I/O | PWM6 的捕获输入和脉冲输出                    |
|              |              | T2        | I   | 定时器 2 外部时钟输入                       |
|              |              | ADC2      | I   | ADC 模拟输入通道 2                       |
| 19           | 15           | Vcc       | Vcc | 电源脚                                |
|              |              | AVcc      | Vcc | ADC 电源脚                            |
| 20           | 16           | ADC_VRef+ | I   | ADC 外部参考电压源输入脚，要求不高时可直接接 MCU 的 VCC |
| 21           | 17           | Gnd       | Gnd | 地线                                 |
|              |              | AGnd      | Gnd | ADC 地线                             |
| 22           | 18           | P4.0      | I/O | 标准 IO 口                            |
|              |              | MOSI_3    | I/O | SPI 主机输出从机输入                       |

| 编号           |              | 名称      | 类型  | 说明                |
|--------------|--------------|---------|-----|-------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |         |     |                   |
| 23           |              | P6.4    | I/O | 标准 IO 口           |
|              |              | PWM3P_3 | I/O | PWM3 的捕获输入和脉冲输出正极 |
| 24           |              | P6.5    | I/O | 标准 IO 口           |
|              |              | PWM3N_3 | I/O | PWM3 的捕获输入和脉冲输出负极 |
| 25           |              | P6.6    | I/O | 标准 IO 口           |
|              |              | PWM4P_3 | I/O | PWM4 的捕获输入和脉冲输出正极 |
| 26           |              | P6.7    | I/O | 标准 IO 口           |
|              |              | PWM4N_3 | I/O | PWM4 的捕获输入和脉冲输出负极 |
| 27           | 19           | P3.0    | I/O | 标准 IO 口           |
|              |              | D-      | I/O | USB 数据口           |
|              |              | RxD     | I   | 串口 1 的接收脚         |
|              |              | INT4    | I   | 外部中断 4            |
| 28           | 20           | P3.1    | I/O | 标准 IO 口           |
|              |              | D+      | I/O | USB 数据口           |
|              |              | TxD     | O   | 串口 1 的发送脚         |
| 29           | 21           | P3.2    | I/O | 标准 IO 口           |
|              |              | INT0    | I   | 外部中断 0            |
|              |              | SCLK_4  | I/O | SPI 的时钟脚          |
|              |              | SCL_4   | I/O | I2C 的时钟线          |
|              |              | PWMETI  | I   | PWM 外部触发输入脚       |
|              |              | PWMETI2 | I   | PWM 外部触发输入脚 2     |
| 30           | 22           | P3.3    | I/O | 标准 IO 口           |
|              |              | INT1    | I   | 外部中断 1            |
|              |              | MISO_4  | I/O | SPI 主机输入从机输出      |
|              |              | SDA_4   | I/O | I2C 接口的数据线        |
|              |              | PWM4N_4 | I/O | PWM4 的捕获输入和脉冲输出负极 |
|              |              | PWM7_2  | I/O | PWM7 的捕获输入和脉冲输出   |
| 31           | 23           | P3.4    | I/O | 标准 IO 口           |
|              |              | T0      | I   | 定时器 0 外部时钟输入      |
|              |              | T1CLKO  | O   | 定时器 1 时钟分频输出      |
|              |              | MOSI_4  | I/O | SPI 主机输出从机输入      |
|              |              | PWM4P_4 | I/O | PWM4 的捕获输入和脉冲输出正极 |
|              |              | PWM8_2  | I/O | PWM8 的捕获输入和脉冲输出   |
|              |              | CMPO    | O   | 比较器输出             |
| 32           | 24           | P5.0    | I/O | 标准 IO 口           |
|              |              | RxD3_2  | I   | 串口 3 的接收脚         |
|              |              | CMP+_2  | I   | 比较器正极输入           |
| 33           | 25           | P5.1    | I/O | 标准 IO 口           |
|              |              | TxD3_2  | O   | 串口 3 的发送脚         |
|              |              | CMP+_3  | I   | 比较器正极输入           |

| 编号           |              | 名称       | 类型  | 说明                |
|--------------|--------------|----------|-----|-------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |          |     |                   |
| 34           | 26           | P3.5     | I/O | 标准 IO 口           |
|              |              | T1       | I   | 定时器 1 外部时钟输入      |
|              |              | T0CLKO   | O   | 定时器 0 时钟分频输出      |
|              |              | SS_4     | I   | SPI 的从机选择脚（主机为输出） |
|              |              | PWMFLT   | I   | 增强 PWMA 的外部异常检测脚  |
|              |              | PWMFLT2  | I   | 增强 PWMB 的外部异常检测脚  |
| 35           | 27           | P3.6     | I/O | 标准 IO 口           |
|              |              | INT2     | I   | 外部中断 2            |
|              |              | RxD_2    | I   | 串口 1 的接收脚         |
|              |              | CMP-     | I   | 比较器负极输入           |
| 36           | 28           | P3.7     | I/O | 标准 IO 口           |
|              |              | INT3     | I   | 外部中断 3            |
|              |              | TxD_2    | O   | 串口 1 的发送脚         |
|              |              | CMP+     | I   | 比较器正极输入           |
| 37           |              | P7.0     | I/O | 标准 IO 口           |
| 38           |              | P7.1     | I/O | 标准 IO 口           |
| 39           |              | P7.2     | I/O | 标准 IO 口           |
| 40           |              | P7.3     | I/O | 标准 IO 口           |
|              |              | PWMETI_3 | I   | PWM 外部触发输入脚       |
| 41           | 29           | P4.1     | I/O | 标准 IO 口           |
|              |              | MISO_3   | I/O | SPI 主机输入从机输出      |
|              |              | CMPO_2   | O   | 比较器输出             |
|              |              | PWMETI_3 | I   | PWM 外部触发输入脚       |
| 42           | 30           | P4.2     | I/O | 标准 IO 口           |
|              |              | WR       | O   | 外部总线的写信号线         |
| 43           | 31           | P4.3     | I/O | 标准 IO 口           |
|              |              | RxD_4    | I   | 串口 1 的接收脚         |
|              |              | SCLK_3   | I/O | SPI 的时钟脚          |
| 44           | 32           | P4.4     | I/O | 标准 IO 口           |
|              |              | RD       | O   | 外部总线的读信号线         |
|              |              | TxD_4    | O   | 串口 1 的发送脚         |
| 45           | 33           | P2.0     | I/O | 标准 IO 口           |
|              |              | A8       | O   | 地址总线              |
|              |              | PWM1P_2  | I/O | PWM1 的捕获输入和脉冲输出正极 |
|              |              | PWM5     | I/O | PWM5 的捕获输入和脉冲输出   |

| 编号           |              | 名称      | 类型  | 说明                |
|--------------|--------------|---------|-----|-------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |         |     |                   |
| 46           | 34           | P2.1    | I/O | 标准 IO 口           |
|              |              | A9      | O   | 地址总线              |
|              |              | PWM1N_2 | I/O | PWM1 的捕获输入和脉冲输出负极 |
|              |              | PWM6    | I/O | PWM6 的捕获输入和脉冲输出   |
| 47           | 35           | P2.2    | I/O | 标准 IO 口           |
|              |              | A10     | O   | 地址总线              |
|              |              | SS_2    | I   | SPI 的从机选择脚（主机为输出） |
|              |              | PWM2P_2 | I/O | PWM2 的捕获输入和脉冲输出正极 |
|              |              | PWM7    | I/O | PWM7 的捕获输入和脉冲输出   |
| 48           | 36           | P2.3    | I/O | 标准 IO 口           |
|              |              | A11     | O   | 地址总线              |
|              |              | MOSI_2  | I/O | SPI 主机输出从机输入      |
|              |              | PWM2N_2 | I/O | PWM2 的捕获输入和脉冲输出负极 |
|              |              | PWM8    | I/O | PWM8 的捕获输入和脉冲输出   |
| 49           | 37           | P2.4    | I/O | 标准 IO 口           |
|              |              | A12     | O   | 地址总线              |
|              |              | MISO_2  | I/O | SPI 主机输入从机输出      |
|              |              | SDA_2   | I/O | I2C 接口的数据线        |
|              |              | PWM3P_2 | I/O | PWM3 的捕获输入和脉冲输出正极 |
| 50           | 38           | P2.5    | I/O | 标准 IO 口           |
|              |              | A13     | O   | 地址总线              |
|              |              | SCLK_2  | I/O | SPI 的时钟脚          |
|              |              | SCL_2   | I/O | I2C 的时钟线          |
|              |              | PWM3N_2 | I/O | PWM3 的捕获输入和脉冲输出负极 |
| 51           | 39           | P2.6    | I/O | 标准 IO 口           |
|              |              | A14     | O   | 地址总线              |
|              |              | PWM4P_2 | I/O | PWM4 的捕获输入和脉冲输出正极 |
| 52           | 40           | P2.7    | I/O | 标准 IO 口           |
|              |              | A15     | O   | 地址总线              |
|              |              | PWM4N_2 | I/O | PWM4 的捕获输入和脉冲输出负极 |
| 53           |              | P7.4    | I/O | 标准 IO 口           |
|              |              | PWM5_4  | I/O | PWM5 的捕获输入和脉冲输出   |
| 54           |              | P7.5    | I/O | 标准 IO 口           |
|              |              | PWM6_4  | I/O | PWM6 的捕获输入和脉冲输出   |
| 55           |              | P7.6    | I/O | 标准 IO 口           |
|              |              | PWM7_4  | I/O | PWM7 的捕获输入和脉冲输出   |
|              |              | SDA_3   | I/O | I2C 接口的数据线        |

| 编号           |              | 名称       | 类型  | 说明              |
|--------------|--------------|----------|-----|-----------------|
| LQFP64/QFN64 | LQFP48/QFN48 |          |     |                 |
| 56           |              | P7.7     | I/O | 标准 IO 口         |
|              |              | PWM8_4   | I/O | PWM8 的捕获输入和脉冲输出 |
|              |              | SCL_3    | I/O | I2C 的时钟线        |
| 57           | 41           | P4.5     | I/O | 标准 IO 口         |
|              |              | ALE      | O   | 地址锁存信号          |
| 58           | 42           | P4.6     | I/O | 标准 IO 口         |
|              |              | RxD2_2   | I   | 串口 2 的接收脚       |
| 59           | 43           | P0.0     | I/O | 标准 IO 口         |
|              |              | AD0      | I/O | 地址/数据总线         |
|              |              | ADC8     | I   | ADC 模拟输入通道 8    |
|              |              | RxD3     | I   | 串口 3 的接收脚       |
|              |              | PWM5_3   | I/O | PWM5 的捕获输入和脉冲输出 |
|              |              | T3_2     | I   | 定时器 3 外部时钟输入    |
| 60           | 44           | P0.1     | I/O | 标准 IO 口         |
|              |              | AD1      | I/O | 地址/数据总线         |
|              |              | ADC9     | I   | ADC 模拟输入通道 9    |
|              |              | TxD3     | O   | 串口 3 的发送脚       |
|              |              | PWM6_3   | I/O | PWM6 的捕获输入和脉冲输出 |
|              |              | T3CLKO_2 | O   | 定时器 3 时钟分频输出    |
| 61           | 45           | P0.2     | I/O | 标准 IO 口         |
|              |              | AD2      | I/O | 地址/数据总线         |
|              |              | ADC10    | I   | ADC 模拟输入通道 10   |
|              |              | RxD4     | I   | 串口 4 的接收脚       |
|              |              | PWM7_3   | I/O | PWM7 的捕获输入和脉冲输出 |
|              |              | T4_2     | I   | 定时器 4 外部时钟输入    |
| 62           | 46           | P0.3     | I/O | 标准 IO 口         |
|              |              | AD3      | I/O | 地址/数据总线         |
|              |              | ADC11    | I   | ADC 模拟输入通道 11   |
|              |              | TxD4     | O   | 串口 4 的发送脚       |
|              |              | PWM8_3   | I/O | PWM8 的捕获输入和脉冲输出 |
|              |              | T4CLKO_2 | O   | 定时器 4 时钟分频输出    |
| 63           | 47           | P0.4     | I/O | 标准 IO 口         |
|              |              | AD4      | I/O | 地址/数据总线         |
|              |              | ADC12    | I   | ADC 模拟输入通道 12   |
|              |              | T3       | I   | 定时器 3 外部时钟输入    |
| 64           | 48           | P5.2     | I/O | 标准 IO 口         |
|              |              | RxD4_2   | I   | 串口 4 的接收脚       |

## 2.6 STC8H4K64TL-45I-LQFP48/32、QFN48、TSSOP20

### 2.6.1 特性及价格(有 16 位硬件乘除法器 MDU16, 准 16 位单片机)

➤ 选型价格(不需要外部晶振、不需要外部复位, 12 位 ADC, 15 通道)

| 价格及封装        | TSSOP20     |  | LQFP32 |  | QFN48 <6mm*6mm> |  | LQFP48 <9mm*9mm> |  | 供货信息              |  |  |  |  |  |  |  |  |                         |  |  |  |  |  |  |  |  |                   |  |  |  |  |  |  |  |  |           |  |  |  |  |  |  |  |  |                |  |  |  |  |  |  |  |  |                           |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |           |  |  |  |  |  |  |  |  |                         |  |  |  |  |  |  |  |  |                                     |  |  |  |  |  |  |  |  |                   |  |  |  |  |  |  |  |  |                                 |  |  |  |  |  |  |  |  |                |  |  |  |  |  |  |  |  |          |  |  |  |  |  |  |  |  |      |  |  |  |  |  |  |  |  |        |  |  |  |  |  |  |  |  |                                      |  |  |  |  |  |  |  |  |              |  |  |  |  |  |  |  |  |                      |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |           |  |  |  |  |  |  |  |  |                 |  |  |  |  |  |  |  |  |                |  |  |  |  |  |  |  |  |                                    |  |  |  |  |  |  |  |  |                                     |  |  |  |  |  |  |  |  |                      |  |  |  |  |  |  |  |  |          |  |  |  |  |  |  |  |  |       |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|--------------|-------------|--|--------|--|-----------------|--|------------------|--|-------------------|--|--|--|--|--|--|--|--|-------------------------|--|--|--|--|--|--|--|--|-------------------|--|--|--|--|--|--|--|--|-----------|--|--|--|--|--|--|--|--|----------------|--|--|--|--|--|--|--|--|---------------------------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|-----------|--|--|--|--|--|--|--|--|-------------------------|--|--|--|--|--|--|--|--|-------------------------------------|--|--|--|--|--|--|--|--|-------------------|--|--|--|--|--|--|--|--|---------------------------------|--|--|--|--|--|--|--|--|----------------|--|--|--|--|--|--|--|--|----------|--|--|--|--|--|--|--|--|------|--|--|--|--|--|--|--|--|--------|--|--|--|--|--|--|--|--|--------------------------------------|--|--|--|--|--|--|--|--|--------------|--|--|--|--|--|--|--|--|----------------------|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|-----------|--|--|--|--|--|--|--|--|-----------------|--|--|--|--|--|--|--|--|----------------|--|--|--|--|--|--|--|--|------------------------------------|--|--|--|--|--|--|--|--|-------------------------------------|--|--|--|--|--|--|--|--|----------------------|--|--|--|--|--|--|--|--|----------|--|--|--|--|--|--|--|--|-------|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
|              | 本身就可在线仿真    |  |        |  |                 |  |                  |  | 支持软件模拟硬件 USB 直接下载 |  |  |  |  |  |  |  |  |                         |  |  |  |  |  |  |  |  |                   |  |  |  |  |  |  |  |  |           |  |  |  |  |  |  |  |  |                |  |  |  |  |  |  |  |  |                           |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |           |  |  |  |  |  |  |  |  |                         |  |  |  |  |  |  |  |  |                                     |  |  |  |  |  |  |  |  |                   |  |  |  |  |  |  |  |  |                                 |  |  |  |  |  |  |  |  |                |  |  |  |  |  |  |  |  |          |  |  |  |  |  |  |  |  |      |  |  |  |  |  |  |  |  |        |  |  |  |  |  |  |  |  |                                      |  |  |  |  |  |  |  |  |              |  |  |  |  |  |  |  |  |                      |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |           |  |  |  |  |  |  |  |  |                 |  |  |  |  |  |  |  |  |                |  |  |  |  |  |  |  |  |                                    |  |  |  |  |  |  |  |  |                                     |  |  |  |  |  |  |  |  |                      |  |  |  |  |  |  |  |  |          |  |  |  |  |  |  |  |  |       |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|              | 支持 RS485 下载 |  |        |  |                 |  |                  |  | 可设置下次更新程序需口令      |  |  |  |  |  |  |  |  |                         |  |  |  |  |  |  |  |  |                   |  |  |  |  |  |  |  |  |           |  |  |  |  |  |  |  |  |                |  |  |  |  |  |  |  |  |                           |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |           |  |  |  |  |  |  |  |  |                         |  |  |  |  |  |  |  |  |                                     |  |  |  |  |  |  |  |  |                   |  |  |  |  |  |  |  |  |                                 |  |  |  |  |  |  |  |  |                |  |  |  |  |  |  |  |  |          |  |  |  |  |  |  |  |  |      |  |  |  |  |  |  |  |  |        |  |  |  |  |  |  |  |  |                                      |  |  |  |  |  |  |  |  |              |  |  |  |  |  |  |  |  |                      |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |           |  |  |  |  |  |  |  |  |                 |  |  |  |  |  |  |  |  |                |  |  |  |  |  |  |  |  |                                    |  |  |  |  |  |  |  |  |                                     |  |  |  |  |  |  |  |  |                      |  |  |  |  |  |  |  |  |          |  |  |  |  |  |  |  |  |       |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 程序加密后传输(防拦截) |             |  |        |  |                 |  |                  |  | 可对外输出时钟及复位        |  |  |  |  |  |  |  |  | 内部高精度时钟 (45MHz 以下可调) 追频 |  |  |  |  |  |  |  |  | 内部高可靠复位(可选复位门檻电压) |  |  |  |  |  |  |  |  | 看门狗 复位定时器 |  |  |  |  |  |  |  |  | 内部低压检测中断并可掉电唤醒 |  |  |  |  |  |  |  |  | 比较器(可当 1 路 A/D, 可作外部掉电检测) |  |  |  |  |  |  |  |  | DMA 15 路高速 ADC (8 路 PWM 可当 8 路 D/A 使用) |  |  |  |  |  |  |  |  | 掉电唤醒专用定时器 |  |  |  |  |  |  |  |  | 16 位高级 PWM 定时器 互补对称死区控制 |  |  |  |  |  |  |  |  | 定时器/计数器 (T0/T1/T2/T3/T4 外部管脚也可掉电唤醒) |  |  |  |  |  |  |  |  | MDU16 硬件 16 位乘除法器 |  |  |  |  |  |  |  |  | I <sup>2</sup> C 并可掉电唤醒(暂无 DMA) |  |  |  |  |  |  |  |  | DMA SPI 并可掉电唤醒 |  |  |  |  |  |  |  |  | RTC 实时时钟 |  |  |  |  |  |  |  |  | 触摸按键 |  |  |  |  |  |  |  |  | LED 驱动 |  |  |  |  |  |  |  |  | DMA 8080/6800 接口 LCM 模块驱动(8 位和 16 位) |  |  |  |  |  |  |  |  | DMA 串口并可掉电唤醒 |  |  |  |  |  |  |  |  | 所有的 I/O 口均支持中断并可掉电唤醒 |  |  |  |  |  |  |  |  | 传统 I/O 中断 (INT0/INT1/INT2/INT3/INT4) 并可掉电唤醒 |  |  |  |  |  |  |  |  | I/O 口最多数量 |  |  |  |  |  |  |  |  | EEPROM 10 万次 字节 |  |  |  |  |  |  |  |  | 强大的双 DPTX 可增可减 |  |  |  |  |  |  |  |  | xdata <sup>*</sup> 内部大容量扩展 SRAM 字节 |  |  |  |  |  |  |  |  | idata <sup>*</sup> 内部传统 8051 RAM 字节 |  |  |  |  |  |  |  |  | Flash 程序存储器 10 万次 字节 |  |  |  |  |  |  |  |  | 工作电压 (V) |  |  |  |  |  |  |  |  | 单片机型号 |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| STC8H4K32TL  |             |  |        |  |                 |  |                  |  | 1.9-5.5           |  |  |  |  |  |  |  |  | 32K                     |  |  |  |  |  |  |  |  | 256               |  |  |  |  |  |  |  |  | 4K        |  |  |  |  |  |  |  |  | 2              |  |  |  |  |  |  |  |  | 32K                       |  |  |  |  |  |  |  |  | 44                                     |  |  |  |  |  |  |  |  | 有         |  |  |  |  |  |  |  |  | 有                       |  |  |  |  |  |  |  |  | 4                                   |  |  |  |  |  |  |  |  | 有                 |  |  |  |  |  |  |  |  | 有                               |  |  |  |  |  |  |  |  | 有              |  |  |  |  |  |  |  |  | 有        |  |  |  |  |  |  |  |  | 有    |  |  |  |  |  |  |  |  | 有      |  |  |  |  |  |  |  |  | 有                                    |  |  |  |  |  |  |  |  | 有            |  |  |  |  |  |  |  |  | 有                    |  |  |  |  |  |  |  |  | 有   |  |  |  |  |  |  |  |  | 有         |  |  |  |  |  |  |  |  | 有               |  |  |  |  |  |  |  |  | 有              |  |  |  |  |  |  |  |  | 有                                  |  |  |  |  |  |  |  |  | 有                                   |  |  |  |  |  |  |  |  | 有                    |  |  |  |  |  |  |  |  | 有        |  |  |  |  |  |  |  |  | 有     |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  | 有 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

#### ➤ 内核

- ✓ 超高速 8051 内核 (1T), 比传统 8051 约快 12 倍以上
- ✓ 指令代码完全兼容传统 8051
- ✓ 41 个中断源, 4 级中断优先级
- ✓ 支持在线仿真

#### ➤ 工作电压

- ✓ 1.9V~5.5V

#### ➤ 工作温度

- ✓ -40℃~85℃ (芯片为-40℃~125℃制程, 超温度范围应用请参考电气特性章节说明)

#### ➤ Flash 存储器

- ✓ 最大 64K 字节 FLASH 程序存储器 (ROM), 用于存储用户代码
- ✓ 支持用户配置 EEPROM 大小, 512 字节单页擦除, 擦写次数可达 10 万次以上
- ✓ 支持在系统编程方式 (ISP) 更新用户应用程序, 无需专用编程器
- ✓ 支持单芯片仿真, 无需专用仿真器, 理论断点个数无限制

#### ➤ SRAM

- ✓ 128 字节内部直接访问 RAM (DATA, C 语言程序中使用 data 关键字进行声明)
- ✓ 128 字节内部间接访问 RAM (IDATA, C 语言程序中使用 idata 关键字进行声明)
- ✓ 4096 字节内部扩展 RAM (内部 XDATA, C 语言程序中使用 xdata 关键字进行声明)

#### ➤ 时钟控制



- ✓ 内部高精度 IRC (4MHz~45MHz, ISP 编程时选择或手动输入, 还可以用户软件分频到较低的频率工作, 如 100KHz)
  - ⊕ 误差±0.3% (常温下 25℃)
  - ⊕ -1.35%~+1.30%温漂 (全温度范围, -40℃~85℃)
  - ⊕ -0.76%~+0.98%温漂 (温度范围, -20℃~65℃)
- ✓ 内部 32KHz 低速 IRC (误差较大)
- ✓ 外部晶振 (4MHz~45MHz) 和外部时钟  
用户可自由选择上面的 3 种时钟源

## ➤ 复位

- ✓ 硬件复位
  - ⊕ 上电复位, 实测电压值为 1.69V~1.82V。(在芯片未使能低压复位功能时有效)  
上电复位电压由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到上电复位的下限门槛电压时, 芯片处于复位状态; 当电压从 0V 上升到上电复位的上限门槛电压时, 芯片解除复位状态。
  - ⊕ 复位脚复位, 出厂时 P5.4 默认为 I/O 口, ISP 下载时可将 P5.4 管脚设置为复位脚 (注意: 当设置 P5.4 管脚为复位脚时, 复位电平为低电平)
  - ⊕ 看门狗溢出复位
  - ⊕ 低压检测复位, 提供 4 级低压检测电压: 1.9V、2.3V、2.8V、3.7V。  
每级低压检测电压都是由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到低压检测的下限门槛电压时, 低压检测生效; 当电压从 0V 上升到低压检测的上限门槛电压时, 低压检测生效。
- ✓ 软件复位
  - ⊕ 软件方式写复位触发寄存器

## ➤ 中断

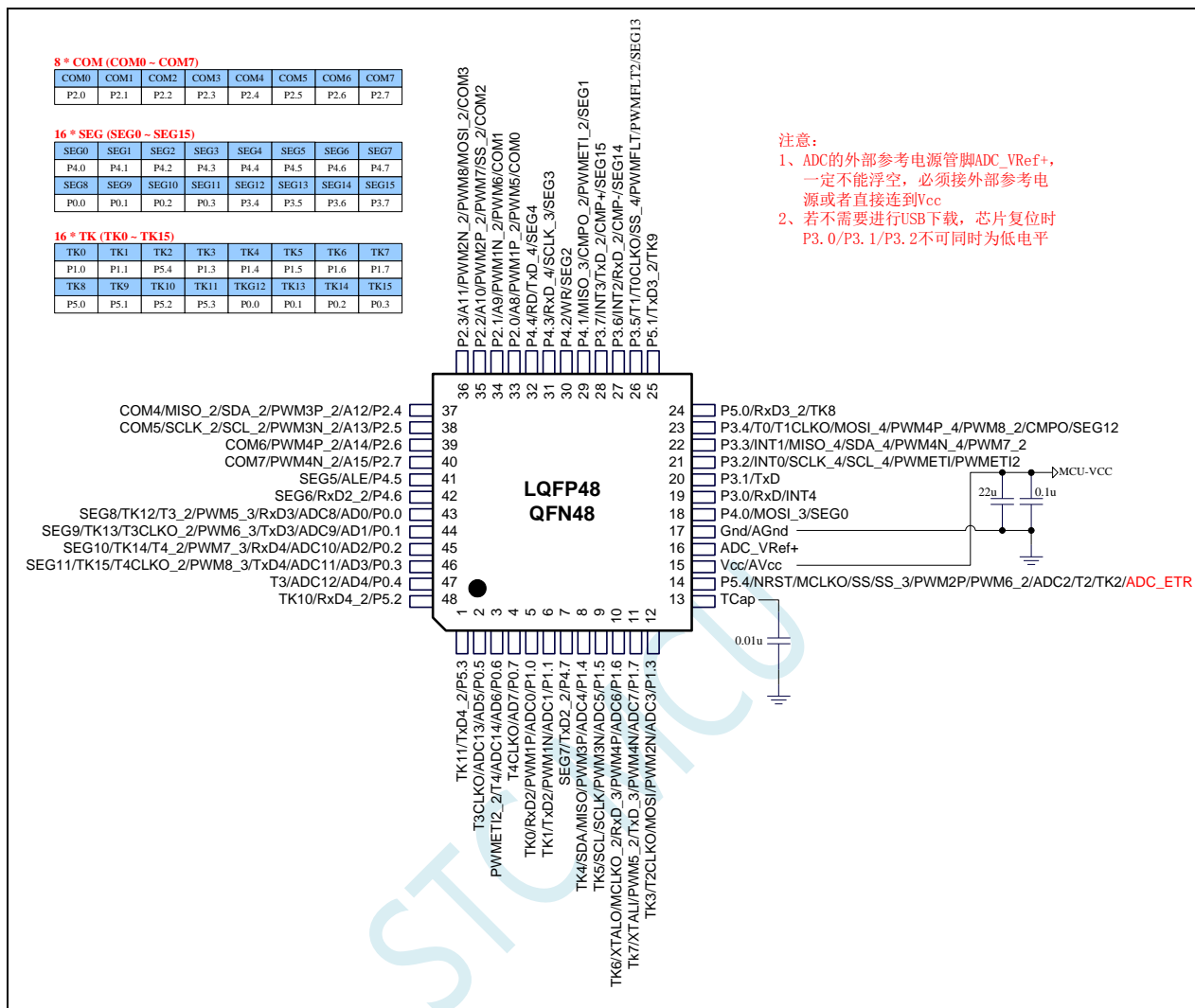
- ✓ 提供 41 个中断源: INT0 (支持上升沿和下降沿中断)、INT1 (支持上升沿和下降沿中断)、INT2 (只支持下降沿中断)、INT3 (只支持下降沿中断)、INT4 (只支持下降沿中断)、定时器 0、定时器 1、定时器 2、定时器 3、定时器 4、串口 1、串口 2、串口 3、串口 4、ADC 模数转换、LVD 低压检测、SPI、I<sup>2</sup>C、比较器、PWMA、PWMB、RTC、TKS、P0 口中断、P1 口中断、P2 口中断、P3 口中断、P4 口中断、P5 口中断、LCM 驱动中断、串口 1 的 DMA 接收和发送中断、串口 2 的 DMA 接收和发送中断、串口 3 的 DMA 接收和发送中断、串口 4 的 DMA 接收和发送中断、SPI 的 DMA 中断、ADC 的 DMA 中断、LCM 驱动的 DMA 中断以及存储器到存储器的 DMA 中断。
- ✓ 提供 4 级中断优先级
- ✓ 时钟停振模式下可以唤醒的中断: INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、T3(P0.4)、T4(P0.6)、RXD(P3.0/P3.6/P1.6/P4.3)、RXD2(P1.0/P4.6)、RXD3(P0.0/P5.0)、RXD4(P0.2/P5.2)、I2C\_SDA(P1.4/P2.4/P3.3)、SPI\_SS(P5.4/P2.2/P3.5)以及比较器中断、低压检测中断、掉电唤醒定时器唤醒以及所有端口的 I/O 中断。

## ➤ 数字外设

- ✓ 5 个 16 位定时器: 定时器 0、定时器 1、定时器 2、定时器 3、定时器 4, 其中定时器 0 的模式 3 具有 NMI (不可屏蔽中断) 功能, 定时器 0 和定时器 1 的模式 0 为 16 位自动重载模式
- ✓ 4 个高速串口: 串口 1、串口 2、串口 3、串口 4, 波特率时钟源最快可为 FOSC/4
- ✓ 8 路/2 组高级 PWM, 可实现带死区的控制信号, 并支持外部异常检测功能, 另外还支持 16 位定时器、8 个外部中断、8 路外部捕获测量脉宽等功能
- ✓ SPI: 支持主机模式和从机模式以及主机/从机自动切换
- ✓ I<sup>2</sup>C: 支持主机模式和从机模式

- ✓ MDU16: 硬件 16 位乘除法器 (支持 32 位除以 16 位、16 位除以 16 位、16 位乘 16 位、数据移位以及数据规格化等运算)
- ✓ RTC: 支持年、月、日、时、分、秒、次秒 (1/128 秒), 并支持时钟中断和一组闹钟
- ✓ I/O 口中断: 所有的 I/O 均支持中断, 每组 I/O 中断有独立的中断入口地址, 所有的 I/O 中断可支持 4 种中断模式: 高电平中断、低电平中断、上升沿中断、下降沿中断。提供 4 级中断优先级并支持掉电唤醒功能。
- ✓ DMA: 支持 Memory-To-Memory、SPI、UART1TX/UART1RX、UART2TX/UART2RX、UART3TX/UART3RX、UART4TX/UART4RX、ADC (自动计算多次 ADC 结果的平均值)、LCM
- ✓ LCM (TFT 彩屏) 驱动: 支持 8080 和 6800 接口, 支持 8 位和 16 位数据宽度
  - ⊕ 8 位 8080 总线: 8 为数据线 (TD0~TD7), 读信号 (TRD), 写信号 (TWR), RS 线 (TRS)
  - ⊕ 16 为 8080 总线: 16 为数据线 (TD0~TD15), 读信号 (TRD), 写信号 (TWR), RS 线 (TRS)
  - ⊕ 8 为 6800 总线: 8 为数据线 (TD0~TD7), 使能信号 (TE), 读写信号 (TRW), RS 线 (TRS)
  - ⊕ 16 位 6800 总线: 16 为数据线 (TD0~TD15), 使能信号 (TE), 读写信号 (TRW), RS 线 (TRS)
  - ⊕ 注意: 如果使用 8 位数据线控制 TFT 屏, 一般需要 TD0~D7, TRD/TWR/TRS, 11 根数据及控制线, 外加 2 个普通 I/O 控制片选及复位 (很多 TFT 彩屏的片选及复位厂商已进行了自动处理, 并不需要软件控制)
- 模拟外设
  - ✓ 超高速 ADC, 支持 12 位高精度 15 通道 (通道 0~通道 14) 的模数转换, 速度最快能达到 800K (每秒进行 80 万次 ADC 转换)
  - ✓ ADC 的通道 15 用于测试内部 1.19V 参考信号源 (芯片在出厂时, 内部参考信号源已调整为 1.19V)
  - ✓ 比较器, 一组比较器 (比较器的正端可选择 CMP+ 端口和所有的 ADC 输入端口, 所以比较器可当作多路比较器进行分时复用)
  - ✓ 触摸按键: 最多支持 16 个触摸按键; 每个触摸按键能够独立使能; 内部参考电压 4 级可调; 灵活的充放电时间设置以及内部工作频率设置; 支持低功耗触摸唤醒
  - ✓ LED 驱动: 最多可驱动 256 (8\*16\*2) 个 LED; 可支持共阴模式、共阳模式以及共阴/共阳模式; 支持 8 级灰度调节 (亮度调节)
  - ✓ DAC: 8 路高级 PWM 定时器可当 8 路 DAC 使用
- GPIO
  - ✓ 最多可达 44 个 GPIO: P0.0~P0.7、P1.0~P1.7 (无 P1.2)、P2.0~P2.7、P3.0~P3.7、P4.0~P4.7、P5.0~P5.4
  - ✓ 所有的 GPIO 均支持如下 4 种模式: 准双向口模式、强推挽输出模式、开漏输出模式、高阻输入模式
  - ✓ 除 P3.0 和 P3.1 外, 其余所有 IO 口上电后的状态均为高阻输入状态, 用户在使用 IO 口时必须先设置 IO 口模式。另外每个 I/O 均可独立使能内部 4K 上拉电阻
- 封装
  - ✓ LQFP48 <9mm\*9mm>、QFN48 <6mm\*6mm>
- ✓ 特别说明
  - ✓ 本系列芯片有 CHIPID 功能
  - ✓ 本系列芯片有所有的 I/O 口中断功能, 有 4 级中断优先级, 可掉电唤醒
  - ✓ 本系列芯片的比较器为 4P+2N 版本

## 2.6.2 管脚图，最小系统

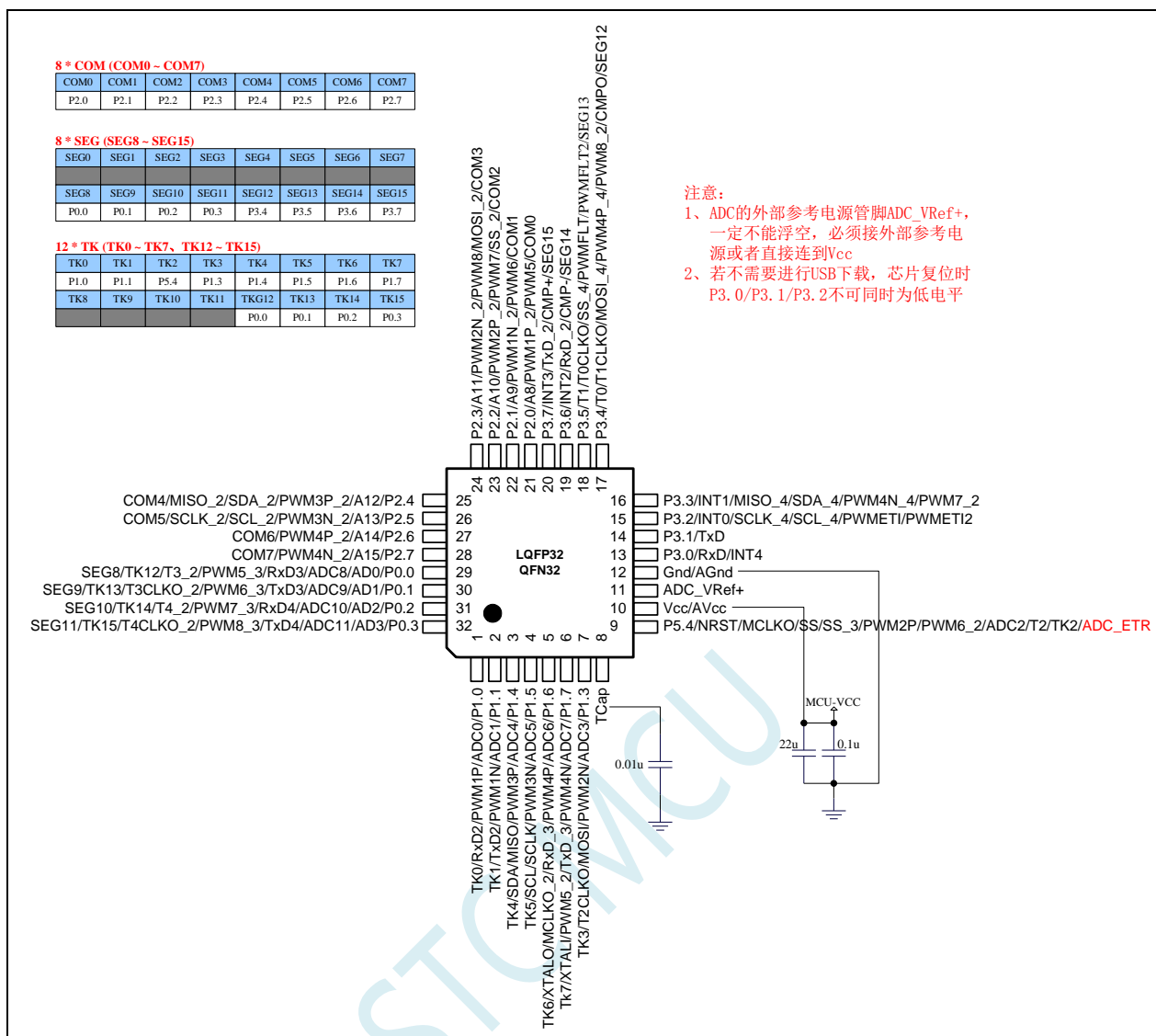


正看芯片丝印左下方小圆点处为第一脚

正看芯片丝印最下面一行最后一个字母为芯片版本号

建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF, 可去除电源线噪声, 提高抗干扰能力

- 注意：**
- 1、除 P3.0 和 P3.1 外, 其余所有 I/O 口上电后的状态均为高阻输入状态, 用户在使用 I/O 口时必须先设置 I/O 口模式
  - 2、所有的 I/O 口均可以设置为准双向口模式、强推挽输出模式、开漏输出模式或者高阻输入模式, 另外每个 I/O 均可独立使能内部 4K 上拉电阻
  - 3、当使能 P5.4 口为复位脚时, 复位电平为低电平



正看芯片丝印左下方小圆点处为第一脚

正看芯片丝印最下面一行最后一个字母为芯片版本号

建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF, 可去除电源线噪声, 提高抗干扰能力

## 0 \* COM

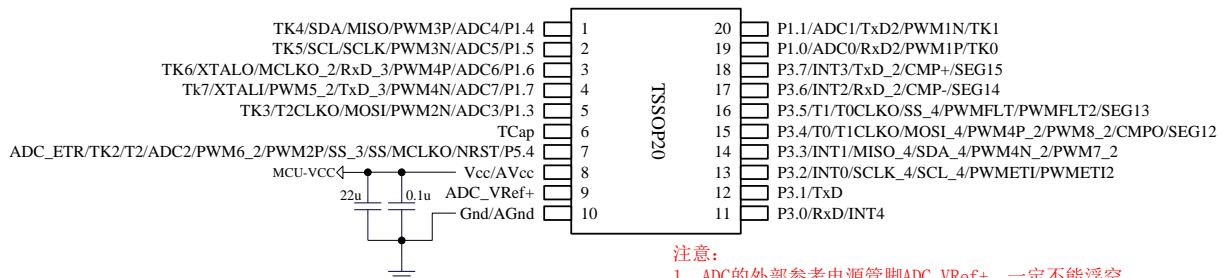
| COM0 | COM1 | COM2 | COM3 | COM4 | COM5 | COM6 | COM7 |
|------|------|------|------|------|------|------|------|
|      |      |      |      |      |      |      |      |

## 4 \* SEG (SEG12 ~ SEG15)

| SEG0 | SEG1 | SEG2  | SEG3  | SEG4  | SEG5  | SEG6  | SEG7  |
|------|------|-------|-------|-------|-------|-------|-------|
|      |      |       |       |       |       |       |       |
| SEG8 | SEG9 | SEG10 | SEG11 | SEG12 | SEG13 | SEG14 | SEG15 |
|      |      |       |       | P3.4  | P3.5  | P3.6  | P3.7  |

## 8 \* TK (TK0 ~ TK7)

| TK0  | TK1  | TK2  | TK3  | TK4  | TK5  | TK6  | TK7  |
|------|------|------|------|------|------|------|------|
| P1.0 | P1.1 | P5.4 | P1.3 | P1.4 | P1.5 | P1.6 | P1.7 |
| TK8  | TK9  | TK10 | TK11 | TK12 | TK13 | TK14 | TK15 |
|      |      |      |      |      |      |      |      |



## 注意:

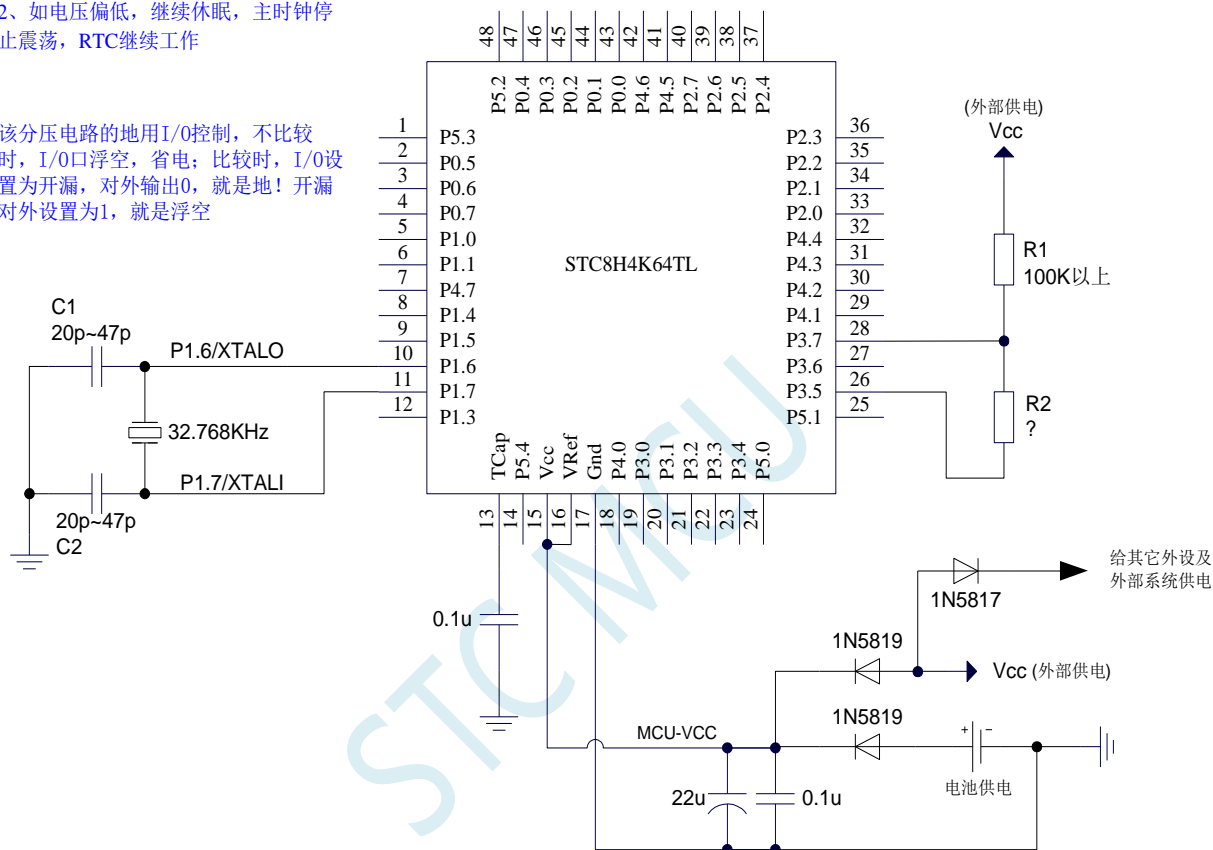
- 1、ADC的外部参考电源管脚ADC\_VRef+, 一定不能浮空, 必须接外部参考电源或者直接连到Vcc
- 2、若不需要进行USB下载, 芯片复位时P3.0/P3.1/P3.2不可同时为低电平

### 2.6.3 RTC 实战线路图

用RTC定时唤醒MCU，如2秒唤醒1次，唤醒后，用比较器判断外部电压：

- 1、正常，正常工作；
- 2、如电压偏低，继续休眠，主时钟停止震荡，RTC继续工作

该分压电路的地用I/O控制，不比较时，I/O口浮空，省电；比较时，I/O设置为开漏，对外输出0，就是地！开漏对外设置为1，就是浮空



## 2.6.4 管脚说明

| 编号           |              | 名称        | 类型  | 说明                |
|--------------|--------------|-----------|-----|-------------------|
| LQFP48/QFN48 | LQFP32/QFN32 |           |     |                   |
| 1            |              | P5.3      | I/O | 标准 IO 口           |
|              |              | TxD4_2    | O   | 串口 4 的发送脚         |
|              |              | TK11      | I   | 触摸按键              |
| 2            |              | P0.5      | I/O | 标准 IO 口           |
|              |              | AD5       | I/O | 地址/数据总线           |
|              |              | ADC13     | I   | ADC 模拟输入通道 13     |
|              |              | T3CLKO    | O   | 定时器 3 时钟分频输出      |
| 3            |              | P0.6      | I/O | 标准 IO 口           |
|              |              | AD6       | I/O | 地址/数据总线           |
|              |              | ADC14     | I   | ADC 模拟输入通道 14     |
|              |              | T4        | I   | 定时器 4 外部时钟输入      |
|              |              | PWMETI2_2 | I   | PWM 外部触发输入脚 2     |
| 4            |              | P0.7      | I/O | 标准 IO 口           |
|              |              | AD7       | I/O | 地址/数据总线           |
|              |              | T4CLKO    | O   | 定时器 4 时钟分频输出      |
| 5            | 1            | P1.0      | I/O | 标准 IO 口           |
|              |              | ADC0      | I   | ADC 模拟输入通道 0      |
|              |              | PWM1P     | I/O | PWM1 的捕获输入和脉冲输出正极 |
|              |              | RxD2      | I   | 串口 2 的接收脚         |
|              |              | TK0       | I   | 触摸按键              |
| 6            | 2            | P1.1      | I/O | 标准 IO 口           |
|              |              | ADC1      | I   | ADC 模拟输入通道 1      |
|              |              | PWM1N     | I/O | PWM1 的捕获输入和脉冲输出负极 |
|              |              | TxD2      | I   | 串口 2 的发送脚         |
|              |              | TK1       | I   | 触摸按键              |
| 7            |              | P4.7      | I/O | 标准 IO 口           |
|              |              | TxD2_2    | I   | 串口 2 的发送脚         |
|              |              | SEG7      | O   | LED 驱动            |
| 8            | 3            | P1.4      | I/O | 标准 IO 口           |
|              |              | ADC4      | I   | ADC 模拟输入通道 4      |
|              |              | PWM3P     | I/O | PWM3 的捕获输入和脉冲输出正极 |
|              |              | MISO      | I/O | SPI 主机输入从机输出      |
|              |              | SDA       | I/O | I2C 接口的数据线        |
|              |              | TK4       | I   | 触摸按键              |

| 编号           |              | 名称      | 类型  | 说明                |
|--------------|--------------|---------|-----|-------------------|
| LQFP48/QFN48 | LQFP32/QFN32 |         |     |                   |
| 9            | 4            | P1.5    | I/O | 标准 IO 口           |
|              |              | ADC5    | I   | ADC 模拟输入通道 5      |
|              |              | PWM3N   | I/O | PWM3 的捕获输入和脉冲输出负极 |
|              |              | SCLK    | I/O | SPI 的时钟脚          |
|              |              | SCL     | I/O | I2C 的时钟线          |
|              |              | TK5     | I   | 触摸按键              |
| 10           | 5            | P1.6    | I/O | 标准 IO 口           |
|              |              | ADC6    | I   | ADC 模拟输入通道 6      |
|              |              | RxD_3   | I   | 串口 1 的接收脚         |
|              |              | PWM4P   | I/O | PWM4 的捕获输入和脉冲输出正极 |
|              |              | MCLKO_2 | O   | 主时钟分频输出           |
|              |              | XTALO   | O   | 外部晶振的输出脚          |
|              |              | TK6     | I   | 触摸按键              |
| 11           | 6            | P1.7    | I/O | 标准 IO 口           |
|              |              | ADC7    | I   | ADC 模拟输入通道 7      |
|              |              | TxD_3   | O   | 串口 1 的发送脚         |
|              |              | PWM4N   | I/O | PWM4 的捕获输入和脉冲输出负极 |
|              |              | PWM5_2  | I/O | PWM5 的捕获输入和脉冲输出   |
|              |              | XTALI   | I   | 外部晶振/外部时钟的输入脚     |
|              |              | TK7     | I   | 触摸按键              |
| 12           | 7            | P1.3    | I/O | 标准 IO 口           |
|              |              | ADC3    | I   | ADC 模拟输入通道 3      |
|              |              | MOSI    | I/O | SPI 主机输出从机输入      |
|              |              | PWM2N   | I/O | PWM2 的捕获输入和脉冲输出负极 |
|              |              | T2CLKO  | O   | 定时器 2 时钟分频输出      |
|              |              | TK3     | I   | 触摸按键              |
| 13           | 8            | TCAP    | I   | 触摸按键充放电电容         |
| 14           | 9            | P5.4    | I/O | 标准 IO 口           |
|              |              | NRST    | I   | 复位引脚（低电平复位）       |
|              |              | MCLKO   | O   | 主时钟分频输出           |
|              |              | SS_3    | I   | SPI 的从机选择脚（主机为输出） |
|              |              | SS      | I   | SPI 的从机选择脚（主机为输出） |
|              |              | PWM2P   | I/O | PWM2 的捕获输入和脉冲输出正极 |
|              |              | PWM6_2  | I/O | PWM6 的捕获输入和脉冲输出   |
|              |              | T2      | I   | 定时器 2 外部时钟输入      |
|              |              | ADC2    | I   | ADC 模拟输入通道 2      |
|              |              | TK2     | I   | 触摸按键              |
|              |              | ADC_ETR | I   | ADC 外部触发脚         |



| 编号           |              | 名称        | 类型  | 说明                                  |
|--------------|--------------|-----------|-----|-------------------------------------|
| LQFP48/QFN48 | LQFP32/QFN32 |           |     |                                     |
| 15           | 10           | Vcc       | Vcc | 电源脚                                 |
|              |              | AVcc      | Vcc | ADC 电源脚                             |
| 16           | 11           | ADC_VRef+ | I   | ADC 外部参考电压源输入脚, 要求不高时可直接接 MCU 的 VCC |
| 17           | 12           | Gnd       | Gnd | 地线                                  |
|              |              | AGnd      | Gnd | ADC 地线                              |
| 18           |              | P4.0      | I/O | 标准 IO 口                             |
|              |              | MOSI_3    | I/O | SPI 主机输出从机输入                        |
|              |              | SEG0      | O   | LED 驱动                              |
| 19           | 13           | P3.0      | I/O | 标准 IO 口                             |
|              |              | RxD       | I   | 串口 1 的接收脚                           |
|              |              | INT4      | I   | 外部中断 4                              |
| 20           | 14           | P3.1      | I/O | 标准 IO 口                             |
|              |              | TxD       | O   | 串口 1 的发送脚                           |
| 21           | 15           | P3.2      | I/O | 标准 IO 口                             |
|              |              | INT0      | I   | 外部中断 0                              |
|              |              | SCLK_4    | I/O | SPI 的时钟脚                            |
|              |              | SCL_4     | I/O | I2C 的时钟线                            |
|              |              | PWMETI    | I   | PWM 外部触发输入脚                         |
|              |              | PWMETI2   | I   | PWM 外部触发输入脚 2                       |
| 22           | 16           | P3.3      | I/O | 标准 IO 口                             |
|              |              | INT1      | I   | 外部中断 1                              |
|              |              | MISO_4    | I/O | SPI 主机输入从机输出                        |
|              |              | SDA_4     | I/O | I2C 接口的数据线                          |
|              |              | PWM4N_4   | I/O | PWM4 的捕获输入和脉冲输出负极                   |
|              |              | PWM7_2    | I/O | PWM7 的捕获输入和脉冲输出                     |
| 23           | 17           | P3.4      | I/O | 标准 IO 口                             |
|              |              | T0        | I   | 定时器 0 外部时钟输入                        |
|              |              | T1CLKO    | O   | 定时器 1 时钟分频输出                        |
|              |              | MOSI_4    | I/O | SPI 主机输出从机输入                        |
|              |              | PWM4P_4   | I/O | PWM4 的捕获输入和脉冲输出正极                   |
|              |              | PWM8_2    | I/O | PWM8 的捕获输入和脉冲输出                     |
|              |              | CMPO      | O   | 比较器输出                               |
|              |              | SEG12     | O   | LED 驱动                              |

| 编号           |              | 名称       | 类型  | 说明                |
|--------------|--------------|----------|-----|-------------------|
| LQFP48/QFN48 | LQFP32/QFN32 |          |     |                   |
| 24           |              | P5.0     | I/O | 标准 IO 口           |
|              |              | RxD3_2   | I   | 串口 3 的接收脚         |
|              |              | TK8      | I   | 触摸按键              |
| 25           |              | P5.1     | I/O | 标准 IO 口           |
|              |              | TxD3_2   | O   | 串口 3 的发送脚         |
|              |              | TK9      | I   | 触摸按键              |
| 26           | 18           | P3.5     | I/O | 标准 IO 口           |
|              |              | T1       | I   | 定时器 1 外部时钟输入      |
|              |              | T0CLKO   | O   | 定时器 0 时钟分频输出      |
|              |              | SS_4     | I   | SPI 的从机选择脚（主机为输出） |
|              |              | PWMFLT   | I   | 增强 PWMA 的外部异常检测脚  |
|              |              | PWMFLT2  | I   | 增强 PWMB 的外部异常检测脚  |
|              |              | SEG13    | O   | LED 驱动            |
| 27           | 19           | P3.6     | I/O | 标准 IO 口           |
|              |              | INT2     | I   | 外部中断 2            |
|              |              | RxD_2    | I   | 串口 1 的接收脚         |
|              |              | CMP-     | I   | 比较器负极输入           |
|              |              | SEG14    | O   | LED 驱动            |
| 28           | 20           | P3.7     | I/O | 标准 IO 口           |
|              |              | INT3     | I   | 外部中断 3            |
|              |              | TxD_2    | O   | 串口 1 的发送脚         |
|              |              | CMP+     | I   | 比较器正极输入           |
|              |              | SEG15    | O   | LED 驱动            |
| 29           |              | P4.1     | I/O | 标准 IO 口           |
|              |              | MISO_3   | I/O | SPI 主机输入从机输出      |
|              |              | CMPO_2   | O   | 比较器输出             |
|              |              | PWMETI_3 | I   | PWM 外部触发输入脚       |
|              |              | SEG1     | O   | LED 驱动            |
| 30           |              | P4.2     | I/O | 标准 IO 口           |
|              |              | WR       | O   | 外部总线的写信号线         |
|              |              | SEG2     | O   | LED 驱动            |
| 31           |              | P4.3     | I/O | 标准 IO 口           |
|              |              | RxD_4    | I   | 串口 1 的接收脚         |
|              |              | SCLK_3   | I/O | SPI 的时钟脚          |
|              |              | SEG3     | O   | LED 驱动            |

| 编号           |              | 名称      | 类型  | 说明                 |
|--------------|--------------|---------|-----|--------------------|
| LQFP48/QFN48 | LQFP32/QFN32 |         |     |                    |
| 32           |              | P4.4    | I/O | 标准 IO 口            |
|              |              | RD      | O   | 外部总线的读信号线          |
|              |              | TxD_4   | O   | 串口 1 的发送脚          |
|              |              | SEG4    | O   | LED 驱动             |
| 33           | 21           | P2.0    | I/O | 标准 IO 口            |
|              |              | A8      | O   | 地址总线               |
|              |              | PWM1P_2 | I/O | PWM1 的捕获输入和脉冲输出正极  |
|              |              | PWM5    | I/O | PWM5 的捕获输入和脉冲输出    |
|              |              | COM0    | O   | LED 驱动             |
| 34           | 22           | P2.1    | I/O | 标准 IO 口            |
|              |              | A9      | O   | 地址总线               |
|              |              | PWM1N_2 | I/O | PWM1 的捕获输入和脉冲输出负极  |
|              |              | PWM6    | I/O | PWM6 的捕获输入和脉冲输出    |
|              |              | COM1    | O   | LED 驱动             |
| 35           | 23           | P2.2    | I/O | 标准 IO 口            |
|              |              | A10     | O   | 地址总线               |
|              |              | SS_2    | I   | SPI 的从机选择脚 (主机为输出) |
|              |              | PWM2P_2 | I/O | PWM2 的捕获输入和脉冲输出正极  |
|              |              | PWM7    | I/O | PWM7 的捕获输入和脉冲输出    |
|              |              | COM2    | O   | LED 驱动             |
| 36           | 24           | P2.3    | I/O | 标准 IO 口            |
|              |              | A11     | O   | 地址总线               |
|              |              | MOSI_2  | I/O | SPI 主机输出从机输入       |
|              |              | PWM2N_2 | I/O | PWM2 的捕获输入和脉冲输出负极  |
|              |              | PWM8    | I/O | PWM8 的捕获输入和脉冲输出    |
|              |              | COM3    | O   | LED 驱动             |

| 编号           |              | 名称      | 类型  | 说明                |
|--------------|--------------|---------|-----|-------------------|
| LQFP48/QFN48 | LQFP32/QFN32 |         |     |                   |
| 37           | 25           | P2.4    | I/O | 标准 IO 口           |
|              |              | A12     | O   | 地址总线              |
|              |              | MISO_2  | I/O | SPI 主机输入从机输出      |
|              |              | SDA_2   | I/O | I2C 接口的数据线        |
|              |              | PWM3P_2 | I/O | PWM3 的捕获输入和脉冲输出正极 |
|              |              | COM4    | O   | LED 驱动            |
| 38           | 26           | P2.5    | I/O | 标准 IO 口           |
|              |              | A13     | O   | 地址总线              |
|              |              | SCLK_2  | I/O | SPI 的时钟脚          |
|              |              | SCL_2   | I/O | I2C 的时钟线          |
|              |              | PWM3N_2 | I/O | PWM3 的捕获输入和脉冲输出负极 |
|              |              | COM5    | O   | LED 驱动            |
| 39           | 27           | P2.6    | I/O | 标准 IO 口           |
|              |              | A14     | O   | 地址总线              |
|              |              | PWM4P_2 | I/O | PWM4 的捕获输入和脉冲输出正极 |
|              |              | COM6    | O   | LED 驱动            |
| 40           | 28           | P2.7    | I/O | 标准 IO 口           |
|              |              | A15     | O   | 地址总线              |
|              |              | PWM4N_2 | I/O | PWM4 的捕获输入和脉冲输出负极 |
|              |              | COM7    | O   | LED 驱动            |
| 41           |              | P4.5    | I/O | 标准 IO 口           |
|              |              | ALE     | O   | 地址锁存信号            |
|              |              | SEG5    | O   | LED 驱动            |
| 42           |              | P4.6    | I/O | 标准 IO 口           |
|              |              | RxD2_2  | I   | 串口 2 的接收脚         |
|              |              | SEG6    | O   | LED 驱动            |

| 编号           |              | 名称       | 类型  | 说明              |
|--------------|--------------|----------|-----|-----------------|
| LQFP48/QFN48 | LQFP32/QFN32 |          |     |                 |
| 43           | 29           | P0.0     | I/O | 标准 IO 口         |
|              |              | AD0      | I/O | 地址/数据总线         |
|              |              | ADC8     | I   | ADC 模拟输入通道 8    |
|              |              | RxD3     | I   | 串口 3 的接收脚       |
|              |              | PWM5_3   | I/O | PWM5 的捕获输入和脉冲输出 |
|              |              | T3_2     | I   | 定时器 3 外部时钟输入    |
|              |              | TK12     | I   | 触摸按键            |
|              |              | SEG8     | O   | LED 驱动          |
| 44           | 30           | P0.1     | I/O | 标准 IO 口         |
|              |              | AD1      | I/O | 地址/数据总线         |
|              |              | ADC9     | I   | ADC 模拟输入通道 9    |
|              |              | TxD3     | O   | 串口 3 的发送脚       |
|              |              | PWM6_3   | I/O | PWM6 的捕获输入和脉冲输出 |
|              |              | T3CLKO_2 | O   | 定时器 3 时钟分频输出    |
|              |              | TK13     | I   | 触摸按键            |
|              |              | SEG9     | O   | LED 驱动          |
| 45           | 31           | P0.2     | I/O | 标准 IO 口         |
|              |              | AD2      | I/O | 地址/数据总线         |
|              |              | ADC10    | I   | ADC 模拟输入通道 10   |
|              |              | RxD4     | I   | 串口 4 的接收脚       |
|              |              | PWM7_3   | I/O | PWM7 的捕获输入和脉冲输出 |
|              |              | T4_2     | I   | 定时器 4 外部时钟输入    |
|              |              | TK14     | I   | 触摸按键            |
|              |              | SEG10    | O   | LED 驱动          |
| 46           | 32           | P0.3     | I/O | 标准 IO 口         |
|              |              | AD3      | I/O | 地址/数据总线         |
|              |              | ADC11    | I   | ADC 模拟输入通道 11   |
|              |              | TxD4     | O   | 串口 4 的发送脚       |
|              |              | PWM8_3   | I/O | PWM8 的捕获输入和脉冲输出 |
|              |              | T4CLKO_2 | O   | 定时器 4 时钟分频输出    |
|              |              | TK15     | I   | 触摸按键            |
|              |              | SEG11    | O   | LED 驱动          |
| 47           |              | P0.4     | I/O | 标准 IO 口         |
|              |              | AD4      | I/O | 地址/数据总线         |
|              |              | ADC12    | I   | ADC 模拟输入通道 12   |
|              |              | T3       | I   | 定时器 3 外部时钟输入    |
| 48           |              | P5.2     | I/O | 标准 IO 口         |
|              |              | RxD4_2   | I   | 串口 4 的接收脚       |



- ✓ 内部高精度 IRC (4MHz~45MHz, ISP 编程时选择或手动输入, 还可以用户软件分频到较低的频率工作, 如 100KHz)
  - ✦ 误差±0.3% (常温下 25℃)
  - ✦ -1.35%~+1.30%温漂 (全温度范围, -40℃~85℃)
  - ✦ -0.76%~+0.98%温漂 (温度范围, -20℃~65℃)
- ✓ 内部 32KHz 低速 IRC (误差较大)
- ✓ 外部晶振 (4MHz~45MHz) 和外部时钟  
用户可自由选择上面的 3 种时钟源

## ➤ 复位

- ✓ 硬件复位
  - ✦ 上电复位, 实测电压值为 1.69V~1.82V。(在芯片未使能低压复位功能时有效)  
上电复位电压由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到上电复位的下限门槛电压时, 芯片处于复位状态; 当电压从 0V 上升到上电复位的上限门槛电压时, 芯片解除复位状态。
  - ✦ 复位脚复位, 出厂时 P5.4 默认为 I/O 口, ISP 下载时可将 P5.4 管脚设置为复位脚 (注意: 当设置 P5.4 管脚为复位脚时, 复位电平为低电平)
  - ✦ 看门狗溢出复位
  - ✦ 低压检测复位, 提供 4 级低压检测电压: 1.9V、2.3V、2.8V、3.7V。  
每级低压检测电压都是由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到低压检测的下限门槛电压时, 低压检测生效; 当电压从 0V 上升到低压检测的上限门槛电压时, 低压检测生效。
- ✓ 软件复位
  - ✦ 软件方式写复位触发寄存器

## ➤ 中断

- ✓ 提供 43 个中断源: INT0 (支持上升沿和下降沿中断)、INT1 (支持上升沿和下降沿中断)、INT2 (只支持下降沿中断)、INT3 (只支持下降沿中断)、INT4 (只支持下降沿中断)、定时器 0、定时器 1、定时器 2、定时器 3、定时器 4、串口 1、串口 2、串口 3、串口 4、ADC 模数转换、LVD 低压检测、SPI、I<sup>2</sup>C、比较器、PWMA、PWMB、RTC、TKS、P0 口中断、P1 口中断、P2 口中断、P3 口中断、P4 口中断、P5 口中断、P6 口中断、P7 口中断、LCM 驱动中断、串口 1 的 DMA 接收和发送中断、串口 2 的 DMA 接收和发送中断、串口 3 的 DMA 接收和发送中断、串口 4 的 DMA 接收和发送中断、SPI 的 DMA 中断、ADC 的 DMA 中断、LCD 驱动的 DMA 中断以及存储器到存储器的 DMA 中断。
- ✓ 提供 4 级中断优先级
- ✓ 时钟停振模式下可以唤醒的中断: INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、T3(P0.4)、T4(P0.6)、RXD(P3.0/P3.6/P1.6/P4.3)、RXD2(P1.0/P4.6)、RXD3(P0.0/P5.0)、RXD4(P0.2/P5.2)、I2C\_SDA(P1.4/P2.4/P3.3)、SPI\_SS(P5.4/P2.2/P3.5)以及比较器中断、低压检测中断、掉电唤醒定时器唤醒以及所有端口的 I/O 中断。

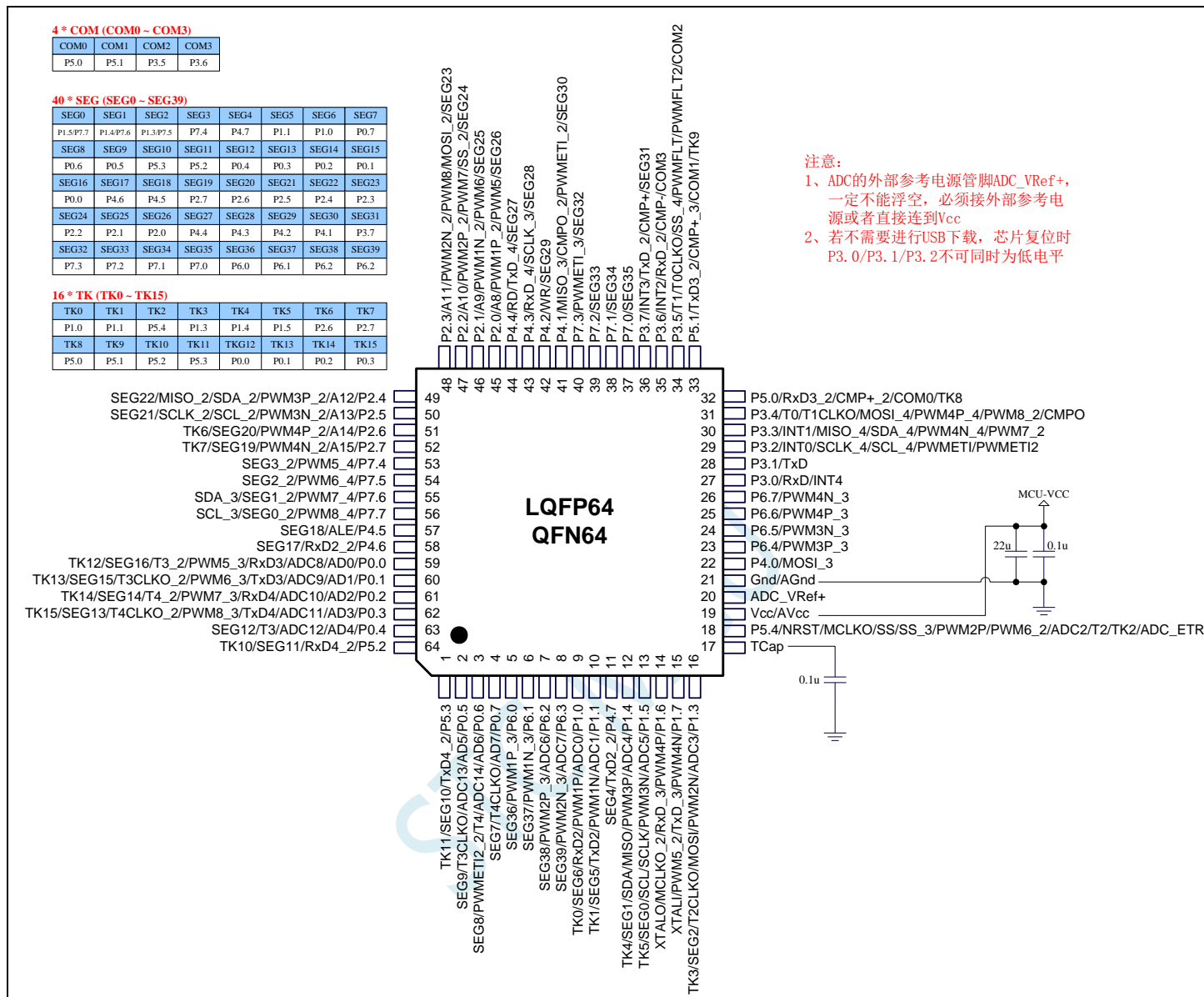
## ➤ 数字外设

- ✓ 5 个 16 位定时器: 定时器 0、定时器 1、定时器 2、定时器 3、定时器 4, 其中定时器 0 的模式 3 具有 NMI (不可屏蔽中断) 功能, 定时器 0 和定时器 1 的模式 0 为 16 位自动重载模式
- ✓ 4 个高速串口: 串口 1、串口 2、串口 3、串口 4, 波特率时钟源最快可为 FOSC/4
- ✓ 8 路/2 组高级 PWM, 可实现带死区的控制信号, 并支持外部异常检测功能, 另外还支持 16 位定时器、8 个外部中断、8 路外部捕获测量脉宽等功能
- ✓ SPI: 支持主机模式和从机模式以及主机/从机自动切换
- ✓ I<sup>2</sup>C: 支持主机模式和从机模式

- ✓ MDU16: 硬件 16 位乘除法器 (支持 32 位除以 16 位、16 位除以 16 位、16 位乘 16 位、数据移位以及数据规格化等运算)
  - ✓ RTC: 支持年、月、日、时、分、秒、次秒 (1/128 秒), 并支持时钟中断和一组闹钟
  - ✓ I/O 口中断: 所有的 I/O 均支持中断, 每组 I/O 中断有独立的中断入口地址, 所有的 I/O 中断可支持 4 种中断模式: 高电平中断、低电平中断、上升沿中断、下降沿中断。提供 4 级中断优先级并支持掉电唤醒功能。
  - ✓ DMA: 支持 Memory-To-Memory、SPI、UART1TX/UART1RX、UART2TX/UART2RX、UART3TX/UART3RX、UART4TX/UART4RX、ADC (自动计算多次 ADC 结果的平均值)、LCM
  - ✓ LCM (TFT 彩屏) 驱动: 支持 8080 和 6800 接口, 支持 8 位和 16 位数据宽度
    - ✦ 8 位 8080 总线: 8 为数据线 (TD0~TD7), 读信号 (TRD), 写信号 (TWR), RS 线 (TRS)
    - ✦ 16 为 8080 总线: 16 为数据线 (TD0~TD15), 读信号 (TRD), 写信号 (TWR), RS 线 (TRS)
    - ✦ 8 为 6800 总线: 8 为数据线 (TD0~TD7), 使能信号 (TE), 读写信号 (TRW), RS 线 (TRS)
    - ✦ 16 位 6800 总线: 16 为数据线 (TD0~TD15), 使能信号 (TE), 读写信号 (TRW), RS 线 (TRS)
    - ✦ 注意: 如果使用 8 位数据线控制 TFT 屏, 一般需要 TD0~D7, TRD/TWR/TRS, 11 根数据及控制线, 外加 2 个普通 I/O 控制片选及复位 (很多 TFT 彩屏的片选及复位厂商已进行了自动处理, 并不需要软件控制)
  - ✓ LCD 液晶驱动: 最大支持 4COM\*40SEG; 支持 8 级灰度调节
- 模拟外设
- ✓ 超高速 ADC, 支持 12 位高精度 15 通道 (通道 0~通道 14) 的模数转换, 速度最快能达到 800K (每秒进行 80 万次 ADC 转换)
  - ✓ ADC 的通道 15 用于测试内部 1.19V 参考信号源 (芯片在出厂时, 内部参考信号源已调整为 1.19V)
  - ✓ 比较器, 一组比较器 (比较器的正端可选择 CMP+ 端口和所有的 ADC 输入端口, 所以比较器可当作多路比较器进行分时复用)
  - ✓ 触摸按键: 最多支持 16 个触摸按键; 每个触摸按键能够独立使能; 内部参考电压 4 级可调; 灵活的充放电时间设置以及内部工作频率设置; 支持低功耗触摸唤醒
  - ✓ DAC: 8 路高级 PWM 定时器可当 8 路 DAC 使用
- GPIO
- ✓ 最多可达 60 个 GPIO: P0.0~P0.7、P1.0~P1.7 (无 P1.2)、P2.0~P2.7、P3.0~P3.7、P4.0~P4.7、P5.0~P5.4、P6.0~P6.7、P7.0~P7.7
  - ✓ 所有的 GPIO 均支持如下 4 种模式: 准双向口模式、强推挽输出模式、开漏输出模式、高阻输入模式
  - ✓ 除 P3.0 和 P3.1 外, 其余所有 IO 口上电后的状态均为高阻输入状态, 用户在使用 IO 口时必须先设置 IO 口模式。另外每个 I/O 均可独立使能内部 4K 上拉电阻
- 封装
- ✓ LQFP64 <12mm\*12mm>、QFN64 <8mm\*8mm>、LQFP48 <9mm\*9mm>、QFN48 <6mm\*6mm>
- ✓ 特别说明
- ✓ 本系列芯片有 CHIPID 功能
  - ✓ 本系列芯片有所有的 I/O 口中断功能, 有 4 级中断优先级, 可掉电唤醒
  - ✓ 本系列芯片的比较器为 4P+2N 版本



## 2.7.2 管脚图，最小系统



正看芯片丝印左下方小圆点处为第一脚

正看芯片丝印最下面一行最后一个字母为芯片版本号

建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF，可去除电源线噪声，提高抗干扰能力

## 4 \* COM (COM0 ~ COM3)

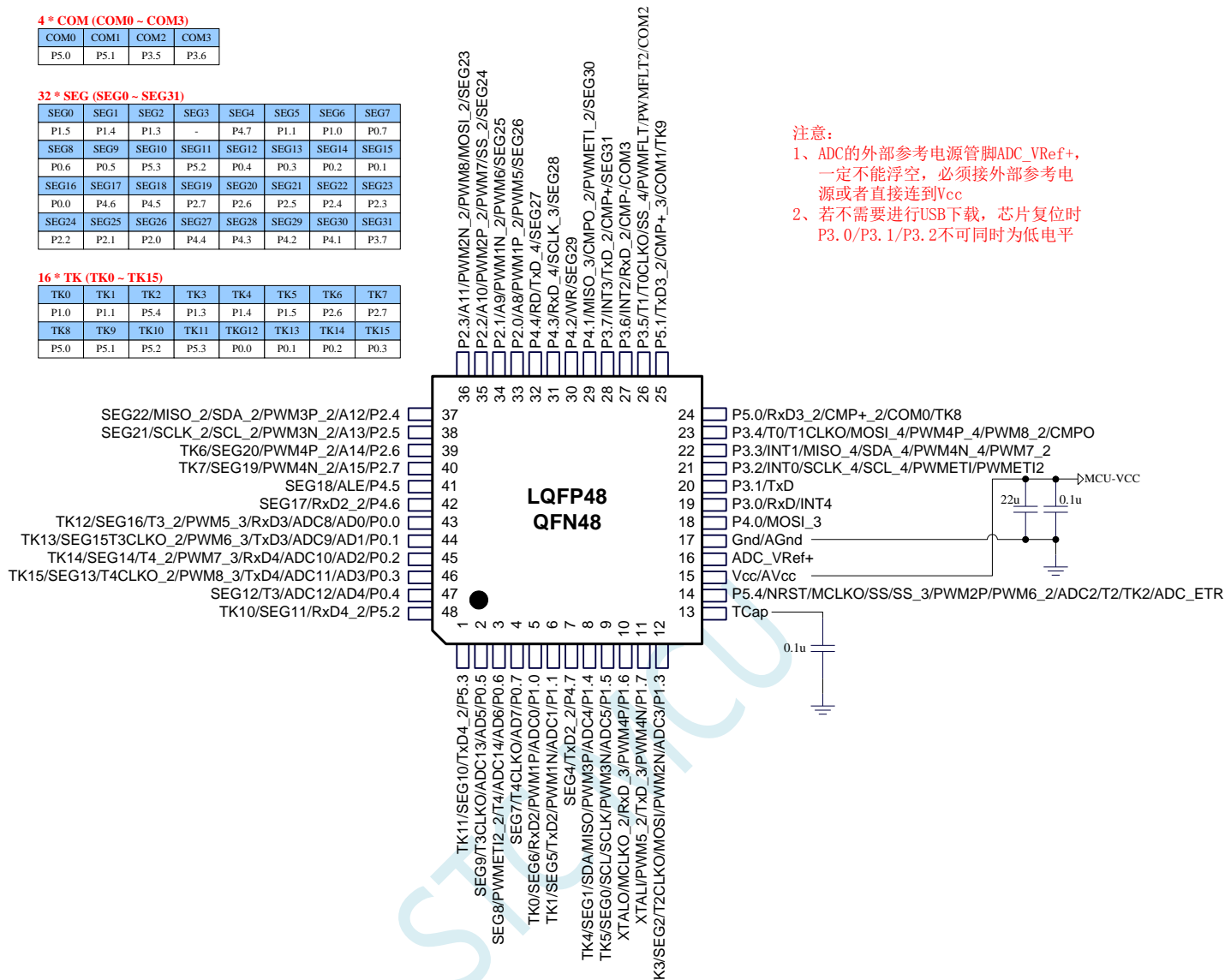
| COM0 | COM1 | COM2 | COM3 |
|------|------|------|------|
| P5.0 | P5.1 | P3.5 | P3.6 |

## 32 \* SEG (SEG0 ~ SEG31)

| SEG0  | SEG1  | SEG2  | SEG3  | SEG4  | SEG5  | SEG6  | SEG7  |
|-------|-------|-------|-------|-------|-------|-------|-------|
| P1.5  | P1.4  | P1.3  | -     | P4.7  | P1.1  | P1.0  | P0.7  |
| SEG8  | SEG9  | SEG10 | SEG11 | SEG12 | SEG13 | SEG14 | SEG15 |
| P0.6  | P0.5  | P5.3  | P5.2  | P0.4  | P0.3  | P0.2  | P0.1  |
| SEG16 | SEG17 | SEG18 | SEG19 | SEG20 | SEG21 | SEG22 | SEG23 |
| P0.0  | P4.6  | P4.5  | P2.7  | P2.6  | P2.5  | P2.4  | P2.3  |
| SEG24 | SEG25 | SEG26 | SEG27 | SEG28 | SEG29 | SEG30 | SEG31 |
| P2.2  | P2.1  | P2.0  | P4.4  | P4.3  | P4.2  | P4.1  | P3.7  |

## 16 \* TK (TK0 ~ TK15)

| TK0  | TK1  | TK2  | TK3  | TK4  | TK5  | TK6  | TK7  |
|------|------|------|------|------|------|------|------|
| P1.0 | P1.1 | P5.4 | P1.3 | P1.4 | P1.5 | P2.6 | P2.7 |
| TK8  | TK9  | TK10 | TK11 | TK12 | TK13 | TK14 | TK15 |
| P5.0 | P5.1 | P5.2 | P5.3 | P0.0 | P0.1 | P0.2 | P0.3 |



正看芯片丝印左下方小圆点处为第一脚

正看芯片丝印最下面一行最后一个字母为芯片版本号

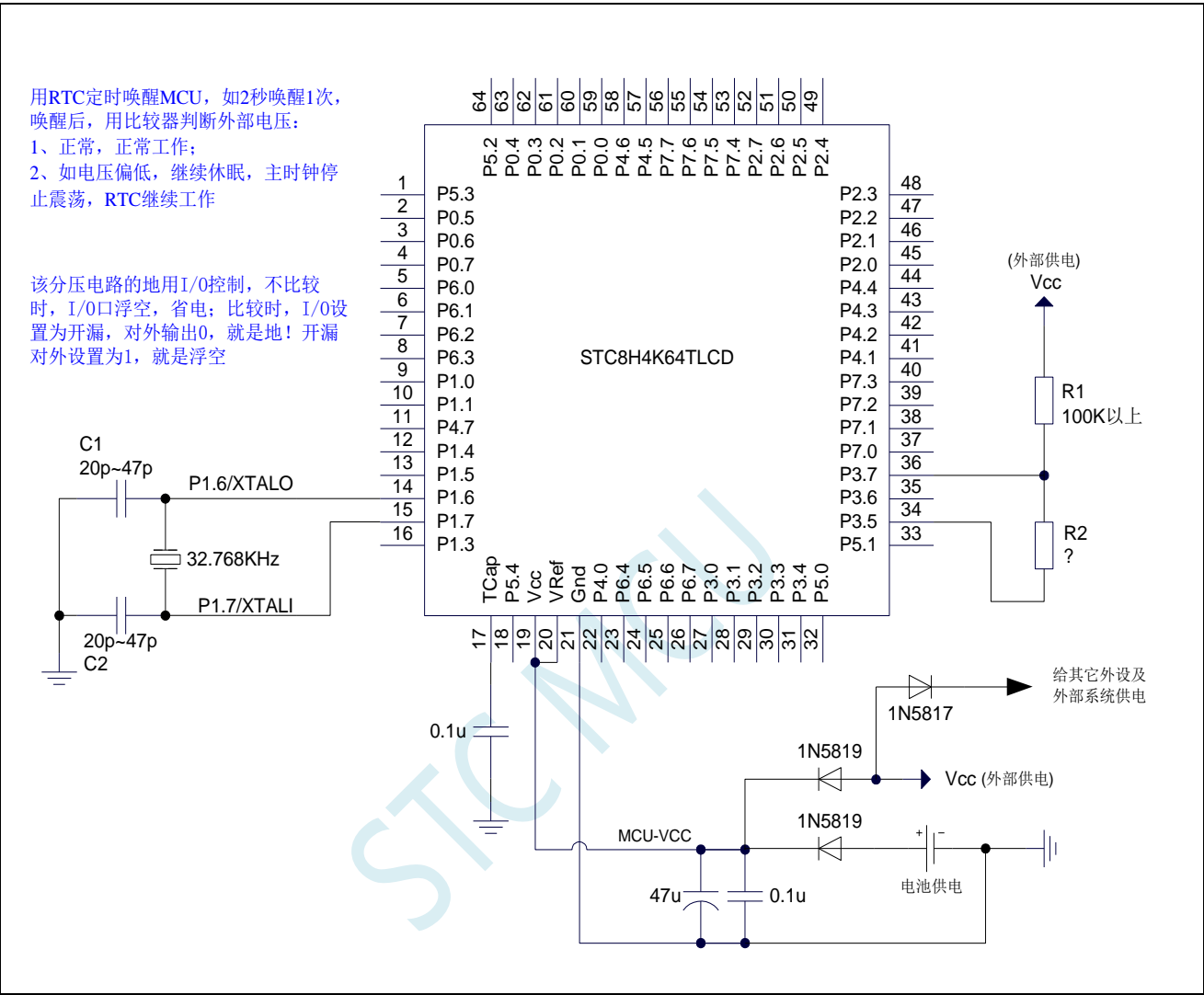
建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF, 可去除电源线噪声, 提高抗干扰能力

注意: 1、除 P3.0 和 P3.1 外, 其余所有 I/O 口上电后的状态均为高阻输入状态, 用户在使用 I/O 口时必须先设置 I/O 口模式

2、所有的 I/O 口均可以设置为准双向口模式、强推挽输出模式、开漏输出模式或者高阻输入模式, 另外每个 I/O 均可独立使能内部 4K 上拉电阻

3、当使能 P5.4 口为复位脚时, 复位电平为低电平

### 2.7.3 RTC 实战线路图



## 2.7.4 管脚说明

| 编号           |              | 名称        | 类型  | 说明                |
|--------------|--------------|-----------|-----|-------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |           |     |                   |
| 1            | 1            | P5.3      | I/O | 标准 IO 口           |
|              |              | TxD4_2    | O   | 串口 4 的发送脚         |
|              |              | SEG10     | O   | LCD 驱动 SEG 线      |
|              |              | TK11      | I   | 触摸按键              |
| 2            | 2            | P0.5      | I/O | 标准 IO 口           |
|              |              | AD5       | I/O | 地址/数据总线           |
|              |              | ADC13     | I   | ADC 模拟输入通道 13     |
|              |              | T3CLKO    | O   | 定时器 3 时钟分频输出      |
|              |              | SEG9      | O   | LCD 驱动 SEG 线      |
| 3            | 3            | P0.6      | I/O | 标准 IO 口           |
|              |              | AD6       | I/O | 地址/数据总线           |
|              |              | ADC14     | I   | ADC 模拟输入通道 14     |
|              |              | T4        | I   | 定时器 4 外部时钟输入      |
|              |              | PWMETI2_2 | I   | PWM 外部触发输入脚 2     |
|              |              | SEG8      | O   | LCD 驱动 SEG 线      |
| 4            | 4            | P0.7      | I/O | 标准 IO 口           |
|              |              | AD7       | I/O | 地址/数据总线           |
|              |              | T4CLKO    | O   | 定时器 4 时钟分频输出      |
|              |              | SEG7      | O   | LCD 驱动 SEG 线      |
| 5            |              | P6.0      | I/O | 标准 IO 口           |
|              |              | PWM1P_3   | I/O | PWM1 的捕获输入和脉冲输出正极 |
|              |              | SEG36     | O   | LCD 驱动 SEG 线      |
| 6            |              | P6.1      | I/O | 标准 IO 口           |
|              |              | PWM1N_3   | I/O | PWM1 的捕获输入和脉冲输出负极 |
|              |              | SEG37     | O   | LCD 驱动 SEG 线      |
| 7            |              | P6.2      | I/O | 标准 IO 口           |
|              |              | ADC6      | I   | ADC 模拟输入通道 6      |
|              |              | PWM2P_3   | I/O | PWM2 的捕获输入和脉冲输出正极 |
|              |              | SEG38     | O   | LCD 驱动 SEG 线      |
| 8            |              | P6.3      | I/O | 标准 IO 口           |
|              |              | ADC7      | I   | ADC 模拟输入通道 7      |
|              |              | PWM2N_3   | I/O | PWM2 的捕获输入和脉冲输出负极 |
|              |              | SEG39     | O   | LCD 驱动 SEG 线      |

| 编号           |              | 名称      | 类型  | 说明                |
|--------------|--------------|---------|-----|-------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |         |     |                   |
| 9            | 5            | P1.0    | I/O | 标准 IO 口           |
|              |              | ADC0    | I   | ADC 模拟输入通道 0      |
|              |              | PWM1P   | I/O | PWM1 的捕获输入和脉冲输出正极 |
|              |              | RxD2    | I   | 串口 2 的接收脚         |
|              |              | SEG6    | O   | LCD 驱动 SEG 线      |
|              |              | TK0     | I   | 触摸按键              |
| 10           | 6            | P1.1    | I/O | 标准 IO 口           |
|              |              | ADC1    | I   | ADC 模拟输入通道 1      |
|              |              | PWM1N   | I/O | PWM1 的捕获输入和脉冲输出负极 |
|              |              | TxD2    | I   | 串口 2 的发送脚         |
|              |              | SEG5    | O   | LCD 驱动 SEG 线      |
|              |              | TK1     | I   | 触摸按键              |
| 11           | 7            | P4.7    | I/O | 标准 IO 口           |
|              |              | TxD2_2  | I   | 串口 2 的发送脚         |
|              |              | SEG4    | O   | LCD 驱动 SEG 线      |
| 12           | 8            | P1.4    | I/O | 标准 IO 口           |
|              |              | ADC4    | I   | ADC 模拟输入通道 4      |
|              |              | PWM3P   | I/O | PWM3 的捕获输入和脉冲输出正极 |
|              |              | MISO    | I/O | SPI 主机输入从机输出      |
|              |              | SDA     | I/O | I2C 接口的数据线        |
|              |              | SEG1    | O   | LCD 驱动 SEG 线      |
|              |              | TK4     | I   | 触摸按键              |
| 13           | 9            | P1.5    | I/O | 标准 IO 口           |
|              |              | ADC5    | I   | ADC 模拟输入通道 5      |
|              |              | PWM3N   | I/O | PWM3 的捕获输入和脉冲输出负极 |
|              |              | SCLK    | I/O | SPI 的时钟脚          |
|              |              | SCL     | I/O | I2C 的时钟线          |
|              |              | SEG0    | O   | LCD 驱动 SEG 线      |
|              |              | TK5     | I   | 触摸按键              |
| 14           | 10           | P1.6    | I/O | 标准 IO 口           |
|              |              | RxD_3   | I   | 串口 1 的接收脚         |
|              |              | PWM4P   | I/O | PWM4 的捕获输入和脉冲输出正极 |
|              |              | MCLKO_2 | O   | 主时钟分频输出           |
|              |              | XTALO   | O   | 外部晶振的输出脚          |

| 编号           |              | 名称        | 类型  | 说明                                 |
|--------------|--------------|-----------|-----|------------------------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |           |     |                                    |
| 15           | 11           | P1.7      | I/O | 标准 IO 口                            |
|              |              | TxD_3     | O   | 串口 1 的发送脚                          |
|              |              | PWM4N     | I/O | PWM4 的捕获输入和脉冲输出负极                  |
|              |              | PWM5_2    | I/O | PWM5 的捕获输入和脉冲输出                    |
|              |              | XTALI     | I   | 外部晶振/外部时钟的输入脚                      |
| 16           | 12           | P1.3      | I/O | 标准 IO 口                            |
|              |              | ADC3      | I   | ADC 模拟输入通道 3                       |
|              |              | MOSI      | I/O | SPI 主机输出从机输入                       |
|              |              | PWM2N     | I/O | PWM2 的捕获输入和脉冲输出负极                  |
|              |              | T2CLKO    | O   | 定时器 2 时钟分频输出                       |
|              |              | TK3       | I   | 触摸按键                               |
| 17           | 13           | TCAP      | I   | 触摸按键充放电电容                          |
| 18           | 14           | P5.4      | I/O | 标准 IO 口                            |
|              |              | NRST      | I   | 复位引脚（低电平复位）                        |
|              |              | MCLKO     | O   | 主时钟分频输出                            |
|              |              | SS_3      | I   | SPI 的从机选择脚（主机为输出）                  |
|              |              | SS        | I   | SPI 的从机选择脚（主机为输出）                  |
|              |              | PWM2P     | I/O | PWM2 的捕获输入和脉冲输出正极                  |
|              |              | PWM6_2    | I/O | PWM6 的捕获输入和脉冲输出                    |
|              |              | T2        | I   | 定时器 2 外部时钟输入                       |
|              |              | ADC2      | I   | ADC 模拟输入通道 2                       |
|              |              | TK2       | I   | 触摸按键                               |
|              |              | ADC_ETR   | I   | ADC 外部触发脚                          |
| 19           | 15           | Vcc       | Vcc | 电源脚                                |
|              |              | AVcc      | Vcc | ADC 电源脚                            |
| 20           | 16           | ADC_VRef+ | I   | ADC 外部参考电压源输入脚，要求不高时可直接接 MCU 的 VCC |
| 21           | 17           | Gnd       | Gnd | 地线                                 |
|              |              | AGnd      | Gnd | ADC 地线                             |
| 22           | 18           | P4.0      | I/O | 标准 IO 口                            |
|              |              | MOSI_3    | I/O | SPI 主机输出从机输入                       |
| 23           |              | P6.4      | I/O | 标准 IO 口                            |
|              |              | PWM3P_3   | I/O | PWM3 的捕获输入和脉冲输出正极                  |
| 24           |              | P6.5      | I/O | 标准 IO 口                            |
|              |              | PWM3N_3   | I/O | PWM3 的捕获输入和脉冲输出负极                  |

| 编号           |              | 名称      | 类型  | 说明                |
|--------------|--------------|---------|-----|-------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |         |     |                   |
| 25           |              | P6.6    | I/O | 标准 IO 口           |
|              |              | PWM4P_3 | I/O | PWM4 的捕获输入和脉冲输出正极 |
| 26           |              | P6.7    | I/O | 标准 IO 口           |
|              |              | PWM4N_3 | I/O | PWM4 的捕获输入和脉冲输出负极 |
| 27           | 19           | P3.0    | I/O | 标准 IO 口           |
|              |              | RxD     | I   | 串口 1 的接收脚         |
|              |              | INT4    | I   | 外部中断 4            |
| 28           | 20           | P3.1    | I/O | 标准 IO 口           |
|              |              | TxD     | O   | 串口 1 的发送脚         |
| 29           | 21           | P3.2    | I/O | 标准 IO 口           |
|              |              | INT0    | I   | 外部中断 0            |
|              |              | SCLK_4  | I/O | SPI 的时钟脚          |
|              |              | SCL_4   | I/O | I2C 的时钟线          |
|              |              | PWMETI  | I   | PWM 外部触发输入脚       |
|              |              | PWMETI2 | I   | PWM 外部触发输入脚 2     |
| 30           | 22           | P3.3    | I/O | 标准 IO 口           |
|              |              | INT1    | I   | 外部中断 1            |
|              |              | MISO_4  | I/O | SPI 主机输入从机输出      |
|              |              | SDA_4   | I/O | I2C 接口的数据线        |
|              |              | PWM4N_4 | I/O | PWM4 的捕获输入和脉冲输出负极 |
|              |              | PWM7_2  | I/O | PWM7 的捕获输入和脉冲输出   |
| 31           | 23           | P3.4    | I/O | 标准 IO 口           |
|              |              | T0      | I   | 定时器 0 外部时钟输入      |
|              |              | T1CLKO  | O   | 定时器 1 时钟分频输出      |
|              |              | MOSI_4  | I/O | SPI 主机输出从机输入      |
|              |              | PWM4P_4 | I/O | PWM4 的捕获输入和脉冲输出正极 |
|              |              | PWM8_2  | I/O | PWM8 的捕获输入和脉冲输出   |
|              |              | CMPO    | O   | 比较器输出             |
| 32           | 24           | P5.0    | I/O | 标准 IO 口           |
|              |              | RxD3_2  | I   | 串口 3 的接收脚         |
|              |              | CMP+_2  | I   | 比较器正极输入           |
|              |              | COM0    | O   | LCD 驱动 COM 线      |
|              |              | TK8     | I   | 触摸按键              |
| 33           | 25           | P5.1    | I/O | 标准 IO 口           |
|              |              | TxD3_2  | O   | 串口 3 的发送脚         |
|              |              | CMP+_3  | I   | 比较器正极输入           |
|              |              | COM1    | O   | LCD 驱动 COM 线      |
|              |              | TK9     | I   | 触摸按键              |

| 编号           |              | 名称       | 类型  | 说明                |
|--------------|--------------|----------|-----|-------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |          |     |                   |
| 34           | 26           | P3.5     | I/O | 标准 IO 口           |
|              |              | T1       | I   | 定时器 1 外部时钟输入      |
|              |              | T0CLKO   | O   | 定时器 0 时钟分频输出      |
|              |              | SS_4     | I   | SPI 的从机选择脚（主机为输出） |
|              |              | PWMFLT   | I   | 增强 PWMA 的外部异常检测脚  |
|              |              | PWMFLT2  | I   | 增强 PWMB 的外部异常检测脚  |
|              |              | COM2     | O   | LCD 驱动 COM 线      |
| 35           | 27           | P3.6     | I/O | 标准 IO 口           |
|              |              | INT2     | I   | 外部中断 2            |
|              |              | RxD_2    | I   | 串口 1 的接收脚         |
|              |              | CMP-     | I   | 比较器负极输入           |
|              |              | COM3     | O   | LCD 驱动 COM 线      |
| 36           | 28           | P3.7     | I/O | 标准 IO 口           |
|              |              | INT3     | I   | 外部中断 3            |
|              |              | TxD_2    | O   | 串口 1 的发送脚         |
|              |              | CMP+     | I   | 比较器正极输入           |
|              |              | SEG31    | O   | LCD 驱动 SEG 线      |
| 37           |              | P7.0     | I/O | 标准 IO 口           |
|              |              | SEG35    | O   | LCD 驱动 SEG 线      |
| 38           |              | P7.1     | I/O | 标准 IO 口           |
|              |              | SEG34    | O   | LCD 驱动 SEG 线      |
| 39           |              | P7.2     | I/O | 标准 IO 口           |
|              |              | SEG33    | O   | LCD 驱动 SEG 线      |
| 40           |              | P7.3     | I/O | 标准 IO 口           |
|              |              | PWMETI_3 | I   | PWM 外部触发输入脚       |
|              |              | SEG32    | O   | LCD 驱动 SEG 线      |
| 41           | 29           | P4.1     | I/O | 标准 IO 口           |
|              |              | MISO_3   | I/O | SPI 主机输入从机输出      |
|              |              | CMPO_2   | O   | 比较器输出             |
|              |              | PWMETI_2 | I   | PWM 外部触发输入脚       |
|              |              | SEG30    | O   | LCD 驱动 SEG 线      |



| 编号           |              | 名称      | 类型  | 说明                |
|--------------|--------------|---------|-----|-------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |         |     |                   |
| 42           | 30           | P4.2    | I/O | 标准 IO 口           |
|              |              | WR      | O   | 外部总线的写信号线         |
|              |              | SEG29   | O   | LCD 驱动 SEG 线      |
| 43           | 31           | P4.3    | I/O | 标准 IO 口           |
|              |              | RxD_4   | I   | 串口 1 的接收脚         |
|              |              | SCLK_3  | I/O | SPI 的时钟脚          |
|              |              | SEG28   | O   | LCD 驱动 SEG 线      |
| 44           | 32           | P4.4    | I/O | 标准 IO 口           |
|              |              | RD      | O   | 外部总线的读信号线         |
|              |              | TxD_4   | O   | 串口 1 的发送脚         |
|              |              | SEG27   | O   | LCD 驱动 SEG 线      |
| 45           | 33           | P2.0    | I/O | 标准 IO 口           |
|              |              | A8      | O   | 地址总线              |
|              |              | PWM1P_2 | I/O | PWM1 的捕获输入和脉冲输出正极 |
|              |              | PWM5    | I/O | PWM5 的捕获输入和脉冲输出   |
|              |              | SEG26   | O   | LCD 驱动 SEG 线      |
| 46           | 34           | P2.1    | I/O | 标准 IO 口           |
|              |              | A9      | O   | 地址总线              |
|              |              | PWM1N_2 | I/O | PWM1 的捕获输入和脉冲输出负极 |
|              |              | PWM6    | I/O | PWM6 的捕获输入和脉冲输出   |
|              |              | SEG25   | O   | LCD 驱动 SEG 线      |
| 47           | 35           | P2.2    | I/O | 标准 IO 口           |
|              |              | A10     | O   | 地址总线              |
|              |              | SS_2    | I   | SPI 的从机选择脚（主机为输出） |
|              |              | PWM2P_2 | I/O | PWM2 的捕获输入和脉冲输出正极 |
|              |              | PWM7    | I/O | PWM7 的捕获输入和脉冲输出   |
|              |              | SEG24   | O   | LCD 驱动 SEG 线      |

| 编号           |              | 名称      | 类型  | 说明                |
|--------------|--------------|---------|-----|-------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |         |     |                   |
| 48           | 36           | P2.3    | I/O | 标准 IO 口           |
|              |              | A11     | O   | 地址总线              |
|              |              | MOSI_2  | I/O | SPI 主机输出从机输入      |
|              |              | PWM2N_2 | I/O | PWM2 的捕获输入和脉冲输出负极 |
|              |              | PWM8    | I/O | PWM8 的捕获输入和脉冲输出   |
|              |              | SEG23   | O   | LCD 驱动 SEG 线      |
| 49           | 37           | P2.4    | I/O | 标准 IO 口           |
|              |              | A12     | O   | 地址总线              |
|              |              | MISO_2  | I/O | SPI 主机输入从机输出      |
|              |              | SDA_2   | I/O | I2C 接口的数据线        |
|              |              | PWM3P_2 | I/O | PWM3 的捕获输入和脉冲输出正极 |
|              |              | SEG22   | O   | LCD 驱动 SEG 线      |
| 50           | 38           | P2.5    | I/O | 标准 IO 口           |
|              |              | A13     | O   | 地址总线              |
|              |              | SCLK_2  | I/O | SPI 的时钟脚          |
|              |              | SCL_2   | I/O | I2C 的时钟线          |
|              |              | PWM3N_2 | I/O | PWM3 的捕获输入和脉冲输出负极 |
|              |              | SEG21   | O   | LCD 驱动 SEG 线      |
| 51           | 39           | P2.6    | I/O | 标准 IO 口           |
|              |              | A14     | O   | 地址总线              |
|              |              | PWM4P_2 | I/O | PWM4 的捕获输入和脉冲输出正极 |
|              |              | SEG20   | O   | LCD 驱动 SEG 线      |
|              |              | TK6     | I   | 触摸按键              |

| 编号           |              | 名称      | 类型  | 说明                |
|--------------|--------------|---------|-----|-------------------|
| LQFP64/QFN64 | LQFP48/QFN48 |         |     |                   |
| 52           | 40           | P2.7    | I/O | 标准 IO 口           |
|              |              | A15     | O   | 地址总线              |
|              |              | PWM4N_2 | I/O | PWM4 的捕获输入和脉冲输出负极 |
|              |              | SEG19   | O   | LCD 驱动 SEG 线      |
|              |              | TK7     | I   | 触摸按键              |
| 53           |              | P7.4    | I/O | 标准 IO 口           |
|              |              | PWM5_4  | I/O | PWM5 的捕获输入和脉冲输出   |
|              |              | SEG3_2  | O   | LCD 驱动 SEG 线      |
| 54           |              | P7.5    | I/O | 标准 IO 口           |
|              |              | PWM6_4  | I/O | PWM6 的捕获输入和脉冲输出   |
|              |              | SEG2    | O   | LCD 驱动 SEG 线      |
| 55           |              | P7.6    | I/O | 标准 IO 口           |
|              |              | PWM7_4  | I/O | PWM7 的捕获输入和脉冲输出   |
|              |              | SDA_3   | I/O | I2C 接口的数据线        |
|              |              | SEG1_2  | O   | LCD 驱动 SEG 线      |
| 56           |              | P7.7    | I/O | 标准 IO 口           |
|              |              | PWM8_4  | I/O | PWM8 的捕获输入和脉冲输出   |
|              |              | SCL_3   | I/O | I2C 的时钟线          |
|              |              | SEG0_2  | O   | LCD 驱动 SEG 线      |
| 57           | 41           | P4.5    | I/O | 标准 IO 口           |
|              |              | ALE     | O   | 地址锁存信号            |
|              |              | SEG18   | O   | LCD 驱动 SEG 线      |
| 58           | 42           | P4.6    | I/O | 标准 IO 口           |
|              |              | RxD2_2  | I   | 串口 2 的接收脚         |
|              |              | SEG17   | O   | LCD 驱动 SEG 线      |
| 59           | 43           | P0.0    | I/O | 标准 IO 口           |
|              |              | AD0     | I/O | 地址/数据总线           |
|              |              | ADC8    | I   | ADC 模拟输入通道 8      |
|              |              | RxD3    | I   | 串口 3 的接收脚         |
|              |              | PWM5_3  | I/O | PWM5 的捕获输入和脉冲输出   |
|              |              | T3_2    | I   | 定时器 3 外部时钟输入      |
|              |              | SEG16   | O   | LCD 驱动 SEG 线      |
|              |              | TK12    | I   | 触摸按键              |

| 编号           |              | 名称       | 类型  | 说明              |
|--------------|--------------|----------|-----|-----------------|
| LQFP64/QFN64 | LQFP48/QFN48 |          |     |                 |
| 60           | 44           | P0.1     | I/O | 标准 IO 口         |
|              |              | AD1      | I/O | 地址/数据总线         |
|              |              | ADC9     | I   | ADC 模拟输入通道 9    |
|              |              | TxD3     | O   | 串口 3 的发送脚       |
|              |              | PWM6_3   | I/O | PWM6 的捕获输入和脉冲输出 |
|              |              | T3CLKO_2 | O   | 定时器 3 时钟分频输出    |
|              |              | SEG15    | O   | LCD 驱动 SEG 线    |
|              |              | TK13     | I   | 触摸按键            |
| 61           | 45           | P0.2     | I/O | 标准 IO 口         |
|              |              | AD2      | I/O | 地址/数据总线         |
|              |              | ADC10    | I   | ADC 模拟输入通道 10   |
|              |              | RxD4     | I   | 串口 4 的接收脚       |
|              |              | PWM7_3   | I/O | PWM7 的捕获输入和脉冲输出 |
|              |              | T4_2     | I   | 定时器 4 外部时钟输入    |
|              |              | SEG14    | O   | LCD 驱动 SEG 线    |
|              |              | TK14     | I   | 触摸按键            |
| 62           | 46           | P0.3     | I/O | 标准 IO 口         |
|              |              | AD3      | I/O | 地址/数据总线         |
|              |              | ADC11    | I   | ADC 模拟输入通道 11   |
|              |              | TxD4     | O   | 串口 4 的发送脚       |
|              |              | PWM8_3   | I/O | PWM8 的捕获输入和脉冲输出 |
|              |              | T4CLKO_2 | O   | 定时器 4 时钟分频输出    |
|              |              | SEG13    | O   | LCD 驱动 SEG 线    |
|              |              | TK15     | I   | 触摸按键            |
| 63           | 47           | P0.4     | I/O | 标准 IO 口         |
|              |              | AD4      | I/O | 地址/数据总线         |
|              |              | ADC12    | I   | ADC 模拟输入通道 12   |
|              |              | T3       | I   | 定时器 3 外部时钟输入    |
|              |              | SEG12    | O   | LCD 驱动 SEG 线    |
| 64           | 48           | P5.2     | I/O | 标准 IO 口         |
|              |              | RxD4_2   | I   | 串口 4 的接收脚       |
|              |              | SEG11    | O   | LCD 驱动 SEG 线    |
|              |              | TK10     | I   | 触摸按键            |

## 2.8 STC8H1K08T-36I-TSSOP20/QFN20

### 2.8.1 特性及价格(有 16 位硬件乘除法器 MDU16, 准 16 位单片机)

- 选型价格(不需要外部晶振、不需要外部复位, 12 位 ADC, 15 通道)

| 价格及封装                                     | QFN20   |    | 供货信息 |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---------|----|------|----|---|----|----|---|---|---|---|---|---|---|---|---|---|---|------|---|---|---|-----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|   | TSSOP20 |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 本身就可在线仿真                                  |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 支持软件模拟硬件 USB 直接下载                         |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 支持 RS485 下载                               |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 可设置下次更新程序需口令                              |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 程序加密后传输（防拦截）                              |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 可对外输出时钟及复位                                |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 内部高精度时钟（36MHz 以下可调）追频                     |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 内部高可靠复位（可选复位门檻电压）                         |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 看门狗 复位定时器                                 |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 内部低压检测中断并可掉电唤醒                            |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 比较器（可当 1 路 A/D，可作外部掉电检测）                  |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| DMA 15 路高速 ADC（8 路 PWM 可当 8 路 D/A 使用）     |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 掉电唤醒专用定时器                                 |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 16 位高级 PWM 定时器 互补对称死区控制                   |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 定时器计数器（T0/T1/T2 外部管脚也可掉电唤醒）               |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MDU16 硬件 16 位乘除法器                         |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| I <sup>2</sup> C 并可掉电唤醒（暂无 DMA）           |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| DMA SPI 并可掉电唤醒                            |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| RTC 实时时钟                                  |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 触摸按键                                      |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| DMA 串口并可掉电唤醒                              |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 所有的 I/O 口均支持中断并可掉电唤醒                      |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 传统 I/O 中断（INT0/INT1/INT2/INT3/INT4）并可掉电唤醒 |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| I/O 口最多数量                                 |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| EEPROM 10 万次 字节                           |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 强大的双 DPTIR 可增可减                           |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| xdata <sup>*</sup> 内部大容量扩展 SRAM 字节        |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| idata <sup>*</sup> 内部传统 8051 RAM 字节       |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Flash 程序存储器 10 万次 字节                      |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 工作电压（V）                                   |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 单片机型号                                     |         |    |      |    |   |    |    |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| STC8H1K08T                                | 1.9-5.5 | 8K | 256  | 1K | 2 | 4K | 16 | 有 | 有 | 2 | 有 | 有 | 有 | 有 | 有 | 3 | 8 | 有 | 12 位 | 有 | 有 | 有 | 4 级 | 有 | 是 | 有 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 | 是 |

➤ 内核

- ✓ 超高速 8051 内核 (1T), 比传统 8051 约快 12 倍以上
- ✓ 指令代码完全兼容传统 8051
- ✓ 29 个中断源, 4 级中断优先级
- ✓ 支持在线仿真

➤ 工作电压

- ✓ 1.9V~5.5V

➤ 工作温度

- ✓ -40℃~85℃ (芯片为 -40℃~125℃ 制程, 超温度范围应用请参考电气特性章节说明)

➤ Flash 存储器

- ✓ 最大 17K 字节 FLASH 程序存储器 (ROM), 用于存储用户代码
- ✓ 支持用户配置 EEPROM 大小, 512 字节单页擦除, 擦写次数可达 10 万次以上
- ✓ 支持在系统编程方式 (ISP) 更新用户应用程序, 无需专用编程器
- ✓ 支持单芯片仿真, 无需专用仿真器, 理论断点个数无限制

➤ SRAM

- ✓ 128 字节内部直接访问 RAM (DATA, C 语言程序中使用 data 关键字进行声明)
- ✓ 128 字节内部间接访问 RAM (IDATA, C 语言程序中使用 idata 关键字进行声明)
- ✓ 1024 字节内部扩展 RAM (内部 XDATA, C 语言程序中使用 xdata 关键字进行声明)

➤ 时钟控制

- ✓ 内部高精度 IRC (4MHz~36MHz, ISP 编程时选择或手动输入, 还可以用户软件分频到较低的频率工作,

如 100KHz)

- ✦ 误差±0.3% (常温下 25℃)
  - ✦ -1.35%~+1.30%温漂 (全温度范围, -40℃~85℃)
  - ✦ -0.76%~+0.98%温漂 (温度范围, -20℃~65℃)
  - ✓ 内部 32KHz 低速 IRC (误差较大)
  - ✓ 外部晶振 (4MHz~36MHz) 和外部时钟
- 用户可自由选择上面的 3 种时钟源

## ➤ 复位

### ✓ 硬件复位

- ✦ 上电复位, 实测电压值为 1.69V~1.82V。(在芯片未使能低压复位功能时有效)

上电复位电压由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到上电复位的下限门槛电压时, 芯片处于复位状态; 当电压从 0V 上升到上电复位的上限门槛电压时, 芯片解除复位状态。

- ✦ 复位脚复位, 出厂时 P5.4 默认为 I/O 口, ISP 下载时可将 P5.4 管脚设置为复位脚 (注意: 当设置 P5.4 管脚为复位脚时, 复位电平为低电平)
- ✦ 看门狗溢出复位
- ✦ 低压检测复位, 提供 4 级低压检测电压: 1.9V、2.3V、2.8V、3.7V。

每级低压检测电压都是由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到低压检测的下限门槛电压时, 低压检测生效; 当电压从 0V 上升到低压检测的上限门槛电压时, 低压检测生效。

### ✓ 软件复位

- ✦ 软件方式写复位触发寄存器

## ➤ 中断

- ✓ 提供 29 个中断源: INT0 (支持上升沿和下降沿中断)、INT1 (支持上升沿和下降沿中断)、INT2 (只支持下降沿中断)、INT3 (只支持下降沿中断)、INT4 (只支持下降沿中断)、定时器 0、定时器 1、定时器 2、串口 1、串口 2、ADC 模数转换、LVD 低压检测、SPI、I<sup>2</sup>C、比较器、PWMA、PWMB、RTC、TKS、P1 口中断、P3 口中断、P5 口中断、串口 1 的 DMA 接收和发送中断、串口 2 的 DMA 接收和发送中断、SPI 的 DMA 中断、ADC 的 DMA 中断以及存储器到存储器的 DMA 中断。
- ✓ 提供 4 级中断优先级
- ✓ 时钟停振模式下可以唤醒的中断: INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、RXD(P3.0/P3.6/P1.6)、RXD2(P1.0)、I2C\_SDA(P1.4/P3.3)、SPI\_SS(P5.4/ P3.5)以及比较器中断、低压检测中断、掉电唤醒定时器唤醒以及所有端口的 I/O 中断。

## ➤ 数字外设

- ✓ 3 个 16 位定时器: 定时器 0、定时器 1、定时器 2, 其中定时器 0 的模式 3 具有 NMI (不可屏蔽中断) 功能, 定时器 0 和定时器 1 的模式 0 为 16 位自动重载模式
- ✓ 2 个高速串口: 串口 1、串口 2, 波特率时钟源最快可为 FOSC/4
- ✓ 8 路/2 组高级 PWM, 可实现带死区的控制信号, 并支持外部异常检测功能, 另外还支持 16 位定时器、8 个外部中断、8 路外部捕获测量脉宽等功能
- ✓ SPI: 支持主机模式和从机模式以及主机/从机自动切换
- ✓ I<sup>2</sup>C: 支持主机模式和从机模式
- ✓ MDU16: 硬件 16 位乘法器 (支持 32 位除以 16 位、16 位除以 16 位、16 位乘 16 位、数据移位以及数据规格化等运算)
- ✓ RTC: 支持年、月、日、时、分、秒、次秒 (1/128 秒), 并支持时钟中断和一组闹钟
- ✓ I/O 口中断: 所有的 I/O 均支持中断, 每组 I/O 中断有独立的中断入口地址, 所有的 I/O 中断可支持 4 种

中断模式: 高电平中断、低电平中断、上升沿中断、下降沿中断。提供 4 级中断优先级并支持掉电唤醒功能。

- ✓ DMA: 支持 Memory-To-Memory、SPI、UART1TX/UART1RX、UART2TX/UART2RX、ADC (自动计算多次 ADC 结果的平均值)

#### ➤ 模拟外设

- ✓ 超高速 ADC, 支持 12 位高精度 15 通道 (通道 0~通道 14) 的模数转换, 速度最快能达到 800K (每秒进行 80 万次 ADC 转换)
- ✓ ADC 的通道 15 用于测试内部 1.19V 参考信号源 (芯片在出厂时, 内部参考信号源已调整为 1.19V)
- ✓ 比较器, 一组比较器 (比较器的正端可选择 CMP+端口和所有的 ADC 输入端口, 所以比较器可当作多路比较器进行分时复用)
- ✓ 触摸按键: 最多支持 16 个触摸按键; 每个触摸按键能够独立使能; 内部参考电压 4 级可调; 灵活的充放电时间设置以及内部工作频率设置; 支持低功耗触摸唤醒
- ✓ DAC: 8 路高级 PWM 定时器可当 8 路 DAC 使用

#### ➤ GPIO

- ✓ 最多可达 16 个 GPIO: P1.0~P1.7 (无 P1.2)、P3.0~P3.7、P5.4
- ✓ 所有的 GPIO 均支持如下 4 种模式: 准双向口模式、强推挽输出模式、开漏输出模式、高阻输入模式
- ✓ 除 P3.0 和 P3.1 外, 其余所有 IO 口上电后的状态均为高阻输入状态, 用户在使用 IO 口时必须先设置 IO 口模式。另外每个 I/O 均可独立使能内部 4K 上拉电阻

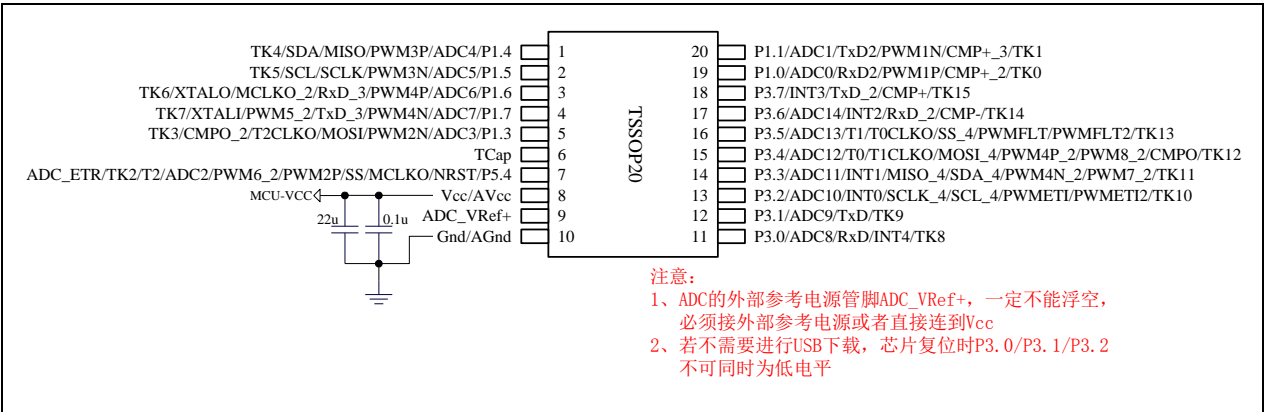
#### ➤ 封装

- ✓ TSSOP20、QFN20

#### ✓ 特别说明

- ✓ 本系列芯片有 CHIPID 功能
- ✓ 本系列芯片有所有的 I/O 口中断功能, 有 4 级中断优先级, 可掉电唤醒
- ✓ 本系列芯片的比较器为 4P+2N 版本

## 2.8.2 管脚图，最小系统



建议在 Vcc 和 Gnd 之间就近加上电源去耦电容 22uF 和 0.1uF，可去除电源线噪声，提高抗干扰能力



## 2.8.3 管脚说明

| 编号      |       | 名称      | 类型  | 说明                |
|---------|-------|---------|-----|-------------------|
| TSSOP20 | QFN20 |         |     |                   |
| 1       | 18    | P1.4    | I/O | 标准 IO 口           |
|         |       | ADC4    | I   | ADC 模拟输入通道 4      |
|         |       | PWM3P   | I/O | PWM3 的捕获输入和脉冲输出正极 |
|         |       | MISO    | I/O | SPI 主机输入从机输出      |
|         |       | SDA     | I/O | I2C 接口的数据线        |
|         |       | TK4     | I   | 触摸按键              |
| 2       | 19    | P1.5    | I/O | 标准 IO 口           |
|         |       | ADC5    | I   | ADC 模拟输入通道 5      |
|         |       | PWM3N   | I/O | PWM3 的捕获输入和脉冲输出负极 |
|         |       | SCLK    | I/O | SPI 的时钟脚          |
|         |       | SCL     | I/O | I2C 的时钟线          |
|         |       | TK5     | I   | 触摸按键              |
| 3       | 20    | P1.6    | I/O | 标准 IO 口           |
|         |       | ADC6    | I   | ADC 模拟输入通道 6      |
|         |       | RxD_3   | I   | 串口 1 的接收脚         |
|         |       | PWM4P   | I/O | PWM4 的捕获输入和脉冲输出正极 |
|         |       | MCLKO_2 | O   | 主时钟分频输出           |
|         |       | XTALO   | O   | 外部晶振的输出脚          |
| 4       | 1     | TK6     | I   | 触摸按键              |
|         |       | P1.7    | I/O | 标准 IO 口           |
|         |       | ADC7    | I   | ADC 模拟输入通道 7      |
|         |       | TxD_3   | O   | 串口 1 的发送脚         |
|         |       | PWM4N   | I/O | PWM4 的捕获输入和脉冲输出负极 |
|         |       | PWM5_2  | I/O | PWM5 的捕获输入和脉冲输出   |
|         |       | XTALI   | I   | 外部晶振/外部时钟的输入脚     |
| 5       | 2     | TK7     | I   | 触摸按键              |
|         |       | P1.3    | I/O | 标准 IO 口           |
|         |       | ADC3    | I   | ADC 模拟输入通道 3      |
|         |       | MOSI    | I/O | SPI 主机输出从机输入      |
|         |       | PWM2N   | I/O | PWM2 的捕获输入和脉冲输出负极 |
|         |       | T2CLKO  | O   | 定时器 2 时钟分频输出      |
|         |       | CMPO_2  | O   | 比较器输出             |
| 6       | 3     | TK3     | I   | 触摸按键              |
|         |       | TCAP    | I   | 触摸按键充放电电容         |

| 编号      |       | 名称        | 类型  | 说明                                  |
|---------|-------|-----------|-----|-------------------------------------|
| TSSOP20 | QFN20 |           |     |                                     |
| 7       | 4     | P5.4      | I/O | 标准 IO 口                             |
|         |       | NRST      | I   | 复位引脚 (低电平复位)                        |
|         |       | MCLKO     | O   | 主时钟分频输出                             |
|         |       | SS        | I   | SPI 的从机选择脚 (主机为输出)                  |
|         |       | PWM2P     | I/O | PWM2 的捕获输入和脉冲输出正极                   |
|         |       | PWM6_2    | I/O | PWM6 的捕获输入和脉冲输出                     |
|         |       | T2        | I   | 定时器 2 外部时钟输入                        |
|         |       | ADC2      | I   | ADC 模拟输入通道 2                        |
|         |       | TK2       | I   | 触摸按键                                |
|         |       | ADC_ETR   | I   | ADC 外部触发脚                           |
| 8       | 5     | Vcc       | Vcc | 电源脚                                 |
|         |       | AVcc      | Vcc | ADC 电源脚                             |
| 9       | 6     | ADC_VRef+ | I   | ADC 外部参考电压源输入脚, 要求不高时可直接接 MCU 的 VCC |
| 10      | 7     | Gnd       | Gnd | 地线                                  |
|         |       | AGnd      | Gnd | ADC 地线                              |
| 11      | 8     | P3.0      | I/O | 标准 IO 口                             |
|         |       | RxD       | I   | 串口 1 的接收脚                           |
|         |       | INT4      | I   | 外部中断 4                              |
|         |       | ADC8      | I   | ADC 模拟输入通道 8                        |
|         |       | TK8       | I   | 触摸按键                                |
| 12      | 9     | P3.1      | I/O | 标准 IO 口                             |
|         |       | TxD       | O   | 串口 1 的发送脚                           |
|         |       | ADC9      | I   | ADC 模拟输入通道 9                        |
|         |       | TK9       | I   | 触摸按键                                |
| 13      | 10    | P3.2      | I/O | 标准 IO 口                             |
|         |       | INT0      | I   | 外部中断 0                              |
|         |       | SCLK_4    | I/O | SPI 的时钟脚                            |
|         |       | SCL_4     | I/O | I2C 的时钟线                            |
|         |       | PWMETI    | I   | PWM 外部触发输入脚                         |
|         |       | PWMETI2   | I   | PWM 外部触发输入脚 2                       |
|         |       | ADC10     | I   | ADC 模拟输入通道 10                       |
|         |       | TK10      | I   | 触摸按键                                |

| 编号      |       | 名称      | 类型  | 说明                |
|---------|-------|---------|-----|-------------------|
| TSSOP20 | QFN20 |         |     |                   |
| 14      | 11    | P3.3    | I/O | 标准 IO 口           |
|         |       | INT1    | I   | 外部中断 1            |
|         |       | MISO_4  | I/O | SPI 主机输入从机输出      |
|         |       | SDA_4   | I/O | I2C 接口的数据线        |
|         |       | PWM4N_4 | I/O | PWM4 的捕获输入和脉冲输出负极 |
|         |       | PWM7_2  | I/O | PWM7 的捕获输入和脉冲输出   |
|         |       | ADC11   | I   | ADC 模拟输入通道 11     |
|         |       | TK11    | I   | 触摸按键              |
| 15      | 12    | P3.4    | I/O | 标准 IO 口           |
|         |       | T0      | I   | 定时器 0 外部时钟输入      |
|         |       | T1CLKO  | O   | 定时器 1 时钟分频输出      |
|         |       | MOSI_4  | I/O | SPI 主机输出从机输入      |
|         |       | PWM4P_4 | I/O | PWM4 的捕获输入和脉冲输出正极 |
|         |       | PWM8_2  | I/O | PWM8 的捕获输入和脉冲输出   |
|         |       | CMPO    | O   | 比较器输出             |
|         |       | ADC12   | I   | ADC 模拟输入通道 12     |
|         |       | TK12    | I   | 触摸按键              |
| 16      | 13    | P3.5    | I/O | 标准 IO 口           |
|         |       | T1      | I   | 定时器 1 外部时钟输入      |
|         |       | T0CLKO  | O   | 定时器 0 时钟分频输出      |
|         |       | SS_4    | I   | SPI 的从机选择脚（主机为输出） |
|         |       | PWMFLT  | I   | 增强 PWMA 的外部异常检测脚  |
|         |       | PWMFLT2 | I   | 增强 PWMB 的外部异常检测脚  |
|         |       | ADC13   | I   | ADC 模拟输入通道 13     |
|         |       | TK13    | I   | 触摸按键              |
| 17      | 14    | P3.6    | I/O | 标准 IO 口           |
|         |       | INT2    | I   | 外部中断 2            |
|         |       | RxD_2   | I   | 串口 1 的接收脚         |
|         |       | CMP-    | I   | 比较器负极输入           |
|         |       | ADC14   | I   | ADC 模拟输入通道 14     |
|         |       | TK14    | I   | 触摸按键              |
| 18      | 15    | P3.7    | I/O | 标准 IO 口           |
|         |       | INT3    | I   | 外部中断 3            |
|         |       | TxD_2   | O   | 串口 1 的发送脚         |
|         |       | CMP+    | I   | 比较器正极输入           |
|         |       | TK15    | I   | 触摸按键              |

| 编号      |       | 名称     | 类型  | 说明                |
|---------|-------|--------|-----|-------------------|
| TSSOP20 | QFN20 |        |     |                   |
| 19      | 16    | P1.0   | I/O | 标准 IO 口           |
|         |       | ADC0   | I   | ADC 模拟输入通道 0      |
|         |       | PWM1P  | I/O | PWM1 的捕获输入和脉冲输出正极 |
|         |       | RxD2   | I   | 串口 2 的接收脚         |
|         |       | TK0    | I   | 触摸按键              |
|         |       | CMP+_2 | I   | 比较器正极输入           |
| 20      | 17    | P1.1   | I/O | 标准 IO 口           |
|         |       | ADC1   | I   | ADC 模拟输入通道 1      |
|         |       | PWM1N  | I/O | PWM1 的捕获输入和脉冲输出负极 |
|         |       | TxD2   | I   | 串口 2 的发送脚         |
|         |       | TK1    | I   | 触摸按键              |
|         |       | CMP+_3 | I   | 比较器正极输入           |

# 3 功能脚切换

STC8H 系列单片机的特殊外设串口、SPI、PWM、I<sup>2</sup>C 以及总线控制脚可以在多个 I/O 直接进行切换, 以实现一个外设当作多个设备进行分时复用。

## 3.1 功能脚切换相关寄存器

| 符号    | 描述          | 地址  | 位地址与符号    |    |            |    |            |      |      |      | 复位值       |
|-------|-------------|-----|-----------|----|------------|----|------------|------|------|------|-----------|
|       |             |     | B7        | B6 | B5         | B4 | B3         | B2   | B1   | B0   |           |
| P_SW1 | 外设端口切换寄存器 1 | A2H | S1_S[1:0] |    | -          | -  | SPI_S[1:0] |      | 0    | -    | nnxx,000x |
| P_SW2 | 外设端口切换寄存器 2 | BAH | EAXFR     | -  | I2C_S[1:0] |    | CMPO_S     | S4_S | S3_S | S2_S | 0x00,0000 |

| 符号         | 描述               | 地址    | 位地址与符号    |               |           |    |           |        |              |         | 复位值       |
|------------|------------------|-------|-----------|---------------|-----------|----|-----------|--------|--------------|---------|-----------|
|            |                  |       | B7        | B6            | B5        | B4 | B3        | B2     | B1           | B0      |           |
| MCLKOCR    | 主时钟输出控制寄存器       | FE05H | MCLKO_S   | MCLKODIV[6:0] |           |    |           |        |              |         | 0000,0000 |
| PWMA_PS    | PWMA 切换寄存器       | FEB2H | C4PS[1:0] |               | C3PS[1:0] |    | C2PS[1:0] |        | C1PS[1:0]    |         | 0000,0000 |
| PWMB_PS    | PWMB 切换寄存器       | FEB6H | C8PS[1:0] |               | C7PS[1:0] |    | C6PS[1:0] |        | C5PS[1:0]    |         | 0000,0000 |
| PWMA_ETRPS | PWMA 的 ETR 选择寄存器 | FEB0H |           |               |           |    |           | BRKAPS | ETRAPPS[1:0] |         | xxxx,x000 |
| PWMB_ETRPS | PWMB 的 ETR 选择寄存器 | FEB4H |           |               |           |    |           | BRKBPS | ETRBPS[1:0]  |         | xxxx,x000 |
| T3T4PIN    | T3/T4 选择寄存器      | FEACH | -         | -             | -         | -  | -         | -      | -            | T3T4SEL | xxxx,xxx0 |

### 3.1.1 外设端口切换控制寄存器 1 (P\_SW1), 串口 1、SPI 切换

| 符号    | 地址  | B7        | B6 | B5 | B4 | B3         | B2 | B1 | B0 |
|-------|-----|-----------|----|----|----|------------|----|----|----|
| P_SW1 | A2H | S1_S[1:0] |    | -  | -  | SPI_S[1:0] |    | 0  | -  |

S1\_S[1:0]: 串口 1 功能脚选择位

| S1_S[1:0] | RxD  | TxD  |
|-----------|------|------|
| 00        | P3.0 | P3.1 |
| 01        | P3.6 | P3.7 |
| 10        | P1.6 | P1.7 |
| 11        | P4.3 | P4.4 |

SPI\_S[1:0]: SPI 功能脚选择位

| SPI_S[1:0] | SS                                      | MOSI | MISO | SCLK |
|------------|---|------|------|------|
| 00         | <a href="#">P1.2/P5.4<sup>[1]</sup></a> | P1.3 | P1.4 | P1.5 |
| 01         | P2.2                                    | P2.3 | P2.4 | P2.5 |
| 10         | P5.4                                    | P4.0 | P4.1 | P4.3 |
| 11         | P3.5                                    | P3.4 | P3.3 | P3.2 |

注<sup>[1]</sup> : 对于部分没有 P1.2 口的单片机型号, 此功能在 P5.4 口上

### 3.1.2 外设端口切换控制寄存器 2 (P\_SW2), 串口 2/3/4、I2C、比较器输出切换

| 符号    | 地址  | B7    | B6 | B5         | B4 | B3     | B2   | B1   | B0   |
|-------|-----|-------|----|------------|----|--------|------|------|------|
| P_SW2 | BAH | EAXFR | -  | I2C_S[1:0] |    | CMPO_S | S4_S | S3_S | S2_S |

EAXFR: 扩展 RAM 区特殊功能寄存器 (XFR) 访问控制寄存器

0: 禁止访问 XFR

1: 使能访问 XFR。

**当需要访问 XFR 时, 必须先将 EAXFR 置 1, 才能对 XFR 进行正常的读写**

I2C\_S[1:0]: I<sup>2</sup>C 功能脚选择位

| I2C_S[1:0] | SCL  | SDA  |
|------------|------|------|
| 00         | P1.5 | P1.4 |
| 01         | P2.5 | P2.4 |
| 10         | P7.7 | P7.6 |
| 11         | P3.2 | P3.3 |

CMPO\_S: 比较器输出脚选择位

| CMPO_S | CMPO |
|--------|------|
| 0      | P3.4 |
| 1      | P4.1 |

S4\_S: 串口 4 功能脚选择位

| S4_S | RxD4 | TxD4 |
|------|------|------|
| 0    | P0.2 | P0.3 |
| 1    | P5.2 | P5.3 |

S3\_S: 串口 3 功能脚选择位

| S3_S | RxD3 | TxD3 |
|------|------|------|
| 0    | P0.0 | P0.1 |
| 1    | P5.0 | P5.1 |

S2\_S: 串口 2 功能脚选择位

| S2_S | RxD2 | TxD2 |
|------|------|------|
| 0    | P1.0 | P1.1 |
| 1    | P4.6 | P4.7 |

### 3.1.3 时钟选择寄存器 (MCLKOCR)

| 符号      | 地址    | B7      | B6            | B5 | B4 | B3 | B2 | B1 | B0 |
|---------|-------|---------|---------------|----|----|----|----|----|----|
| MCLKOCR | FE05H | MCLKO_S | MCLKODIV[6:0] |    |    |    |    |    |    |

MCLKO\_S: 主时钟输出脚选择位

| MCLKO_S | MCLKO |
|---------|-------|
| 0       | P5.4  |
| 1       | P1.6  |

### 3.1.4 T3/T4 选择寄存器 (T3T4PIN)

| 符号 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----|----|----|----|----|----|----|----|----|----|
|----|----|----|----|----|----|----|----|----|----|

|         |       |   |   |   |   |   |   |   |         |
|---------|-------|---|---|---|---|---|---|---|---------|
| T3T4PIN | FEACH | - | - | - | - | - | - | - | T3T4SEL |
|---------|-------|---|---|---|---|---|---|---|---------|

T3T4SEL: T3/T3CLKO/T4/T4CLKO 脚选择位

| T3T4SEL | T3   | T3CLKO | T4   | T4CLKO |
|---------|------|--------|------|--------|
| 0       | P0.4 | P0.5   | P0.6 | P0.7   |
| 1       | P0.0 | P0.1   | P0.2 | P0.3   |

### 3.1.5 高级 PWM 选择寄存器 (PWMx\_PS)

| 符号      | 地址    | B7        | B6 | B5        | B4 | B3        | B2 | B1        | B0 |
|---------|-------|-----------|----|-----------|----|-----------|----|-----------|----|
| PWMA_PS | FEB2H | C4PS[1:0] |    | C3PS[1:0] |    | C2PS[1:0] |    | C1PS[1:0] |    |
| PWMB_PS | FEB6H | C8PS[1:0] |    | C7PS[1:0] |    | C6PS[1:0] |    | C5PS[1:0] |    |

C1PS[1:0]: 高级 PWM 通道 1 输出脚选择位

| C1PS[1:0] | PWM1P | PWM1N |
|-----------|-------|-------|
| 00        | P1.0  | P1.1  |
| 01        | P2.0  | P2.1  |
| 10        | P6.0  | P6.1  |
| 11        | -     | -     |

C2PS[1:0]: 高级 PWM 通道 2 输出脚选择位

| C2PS[1:0] | PWM2P                    | PWM2N |
|-----------|--------------------------|-------|
| 00        | P1.2/P5.4 <sup>[1]</sup> | P1.3  |
| 01        | P2.2                     | P2.3  |
| 10        | P6.2                     | P6.3  |
| 11        | -                        | -     |

注<sup>[1]</sup>: 对于部分没有 P1.2 口的单片机型号, 此功能在 P5.4 口上

C3PS[1:0]: 高级 PWM 通道 3 输出脚选择位

| C3PS[1:0] | PWM3P | PWM3N |
|-----------|-------|-------|
| 00        | P1.4  | P1.5  |
| 01        | P2.4  | P2.5  |
| 10        | P6.4  | P6.5  |
| 11        | -     | -     |

C4PS[1:0]: 高级 PWM 通道 4 输出脚选择位

| C4PS[1:0] | PWM4P | PWM4N |
|-----------|-------|-------|
| 00        | P1.6  | P1.7  |
| 01        | P2.6  | P2.7  |
| 10        | P6.6  | P6.7  |
| 11        | P3.4  | P3.3  |



C5PS[1:0]: 高级 PWM 通道 5 输出脚选择位

| C5PS[1:0] | PWM5 |
|-----------|------|
| 00        | P2.0 |
| 01        | P1.7 |
| 10        | P0.0 |
| 11        | P7.4 |

C6PS[1:0]: 高级 PWM 通道 6 输出脚选择位

| C6PS[1:0] | PWM6 |
|-----------|------|
| 00        | P2.1 |
| 01        | P5.4 |
| 10        | P0.1 |
| 11        | P7.5 |

C7PS[1:0]: 高级 PWM 通道 7 输出脚选择位

| C7PS[1:0] | PWM7 |
|-----------|------|
| 00        | P2.2 |
| 01        | P3.3 |
| 10        | P0.2 |
| 11        | P7.6 |

C8PS[1:0]: 高级 PWM 通道 8 输出脚选择位

| C8PS[1:0] | PWM8 |
|-----------|------|
| 00        | P2.3 |
| 01        | P3.4 |
| 10        | P0.3 |
| 11        | P7.7 |

### 3.1.6 高级 PWM 功能脚选择寄存器 (PWMx\_ETRPS)

| 符号         | 地址    | B7 | B6 | B5 | B4 | B3 | B2     | B1          | B0 |
|------------|-------|----|----|----|----|----|--------|-------------|----|
| PWMA_ETRPS | FEB0H |    |    |    |    |    | BRKAPS | ETRAPS[1:0] |    |
| PWMB_ETRPS | FEB4H |    |    |    |    |    | BRKBPS | ETRBPS[1:0] |    |

ETRAPS[1:0]: 高级 PWMA 的外部触发脚 ERI 选择位

| ETRAPS [1:0] | PWMETI |
|--------------|--------|
| 00           | P3.2   |
| 01           | P4.1   |
| 10           | P7.3   |
| 11           | -      |

ETRBPS[1:0]: 高级 PWMB 的外部触发脚 ERIB 选择位

| ETRBPS [1:0] | PWMETI2 |
|--------------|---------|
| 00           | P3.2    |
| 01           | P0.6    |
| 10           | -       |
| 11           | -       |

BRKAPS: 高级 PWMA 的刹车脚 PWMFLT 选择位

| BRKAPS | PWMFLT |
|--------|--------|
| 0      | P3.5   |
| 1      | 比较器的输出 |

BRKBPS: 高级 PWMB 的刹车脚 PWMFLT2 选择位

| BRKBPS | PWMFLT2 |
|--------|---------|
| 0      | P3.5    |
| 1      | 比较器的输出  |

## 3.2 范例程序

### 3.2.1 串口 1 切换

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"

void main()
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW1 = 0x00;                //RXD/P3.0, TXD/P3.1
    // P_SW1 = 0x40;              //RXD_2/P3.6, TXD_2/P3.7
    // P_SW1 = 0x80;              //RXD_3/P1.6, TXD_3/P1.7
    // P_SW1 = 0xc0;              //RXD_4/P4.3, TXD_4/P4.4

    while (1);
}
```

#### 汇编代码

```
;测试工作频率为 11.0592MHz

P_SW1      DATA      0A2H
P_SW2      DATA      0BAH

P1M1       DATA      091H
P1M0       DATA      092H
P0M1       DATA      093H
P0M0       DATA      094H
P2M1       DATA      095H
P2M0       DATA      096H
P3M1       DATA      0B1H
P3M0       DATA      0B2H
P4M1       DATA      0B3H
P4M0       DATA      0B4H
P5M1       DATA      0C9H
P5M0       DATA      0CAH

            ORG         0000H
```

```

        LJMP      MAIN

MAIN:
        ORG      0100H

        MOV      SP, #5FH
        ORL      P_SW2, #80H           ;使能访问 XFR

        MOV      P0M0, #00H
        MOV      P0M1, #00H
        MOV      P1M0, #00H
        MOV      P1M1, #00H
        MOV      P2M0, #00H
        MOV      P2M1, #00H
        MOV      P3M0, #00H
        MOV      P3M1, #00H
        MOV      P4M0, #00H
        MOV      P4M1, #00H
        MOV      P5M0, #00H
        MOV      P5M1, #00H

        MOV      P_SW1, #00H           ;RXD/P3.0, TXD/P3.1
;      MOV      P_SW1, #40H           ;RXD_2/P3.6, TXD_2/P3.7
;      MOV      P_SW1, #80H           ;RXD_3/P1.6, TXD_3/P1.7
;      MOV      P_SW1, #0C0H          ;RXD_4/P4.3, TXD_4/P4.4

        SJMP     $

END
    
```

### 3.2.2 串口 2 切换

#### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"

void main()
{
    P_SW2 /= 0x80;           //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 /= 0x00;           //RXD2/P1.0, TXD2/P1.1
//    P_SW2 /= 0x01;         //RXD2_2/P4.6, TXD2_2/P4.7
    
```

```
while (1);  
}
```

汇编代码

```
;测试工作频率为 11.0592MHz
```

```
P_SW2      DATA      0BAH  
  
P1M1      DATA      091H  
P1M0      DATA      092H  
P0M1      DATA      093H  
P0M0      DATA      094H  
P2M1      DATA      095H  
P2M0      DATA      096H  
P3M1      DATA      0B1H  
P3M0      DATA      0B2H  
P4M1      DATA      0B3H  
P4M0      DATA      0B4H  
P5M1      DATA      0C9H  
P5M0      DATA      0CAH  
  
          ORG          0000H  
          LJMP         MAIN  
  
MAIN:     ORG          0100H  
  
          MOV          SP, #5FH  
          ORL          P_SW2, #80H      ;使能访问 XFR  
  
          MOV          P0M0, #00H  
          MOV          P0M1, #00H  
          MOV          P1M0, #00H  
          MOV          P1M1, #00H  
          MOV          P2M0, #00H  
          MOV          P2M1, #00H  
          MOV          P3M0, #00H  
          MOV          P3M1, #00H  
          MOV          P4M0, #00H  
          MOV          P4M1, #00H  
          MOV          P5M0, #00H  
          MOV          P5M1, #00H  
  
          ORL          P_SW2, #00H      ;RXD2/P1.0, TXD2/P1.1  
;          ORL          P_SW2, #01H      ;RXD2_2/P4.0, TXD2_2/P4.2  
  
          SJMP         $  
  
          END
```

3.2.3 串口 3 切换

C 语言代码

```
//测试工作频率为 11.0592MHz
```

```
#include "stc8h.h"
```

```
void main()
{
    P_SW2 /= 0x80;                //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2/= 0x02;                //RXD3/P0.0, TXD3/P0.1
    // P_SW2 /= 0x02;           //RXD3_2/P5.0, TXD3_2/P5.1

    while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

|       |      |             |          |
|-------|------|-------------|----------|
| P_SW2 | DATA | 0BAH        |          |
| P0M1  | DATA | 093H        |          |
| P0M0  | DATA | 094H        |          |
| P1M1  | DATA | 091H        |          |
| P1M0  | DATA | 092H        |          |
| P2M1  | DATA | 095H        |          |
| P2M0  | DATA | 096H        |          |
| P3M1  | DATA | 0B1H        |          |
| P3M0  | DATA | 0B2H        |          |
| P4M1  | DATA | 0B3H        |          |
| P4M0  | DATA | 0B4H        |          |
| P5M1  | DATA | 0C9H        |          |
| P5M0  | DATA | 0CAH        |          |
|       | ORG  | 0000H       |          |
|       | LJMP | MAIN        |          |
|       | ORG  | 0100H       |          |
| MAIN: | MOV  | SP, #5FH    |          |
|       | ORL  | P_SW2, #80H | ;使能访问XFR |
|       | MOV  | P0M0, #00H  |          |
|       | MOV  | P0M1, #00H  |          |
|       | MOV  | P1M0, #00H  |          |
|       | MOV  | P1M1, #00H  |          |
|       | MOV  | P2M0, #00H  |          |
|       | MOV  | P2M1, #00H  |          |
|       | MOV  | P3M0, #00H  |          |
|       | MOV  | P3M1, #00H  |          |

```
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

ORL      P_SW2, #00H      ;RXD3/P0.0, TXD3/P0.1
; ORL      P_SW2, #02H      ;RXD3_2/P5.0, TXD3_2/P5.1

SJMP     $

END
```

### 3.2.4 串口 4 切换

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"

void main()
{
    P_SW2 /= 0x80;      //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 /= 0x00;      //RXD4/P0.2, TXD4/P0.3
    // P_SW2 /= 0x04;    //RXD4_2/P5.2, TXD4_2/P5.3

    while (1);
}
```

#### 汇编代码

```
;测试工作频率为 11.0592MHz

P_SW2      DATA      0BAH

P0M1        DATA      093H
P0M0        DATA      094H
P1M1        DATA      091H
P1M0        DATA      092H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
```

```
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN

MAIN:      ORG          0100H

          MOV          SP, #5FH
          ORL          P_SW2, #80H          ;使能访问 XFR

          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          ORL          P_SW2, #00H          ;RXD4/P0.2, TXD4/P0.3
;          ORL          P_SW2, #04H          ;RXD4_2/P5.2, TXD4_2/P5.3

          SJMP         $

          END
```

3.2.5 SPI 切换

C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"

void main()
{
    P_SW2 |= 0x80;          //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;
```



```

        P_SW1 = 0x00;                                //SS/P1.2, MOSI/P1.3, MISO/P1.4, SCLK/P1.5
//      P_SW1 = 0x04;                                //SS_2/P2.2, MOSI_2/P2.3, MISO_2/P2.4, SCLK_2/P2.5
//      P_SW1 = 0x08;                                //SS_3/P5.4, MOSI_3/P4.0, MISO_3/P4.1, SCLK_3/P4.3
//      P_SW1 = 0x0c;                                //SS_4/P3.5, MOSI_4/P3.4, MISO_4/P3.3, SCLK_4/P3.2

        while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

```

P_SW1      DATA      0A2H
P_SW2      DATA      0BAH

P1M1       DATA      091H
P1M0       DATA      092H
P0M1       DATA      093H
P0M0       DATA      094H
P2M1       DATA      095H
P2M0       DATA      096H
P3M1       DATA      0B1H
P3M0       DATA      0B2H
P4M1       DATA      0B3H
P4M0       DATA      0B4H
P5M1       DATA      0C9H
P5M0       DATA      0CAH

                ORG      0000H
                LJMP     MAIN

MAIN:         ORG      0100H

                MOV      SP, #5FH
                ORL      P_SW2, #80H                ;使能访问 XFR

                MOV      P0M0, #00H
                MOV      P0M1, #00H
                MOV      P1M0, #00H
                MOV      P1M1, #00H
                MOV      P2M0, #00H
                MOV      P2M1, #00H
                MOV      P3M0, #00H
                MOV      P3M1, #00H
                MOV      P4M0, #00H
                MOV      P4M1, #00H
                MOV      P5M0, #00H
                MOV      P5M1, #00H

                MOV      P_SW1, #00H                ;SS/P1.2, MOSI/P1.3, MISO/P1.4, SCLK/P1.5
;              MOV      P_SW1, #04H                ;SS_2/P2.2, MOSI_2/P2.3, MISO_2/P2.4, SCLK_2/P2.5
;              MOV      P_SW1, #08H                ;SS_3/P5.4, MOSI_3/P4.0, MISO_3/P4.1, SCLK_3/P4.3
;              MOV      P_SW1, #0CH                ;SS_4/P3.5, MOSI_4/P3.4, MISO_4/P3.3, SCLK_4/P3.2

                SJMP     $

                END

```

### 3.2.6 I2C 切换

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"

void main()
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 /= 0x00;                //SCL/P1.5, SDA/P1.4
    // P_SW2 /= 0x10;            //SCL_2/P2.5, SDA_2/P2.4
    // P_SW2 /= 0x20;            //SCL_3/P7.7, SDA_3/P7.6
    // P_SW2 /= 0x30;            //SCL_4/P3.2, SDA_4/P3.3

    while (1);
}
```

#### 汇编代码

```
;测试工作频率为 11.0592MHz

P_SW2      DATA      0BAH

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

            ORG          0000H
            LJMP         MAIN

            ORG          0100H

MAIN:
```

```

MOV    SP, #5FH
ORL    P_SW2, #80H           ;使能访问 XFR

MOV    P0M0, #00H
MOV    P0M1, #00H
MOV    P1M0, #00H
MOV    P1M1, #00H
MOV    P2M0, #00H
MOV    P2M1, #00H
MOV    P3M0, #00H
MOV    P3M1, #00H
MOV    P4M0, #00H
MOV    P4M1, #00H
MOV    P5M0, #00H
MOV    P5M1, #00H

ORL    P_SW2, #00H           ;SCL/P1.5, SDA/P1.4
; ORL    P_SW2, #10H         ;SCL_2/P2.5, SDA_2/P2.4
; ORL    P_SW2, #20H         ;SCL_3/P7.7, SDA_3/P7.6
; ORL    P_SW2, #30H         ;SCL_4/P3.2, SDA_4/P3.3

SJMP    $

END

```

### 3.2.7 比较器输出切换

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
```

```
void main()
```

```

{
    P_SW2 /= 0x80;           //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 /= 0x00;           //CMPO/P3.4
    // P_SW2 /= 0x08;        //CMPO_2/P4.1

    while (1);
}

```

汇编代码

;测试工作频率为 11.0592MHz

```
P_SW2      DATA      0BAH

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

                ORG      0000H
                LJMP     MAIN

MAIN:                ORG      0100H

                MOV      SP, #5FH
                ORL      P_SW2, #80H                ;使能访问 XFR

                MOV      P0M0, #00H
                MOV      P0M1, #00H
                MOV      P1M0, #00H
                MOV      P1M1, #00H
                MOV      P2M0, #00H
                MOV      P2M1, #00H
                MOV      P3M0, #00H
                MOV      P3M1, #00H
                MOV      P4M0, #00H
                MOV      P4M1, #00H
                MOV      P5M0, #00H
                MOV      P5M1, #00H

                ORL      P_SW2, #00H                ;CMPO/P3.4
;                ORL      P_SW2, #08H                ;CMPO_2/P4.1

                SJMP     $

                END
```

3.2.8 主时钟输出切换

C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
```

```
void main()
{
```

```

P_SW2 /= 0x80;                                     //使能访问XFR

P0M0 = 0x00;
P0M1 = 0x00;
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

CLKOCR = 0x04;                                     //IRC/4 output via MCLKO/P5.4
// CLKOCR = 0x84;                                 //IRC/4 output via MCLKO_2/P1.6

while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|        |      |             |          |
|--------|------|-------------|----------|
| P_SW2  | DATA | 0BAH        |          |
| CLKOCR | EQU  | 0FE05H      |          |
| P1M1   | DATA | 091H        |          |
| P1M0   | DATA | 092H        |          |
| P0M1   | DATA | 093H        |          |
| P0M0   | DATA | 094H        |          |
| P2M1   | DATA | 095H        |          |
| P2M0   | DATA | 096H        |          |
| P3M1   | DATA | 0B1H        |          |
| P3M0   | DATA | 0B2H        |          |
| P4M1   | DATA | 0B3H        |          |
| P4M0   | DATA | 0B4H        |          |
| P5M1   | DATA | 0C9H        |          |
| P5M0   | DATA | 0CAH        |          |
|        | ORG  | 0000H       |          |
|        | LJMP | MAIN        |          |
|        | ORG  | 0100H       |          |
| MAIN:  | MOV  | SP, #5FH    |          |
|        | ORL  | P_SW2, #80H | ;使能访问XFR |
|        | MOV  | P0M0, #00H  |          |
|        | MOV  | P0M1, #00H  |          |
|        | MOV  | P1M0, #00H  |          |
|        | MOV  | P1M1, #00H  |          |
|        | MOV  | P2M0, #00H  |          |
|        | MOV  | P2M1, #00H  |          |
|        | MOV  | P3M0, #00H  |          |
|        | MOV  | P3M1, #00H  |          |
|        | MOV  | P4M0, #00H  |          |

---

```
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      A, #04H           ;IRC/4 output via MCLKO/P5.4
; MOV     A, #84H           ;IRC/4 output via MCLKO_2/P1.6
MOV      DPTR, #CLKOCR
MOVX     @DPTR, A

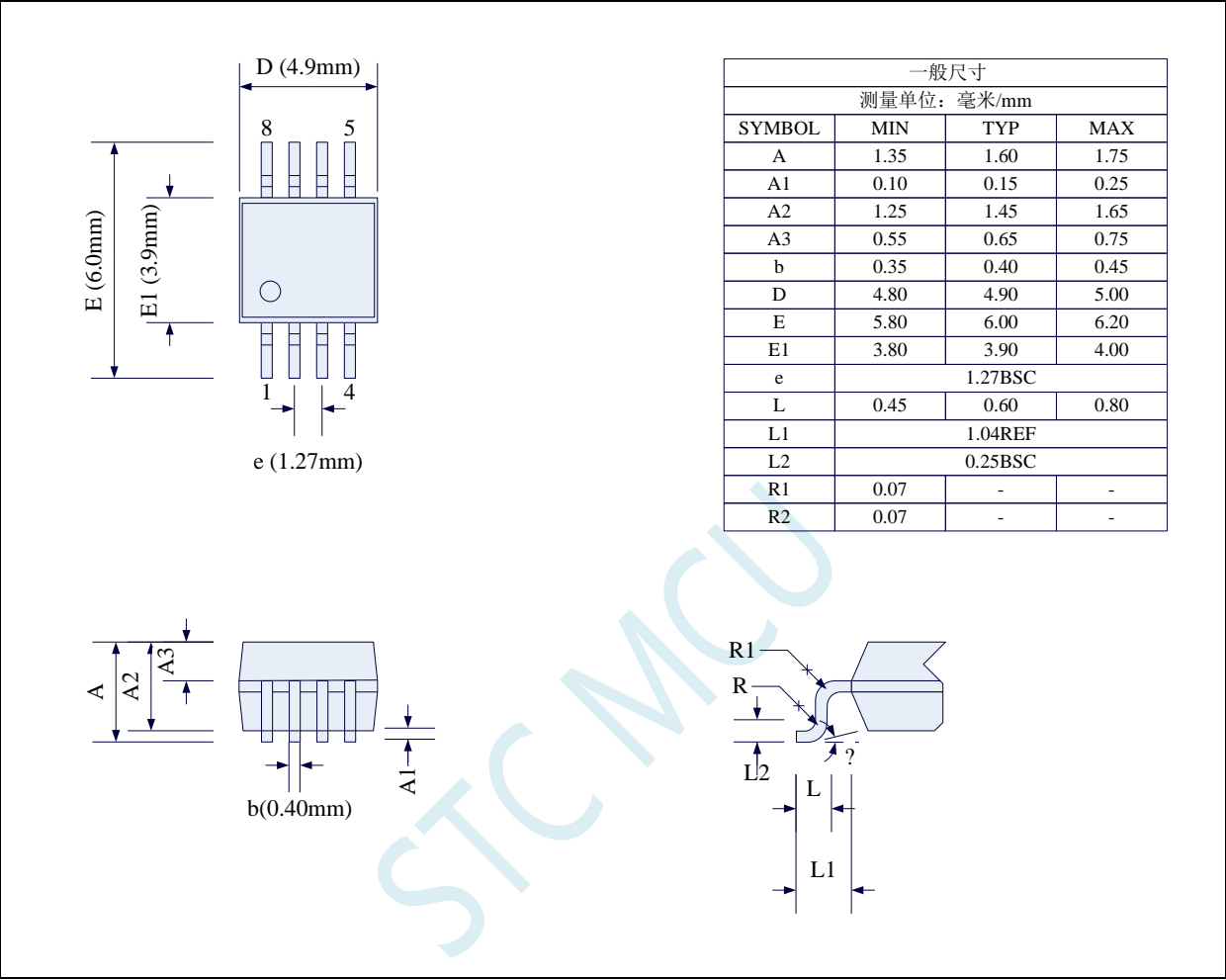
SJMP     $

END
```

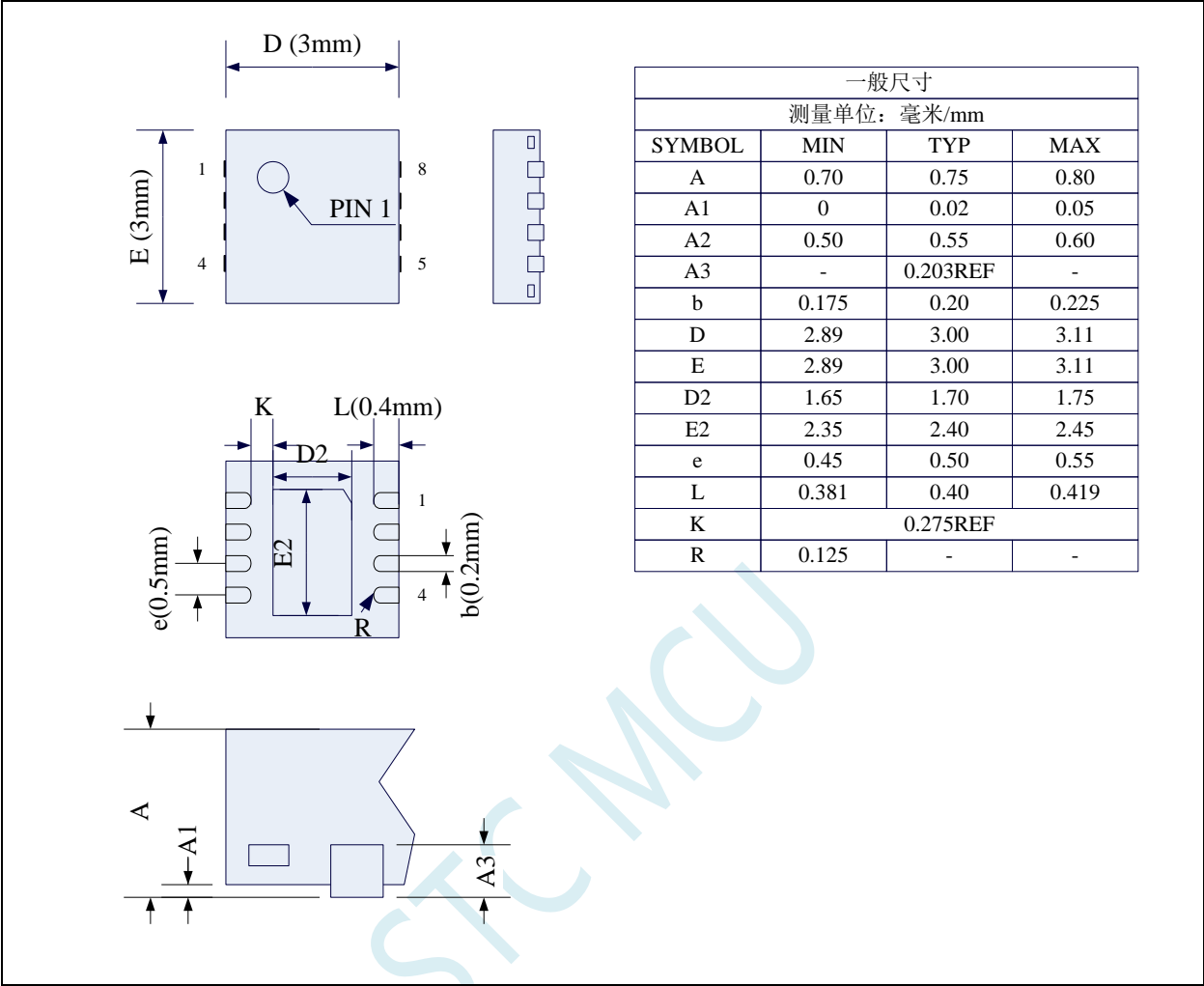
---

# 4 封装尺寸图

## 4.1 SOP8 封装尺寸图



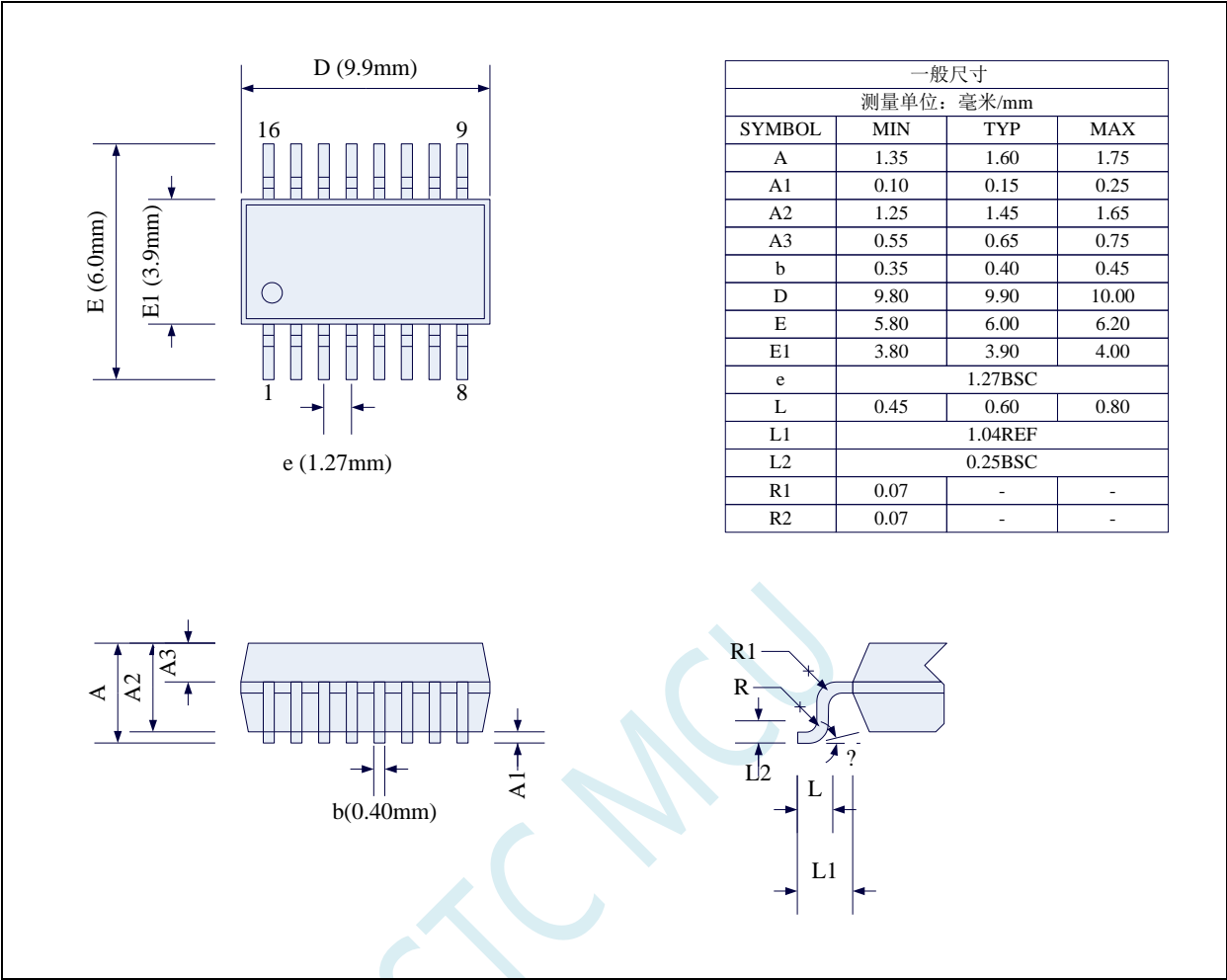
4.2 DFN8 封装尺寸图 (3mm\*3mm)



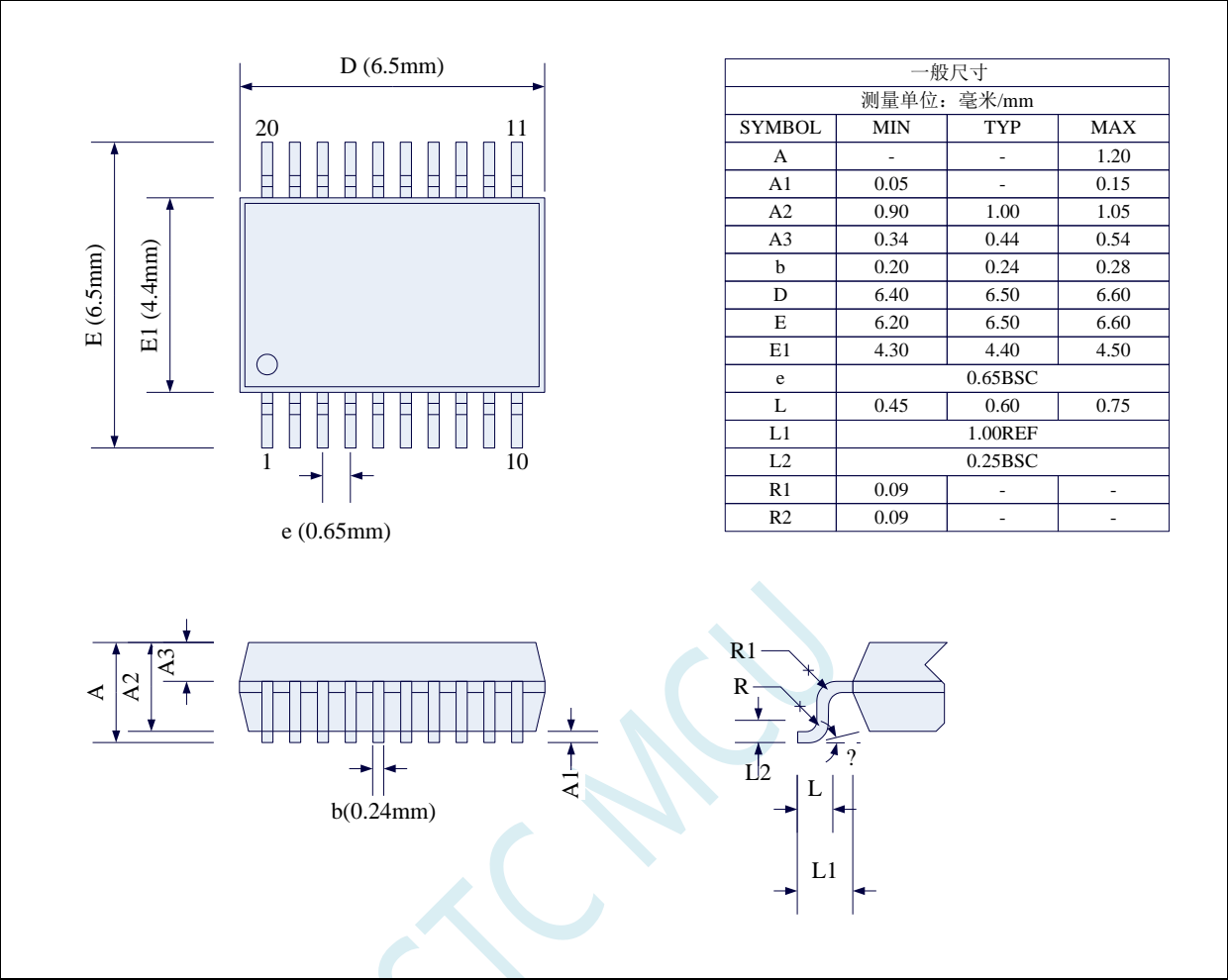
STC 现有 DFN8 封装芯片的背面金属片（衬底），在芯片内部并未接地，在用户的 PCB 板上可以接地，也可以不接地，不会对芯片性能造成影响



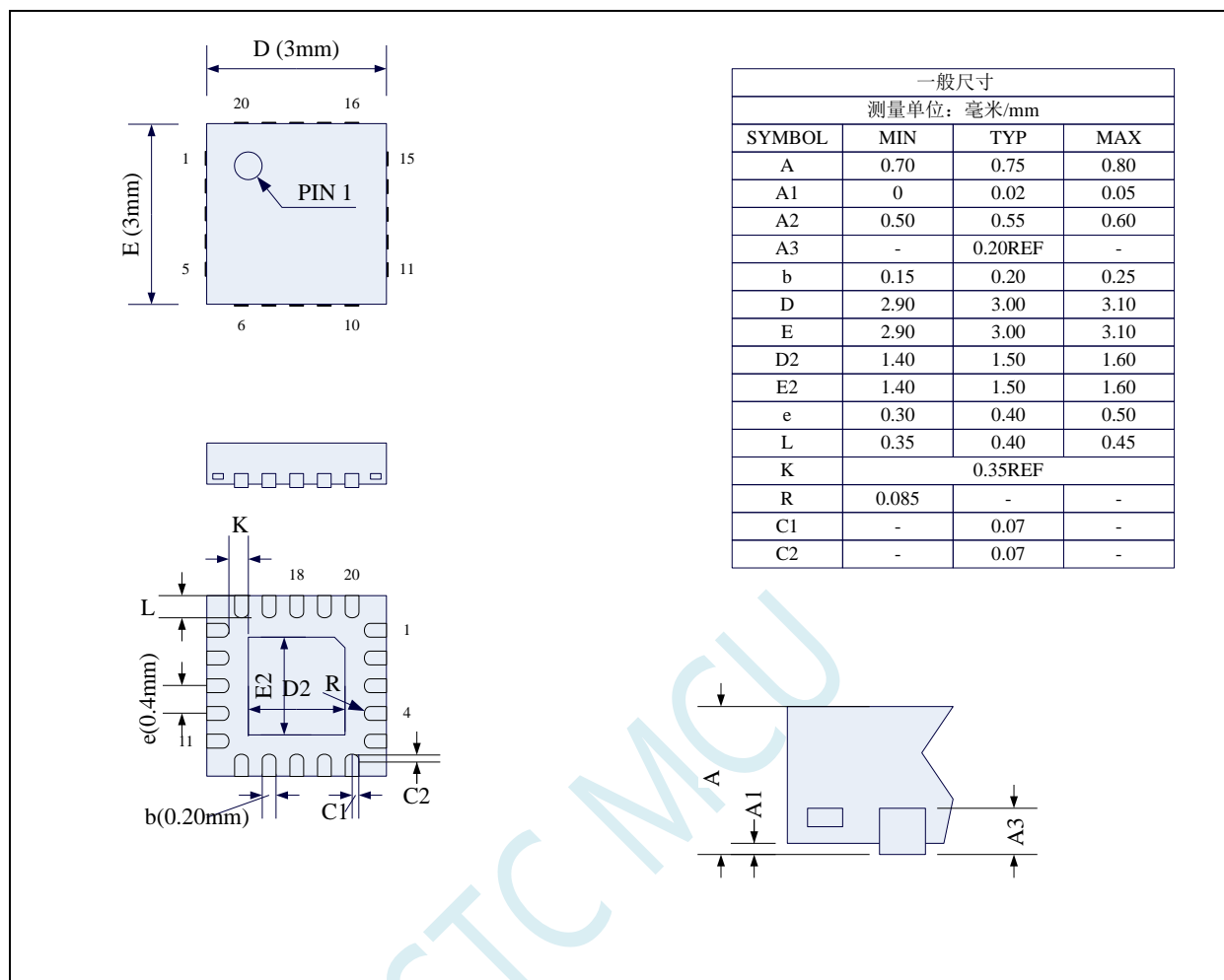
4.3 SOP16 封装尺寸图



4.4 TSSOP20 封装尺寸图

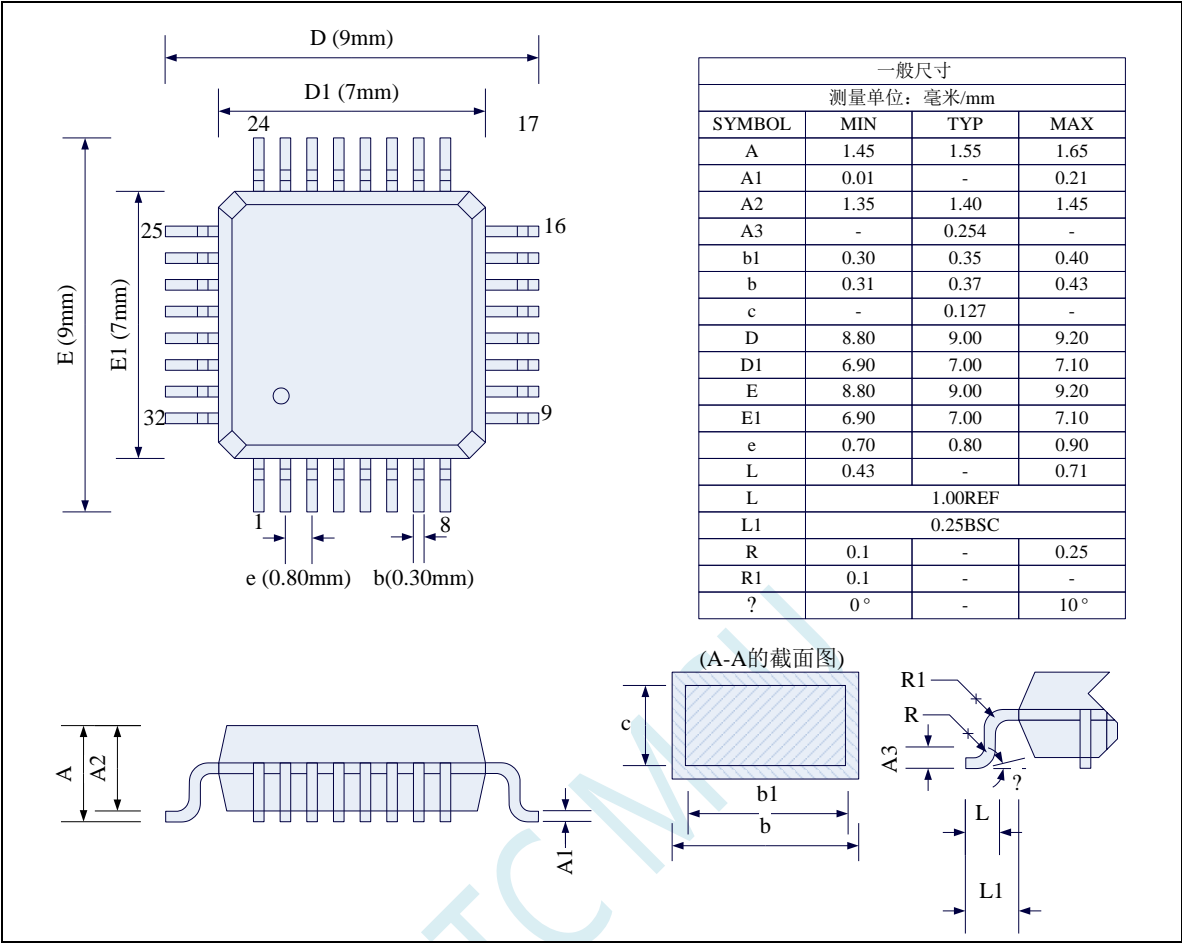


## 4.5 QFN20 封装尺寸图 (3mm\*3mm)

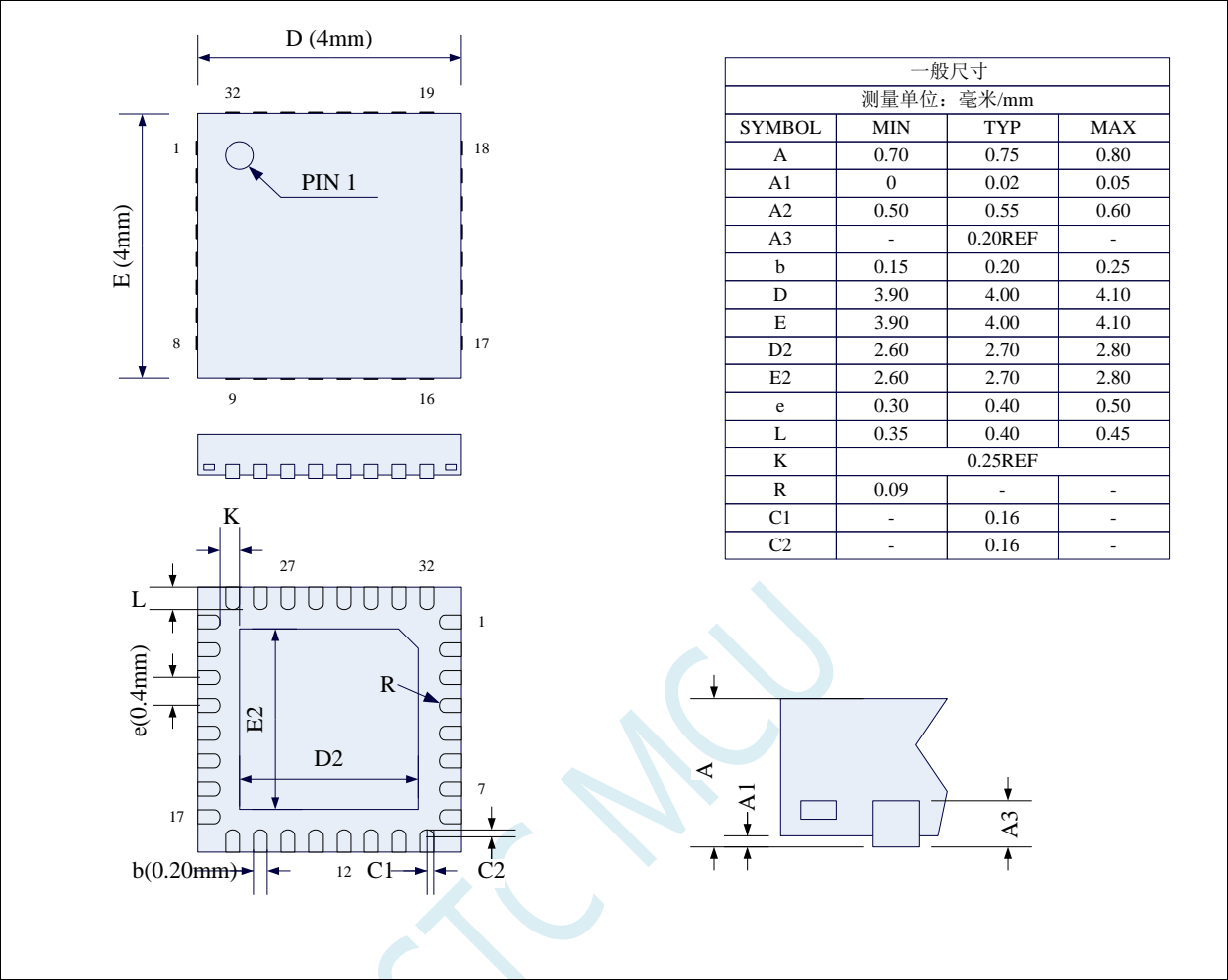


STC 现有 QFN20 封装芯片的背面金属片 (衬底), 在芯片内部并未接地, 在用户的 PCB 板上可以接地, 也可以不接地, 不会对芯片性能造成影响

4.6 LQFP32 封装尺寸图 (9mm\*9mm)

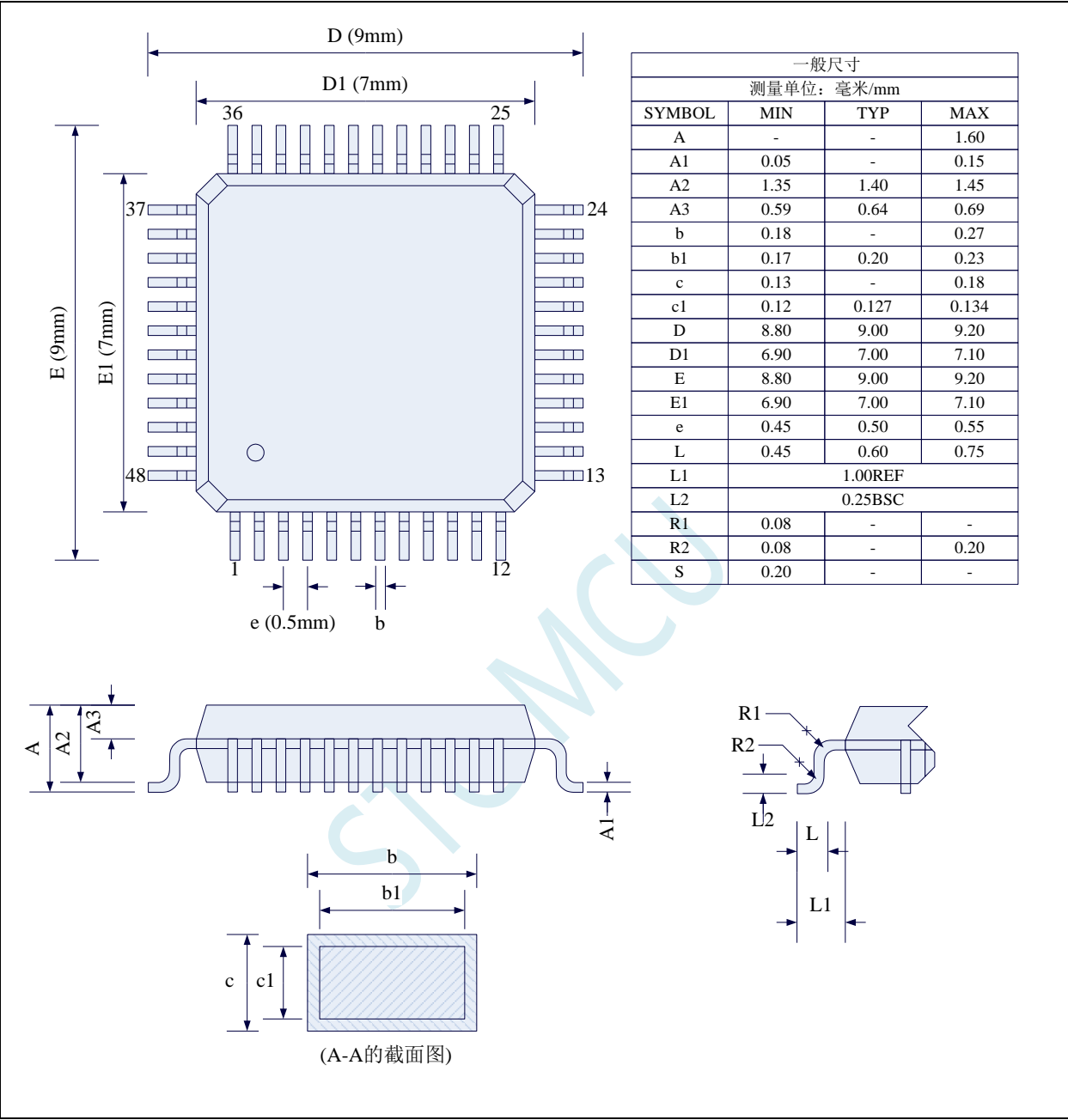


4.7 QFN32 封装尺寸图 (4mm\*4mm)

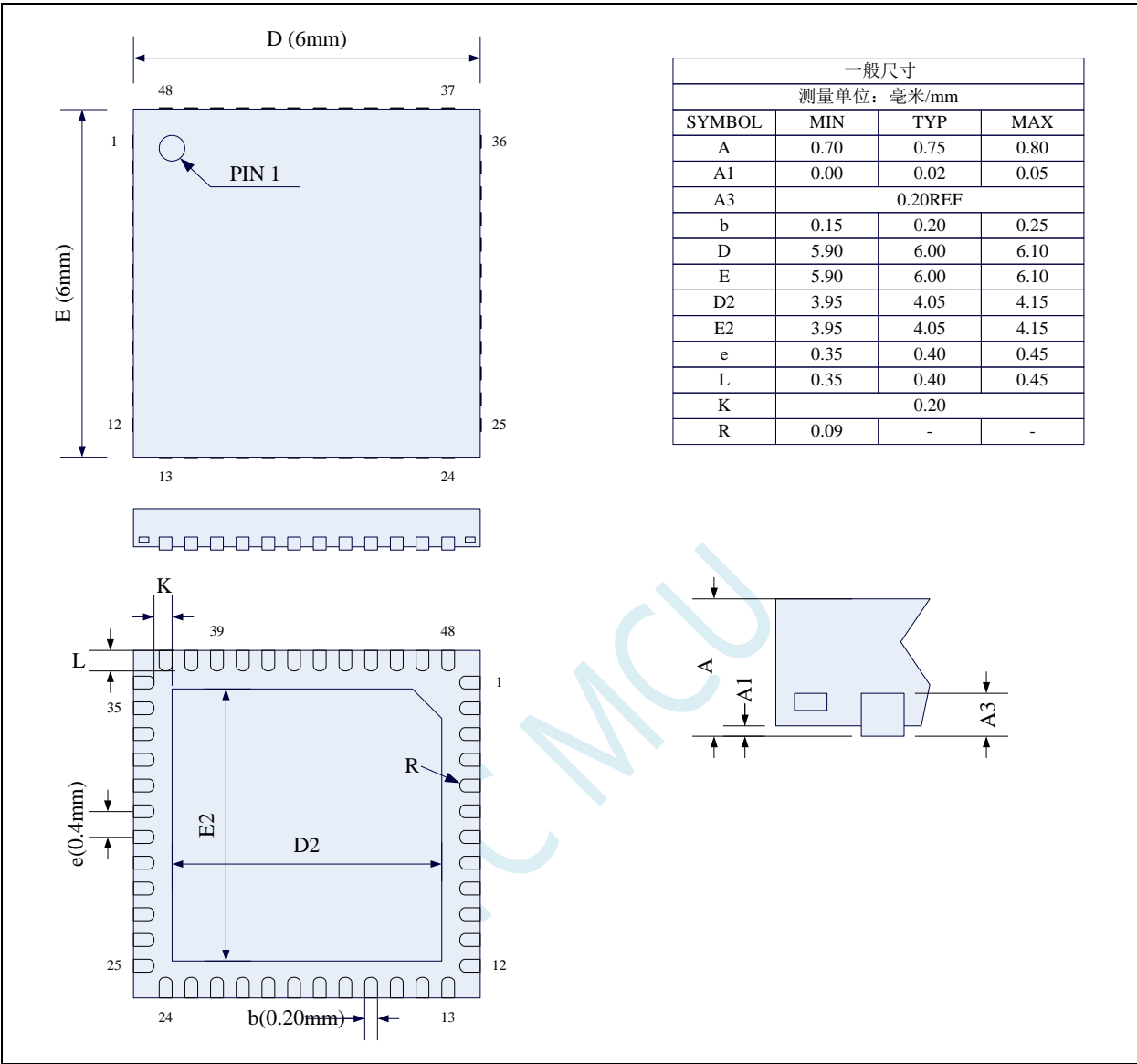


STC 现有 QFN32 封装芯片的背面金属片（衬底），在芯片内部并未接地，在用户的 PCB 板上可以接地，也可以不接地，不会对芯片性能造成影响

4.8 LQFP48 封装尺寸图 (9mm\*9mm)

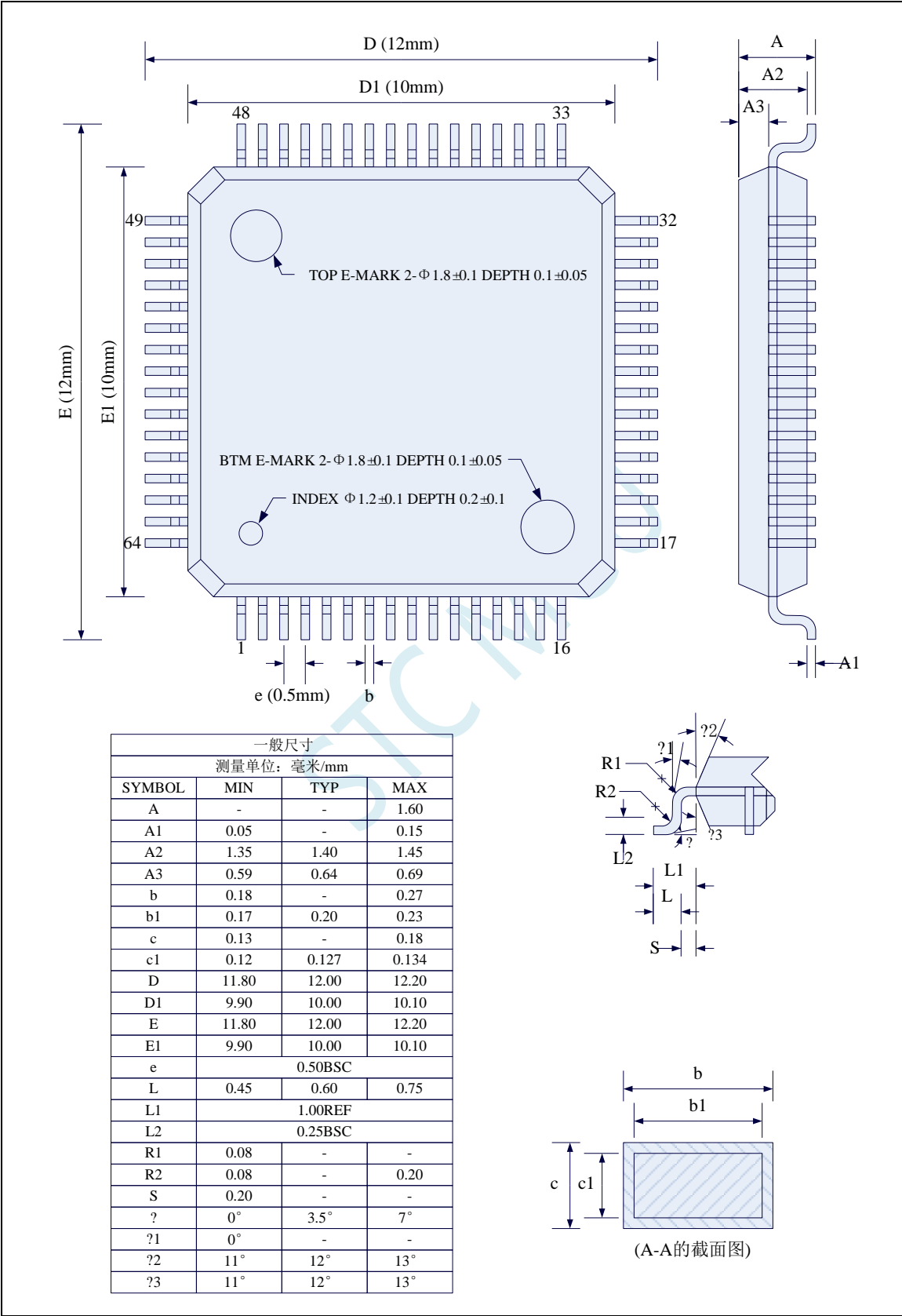


4.9 QFN48 封装尺寸图 (6mm\*6mm)



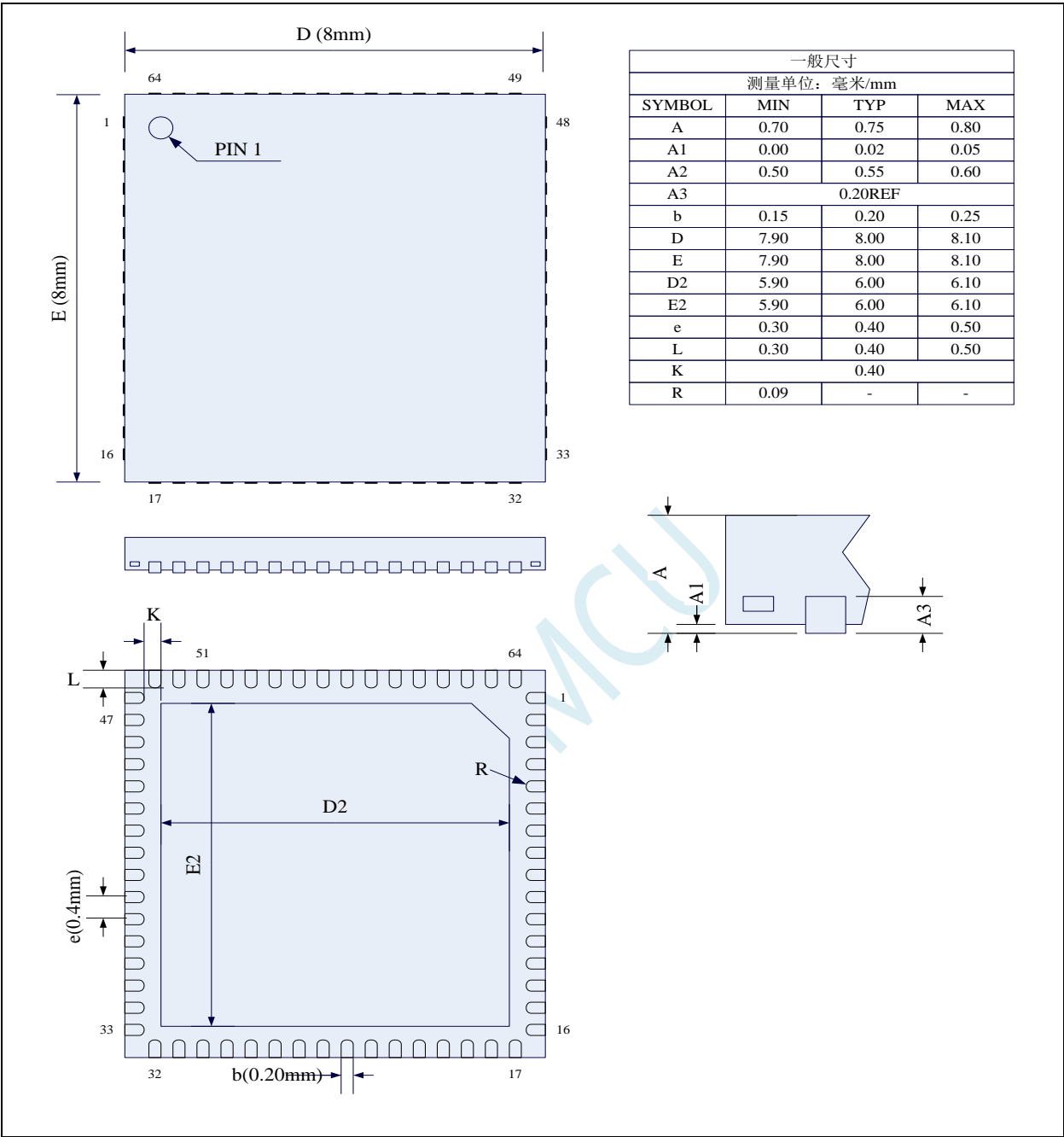
STC 现有 QFN48 封装芯片的背面金属片（衬底），在芯片内部并未接地，在用户的 PCB 板上可以接地，也可以不接地，不会对芯片性能造成影响

# 4.10 LQFP64S 封装尺寸图 (12mm\*12mm)



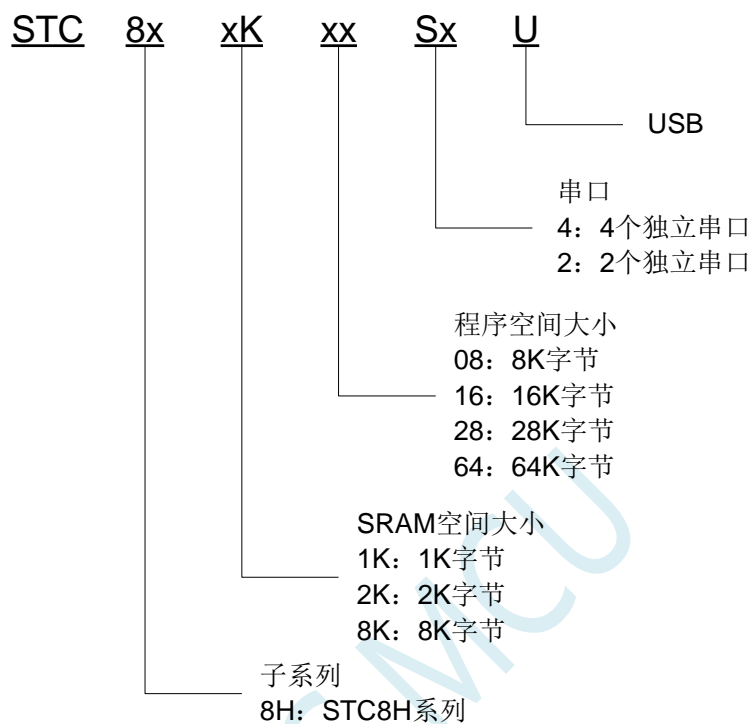


4.11 QFN64 封装尺寸图 (8mm\*8mm)



STC 现有 QFN64 封装芯片的背面金属片 (衬底), 在芯片内部并未接地, 在用户的 PCB 板上可以接地, 也可以不接地, 不会对芯片性能造成影响

## 4.12 STC8H 系列单片机命名规则

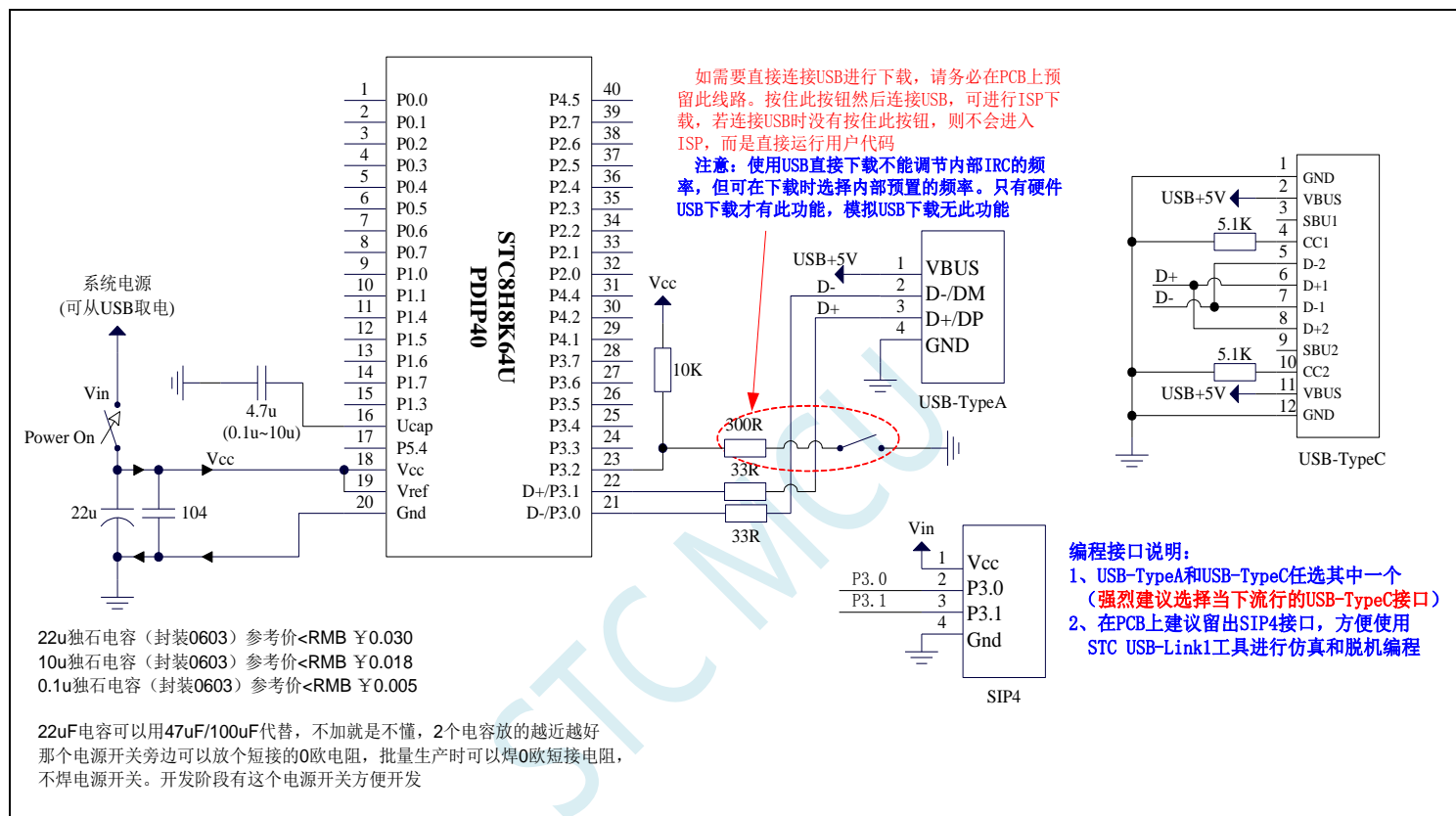


## 5 ISP 下载及典型应用线路图

### 5.1 STC8H 系列 ISP 下载应用线路图

#### 5.1.1 硬件 USB 直接 ISP 下载 (5V 系统)

注 1: STC8H8K64U-B 版芯片开始可支持 USB 模式的 ISP 下载和 USB 仿真



**项目开发温馨提示:** 一般 USB 直接进行 ISP 下载是提供给您的客户升级代码时使用的, 而项目开发阶段应该使用 (强烈建议) 我公司提供的 STC USB-Link1D 工具。STC USB-Link1D 工具给项目开发可提供如下便利:

- 1、ISP 下载时, 工具能够自动停电和上电, 可免去手动给目标芯片上电的麻烦
- 2、工具能够根据选择的目标单片机智能的提供 3.3V 或者 5V 的 VCC 电源
- 3、可直接使用工具对目标芯片进行串口模式仿真
- 4、在不进行 ISP 下载时, 下载口就是一个 USB-CDC 串口 1, 可协助工程师调试程序
- 5、另外工具还额外送一个独立的 USB-CDC 串口 2, 当使用 USB-CDC 串口 1 进行仿真的同时还可以使用 USB-CDC 串口 2 调试程序中的串口模块

所以, 对于一个专业的企业级公司, 应给您的软件工程师人手一个 STC USB-Link1D 工具, 从而极大提高项目开发进度。

现在 STC 的带硬件 USB 的 MCU 支持用硬件 USB 下载, 因为用的是 USB-HID 通信协议, 不需要安装任何驱动。STC 打狗棒、降龙棍、开天斧、屠龙刀核心板以及 STC 开源示波器、STC 实验箱在 D-/P3.0, D+/P3.1 与 PC-USB 端口连接好的状况下, USB-ISP 下载程序步骤:

1、按下板子上的 P3.2/INT0 按键，就是 P3.2 接地

2、给目标芯片重新上电，不管之前是否已通电。

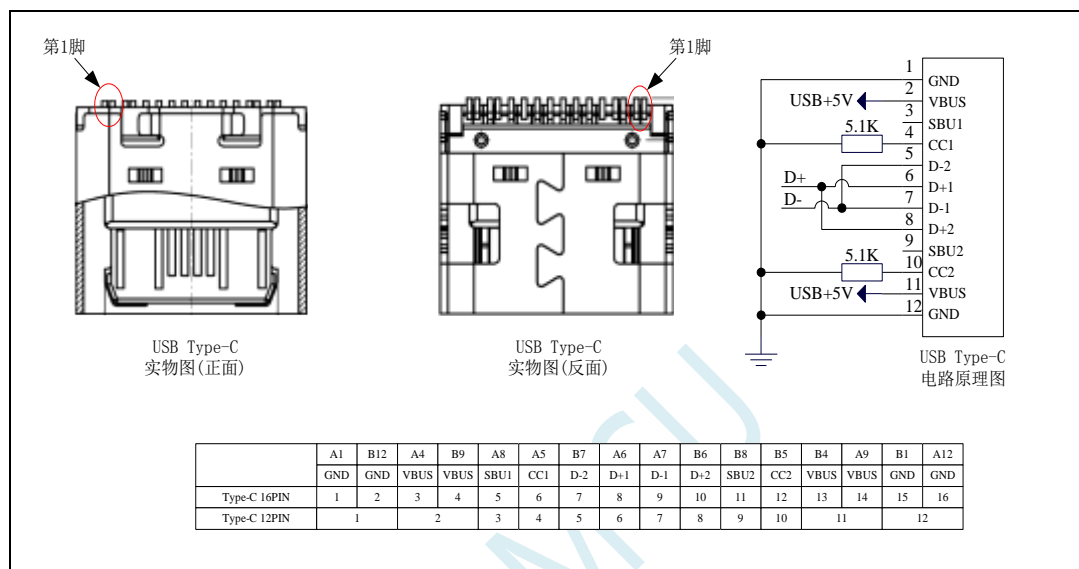
===电子开关是按下停电后再松开就是上电

等待 STC-ISP 下载软件中自动识别出“STC USB Writer (HID1)”，识别出来后，就与 P3.2 状态无关了，这时可以松开 P3.2 按键

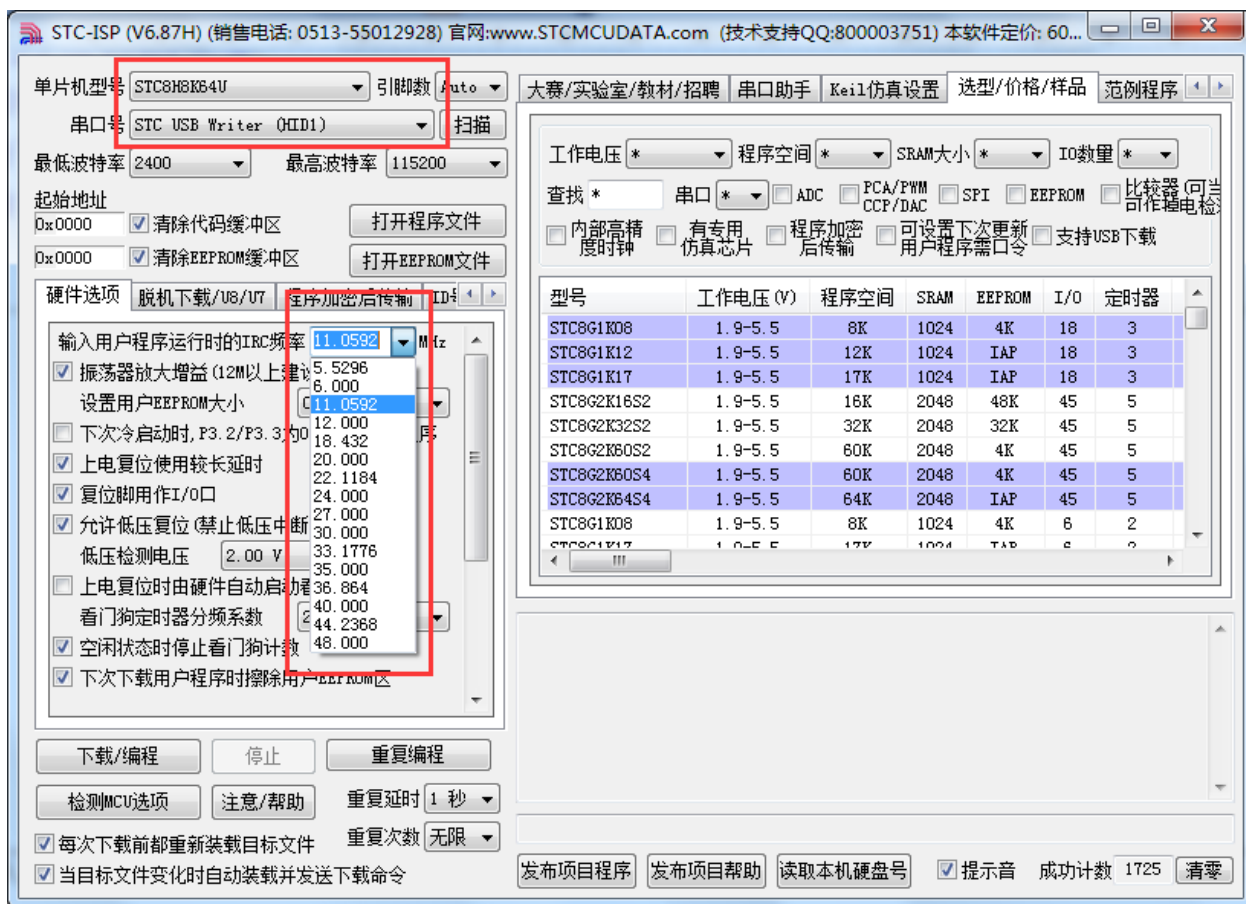
===传统的机械自锁紧开关是按上来停电，按下去是上电

3、点击下载软件中的“下载/编程”按钮（注意：USB 下载与串口下载的操作顺序不同）  
下载成功！

===另外从用户区软复位到系统区也是等待 USB 下载。



当用户使用硬件 USB 对 STC8H8K64U 系列进行 ISP 下载时不能调节内部 IRC 的频率，但用户可以选择芯片出厂时内部预置的 16 个频率(分别是 **5.5296M**、**6M**、**11.0592M**、**12M**、**18.432M**、**20M**、**22.1184M**、**24M**、**27M**、**30M**、**33.1776M**、**35M**、**36.864M**、**40M**、**44.2368M** 和 **48M**)。下载时用户只能从频率下拉列表中进行选择其中之一，而不能手动输入其他频率。(使用串口下载则可用输入 4M~48M 之间的任意频率)。详情见下页的图示

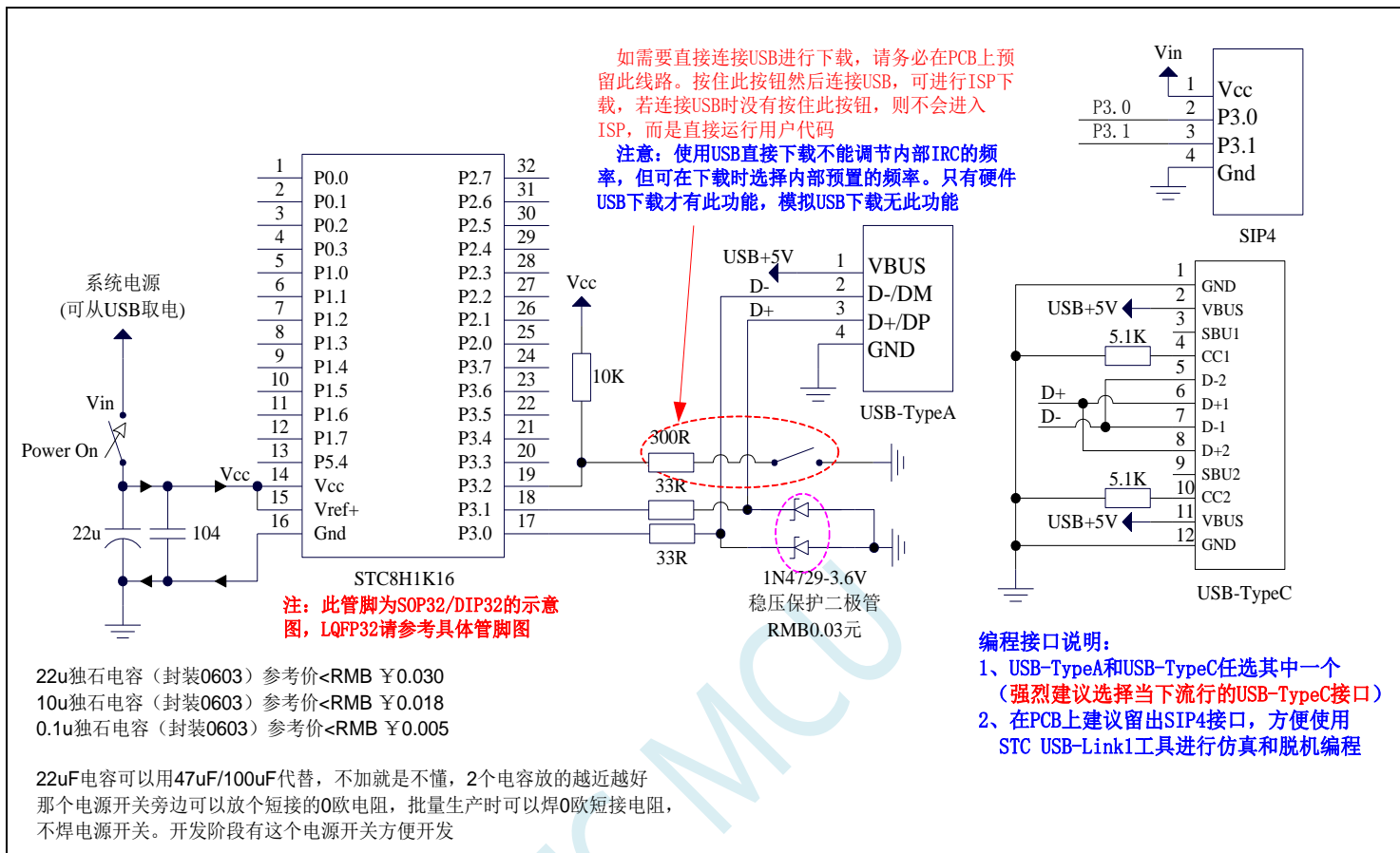




- 3、点击下载软件中的“下载/编程”按钮（注意：USB 下载与串口下载的操作顺序不同）  
下载成功！  
===另外从用户区软复位到系统区也是等待 USB 下载。

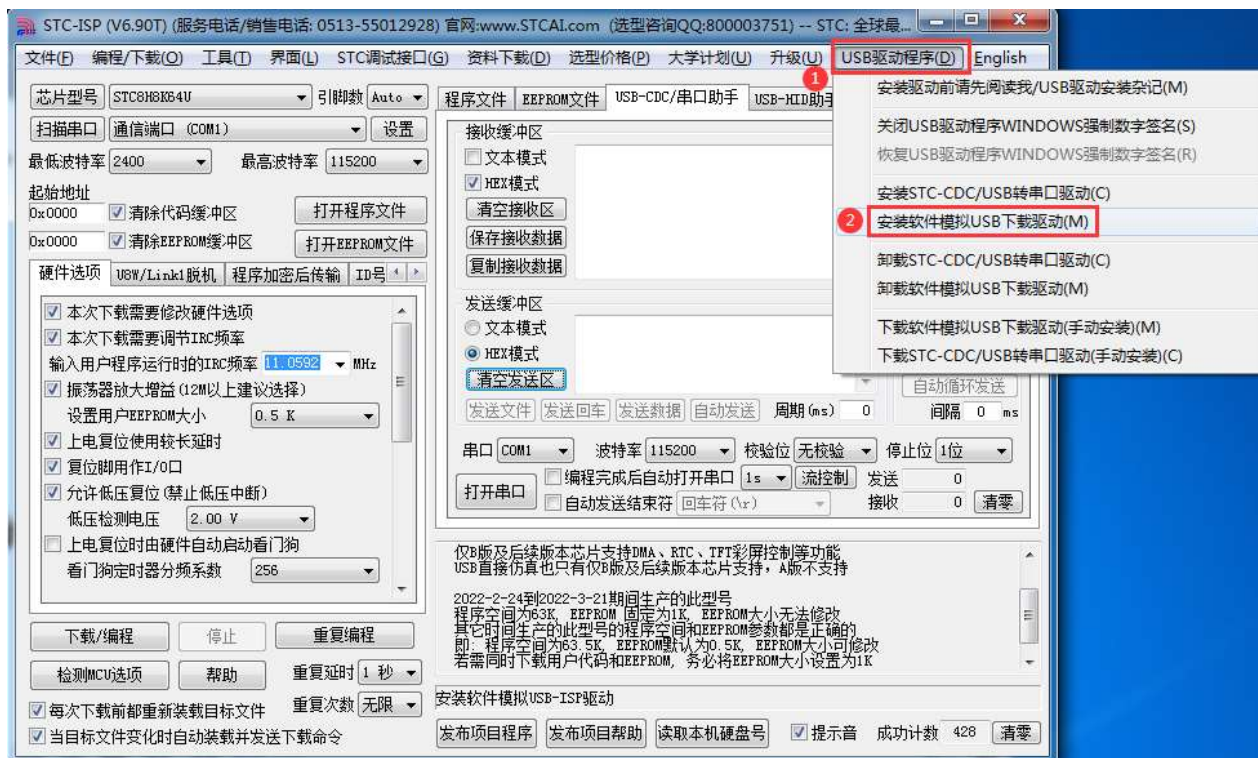
STC MCU

### 5.1.3 软件模拟硬件 USB 直接 ISP 下载，建议尝试，不支持仿真（5V 系统）



现在 STC 的不带硬件 USB 的 STC8G/STC8H 的 MCU，基本都支持用软件模拟硬件 USB 下载用户程序，因为走的是 USB-SCAN 通信协议，不管任何版本操作系统，都要安装驱动。在 STC-ISP 下载软件如下图所示的地方安装驱动。

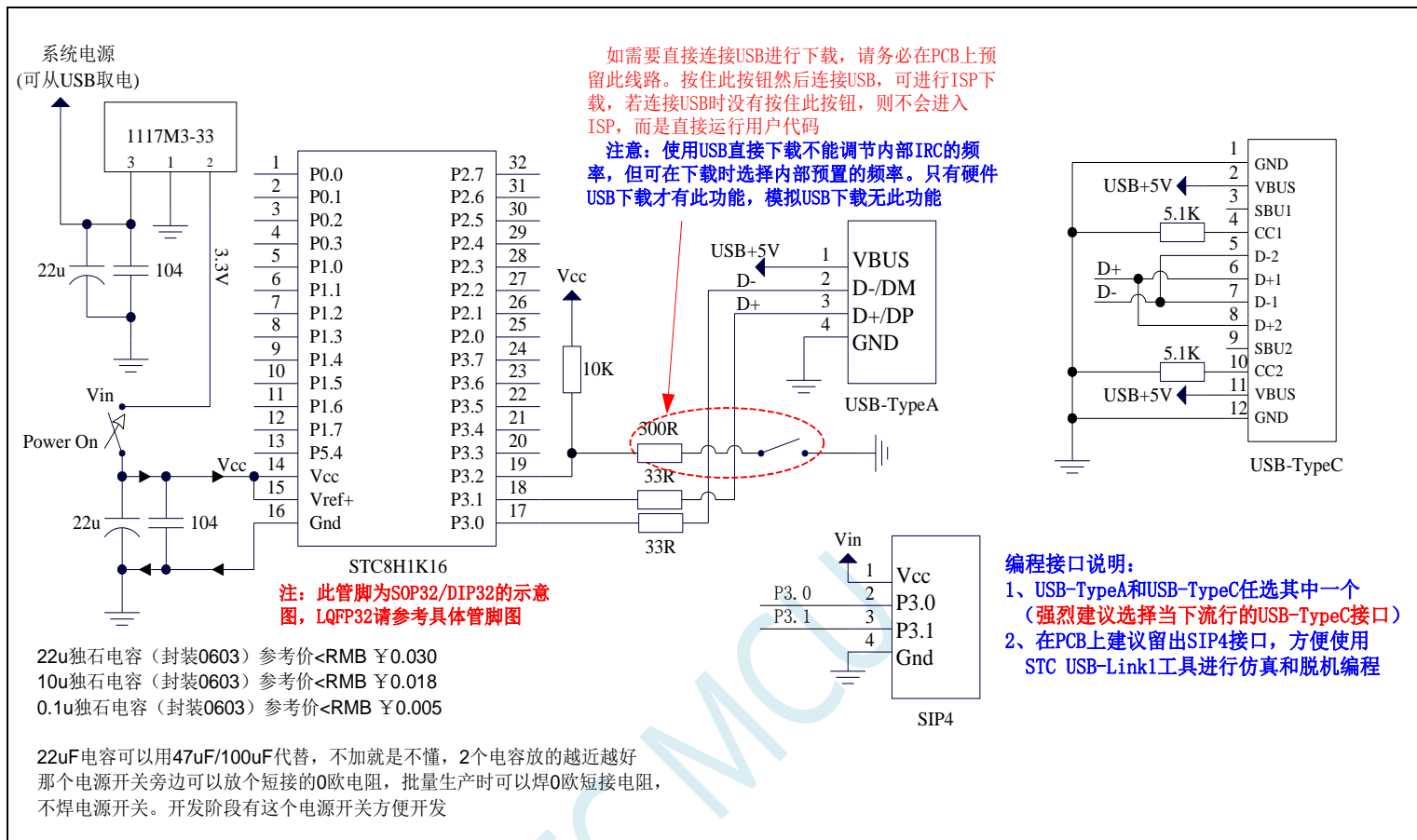




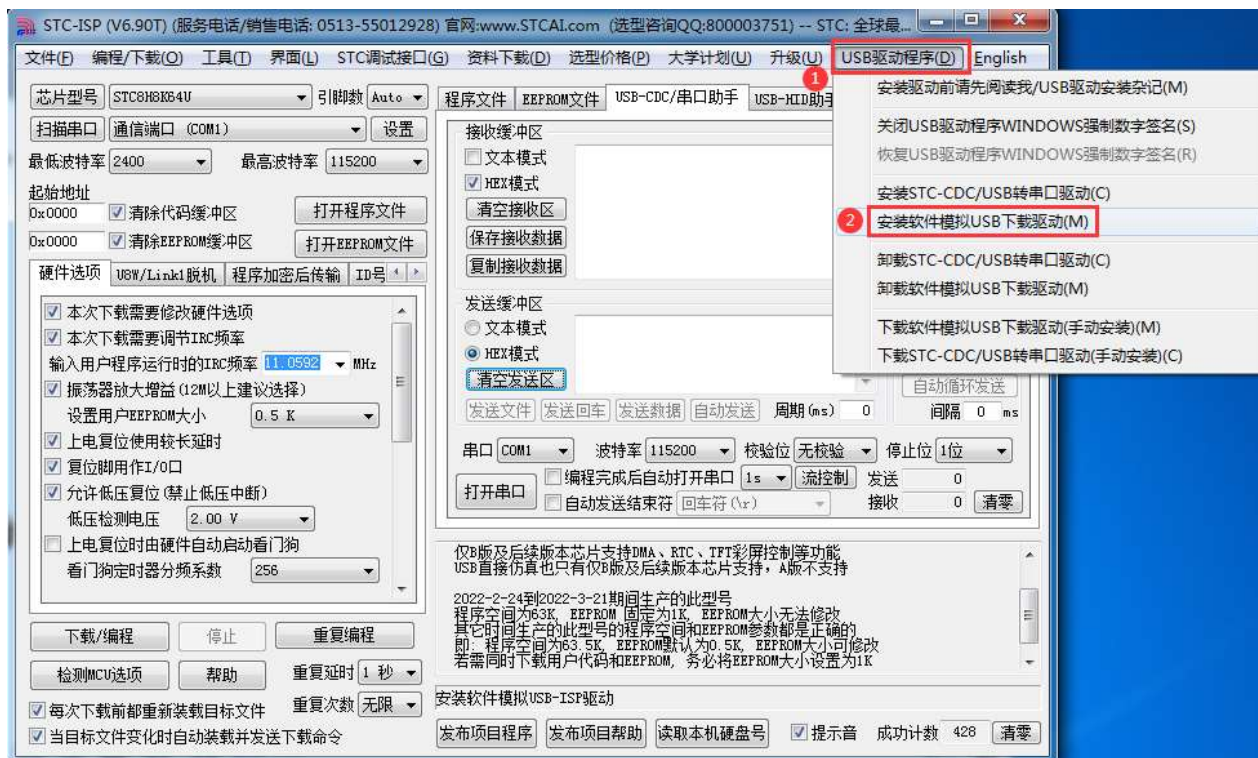
#### ISP 下载步骤:

- 1、D-/P3.0, D+/P3.1 与 PC-USB 端口连接好
- 2、将 P3.2 与 GND 短接, 实验箱板子上的 P3.2/INT0 按键按下
- 3、给目标芯片重新上电。若目标芯片已经停电, 直接上电即可; 若目标芯片处于通电状态, 则需给目标芯片停电再上电 (冷启动)。等待 STC-ISP 下载软件中自动识别出 “STC USB Writer (HID1)” 识别出来后, 就与 P3.2 状态无关了 (此时就不需要一直按着 P3.2 口了, 一直按着把手累坏了不要紧, 将按键按坏就麻烦了)。
- 4、点击下载软件中的 “下载/编程” 按钮 (注意: USB 下载与串口下载的操作顺序不同, 千万千万不要先点击下载按钮, 一定到等到电脑端识别出 “STC USB Writer (HID1)” 设备后, 才能点击下载按钮开始下载)

## 5.1.4 软件模拟硬件 USB 直接 ISP 下载, 建议尝试, 不支持仿真(3.3V 系统)



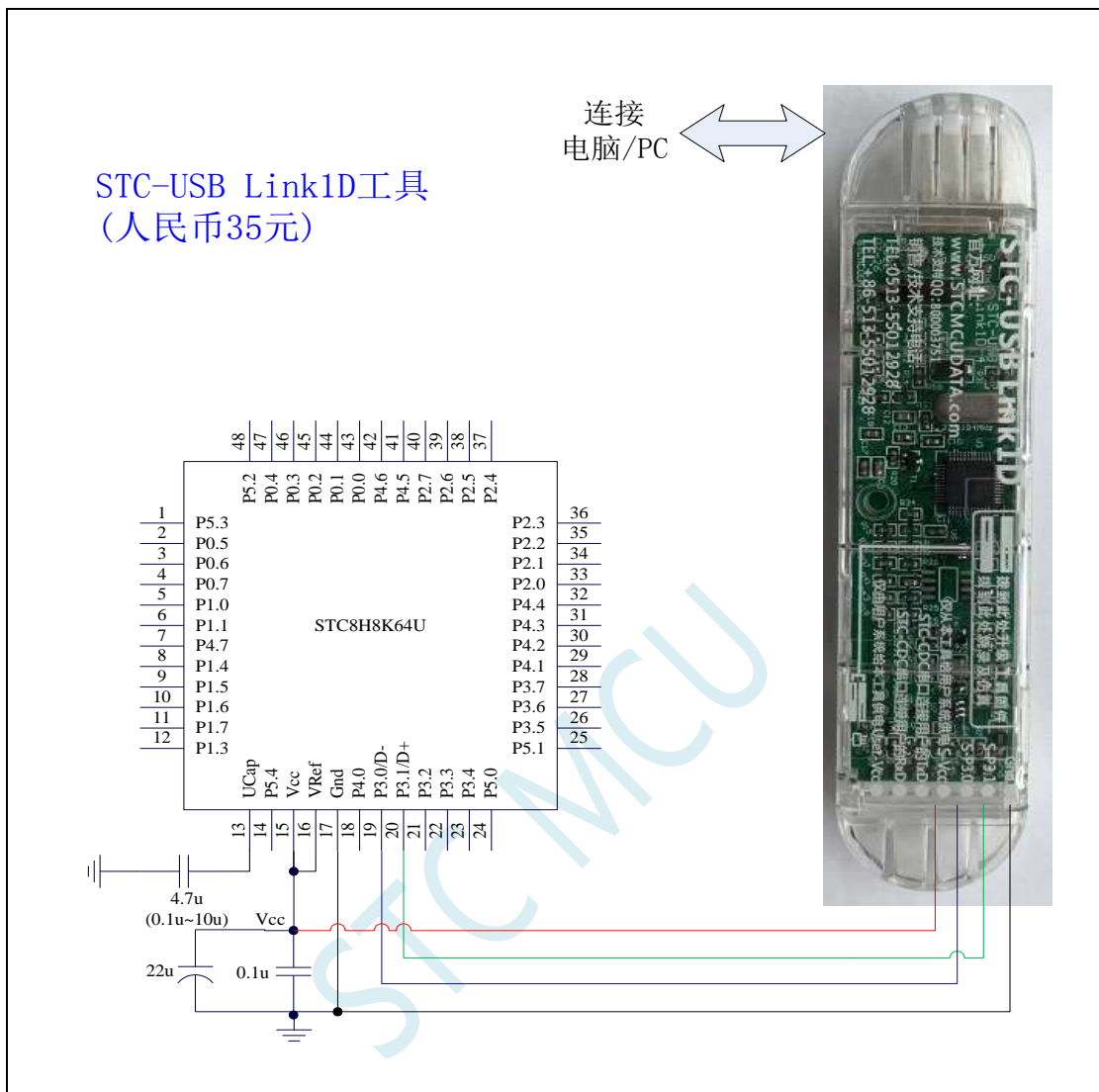
现在 STC 的不带硬件 USB 的 STC8G/STC8H 的 MCU, 基本都支持用软件模拟硬件 USB 下载用户程序, 因为走的是 USB-SCAN 通信协议, 不管任何版本操作系统, 都要安装驱动。在 STC-ISP 下载软件如下图所示的地方安装驱动。



#### ISP 下载步骤:

- 1、D-/P3.0, D+/P3.1 与 PC-USB 端口连接好
- 2、将 P3.2 与 GND 短接, 实验箱板子上的 P3.2/INT0 按键按下
- 3、给目标芯片重新上电。若目标芯片已经停电, 直接上电即可; 若目标芯片处于通电状态, 则需给目标芯片停电再上电 (冷启动)。等待 STC-ISP 下载软件中自动识别出 “STC USB Writer (HID1)” 识别出来后, 就与 P3.2 状态无关了 (此时就不需要一直按着 P3.2 口了, 一直按着把手累坏了不要紧, 将按键按坏就麻烦了)。
- 4、点击下载软件中的 “下载/编程” 按钮 (注意: USB 下载与串口下载的操作顺序不同, 千万千万不要先点击下载按钮, 一定到等到电脑端识别出 “STC USB Writer (HID1)” 设备后, 才能点击下载按钮开始下载)

### 5.1.5 使用 STC-USB Link1D 工具下载，支持在线和脱机下载



#### ISP 下载步骤:

- 1、按照如图所示的连接方式将 STC-USB Link1D 和目标芯片连接
- 2、点击 STC-ISP 下载软件中的“下载/编程”按钮
- 3、开始 ISP 下载

注意：若是使用 STC-USB Link1D 给目标系统供电，目标系统的总电流不能大于 200mA，否则会导致下载失败。

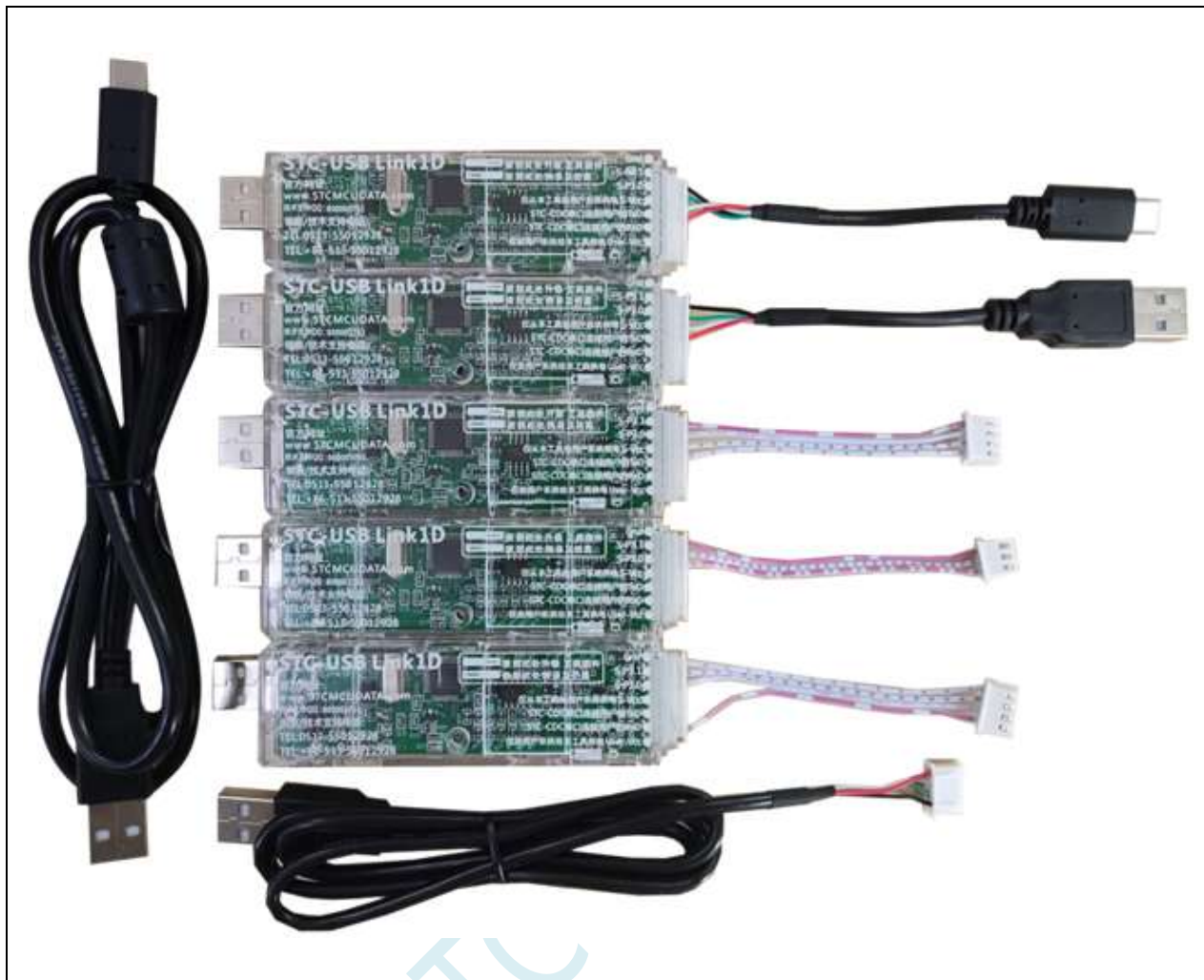
**项目开发温馨提示：**一般 USB 直接进行 ISP 下载是提供给您的客户升级代码时使用的，而项目开发阶段应该使用（强烈建议）我公司提供的 STC USB-Link1D 工具。STC USB-Link1D 工具给项目开发可提供如下便利：

- 1、ISP 下载时，工具能够自动停电和上电，可免去手动给目标芯片上电的麻烦
- 2、工具能够根据选择的目标单片机智能的提供 3.3V 或者 5V 的 VCC 电源
- 3、可直接使用工具对目标芯片进行串口模式仿真
- 4、在不进行 ISP 下载时，下载口就是一个 USB-CDC 串口 1，可协助工程师调试程序
- 5、另外工具还额外送一个独立的 USB-CDC 串口 2，当使用 USB-CDC 串口 1 进行仿真的同时还可以使用 USB-CDC 串口 2 调试程序中的串口模块

所以，对于一个专业的企业级公司，应给您的软件工程师人手一个 STC USB-Link1D 工具，从而极大提高项目开发进度。

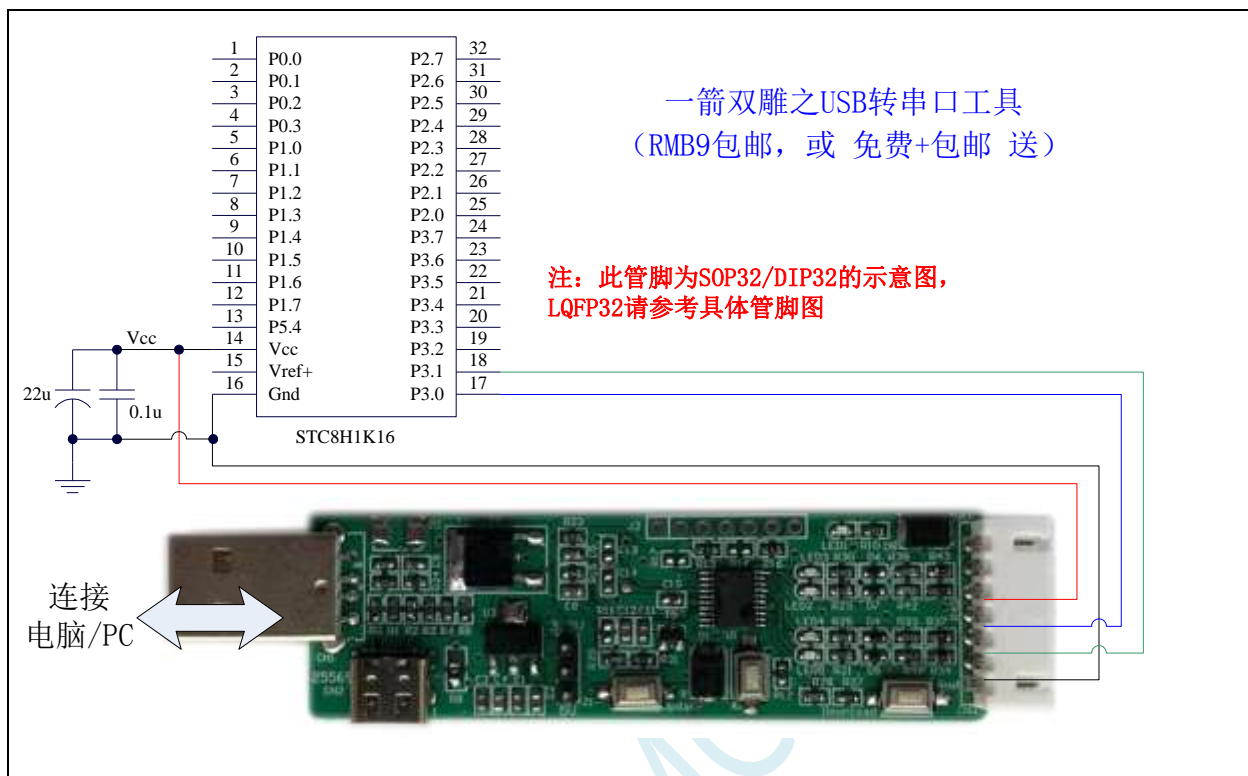
STC MCU





上面 **RMB35** 是配上面全部的线，是亏本补助大家的

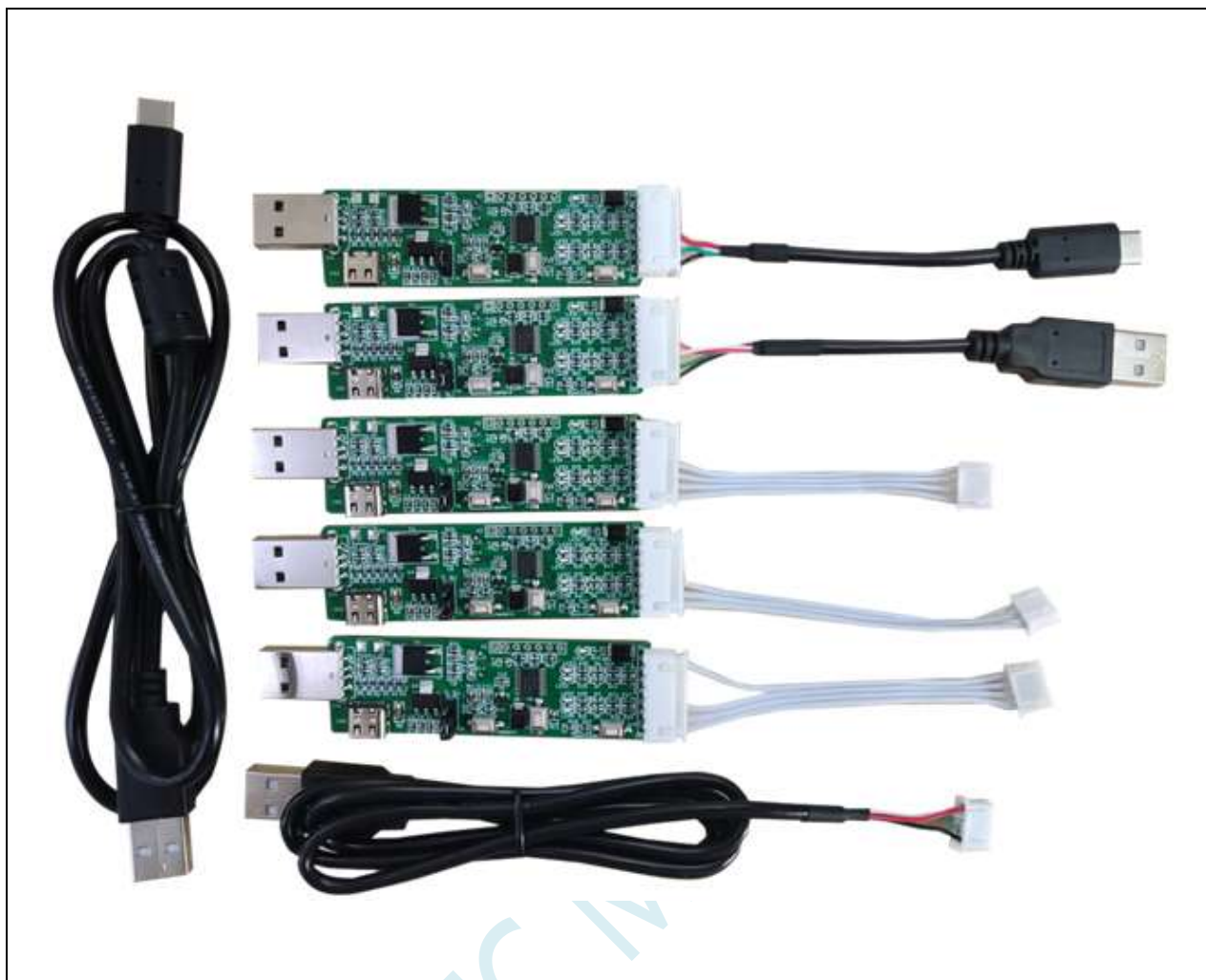
## 5.1.6 使用一箭双雕之 USB 转串口工具下载



### ISP 下载步骤:

- 1、按照如图所示的连接方式将 USB 转串口工具和目标芯片连接
- 2、点击 STC-ISP 下载软件中的“下载/编程”按钮
- 3、开始 ISP 下载

**注意:** 目前有发现使用 USB 线供电进行 ISP 下载时, 由于 USB 线太细, 在 USB 线上的压降过大, 导致 ISP 下载时供电不足, 所以请在使用 USB 线供电进行 ISP 下载时, 务必使用 USB 加强线。

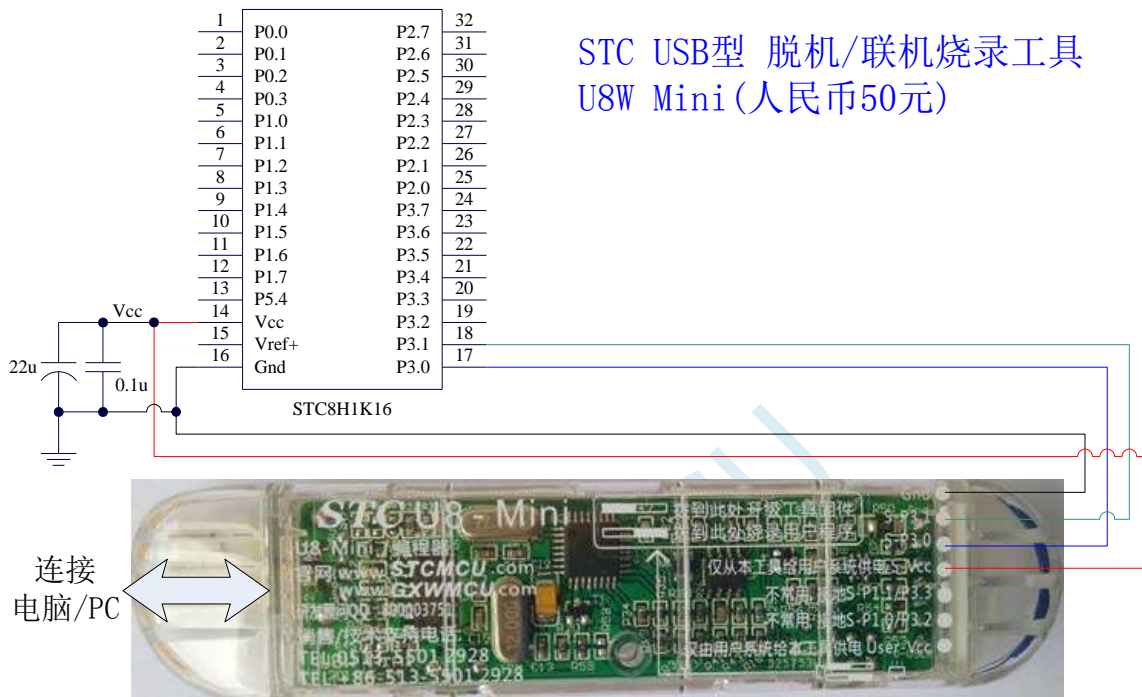


一箭双雕之USB转串口工具  
(人民币9元包邮销售, 只含一条SIP7-SIP4的线, 亏本补助大家)



## 5.1.7 使用 U8-Mini 工具下载, 支持 ISP 在线和脱机下载, 也可支持仿真

注: 此管脚为SOP32/DIP32的示意图,  
LQFP32请参考具体管脚图



### ISP 下载步骤:

- 4、按照如图所示的连接方式将 U8-Mini 和目标芯片连接
- 5、点击 STC-ISP 下载软件中的“下载/编程”按钮
- 6、开始 ISP 下载

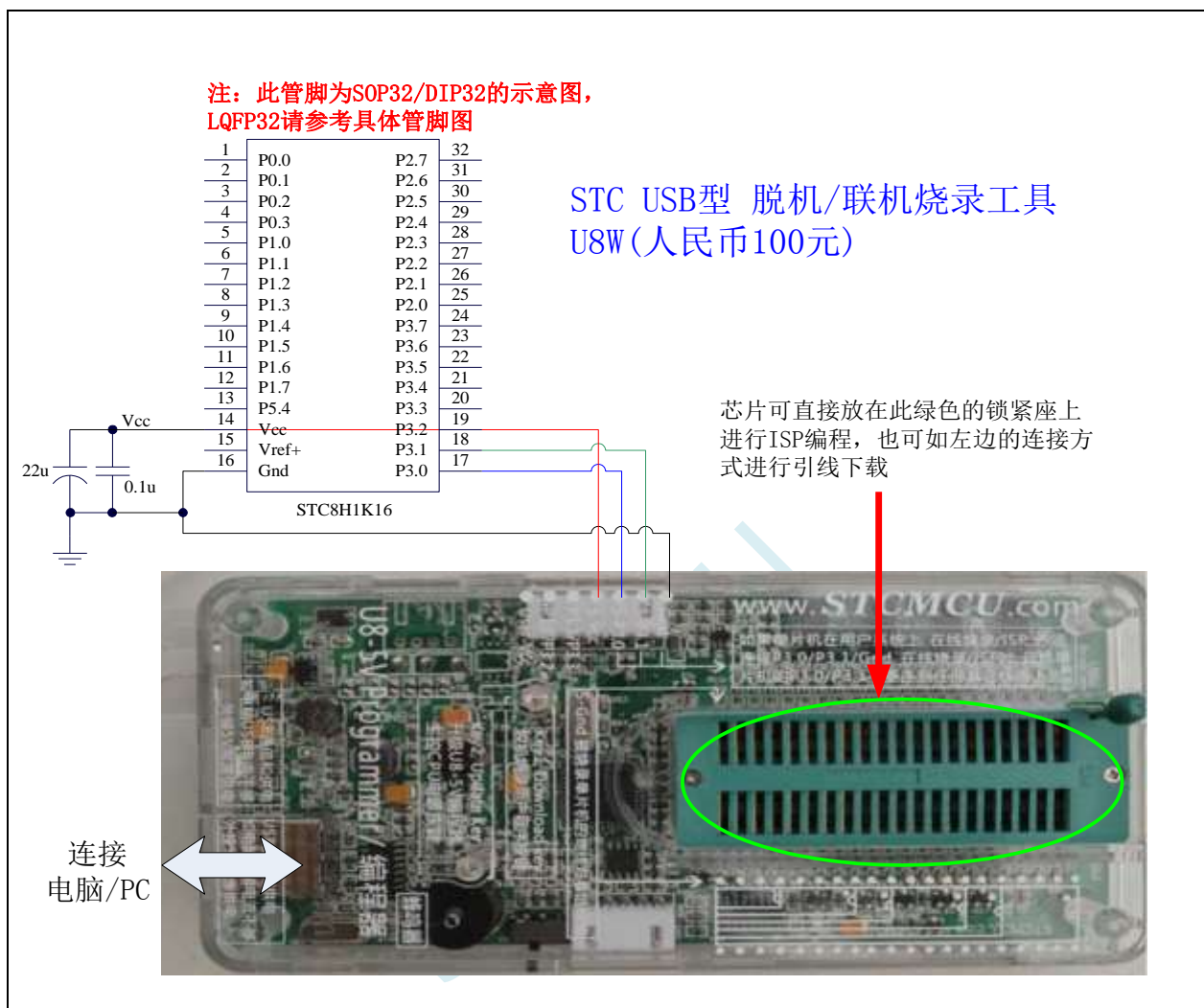
注意: 若是使用 U8-Mini 给目标系统供电, 目标系统的总电流不能大于 200mA, 否则会导致下载失败。

注意: 目前有发现使用 USB 线供电进行 ISP 下载时, 由于 USB 线太细, 在 USB 线上的压降过大, 导致 ISP 下载时供电不足, 所以请在使用 USB 线供电进行 ISP 下载时, 务必使用 USB 加强线。

若要使用 U8-Mini 进行仿真, 首先必须将 U8-Mini 设置为直通模式。U8W/U8W-Mini 实现 USB 转串口直通模式的方法如下:

- 1、首先 U8W/U8W-Mini 固件必须升级到 v1.37 及以上版本
- 2、U8W/U8W-Mini 上电后为正常下载模式, 此时按住工具上的 Key1 (下载) 按键不要松开, 再按一下 Key2 (电源) 按键, 然后松开 Key2 (电源) 按键, 再松开 Key1 (下载) 按键, U8W/U8W-Mini 会进入 USB 转串口直通模式。(按下 Key1 → 按下 Key2 → 松开 Key2 → 松开 Key1)
- 3、进入直通模式的 U8W/U8W-Mini 工具只是简单的 USB 转串口不具备脱机下载功能, 若需要恢复 U8W/U8W-Mini 的原有功能, 只需要再次单独按一下 Key2 (电源) 按键 即可

### 5.1.8 使用 U8W 工具下载，支持 ISP 在线和脱机下载，也可支持仿真



#### ISP 下载步骤（连线方式）：

- 1、按照如图所示的连接方式将 U8W 和目标芯片连接
- 2、点击 STC-ISP 下载软件中的“下载/编程”按钮
- 3、开始 ISP 下载

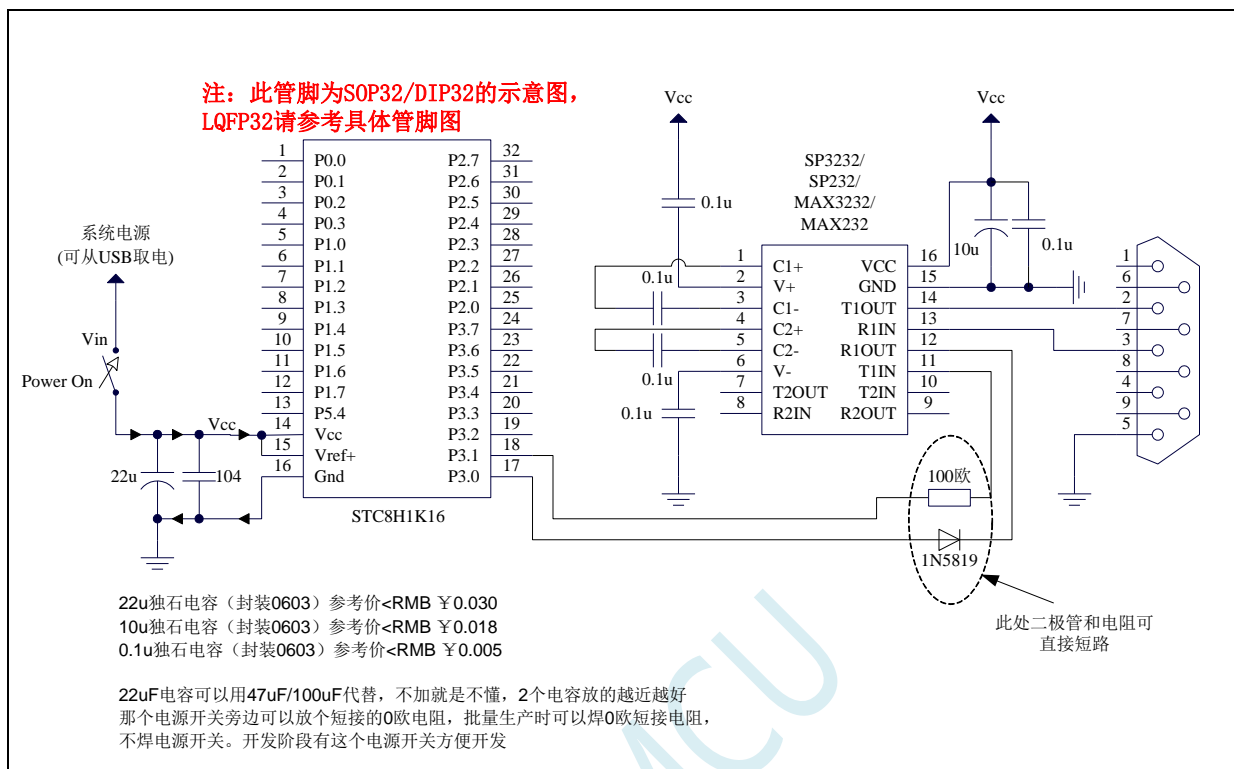
注意：若是使用 U8W 给目标系统供电，目标系统的总电流不能大于 200mA，否则会导致下载失败。

#### ISP 下载步骤（在板方式）：

- 1、将芯片按照 1 脚靠近锁紧扳手、管脚向下靠齐的方向放置好目标芯片
- 2、点击 STC-ISP 下载软件中的“下载/编程”按钮
- 3、开始 ISP 下载

注意：目前有发现使用 USB 线供电进行 ISP 下载时，由于 USB 线太细，在 USB 线上的压降过大，导致 ISP 下载时供电不足，所以请在使用 USB 线供电进行 ISP 下载时，务必使用 USB 加强线。

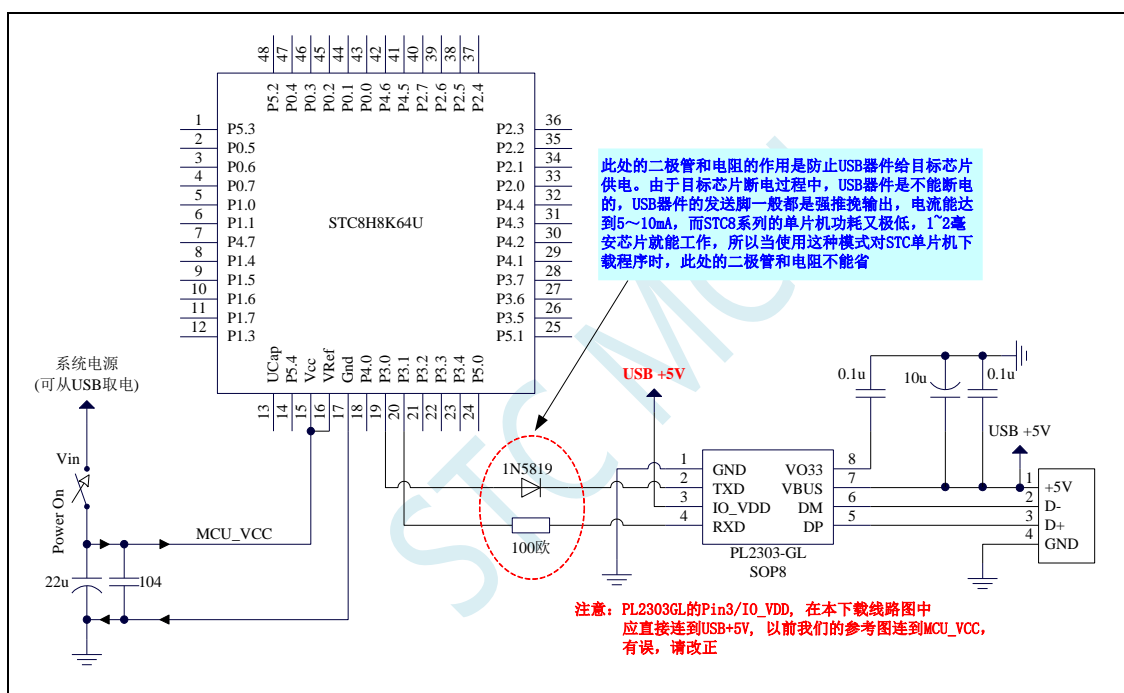
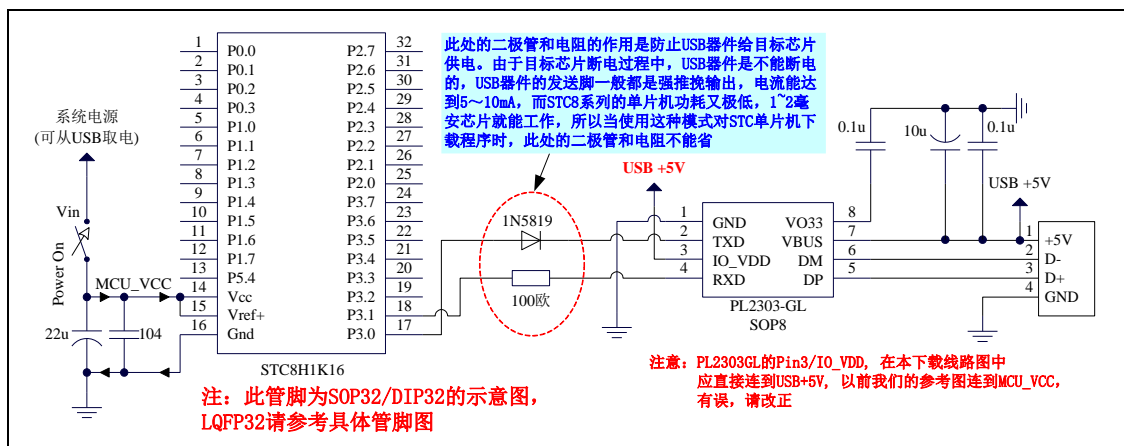
## 5.1.9 使用 RS-232 转换器下载，也可支持仿真



### ISP 下载步骤:

- 1、给目标芯片停电
- 2、点击 STC-ISP 下载软件中的“下载/编程”按钮
- 3、给目标芯片上电
- 4、开始 ISP 下载

### 5.1.10 使用 PL2303-GL 下载，也可支持仿真



#### ISP 下载步骤:

- 1、给目标芯片停电，注意不能给 USB 转串口芯片停电（如：CH340、PL2303-GL 等）  
注意：PL2303-SA 的部分波特率误差非常大，建议使用 PL2303-GL
- 2、由于 USB 转串口芯片的发送脚一般都是强推挽输出，必须在目标芯片的 P3.0 口和 USB 转串口芯片的发送脚之间串接一个二极管，否则目标芯片无法完全断电，达不到给目标芯片停电的目标。
- 3、点击 STC-ISP 下载软件中的“下载/编程”按钮
- 4、给目标芯片上电
- 5、开始 ISP 下载

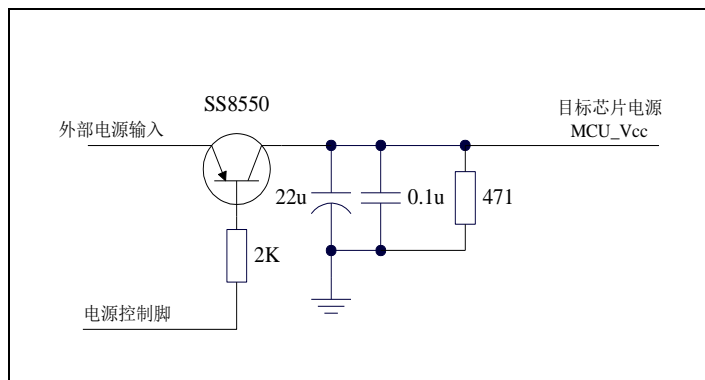
注意：目前有发现使用 USB 线供电进行 ISP 下载时，由于 USB 线太细，在 USB 线上的压降过大，导致 ISP 下载时供电不足，所以请在使用 USB 线供电进行 ISP 下载时，务必使用 USB 加强线。

若要使用 U8W 进行仿真，首先必须将 U8W 设置为直通模式。U8W/U8W-Mini 实现 USB 转串口直通模式的方法如下：

- 1、首先 U8W/U8W-Mini 固件必须升级到 v1.37 及以上版本
- 2、U8W/U8W-Mini 上电后为正常下载模式，此时按住工具上的 Key1（下载）按键不要松开，再按一下 Key2（电源）按键，然后松开 Key2（电源）按键 后，再松开 Key1（下载）按键，U8W/U8W-Mini 会进入 USB 转串口直通模式。（按下 Key1 → 按下 Key2 → 松开 Key2 → 松开 Key1）
- 3、进入直通模式的 U8W/U8W-Mini 工具只是简单的 USB 转串口不具备脱机下载功能，若需要恢复 U8W/U8W-Mini 的原有功能，只需要再次单独按一下 Key2（电源）按键 即可

STC MCU

### 5.1.11 单机电源控制参考电路

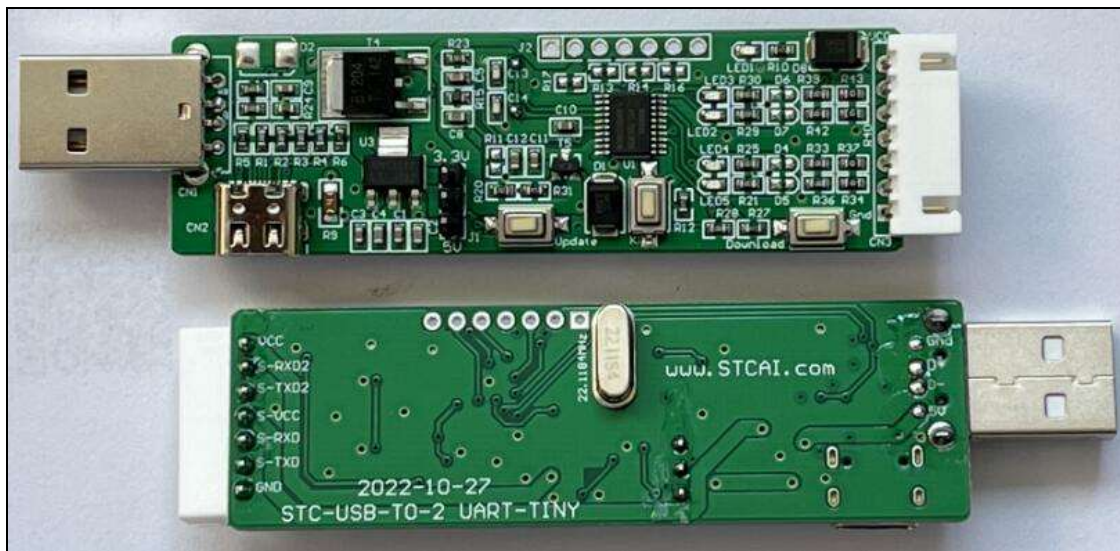




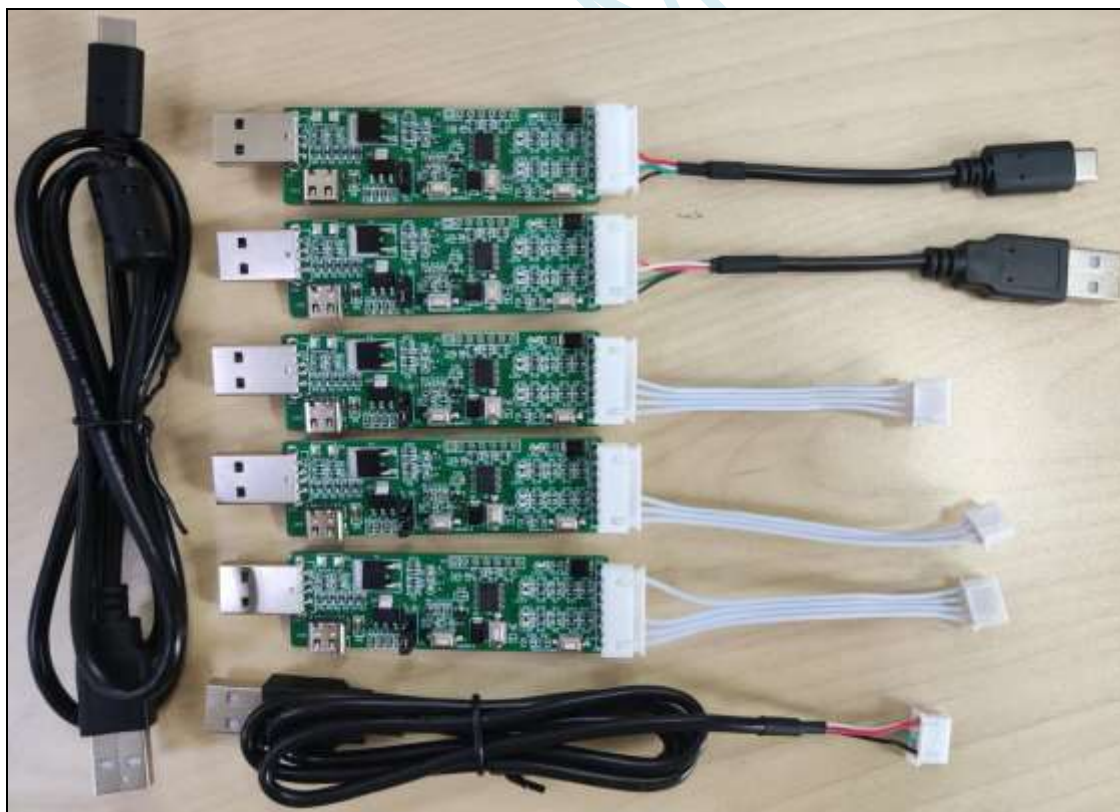
## 5.2 用 STC 一箭双雕之 USB 转双串口仿真 STC8 系列 MCU

先简单介绍下一箭双雕之 USB 转双串口工具

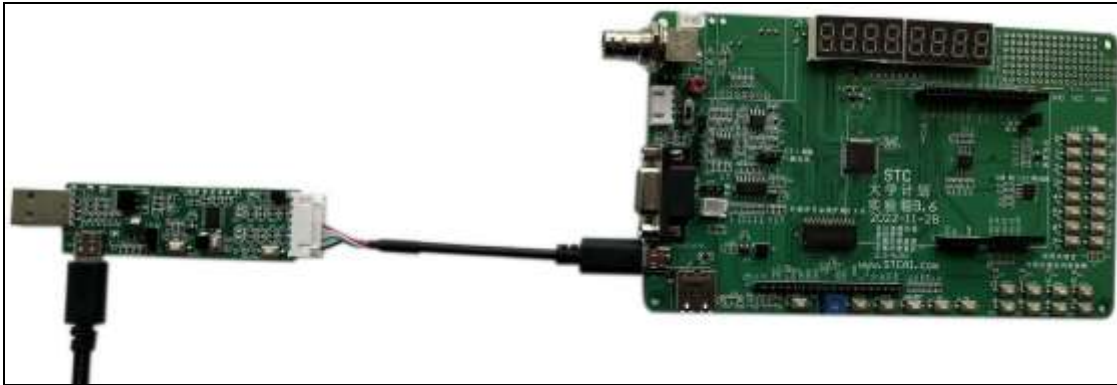
1、一箭双雕之 USB 转双串口工具外观图:



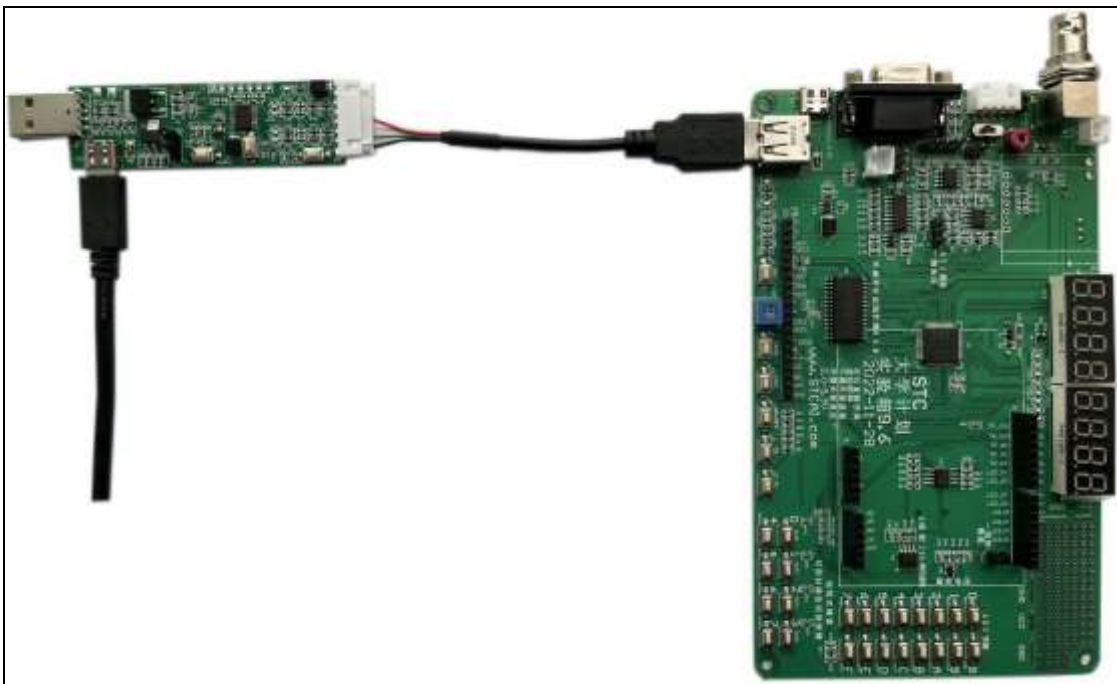
2、一箭双雕之 USB 转双串口工具几种常用连接线:



3、一箭双雕用 SIP7-USB-TypeC 对 STC8 系列 MCU 进行仿真/烧录，硬件连接图如下：



4、一箭双雕用 SIP7-USB-TypeA 对 STC8 系列 MCU 进行仿真/烧录，硬件连接图如下：

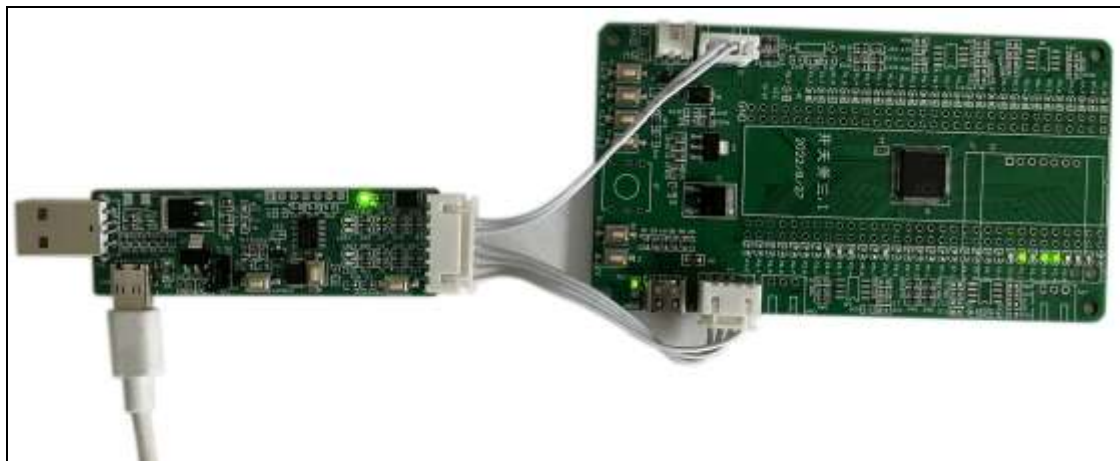


5、一箭双雕用 SIP7-SIP4/2.54mm 普通插座对 STC8 系列 MCU 进行仿真/烧录，硬件连接图如下：





6、一箭双雕 ,USB 扩展的 USB-CDC 串口 1 仿真; 扩展的 USB-CDC 串口 2 与其他串口进行通信, 硬件连接图如下:



7、将一箭双雕设置成普通的下载工具，可以参考这个官网论坛的这个帖子：

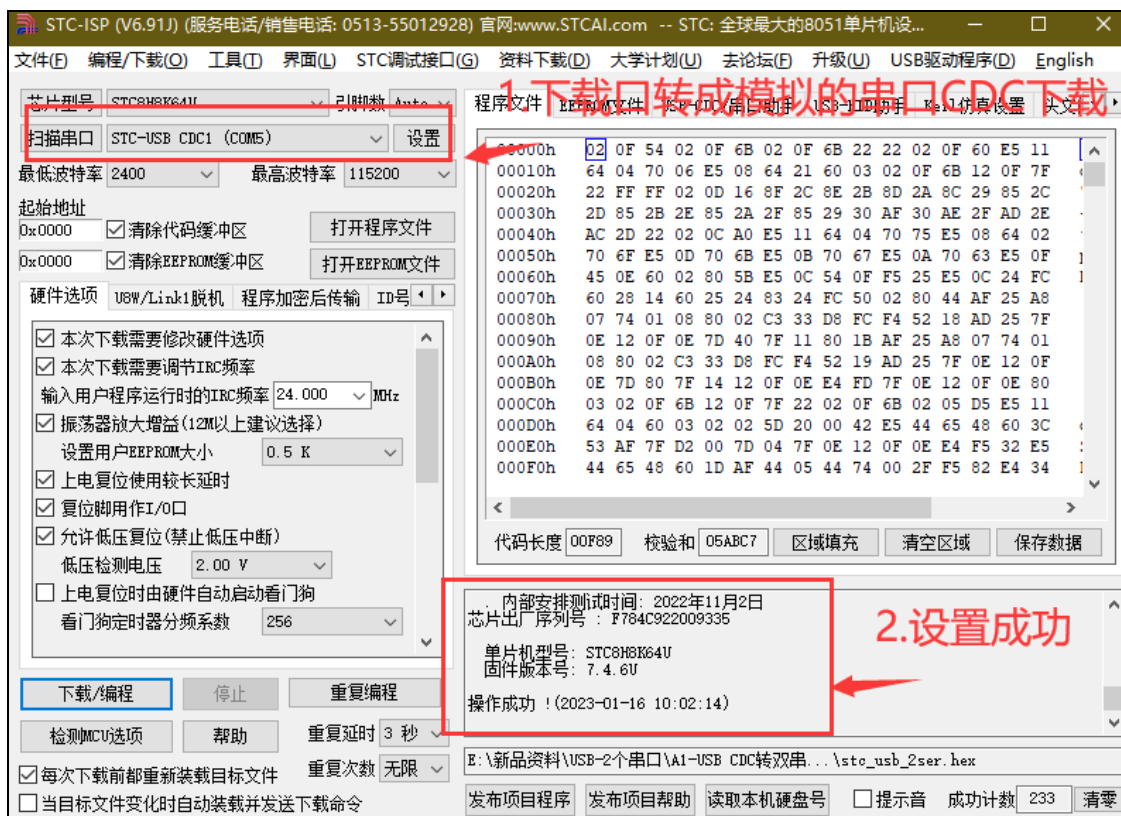
<https://www.stcaimcu.com/forum.php?mod=viewthread&tid=240&highlight=%E4%B8%80%E7%AE%AD%E5%8F%8C%E9%9B%95>

拿到 USB 转双串口工具后可对其烧录不同的固件来实现不同的功能，例如做串口工具、做烧录工具、做 OLED 示波器等。固件烧录流程如下：

- 1) 使用 USB-TypeC 数据线或者通过 USB-TypeA 接口连接核心板到电脑；
- 2) 按住 P3.2 口按键不放；
- 3) 按一下电源开关按键（按下-松开），然后可松开 P3.2 口按键；

正常情况下在 STC-ISP 软件上就可以识别出“STC USB Writer (HID1)”设备：





软件设置如下:

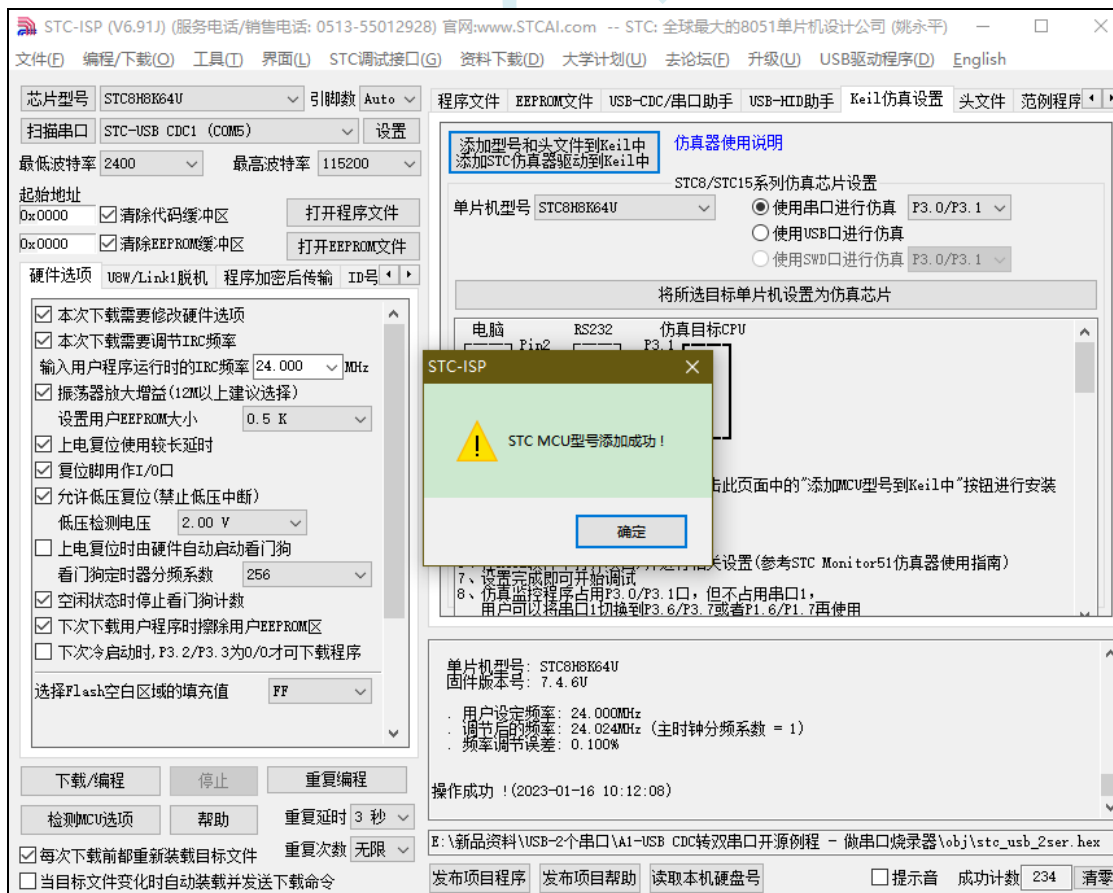
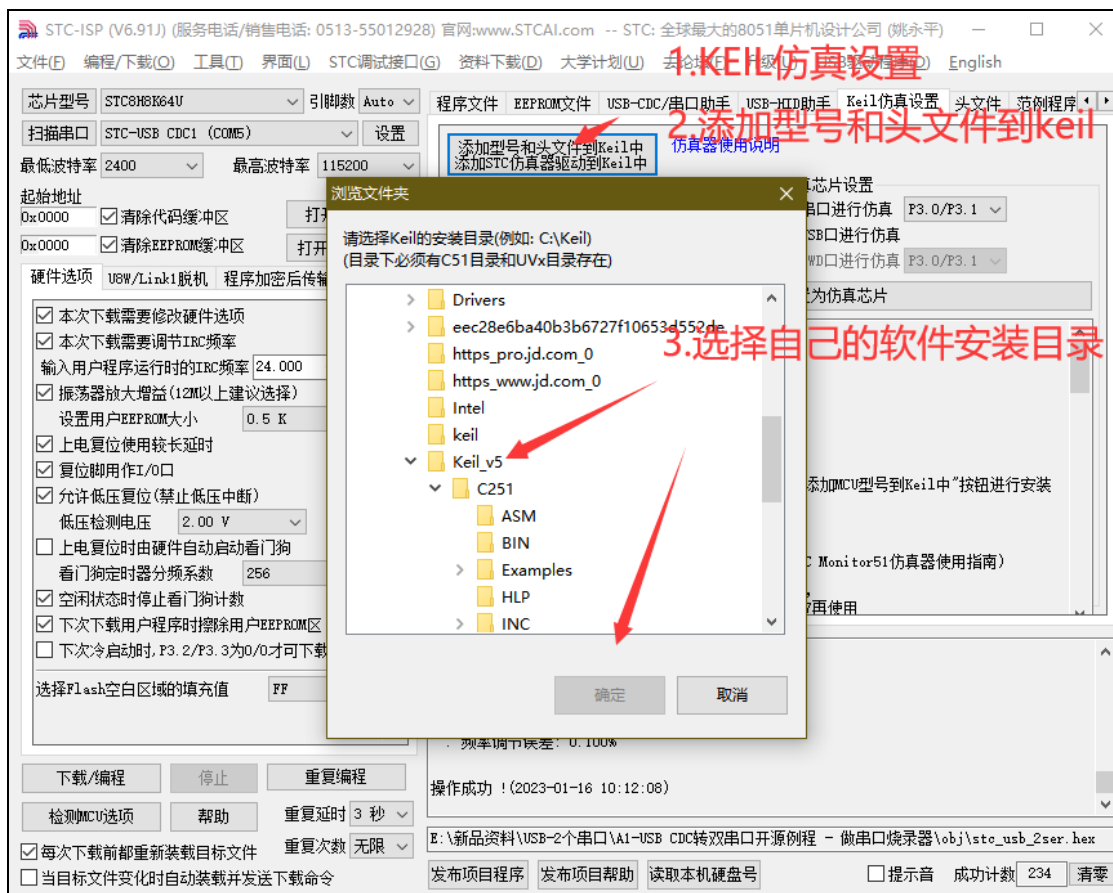
8、先去官网下载最新的 STC-ISP 软件, 截止至目前最新版本是 STC-ISP (6.91J), 特别是仿真这块, STC-ISP (6.91J) 的 stcmon51 仿真驱动程序版本已更新至 v1.18, 经内部反复测试已经非常稳定。

(下载地址: 工具软件-深圳国芯人工智能有限公司 <https://www.stcai.com/gjrz>)



## 9、更新 KEIL 中的 STC 的资源：添加 STC 仿真器的固件和芯片型号到 KEIL 中。

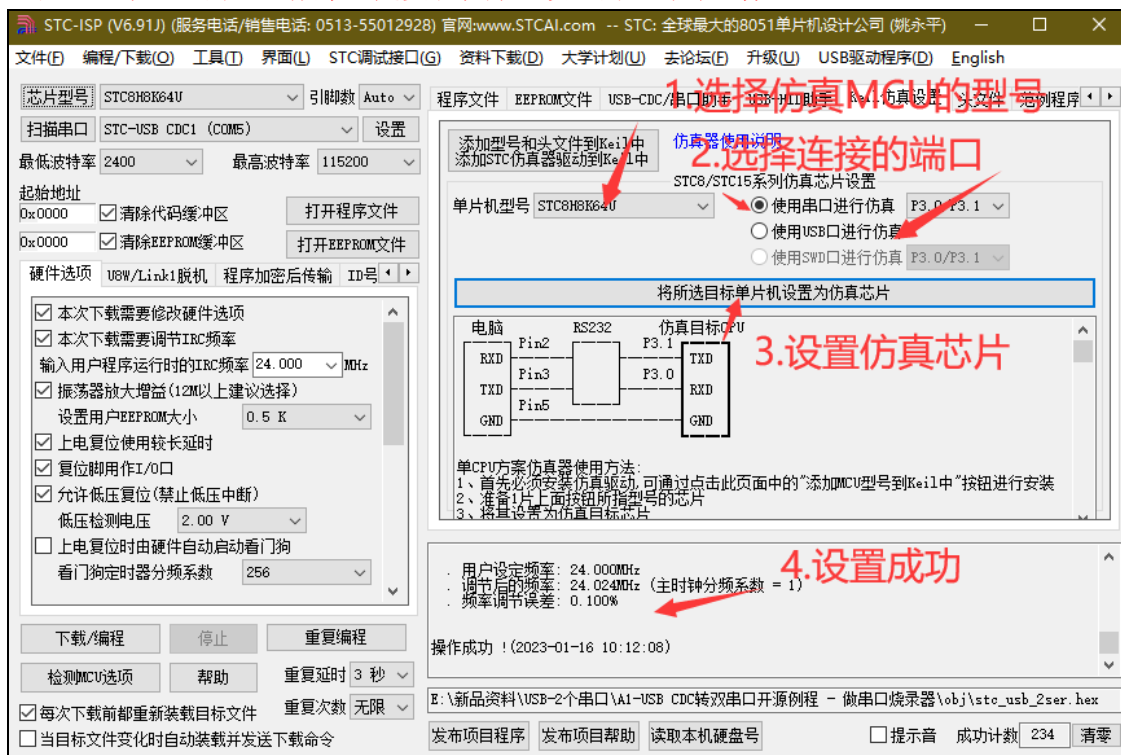
(此步骤建议在每次 ISP 下载软件更新时都重新添加一次，保证仿真驱动都是最新的版本)



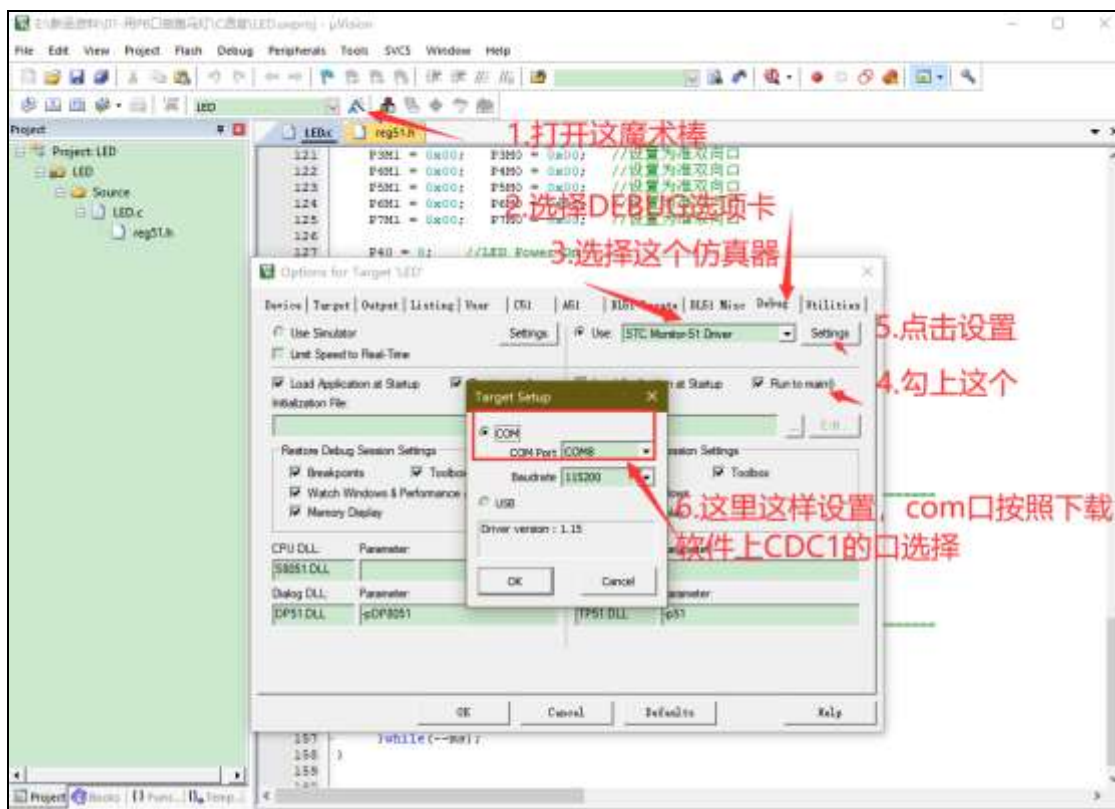


10、现在开始进行 KEIL 仿真的步骤，先将 STC8H8K64U 设置为仿真芯片，STC8H8K64U 目前仅支持串口和 USB 直接仿真。（这里选择了 P3.0/3.1 作为仿真端口，所以程序里不能出现任何占用 3.0 和 3.1 引脚的功能，仿真注意事项贴中也会说明，**先用点亮一个 LED 的程序进行测试，比较容易观察结果！**）此时连接 STC8 的芯片，然后进行如下的设置将开天斧设置成仿真的主控芯片。

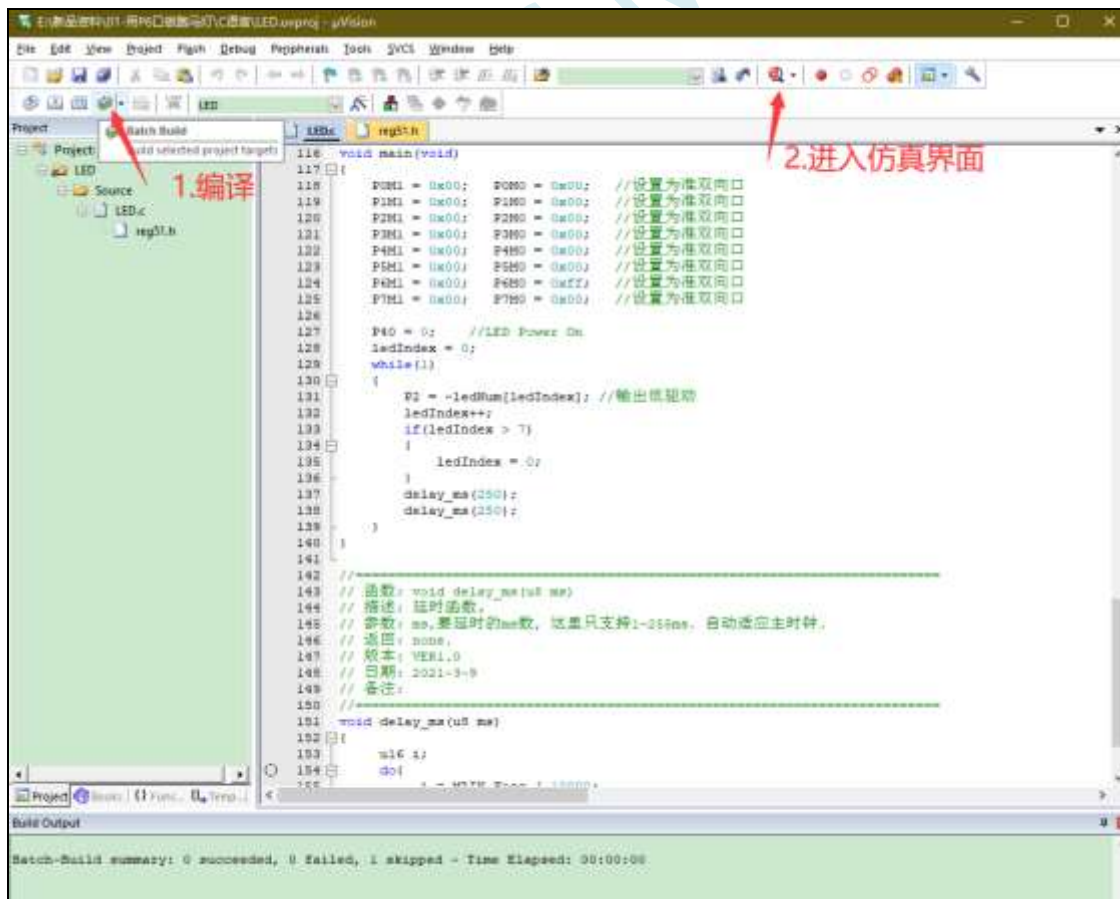
（注意一下这里的 IRC 频率一定要和程序里设置的主时钟一样！！）



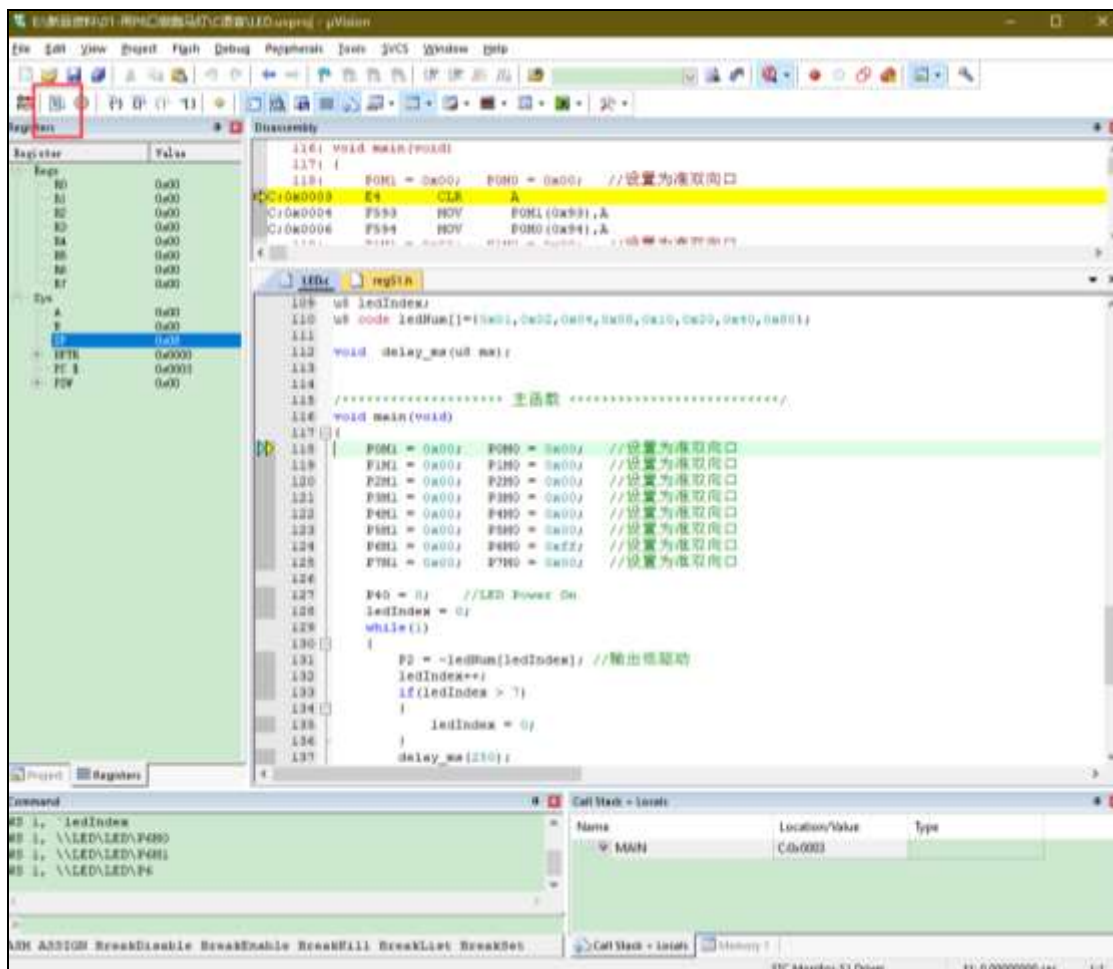
11、开天斧设置为仿真芯片（如何在 keil 中创建工程不在此贴详述），打开 KEIL 中建好的工程项目，直接去 KEIL 中设置仿真的路径。



12、这样就可以编译并且调试了。



13、出现下面这个界面，说明已经成功的进入了仿真模式，然后就可以用变量监测，断点等等的功能。



## 5.3 STC-ISP 下载软件高级应用

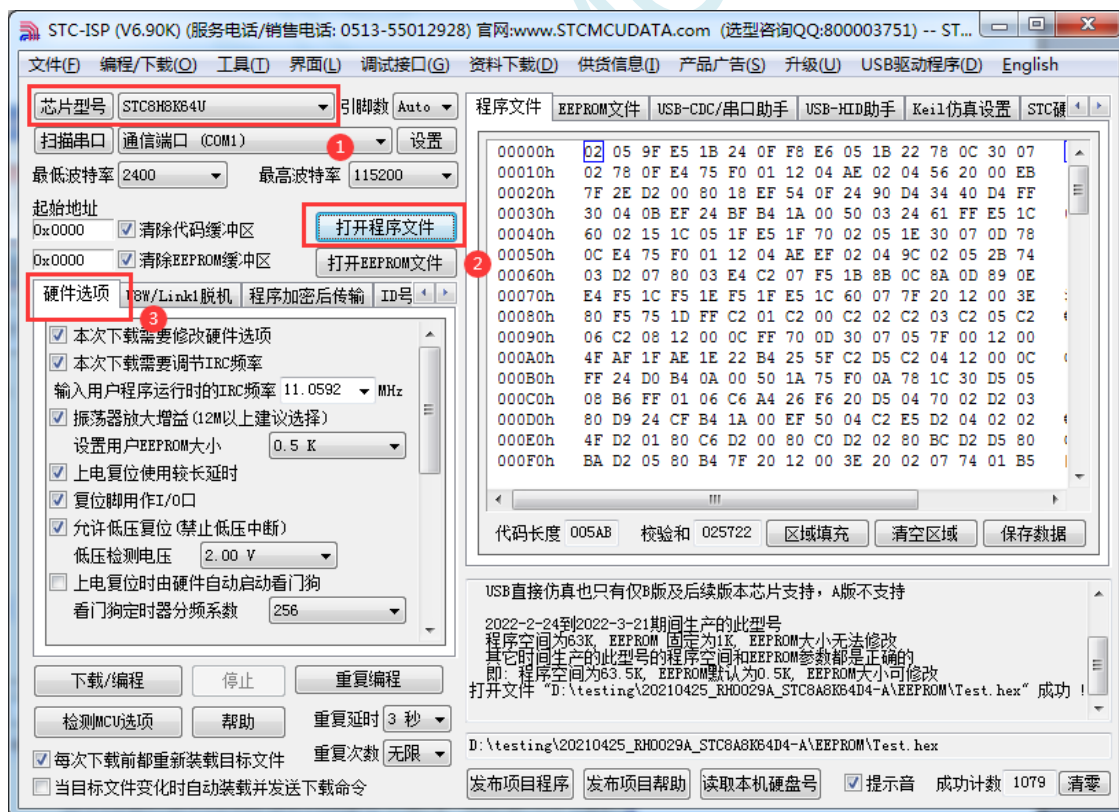
### 5.3.1 发布项目程序

发布项目程序功能主要是将用户的程序代码与相关的选项设置打包成为一个可以直接对目标芯片进行下载编程的超级简单的用户自己界面的可执行文件。

关于界面, 用户可以自己进行定制 (用户可以自行修改发布项目程序的标题、按钮名称以及帮助信息), 同时用户还可以指定目标电脑的硬盘号和目标芯片的 ID 号, 指定目标电脑的硬盘号后, 便可以控制发布应用程序只能在指定的电脑上运行 (防止烧录人员将程序轻易从电脑盗走, 如通过网络发走, 如通过 U 盘拷走, 防不胜防, 当然盗走你的电脑那就没办法那, 所以 STC 的脱机下载工具比电脑烧录安全, 能限制可烧录芯片数量, 让前台文员小姐烧, 让老板娘烧都可以), 拷贝到其它电脑, 应用程序不能运行。同样的, 当指定了目标芯片的 ID 号后, 那么用户代码只能下载到具有相应 ID 号的目标芯片中 (对于一台设备要卖几千万的产品特别有用---坦克, 可以发给客户自己升级, 不需冒着生命危险跑到战火纷飞的伊拉克升级软件啦), 对于 ID 号不一致的其它芯片, 不能进行下载编程。

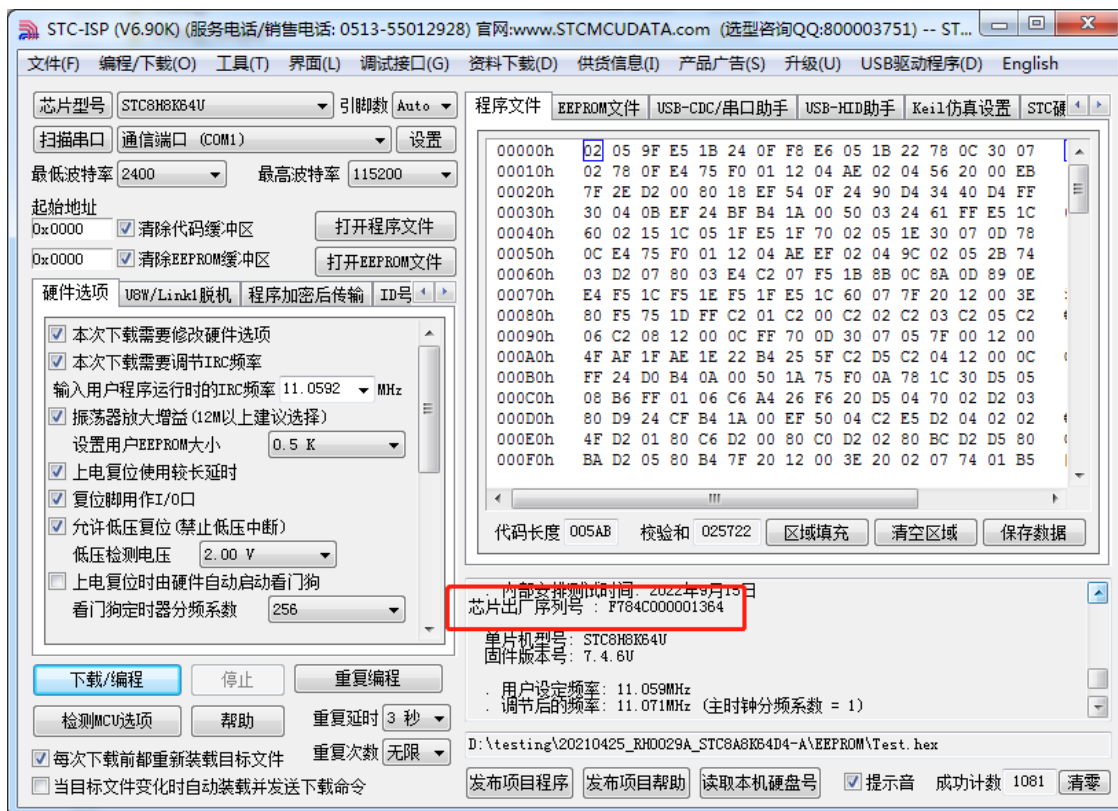
发布项目程序详细的操作步骤如下:

- 1、首先选择目标芯片的型号
- 2、打开程序代码文件
- 3、设置好相应的硬件选项

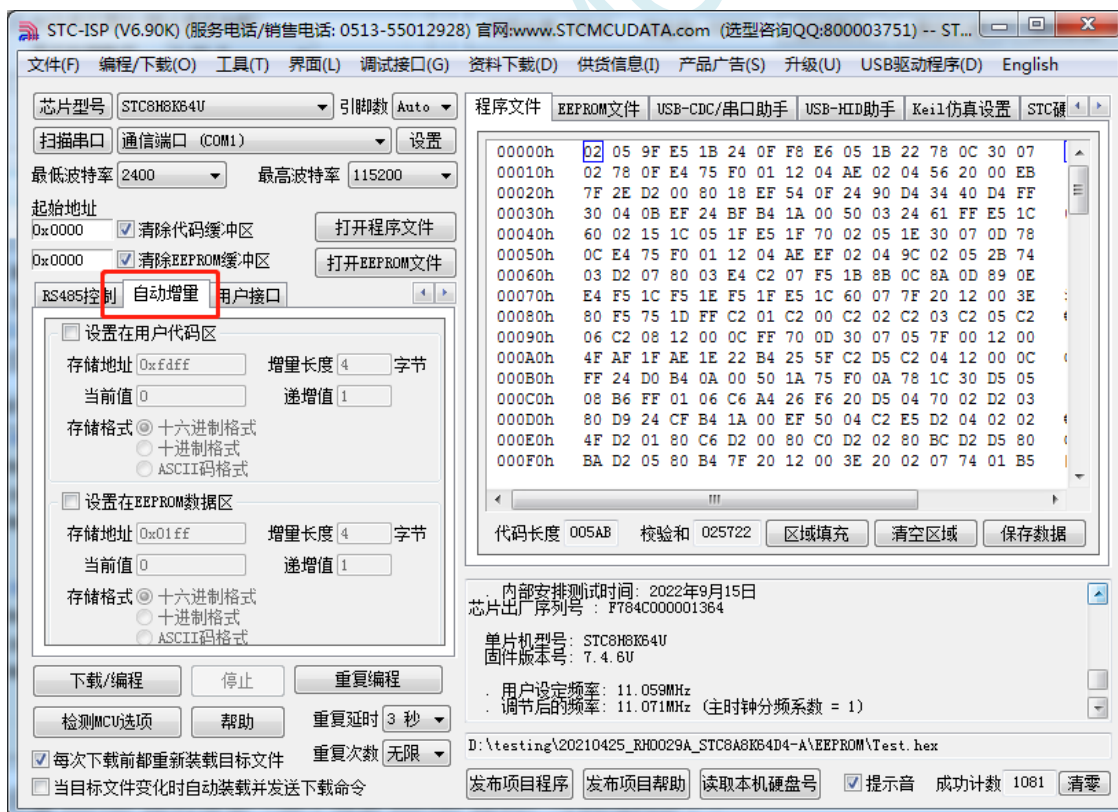


- 4、试烧一下芯片, 并记下目标芯片的 ID 号, 如下图所示, 该芯片的 ID 号即为 “F784C000001364” (如不需要对目标芯片的 ID 号进行校验, 可跳过此步)

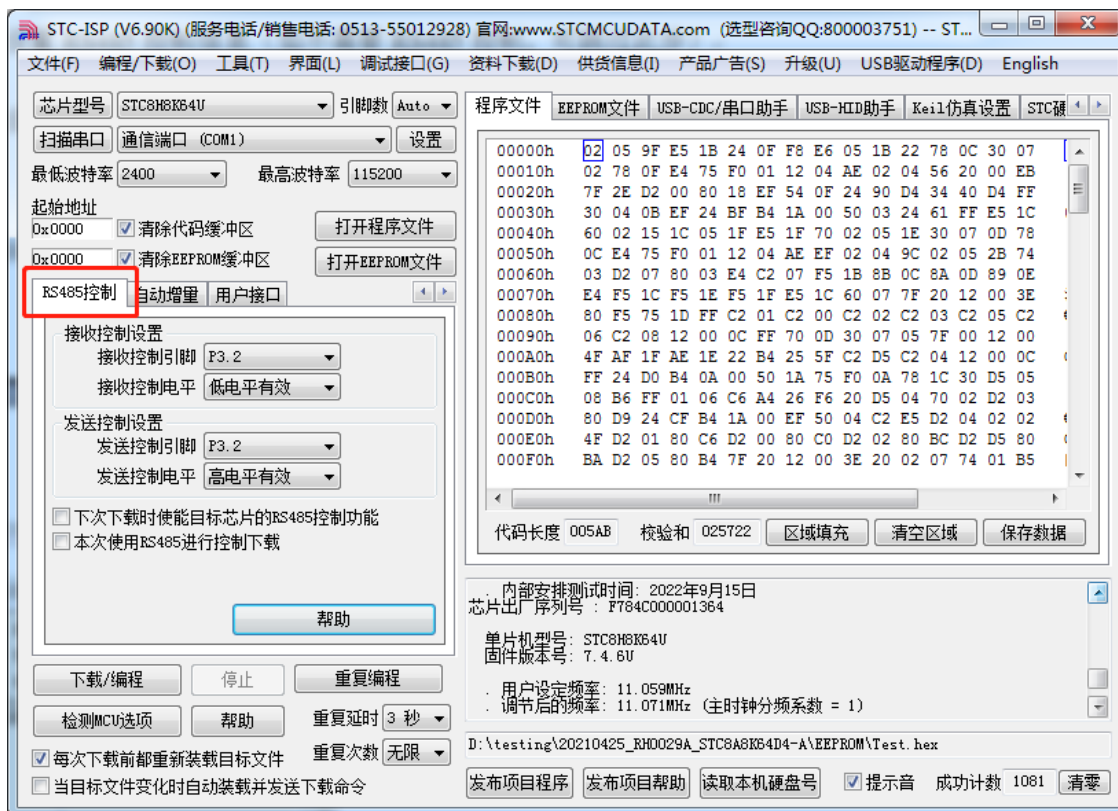




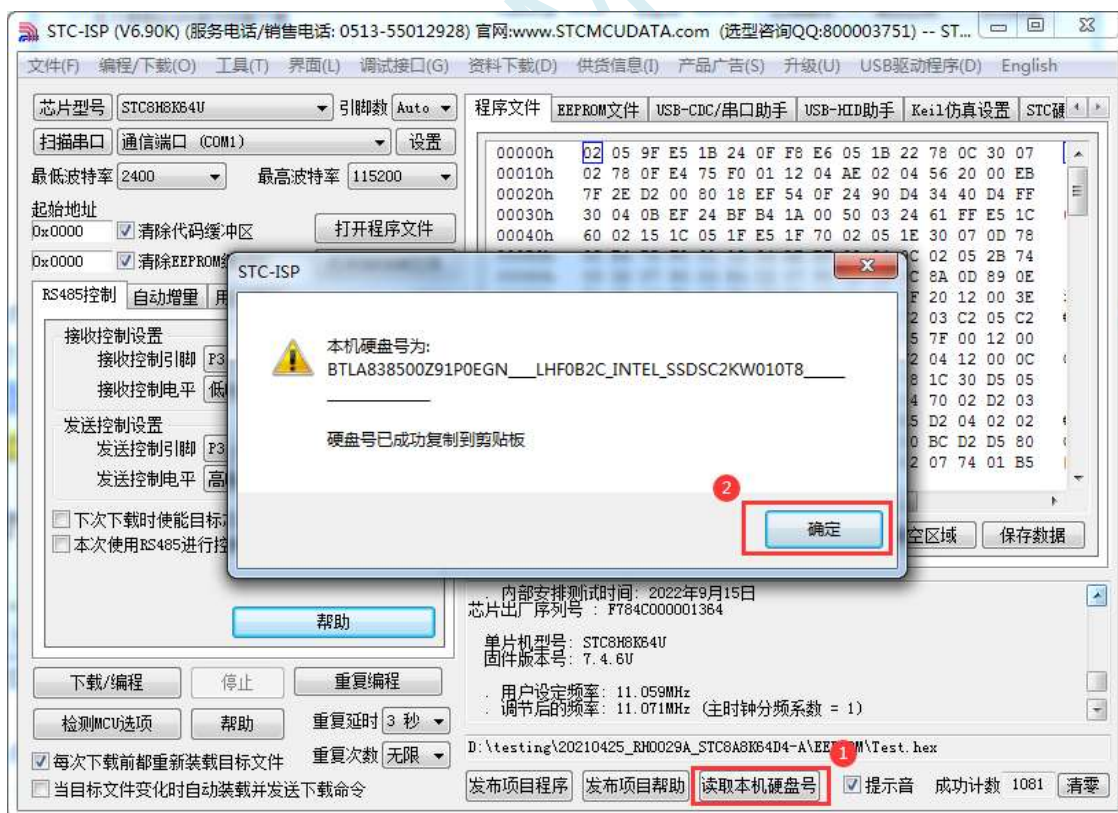
##### 5、设置自动增量（如不需要自动增量，可跳过此步）



##### 6、设置 RS485 控制信息（如不需要 RS485 控制，可跳过此步）



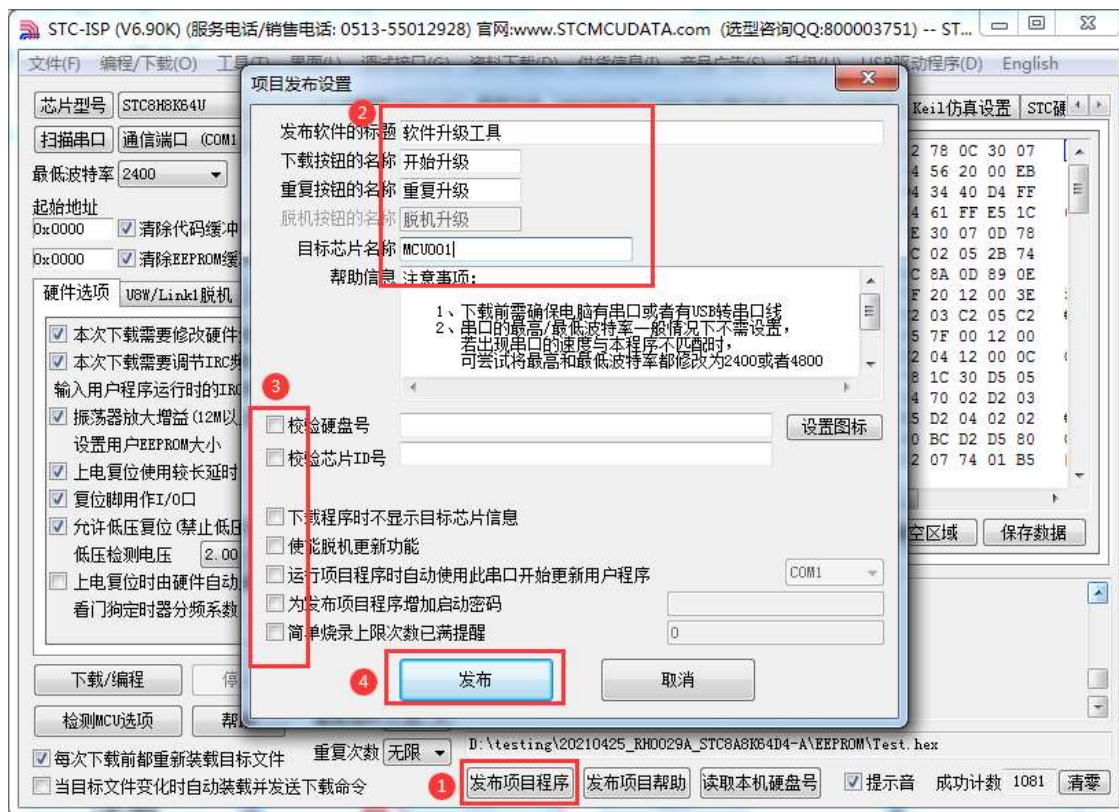
- 7、点击界面上的“读取本机硬盘号”按钮，并记下目标电脑的硬盘号（如不需要对目标电脑的硬盘号进行校验，可跳过此步）



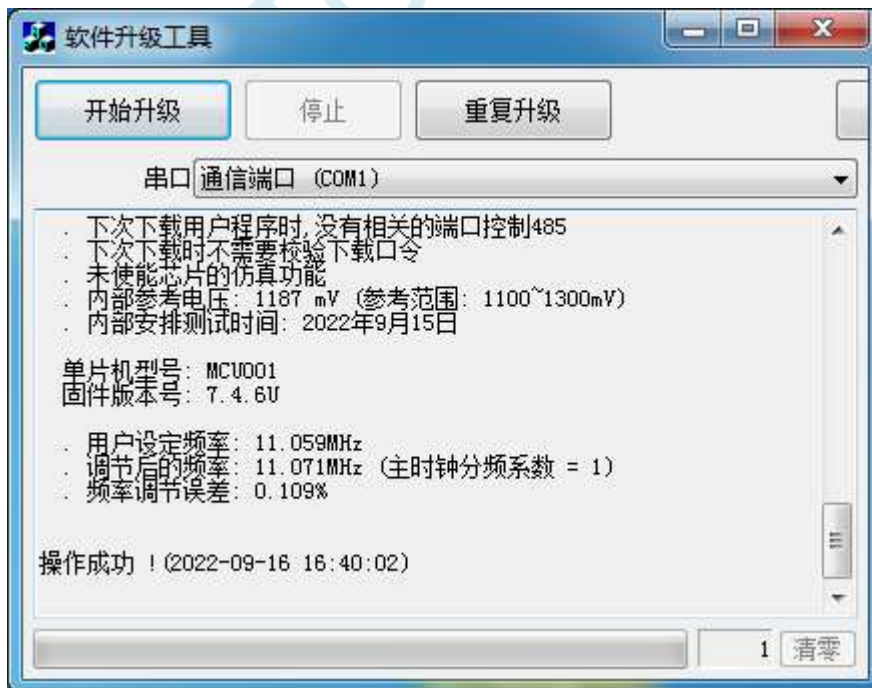
- 8、点击“发布项目程序”按钮，进入发布应用程序的设置界面。
- 9、根据各自的需要，修改发布软件的标题、下载按钮的名称、重复下载按钮的名称、自动增量的名称以及帮助信息
- 10、若需要校验目标电脑的硬盘号,则需要勾选上“校验硬盘号”，并在后面的文本框内输入前面所记

下的目标电脑的硬盘号

- 11、若需要校验目标芯片的 ID 号,则需要勾选上“校验芯片 ID 号”,并在后面的文本框内输入前面所记下的目标芯片的 ID 号



- 12、最后点击发布按钮,将项目发布程序保存,即可得到相应的可执行文件。发布的项目程序界面如下图





### 5.3.2 程序加密后传输（防烧录时串口分析出程序）

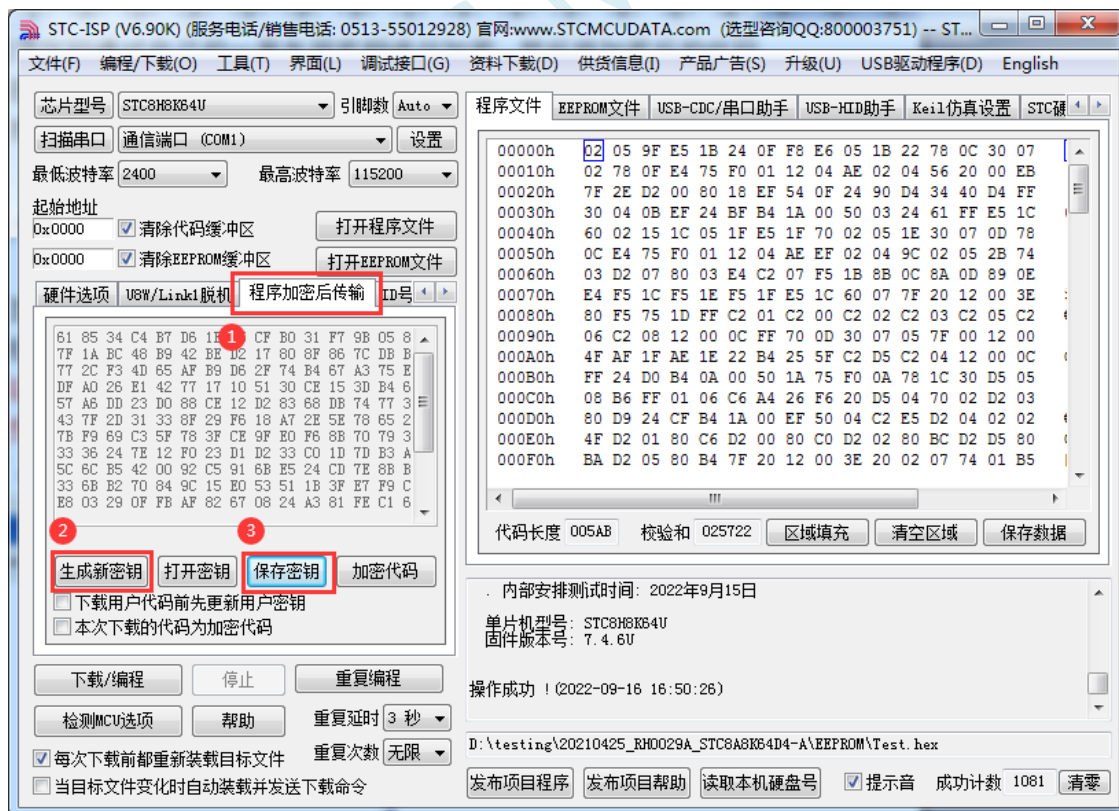
目前，所有的普通串口下载烧录编程都是采用明码通信的（电脑和目标芯片通信时，或脱机下载板和目标芯片通信时），问题：如果烧录人员通过分析下载烧录编程时串口通信的数据，高手是可以在烧录时在串口上引 2 根线出来，通过分析串口通信的数据分析出实际的用户程序代码的。当然用 STC 的脱机下载板烧程序总比用电脑烧程序强（防止烧录人员将程序轻易从电脑盗走，如通过网络发走，如通过 U 盘拷走，防不胜防，当然盗走你的电脑那就没办法那，所以 STC 的脱机下载工具比电脑烧录安全，让前台文员小姐烧，让老板娘烧都可以）。即使是 STC 全球首创的脱机下载工具，对于要防止天才的不法分子在脱机下载工具烧录的过程中通过分析串口通信的数据分析出实际的用户程序代码，也是没有办法达到要求的，这就需要用到最新的 STC 单片机所提供的程序加密后传输功能。

程序加密后传输下载是用户先将程序代码通过自己的一套专用密钥进行加密，然后将加密后的代码再通过串口下载，此时下载传输的是加密文件，通过串口分析出来的是加密后的乱码，如不通过派人潜入你公司盗窃你电脑里面的加密密钥，就无任何价值，便可起到防止在烧录程序时被烧录人员通过监测串口分析出代码的目的。

程序加密后传输功能的使用需要如下的几个步骤：

#### 1、生成并保存新的密钥

如下图，进入到“程序加密后传输”页面，点击“生成新密钥”按钮，即可在缓冲区显示新生成的 256 字节的密钥。然后点击“保存密钥”按钮，即可将生成的新密钥保存为以“.K”为扩展名的密钥文件（注意：这个密钥文件一定要保存好，以后发布的代码文件都需要使用这个密钥加密，而且这个密钥的生成是非重复的，即任何时候都不可能生成两个完全相同的密钥，所以一旦密钥文件丢失将无法重新获得）。例如我们将密钥保存为“New.k”。

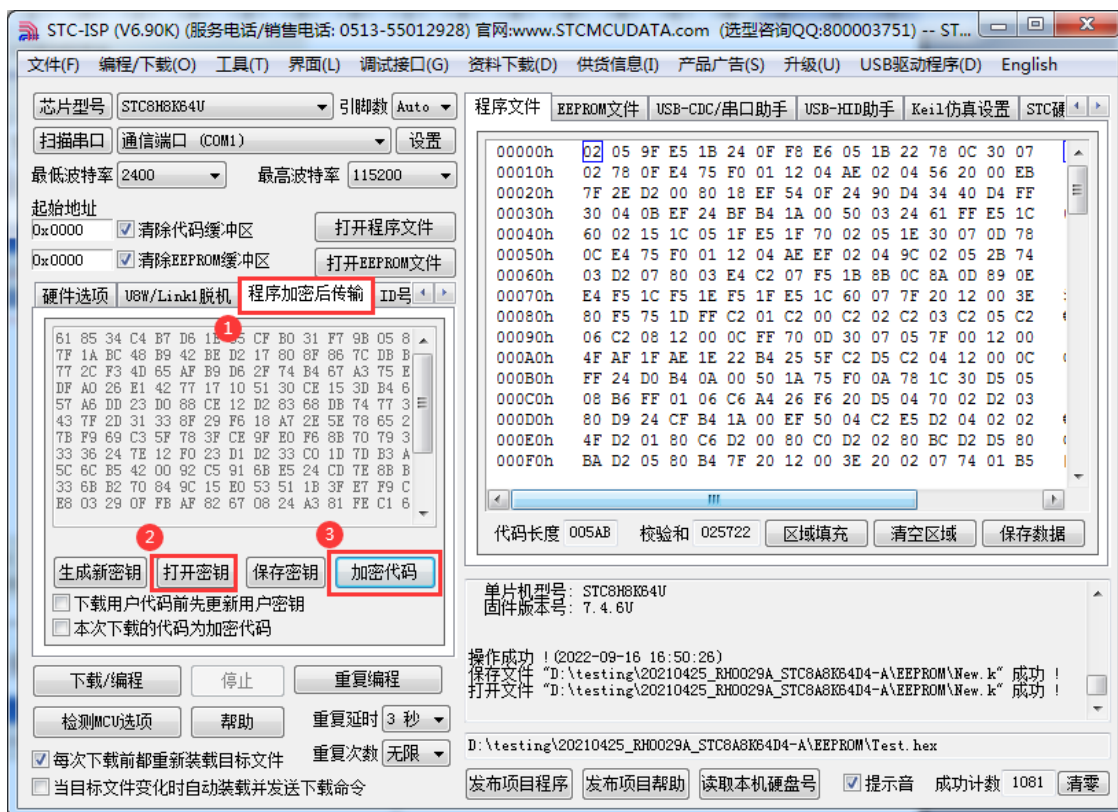


#### 2、对代码文件加密

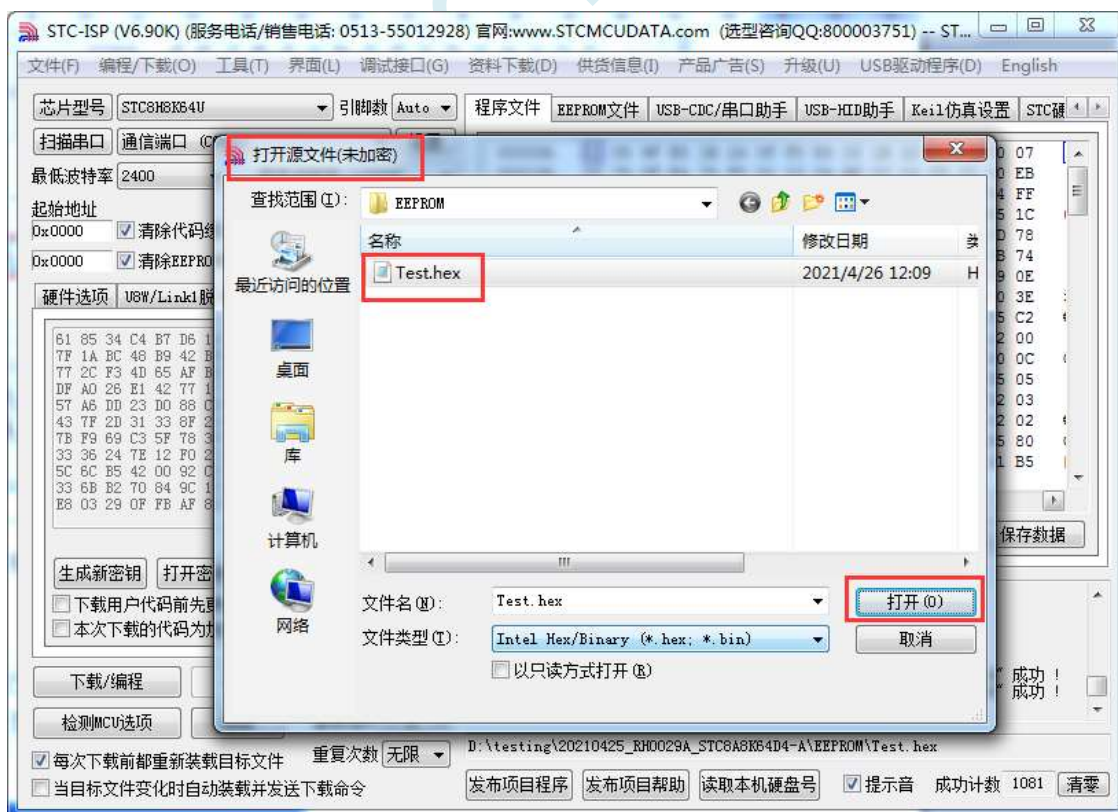
加密文件前，需要先打开我们自己的密钥。若缓冲区中存放的已经是我们的密钥，则不要再打开。

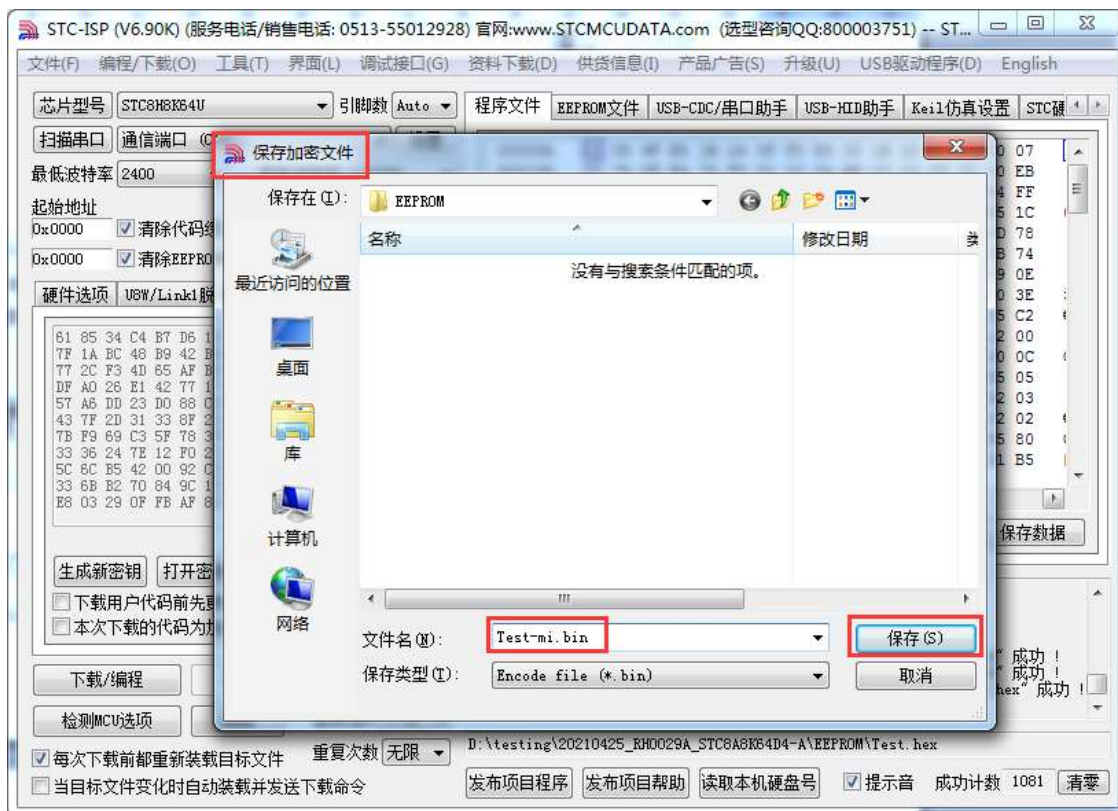
如下图，在“程序加密后传输”页面中点击“打开密钥”按钮，打开我们之前保存的密钥文件，

例如“New.k”，然后返回到“程序加密后传输”页面中点击“加密代码”按钮，如下图所示，首先会弹出“打开源文件（未加密）”的对话框，此时选择的是原始的未加密的代码文件



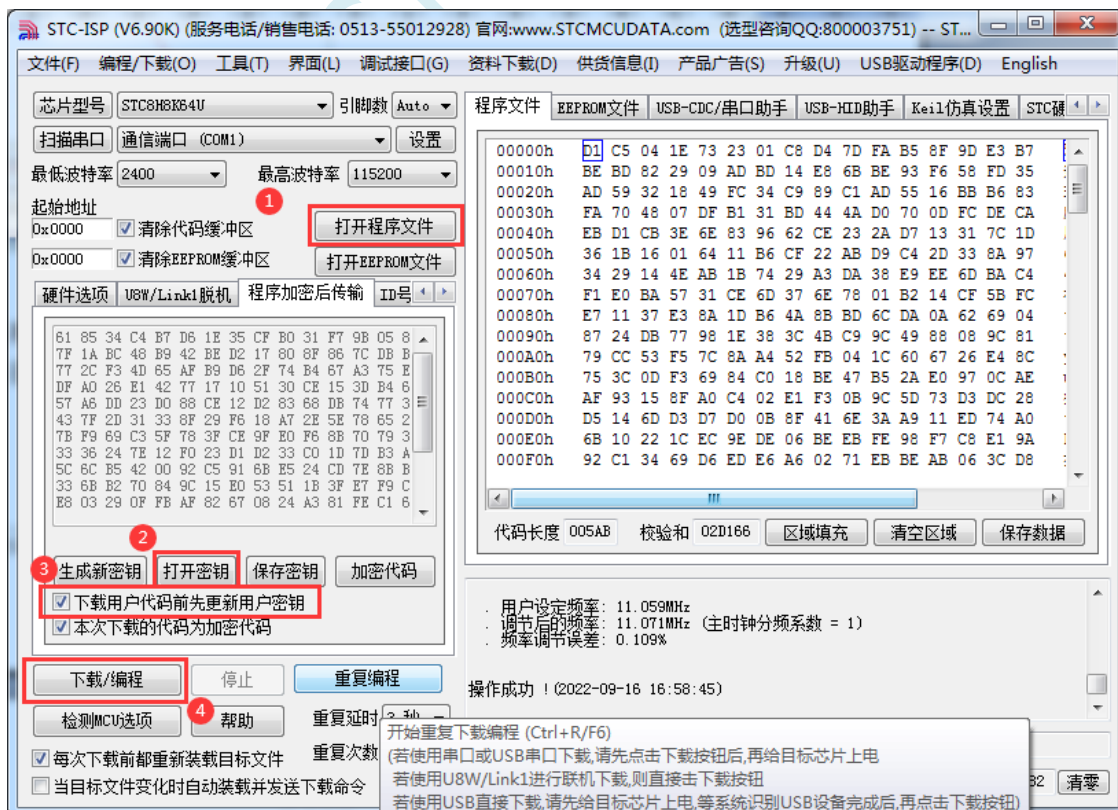
点击打开按钮后，马上会有会弹出一个类似的对话框，但此时是对加密后的文件进行保存的对话框。如下图所示，点击保存按钮即可保存加密后的文件。





### 3、将用户密钥更新到目标芯片中

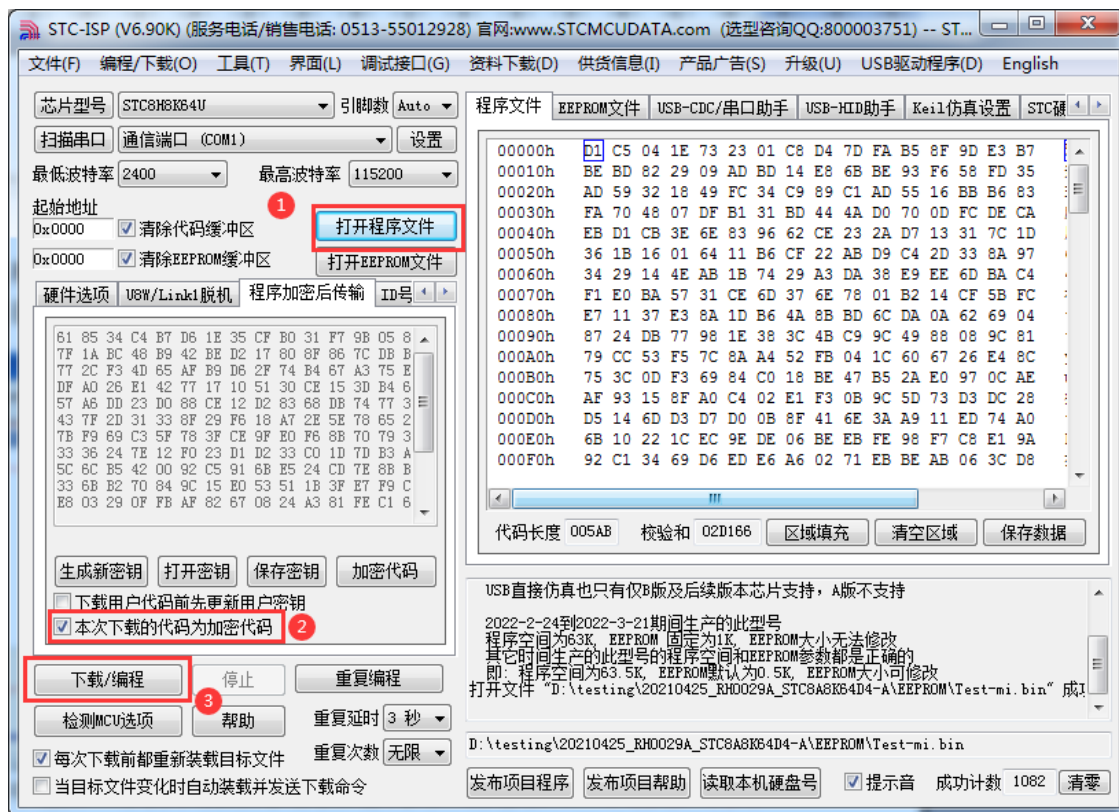
更新密钥前，需要先打开我们自己的密钥。若缓冲区中存放的已经是我们的密钥，则不要再打开。如下图，在“自定义加密下载”页面中点击“打开密钥”按钮，打开我们之前保存的密钥文件，例如“New.k”。密钥打开后，如下图所示，勾选上“下载用户代码前先更新用户密钥”选项和“本次下载的代码为加密码码”的选项，然后打开我们之前加密过后的文件，打开后点击界面左下角的“下载/编程”按钮，按正常方式对目标芯片下载完成即可更新用户密钥。





#### 4、加密更新用户代码

密钥更新成功后, 目标芯片便具有接收加密代码并还原的功能。此时若需要再次升级/更新代码, 则只需要参考第二步的方法, 将目标代码进行加密, 然后如下图



对于一片新的 STC 单片机, 可将步骤 3 和步骤 4 合并完成, 即将密钥更新到目标单片机的同时也可将加密后的代码一并下载到单片机中, 若已经执行过步骤 3 (即已经将密钥更新到目标芯片中了), 则后续的代码更新就只需要按照步骤 4, 只需要在“程序加密后传输”页面中选择“本次下载的代码为加密代码”的选项 (“下载用户代码前先更新用户密钥”选项不需要选了), 然后打开我们之前加过密后的文件, 打开后点击界面左下角的“下载/编程”按钮, 按正常方式对目标芯片下载即可完成用用户自己专用的加密文件更新用户代码的目的 (防止在烧录程序时被烧录人员通过监测串口分析出代码的目的)。

### 5.3.3 发布项目程序+程序加密后传输结合使用

发布项目程序与程序加密后传输两项新的特殊功能可以结合在一起使用。首先程序加密后传输可以确保用户代码在烧录编程时串口通信传输过程当中的保密性，而发布项目程序可实现让最终使用者远程升级功能（方案公司的人员不需要亲自到场）。所以两项功能结合起来使用，非常适用于方案公司/生产商在软件需要更新时，让最终使用者自己对终端产品进行软件更新的目的，又确保现场烧录人员无法通过串口分析出有用程序，强烈建议方案公司使用。

STC MCU

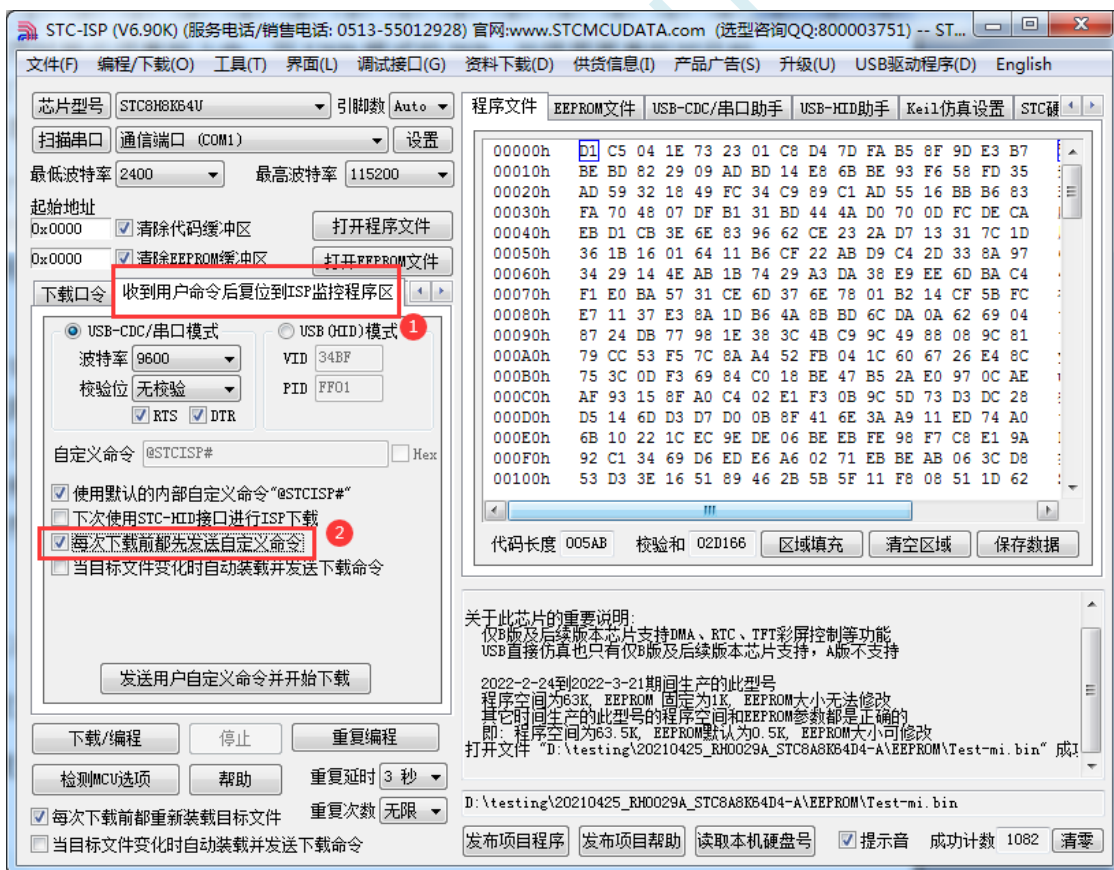


### 5.3.4 用户自定义下载（实现不停电下载）

将用户的目标程序下载到STC单片机是通过执行单片机内部的ISP系统代码和上位机进行串口或者USB通讯来实现的。但STC单片机内部的ISP系统代码只有在每次重新停电再上电时才会被执行，这就要求用户每次需要对目标单片机更新程序时必须重新上电，而USB模式的ISP，处理需要重新对目标芯片上电外，还需要在上电时将P3.2口下拉到GND。对于处于开发阶段的项目，需要频繁的修改代码、更新代码，每次下载都需要重新上电会导致操作非常麻烦。

STC单片机在硬件设计时，增加了一个软复位寄存器（IAP\_CONTR），让用户可以通过设置此寄存器来决定CPU复位后重新执行用户代码还是复位到ISP区执行ISP系统代码。当向IAP\_CONTR寄存器写入0x20时，CPU复位后重新执行用户代码；当向IAP\_CONTR寄存器写入0x60时，CPU复位后复位到ISP区执行ISP系统代码。

要实现不停电进行ISP下载，用户可以在程序中设计一段代码，例如检测一个特殊的按键、或者监控串口等待一个特殊的串口命令，当检测到满足下载条件时，就通过软件触发软复位寄存器复位到ISP区执行ISP系统代码，从而实现不停电ISP下载。当触发条件是外部按键时，则在用户代码中实时监控按键状态即可。若要实现STC-ISP软件和用户触发软复位完全同步，则需要使用STC-ISP软件中所提供的“收到用户命令后复位到ISP监控程序区”这个功能。



实现不停电 ISP 下载的步骤如下:

1、编写用户代码,并在用户代码中添加串口命令监控程序

(参考代码如下,测试单片机型号为 STC8H8K64U)

```
#include "stc8h.h"

#define FOSC 11059200UL
#define BAUD (65536 - (FOSC/115200+2)/4) //加 2 操作是为了让 Keil 编译器
//自动实现四舍五入运算

char code *STCISPCMD = "@STCISP#"; //自定义下载命令
char index;

void uart_isr() interrupt 4
{
    char dat;

    if (TI)
    {
        TI = 0;
    }

    if (RI)
    {
        RI = 0;
        dat = SBUF; //接收串口数据

        if (dat == STCISPCMD[index]) //判断接收的数据和当前的命令字符是否匹配
        {
            index++; //若匹配则索引+1
            if (STCISPCMD[index] == '\0') //判断命令是否配完成
                IAP_CONTR = 0x60; //若匹配完成则软复位到 ISP
        }
        else
        {
            index = 0; //若不匹配,则需要从头开始
            if (dat == STCISPCMD[index])
                index++;
        }
    }
}

void main()
{
```

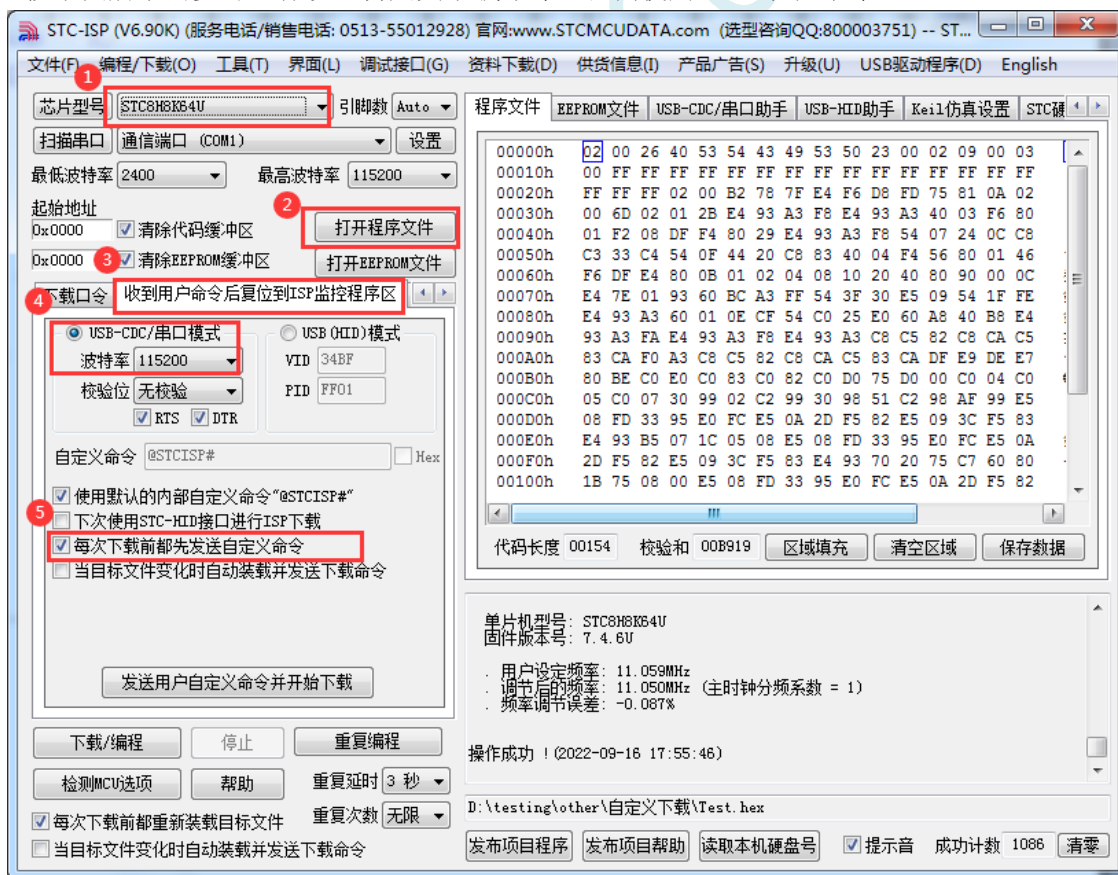
```
POM0 = 0x00; POM1 = 0x00;
P1M0 = 0x00; P1M1 = 0x00;
P2M0 = 0x00; P2M1 = 0x00;
P3M0 = 0x00; P3M1 = 0x00;
```

```
SCON = 0x50; //串口初始化
AUXR = 0x40;
TMOD = 0x00;
TH1 = BAUD >> 8;
TL1 = BAUD;
TR1 = 1;
ES = 1;
EA = 1;
```

```
index = 0; //初始化命令
```

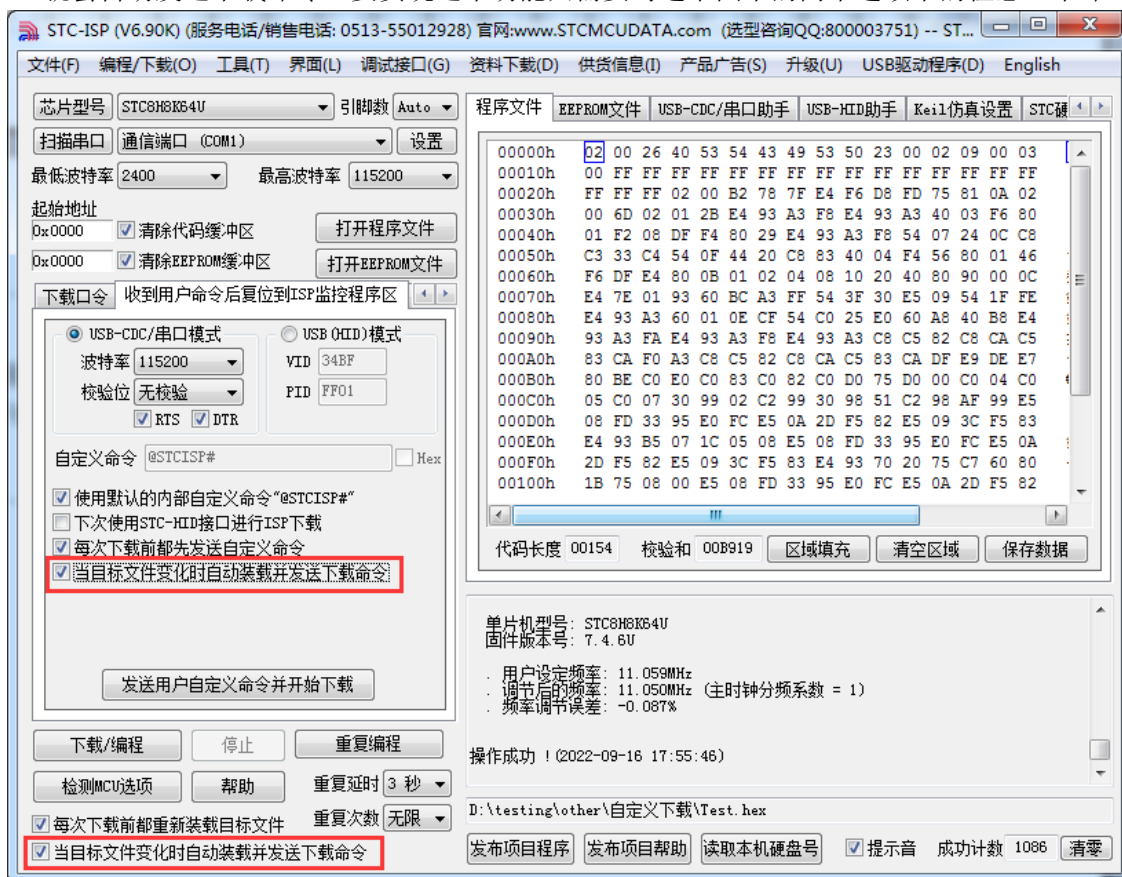
```
while (1);
}
```

## 2、按下图所示的步骤进行设置自定义下载命令（范例使用 STC 默认命令 “@STCISP#”）



## 3、第一次下载时需要目标单片机重新上电，之后的每次更新只需要点击下载软件中的“下载/编程”按钮，下载软件自动将下载命令发送给目标单片机，目标单片机接收到命令后自动复位到系统 ISP 区，即可实现不停电更新用户代码。

4、STC-ISP 还可实现项目开发阶段，完全自动下载功能，即当下载软件侦测到目标代码被更新了，就会自动发送下载命令。要实现这个功能只需要勾选下图中的两个选项中的任意一个即可

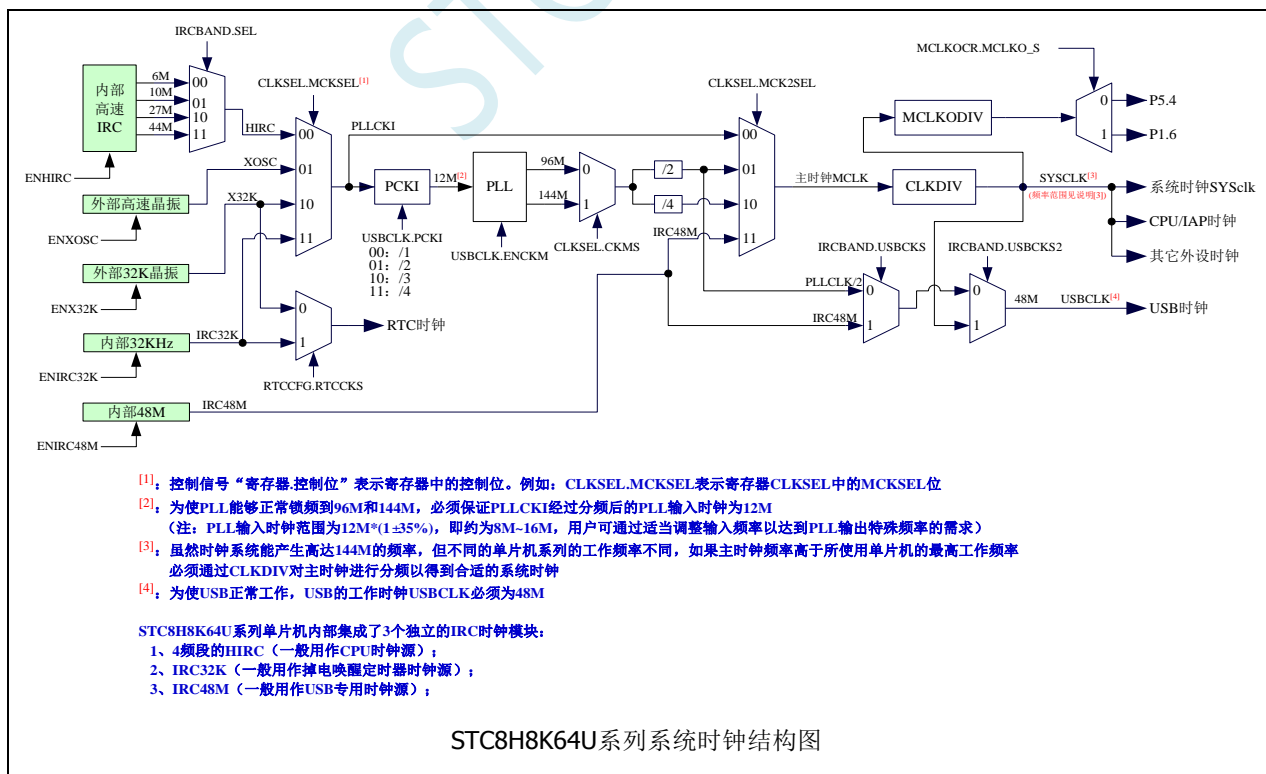
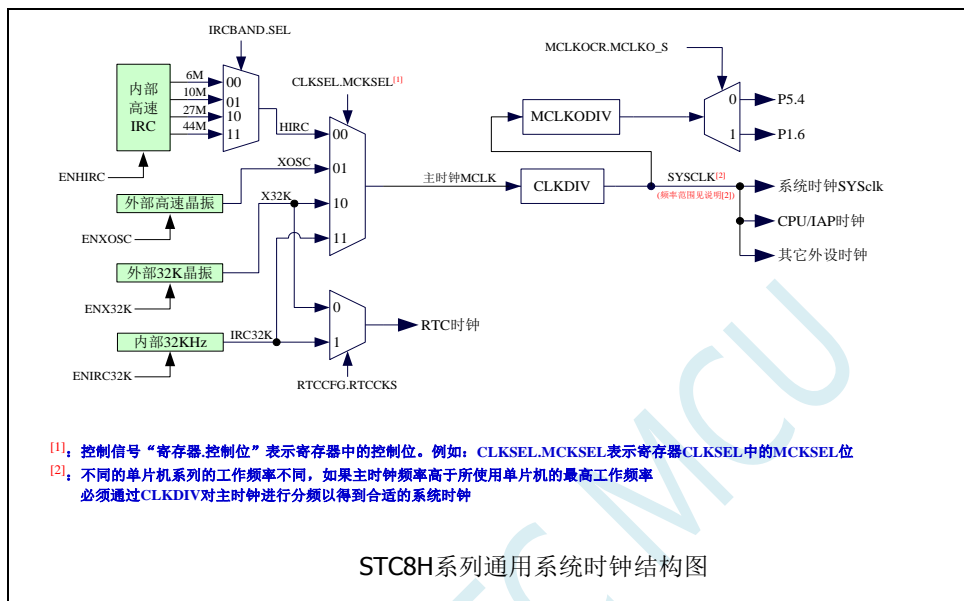


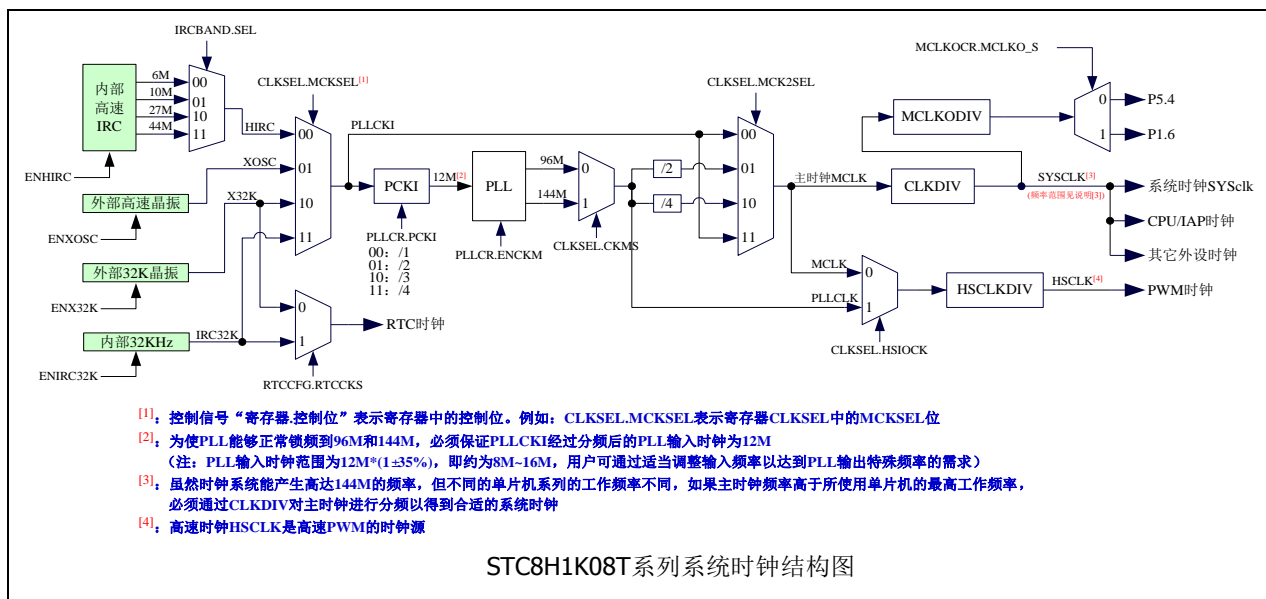
## 6 时钟、复位、看门狗与电源管理

### 6.1 系统时钟控制

系统时钟控制器为单片机的 CPU 和所有外设系统提供时钟源, 系统时钟有 3 个时钟源可供选择: 内部高精度 IRC、内部 32KHz 的 IRC (误差较大)、外部晶振。用户可通过程序分别使能和关闭各个时钟源, 以及内部提供时钟分频以达到降低功耗的目的。

单片机进入掉电模式后, 时钟控制器将会关闭所有的时钟源





## 相关寄存器

| 符号     | 描述          | 地址  | 位地址与符号 |           |    |     |         |         |             |    | 复位值       |
|--------|-------------|-----|--------|-----------|----|-----|---------|---------|-------------|----|-----------|
|        |             |     | B7     | B6        | B5 | B4  | B3      | B2      | B1          | B0 |           |
| USBCLK | USB 时钟控制寄存器 | DCH | ENCKM  | PCKI[1:0] |    | CRE | TST_USB | TST_PHY | PHYTST[1:0] |    | 0010,0000 |
| PLLCR  | PLL 时钟控制寄存器 | DCH | ENCKM  | PCKI[1:0] |    | -   | -       | -       | -           | -  | 000x,xxxx |

| 符号       | 描述              | 地址    | 位地址与符号   |               |               |    |              |    |             |          | 复位值       |
|----------|-----------------|-------|----------|---------------|---------------|----|--------------|----|-------------|----------|-----------|
|          |                 |       | B7       | B6            | B5            | B4 | B3           | B2 | B1          | B0       |           |
| CLKSEL   | 时钟选择寄存器         | FE00H | CKMS     | HSIOCK        | -             | -  | MCK2SEL[1:0] |    | MCKSEL[1:0] |          | 00xx,0000 |
| CLKDIV   | 时钟分频寄存器         | FE01H |          |               |               |    |              |    |             |          | nnnn,nnnn |
| HIRCCR   | 内部高速振荡器控制寄存器    | FE02H | ENHIRC   | -             | -             | -  | -            | -  | -           | HIRCST   | 1xxx,xxx0 |
| XOSCCR   | 外部晶振控制寄存器       | FE03H | ENXOSC   | XITYPE        | XCFILTER[1:0] |    | GAIN         | -  | -           | XOSCST   | 000x,xxx0 |
| IRC32KCR | 内部 32K 振荡器控制寄存器 | FE04H | ENIRC32K | -             | -             | -  | -            | -  | -           | IRC32KST | 0xxx,xxx0 |
| MCLKOCR  | 主时钟输出控制寄存器      | FE05H | MCKLO_S  | MCLKODIV[6:0] |               |    |              |    |             |          | 0000,0000 |
| X32KCR   | 外部 32K 晶振控制寄存器  | FE08H | ENX32K   | GAIN32K       |               | -  | -            | -  | -           | X32KST   | 00xx,xxx0 |

## 6.1.1 系统时钟选择寄存器 (CLKSEL)

| 符号     | 地址    | B7   | B6     | B5 | B4 | B3           | B2 | B1          | B0 |
|--------|-------|------|--------|----|----|--------------|----|-------------|----|
| CLKSEL | FE00H | CKMS | HSIOCK | -  | -  | MCK2SEL[1:0] |    | MCKSEL[1:0] |    |

CKMS: 内部 PLL 输出时钟选择

0: PLL 输出 96MHz

1: PLL 输出 144MHz

HSIOCK: 高速 I/O 时钟源选择 (注意: 此位为只写位, 不可读)

0: 主时钟 MCLK 为高速 I/O 时钟源

1: PLL 输出 96MHz/144MHz 的 PLLCLK 为高速 I/O 时钟源

MCK2SEL[1:0]: 主时钟源选择

| MCK2SEL[1:0] | 主时钟源          |
|--------------|---------------|
| 00           | MCKSEL 选择的时钟源 |
| 01           | 内部 PLL 输出/2   |
| 10           | 内部 PLL 输出/4   |
| 11           | 保留            |

MCKSEL[1:0]: 主时钟源选择

| MCKSEL[1:0] | 主时钟源            |
|-------------|-----------------|
| 00          | 内部高速高精度 IRC     |
| 01          | 外部高速晶振          |
| 10          | 外部 32KHz 晶振     |
| 11          | 内部 32KHz 低速 IRC |

## 6.1.2 时钟分频寄存器 (CLKDIV)

| 符号     | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-------|----|----|----|----|----|----|----|----|
| CLKDIV | FE01H |    |    |    |    |    |    |    |    |



CLKDIV: 主时钟分频系数。系统时钟 SYSCLK 是对主时钟 MCLK 进行分频后的时钟信号。

| CLKDIV | 系统时钟频率   |
|--------|----------|
| 0      | MCLK/1   |
| 1      | MCLK/1   |
| 2      | MCLK/2   |
| 3      | MCLK/3   |
| ...    | ...      |
| x      | MCLK/x   |
| ...    | ...      |
| 255    | MCLK/255 |

注意: 用户程序复位后, 系统会自动根据上次 ISP 下载时所设定工作频率所需的分频系数来设置此寄存器的初始值

### 6.1.3 内部高速高精度 IRC 控制寄存器 (HIRCCR)

| 符号     | 地址    | B7     | B6 | B5 | B4 | B3 | B2 | B1 | B0     |
|--------|-------|--------|----|----|----|----|----|----|--------|
| HIRCCR | FE02H | ENHIRC | -  | -  | -  | -  | -  | -  | HIRCST |

ENHIRC: 内部高速高精度 IRC 使能位

0: 关闭内部高精度 IRC

1: 使能内部高精度 IRC

HIRCST: 内部高速高精度 IRC 频率稳定标志位。(只读位)

当内部的 IRC 从停振状态开始使能后, 必须经过一段时间, 振荡器的频率才会稳定, 当振荡器频率稳定后, 时钟控制器会自动将 HIRCST 标志位置 1。所以当用户程序需要将时钟切换到使用内部 IRC 时, 首先必须设置 ENHIRC=1 使能振荡器, 然后一直查询振荡器稳定标志位 HIRCST, 直到标志位变为 1 时, 才可进行时钟源切换。

### 6.1.4 外部振荡器控制寄存器 (XOSCCR)

STC8H8K86U 系列 (注意: SFR 分配和 STC8H 其它系列不兼容)

| 符号     | 地址    | B7     | B6     | B5   | B4 | B3            | B2 | B1 | B0     |
|--------|-------|--------|--------|------|----|---------------|----|----|--------|
| XOSCCR | FE03H | ENXOSC | XITYPE | GAIN | -  | XCFILTER[1:0] |    | -  | XOSCST |

STC8H 其它系列

| 符号     | 地址    | B7     | B6     | B5            | B4 | B3   | B2 | B1 | B0     |
|--------|-------|--------|--------|---------------|----|------|----|----|--------|
| XOSCCR | FE03H | ENXOSC | XITYPE | XCFILTER[1:0] |    | GAIN | -  | -  | XOSCST |

ENXOSC: 外部晶体振荡器使能位

0: 关闭外部晶体振荡器

1: 使能外部晶体振荡器

XITYPE: 外部时钟源类型

0: 外部时钟源是外部时钟信号 (或有源晶振)。信号源只需连接单片机的 XTALI (P1.7) (此时 P1.6 口固定为高阻输入模式, 可用于读取外部数字信号或当作 ADC 输入, 但一般不建议使用, 因为旁边的 P1.7 口有高频振荡信号会对 P1.6 的信号有影响)

1: 外部时钟源是晶体振荡器。信号源连接单片机的 XTALI (P1.7) 和 XTALO (P1.6)

XCFILTER[1:0]: 外部晶体振荡器抗干扰控制寄存器

00: 外部晶体振荡器频率在 48M 及以下时可选择此项



01: 外部晶体振荡器频率在 24M 及以下时可选择此项

1x: 外部晶体振荡器频率在 12M 及以下时可选择此项

**注: 此寄存器目前只有 STC8H3K64S4 系列的 B 版芯片、STC8H3K64S2 系列的 B 版芯片、STC8A8K64D4 系列和 STC8H8K64U 系列的 B/C/D 版芯片才有效。需谨慎设置, 设置不当可能会导致外部晶振供给内部 MCU 的时钟不正常。**

GAIN: 外部晶体振荡器振荡增益控制位

0: 关闭振荡增益 (低增益)

1: 使能振荡增益 (高增益)

XOSCST: 外部晶体振荡器频率稳定标志位。(只读位)

当外部晶体振荡器从停振状态开始使能后, 必须经过一段时间, 振荡器的频率才会稳定, 当振荡器频率稳定后, 时钟控制器会自动将 XOSCST 标志位置 1。所以当用户程序需要将时钟切换到使用外部晶体振荡器时, 首先必须设置 ENXOSC=1 使能振荡器, 然后一直查询振荡器稳定标志位 XOSCST, 直到标志位变为 1 时, 才可进行时钟源切换。

## 6.1.5 外部 32K 振荡器控制寄存器 (X32KCR)

| 符号     | 地址    | B7     | B6      | B5 | B4 | B3 | B2 | B1 | B0     |
|--------|-------|--------|---------|----|----|----|----|----|--------|
| X32KCR | FE08H | ENX32K | GAIN32K | -  | -  | -  | -  | -  | X32KST |

ENX32K: 外部 32K 晶体振荡器使能位

0: 关闭外部 32K 晶体振荡器

1: 使能外部 32K 晶体振荡器

GAIN32K: 外部 32K 晶体振荡器振荡增益控制位

0: 关闭 32K 振荡增益 (低增益)

1: 使能 32K 振荡增益 (高增益)

X32KST: 外部 32K 晶体振荡器频率稳定标志位。(只读位)

6.1.6 内部 32KHz 低速 IRC 控制寄存器（IRC32KCR）

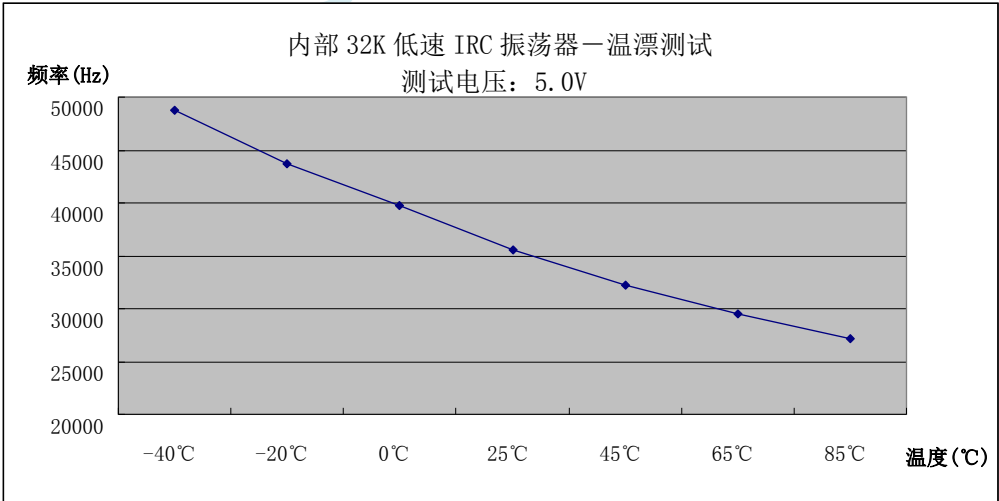
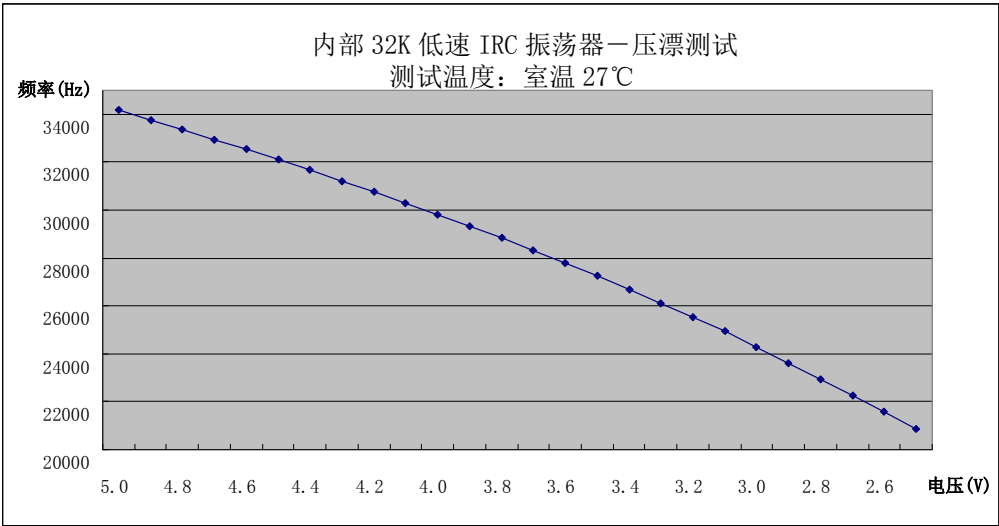
| 符号       | 地址    | B7       | B6 | B5 | B4 | B3 | B2 | B1 | B0       |
|----------|-------|----------|----|----|----|----|----|----|----------|
| IRC32KCR | FE04H | ENIRC32K | -  | -  | -  | -  | -  | -  | IRC32KST |

ENIRC32K：内部 32K 低速 IRC 使能位

- 0：关闭内部 32K 低速 IRC
- 1：使能内部 32K 低速 IRC

IRC32KST：内部 32K 低速 IRC 频率稳定标志位。（只读位）

当内部 32K 低速 IRC 从停振状态开始使能后，必须经过一段时间，振荡器的频率才会稳定，当振荡器频率稳定后，时钟控制器会自动将 IRC32KST 标志位置 1。所以当用户程序需要将时钟切换到使用内部 32K 低速 IRC 时，首先必须设置 ENIRC32K=1 使能振荡器，然后一直查询振荡器稳定标志位 IRC32KST，直到标志位变为 1 时，才可进行时钟源切换。



### 6.1.7 主时钟输出控制寄存器 (MCLKOCR)

| 符号      | 地址    | B7     | B6            | B5 | B4 | B3 | B2 | B1 | B0 |
|---------|-------|--------|---------------|----|----|----|----|----|----|
| MCLKOCR | FE05H | MCKO_S | MCLKODIV[6:0] |    |    |    |    |    |    |

MCLKODIV[6:0]: 主时钟输出分频系数

(注意: 主时钟分频输出的时钟源是经过 CLKDIV 分频后的系统时钟)

| MCLKODIV[6:0] | 系统时钟分频输出频率  |
|---------------|-------------|
| 0000000       | 不输出时钟       |
| 0000001       | SYSClk /1   |
| 0000010       | SYSClk /2   |
| 0000011       | SYSClk /3   |
| ...           | ...         |
| 1111110       | SYSClk /126 |
| 1111111       | SYSClk /127 |

MCKO\_S: 系统时钟输出管脚选择

0: 系统时钟分频输出到 P5.4 口

1: 系统时钟分频输出到 P1.6 口

### 6.1.8 USB 时钟控制寄存器 (USBCLK)

| 符号     | 地址  | B7    | B6        | B5 | B4  | B3      | B2      | B1          | B0 |
|--------|-----|-------|-----------|----|-----|---------|---------|-------------|----|
| USBCLK | DCH | ENCKM | PCKI[1:0] |    | CRE | TST_USB | TST_PHY | PHYTST[1:0] |    |

ENCKM: PLL 倍频控制

0: 禁止 PLL 倍频

1: 使能 PLL 倍频

PCKI[1:0]: PLL 时钟选择

| PCKI[1:0] | PLL 时钟源 |
|-----------|---------|
| 00        | 12M     |
| 01        | 24M     |
| 10        | 36M     |
| 11        | 48M     |

### 6.1.9 PLL 时钟控制寄存器 (PLLCCR)

| 符号     | 地址  | B7    | B6        | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-----|-------|-----------|----|----|----|----|----|----|
| PLLCCR | DCH | ENCKM | PCKI[1:0] |    | -  | -  | -  | -  | -  |

ENCKM: PLL 倍频控制

0: 禁止 PLL 倍频

1: 使能 PLL 倍频

PCKI[1:0]: PLL 输入时钟选择

| PCKI[1:0] | PLL 时钟源 |
|-----------|---------|
| 00        | 12M     |
| 01        | 24M     |
| 10        | 36M     |

|    |     |
|----|-----|
| 11 | 48M |
|----|-----|

STC MCU

## 6.2 STC8H 系列内部 IRC 频率调整

STC8H 系列单片机内部均集成有一颗高精度内部 IRC 振荡器。在用户使用 ISP 下载软件进行下载时，ISP 下载软件会根据用户所选择/设置的频率自动进行调整，一般频率值可调整到±0.3%以下，调整后的频率在全温度范围内（-40℃~85℃）的温漂可达-1.35%~1.30%。

STC8H1K08 系列和 STC8H1K28 系列内部 IRC 有两个频段，频段的中心频率分别为 20MHz 和 35MHz，20M 频段的调节范围约为 15.5MHz~27MHz，35M 频段的调节范围约为 27.5MHz~47MHz（注意：不同的芯片以及不同的生成批次可能会有约 5%左右的制造误差）。**经实际测试，部分芯片的最高工作频率只能为 39.5MHz，所以为了安全起见，建议用户在 ISP 下载时设置 IRC 频率不要高于 35MHz。**

除 STC8H1K08 系列和 STC8H1K28 系列以外的其他的 STC8H 系列内部 IRC 有四个频段，频段的中心频率分别为 6MHz、10MHz、27MHz 和 44MHz，每个频段的调节范围约为±27%（注意：不同的芯片以及不同的生成批次可能会有约 5%左右的制造误差）。

**注意：对于一般用户，内部 IRC 频率的调整可以不用关心，因为频率调整工作在进行 ISP 下载时已经自动完成了。所以若用户不需要自行调整频率，那么下面相关的 4 个寄存器也不能随意修改，否则可能会导致工作频率变化。**

**若用户需要在自己的代码中动态选择芯片预置的频率，请参考预置频率列表以及“用户自定义内部 IRC 频率”的范例程序**

内部 IRC 频率调整主要使用下面的 4 个寄存器进行调整

### 相关寄存器

| 符号      | 描述          | 地址    | 位地址与符号      |         |    |    |    |    |              |    | 复位值       |
|---------|-------------|-------|-------------|---------|----|----|----|----|--------------|----|-----------|
|         |             |       | B7          | B6      | B5 | B4 | B3 | B2 | B1           | B0 |           |
| IRCBAND | IRC 频段选择    | 9DH   | USBCKS      | USBCKS2 | -  | -  | -  | -  | SEL[1:0]     |    | 10xx,xxnn |
| LIRTRIM | IRC 频率微调寄存器 | 9EH   | -           | -       | -  | -  | -  | -  | LIRTRIM[1:0] |    | xxxx,xxnn |
| IRTRIM  | IRC 频率调整寄存器 | 9FH   | IRTRIM[7:0] |         |    |    |    |    |              |    | nnnn,nnnn |
| CLKDIV  | 时钟分频寄存器     | FE01H | CLKDIV[7:0] |         |    |    |    |    |              |    | nnnn,nnnn |

### 6.2.1 IRC 频段选择寄存器（IRCBAND）

| 符号      | 地址  | B7     | B6      | B5 | B4 | B3 | B2 | B1       | B0 |
|---------|-----|--------|---------|----|----|----|----|----------|----|
| IRCBAND | 9DH | USBCKS | USBCKS2 | -  | -  | -  | -  | SEL[1:0] |    |

USBCKS/USBCKS2: USB 时钟选择寄存器

**如需使能 USB 功能，请保持 USBCKS=1; USBCKS2=0;**

**特别是在通过改变 SEL[1:0]来修改 CPU 主频时，务必不能修改 USBCKS 和 USBCKS2 的值**

SEL[0]: 频段选择(两频段)

0: 选择 20MHz 频段

1: 选择 35MHz 频段

SEL[1:0]: 频段选择(四频段)

00: 选择 6MHz 频段

01: 选择 10MHz 频段

10: 选择 27MHz 频段

11: 选择 44MHz 频段

# 6.2.2 内部 IRC 频率调整寄存器 (IRTRIM)

| 符号     | 地址  | B7          | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-----|-------------|----|----|----|----|----|----|----|
| IRTRIM | 9FH | IRTRIM[7:0] |    |    |    |    |    |    |    |

IRTRIM[7:0]: 内部高精度 IRC 频率调整寄存器

IRTRIM 可对 IRC 频率进行 256 个等级的调整，每个等级所调整的频率值在整体上呈线性分布，局部会有波动。宏观上，每一级所调整的频率约为 0.24%，即 IRTRIM 为 (n+1) 时的频率比 IRTRIM 为 (n) 时的频率约快 0.24%。但由于 IRC 频率调整并非每一级都是 0.24%（每一级所调整频率的最大值约为 0.55%，最小值约为 0.02%，整体平均值约为 0.24%），所以会造成局部波动。

STC MCU

### 6.2.3 内部 IRC 频率微调寄存器 (LIRTRIM)

| 符号      | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1          | B0 |
|---------|-----|----|----|----|----|----|----|-------------|----|
| LIRTRIM | 9EH | -  | -  | -  | -  | -  | -  | IRTRIM[1:0] |    |

LIRTRIM[1:0]: 内部高精度 IRC 频率微调寄存器

LIRTRIM 可对 IRC 频率进行 3 个等级的调整, 3 个等级所调整的频率范围如下表所示:

| LIRTRIM[1:0] | 调整的频率范围   |
|--------------|-----------|
| 00           | 不微调       |
| 01           | 调整约 0.10% |
| 10           | 调整约 0.04% |
| 11           | 调整约 0.10% |

### 6.2.4 时钟分频寄存器 (CLKDIV)

| 符号     | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-------|----|----|----|----|----|----|----|----|
| CLKDIV | FE01H |    |    |    |    |    |    |    |    |

CLKDIV: 主时钟分频系数。系统时钟 SYSCLK 是对主时钟 MCLK 进行分频后的时钟信号。

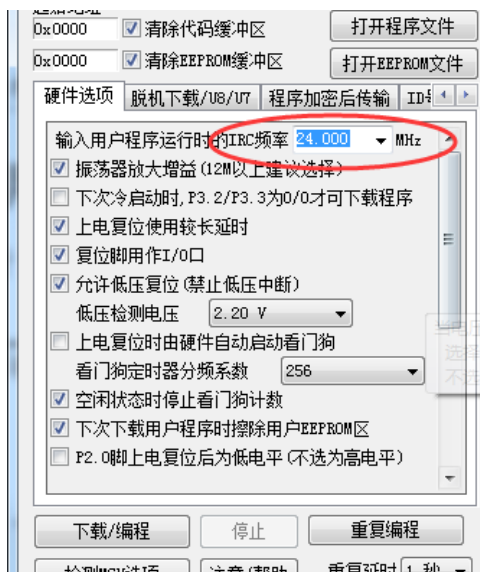
| CLKDIV | 系统时钟频率   |
|--------|----------|
| 0      | MCLK/1   |
| 1      | MCLK/1   |
| 2      | MCLK/2   |
| 3      | MCLK/3   |
| ...    | ...      |
| x      | MCLK/x   |
| ...    | ...      |
| 255    | MCLK/255 |

STC8H 系列内部的两个频段的可调范围分别为 15.5MHz~27MHz 和 25.3MHz~43.6MHz。虽然 35MHz 频段的上限可调到 40MHz 以上, 但芯片内部的程序存储器无法运行到 40MHz 以上的速度, 所以用户在 ISP 下载时设置内部 IRC 频率不能高于 40MHz, 一般建议用户设置为 35MHz 以下。若用户需要较低的工作频率时, 可使用 CLKDIV 寄存器对调节后的频率进行分频, 例如用户需要 11.0592MHz 的频率, 使用内部 IRC 直接调整是无法得到这个频率的, 但可将内部 IRC 调整到 22.1184MHz, 在使用 CLKDIV 进行 2 分频即可得到 11.0592MHz。

## 6.2.5 分频出 3MHz 用户工作频率，并用户动态改变频率追频示例

为得到 3MHz 的频率，可使用  $24\text{MHz} \div 8$  的方法。

首先在进行 ISP 下载时选择内部 IRC 工作频率为 24MHz，如下图所示，



然后在代码中选择时钟源为内部 IRC，并使用 CLKDIV 寄存器进行 8 分频。

### C 语言代码

//测试工作频率为24MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void main()
```

```
{
```

```
    P_SW2 /= 0x80;
```

```
//使能访问XFR
```

```
    P0M0 = 0x00;
```

```
    P0M1 = 0x00;
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P2M0 = 0x00;
```

```
    P2M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P4M0 = 0x00;
```

```
    P4M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
    CLKSEL = 0x00;
```

```
//选择内部IRC(默认)
```

```
    CLKDIV = 0x08;
```

```
//时钟8分频
```

```
    IRTRIM++;
```

```
//IRC 频率向上3%进行微调(注意判断边界)
```

```
//    IRTRIM--;
```

```
//IRC 频率向下3%进行微调(注意判断边界)
```

```
    while(1);
```

```
}
```



## 汇编代码

;测试工作频率为24MHz

```

P_SW2      DATA      0BAH
IRTRIM      DATA      09FH

CLKSEL      EQU        0FE00H
CLKDIV      EQU        0FE01H
HIRCCR      EQU        0FE02H
XOSCCR      EQU        0FE03H
IRC32KCR    EQU        0FE04H

P0M1        DATA      093H
P0M0        DATA      094H
P1M1        DATA      091H
P1M0        DATA      092H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

                ORG      0000H
                LJMP     MAIN

MAIN:         ORG      0100H

                MOV      SP, #5FH
                ORL      P_SW2, #80H                ;使能访问 XFR

                MOV      P0M0, #00H
                MOV      P0M1, #00H
                MOV      P1M0, #00H
                MOV      P1M1, #00H
                MOV      P2M0, #00H
                MOV      P2M1, #00H
                MOV      P3M0, #00H
                MOV      P3M1, #00H
                MOV      P4M0, #00H
                MOV      P4M1, #00H
                MOV      P5M0, #00H
                MOV      P5M1, #00H

                MOV      A, #00H                    ;选择内部 IRC
                MOV      DPTR, #CLKSEL
                MOVX     @DPTR, A
                MOV      A, #08H                    ;时钟 8 分频
                MOV      DPTR, #CLKDIV
                MOVX     @DPTR, A

                INC      IRTRIM                    ;IRC 频率向上 3%进行微调 (注意判断边界)
;                DEC      IRTRIM                    ;IRC 频率向下 3%进行微调 (注意判断边界)

                JMP      $

```

*END*

---

---

STC MCU

# 6.3 系统复位

STC8 系列单片机的复位分为硬件复位和软件复位两种。

硬件复位时，所有的寄存器的值会复位到初始值，系统会重新读取所有的硬件选项。同时根据硬件选项所设置的上电等待时间进行上电等待。硬件复位主要包括：

- 上电复位，POR，1.7V 附近
- 低压复位，LVD-RESET（2.0V，2.4V，2.7V，3.0V 附近）
- 复位脚复位（低电平复位）
- 看门狗复位

软件复位时，除与时钟相关的寄存器保持不变外，其余的所有寄存器的值会复位到初始值，软件复位不会重新读取所有的硬件选项。软件复位主要包括：

- 写 IAP\_CONTR 的 SWRST 所触发的复位

## 相关寄存器

| 符号        | 描述        | 地址  | 位地址与符号   |       |        |          |         |             |           |    | 复位值       |
|-----------|-----------|-----|----------|-------|--------|----------|---------|-------------|-----------|----|-----------|
|           |           |     | B7       | B6    | B5     | B4       | B3      | B2          | B1        | B0 |           |
| WDT_CONTR | 看门狗控制寄存器  | C1H | WDT_FLAG | -     | EN_WDT | CLR_WDT  | IDL_WDT | WDT_PS[2:0] |           |    | 0x00,0000 |
| IAP_CONTR | IAP 控制寄存器 | C7H | IAPEN    | SWBS  | SWRST  | CMD_FAIL | -       | IAP_WT[2:0] |           |    | 0000,x000 |
| RSTCFG    | 复位配置寄存器   | FFH | -        | ENLVR | -      | P54RST   | -       | -           | LVDS[1:0] |    | x0x0,xx00 |

| 符号      | 描述      | 地址    | 位地址与符号 |    |    |         |         |        |        |        | 复位值       |
|---------|---------|-------|--------|----|----|---------|---------|--------|--------|--------|-----------|
|         |         |       | B7     | B6 | B5 | B4      | B3      | B2     | B1     | B0     |           |
| RSTFLAG | 复位标志寄存器 | FE99H | -      | -  | -  | LVDRSTF | WDTRSTF | SWRSTF | ROMOVF | EXRSTF | xxx1,0100 |

### 6.3.1 看门狗复位 (WDT\_CONTR)

在工业控制/汽车电子/航空航天等需要高可靠性的系统中, 为了防止“系统在异常情况下, 受到干扰, MCU/CPU 程序跑飞, 导致系统长时间异常工作”, 通常是引进看门狗, 如果 MCU/CPU 不在规定的时间内按要求访问看门狗, 就认为 MCU/CPU 处于异常状态, 看门狗就会强制 MCU/CPU 复位, 使系统重新从头开始执行用户程序。

STC8 系列的看门狗复位是热启动复位中的硬件复位之一。STC8 系列单片机引进此功能, 使单片机系统可靠性设计变得更加方便、简洁。STC8 系列看门狗复位状态结束后, 系统固定从 ISP 监控程序区启动, 与看门狗复位前 IAP\_CONTR 寄存器的 SWBS 无关 (**注意: 此处与 STC15 系列 MCU 不同**)

#### WDT\_CONTR (看门狗控制寄存器)

| 符号        | 地址  | B7       | B6 | B5     | B4      | B3      | B2          | B1 | B0 |
|-----------|-----|----------|----|--------|---------|---------|-------------|----|----|
| WDT_CONTR | C1H | WDT_FLAG | -  | EN_WDT | CLR_WDT | IDL_WDT | WDT_PS[2:0] |    |    |

WDT\_FLAG: 看门狗溢出标志

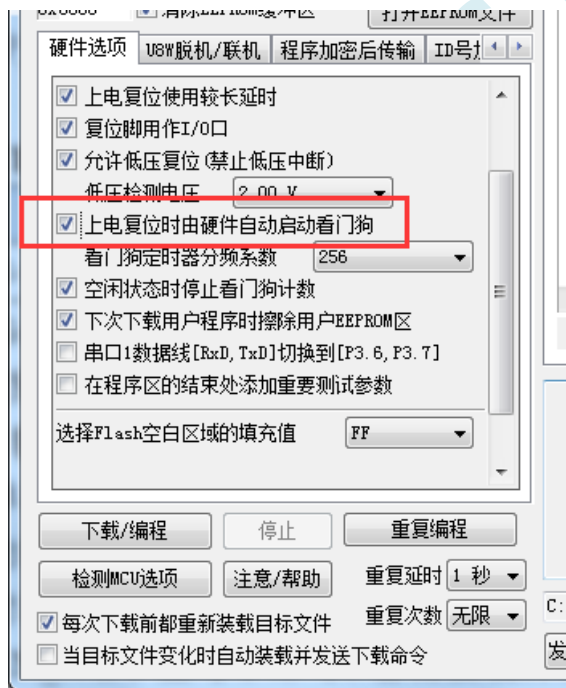
看门狗发生溢出时, 硬件自动将此位置 1, 需要软件清零。

EN\_WDT: 看门狗使能位

0: 对单片机无影响

1: 启动看门狗定时器。

**注意:** 看门狗定时器可使用软件方式启动, 也可硬件自动启动, 一旦看门狗定时器启动后, 软件将无法关闭, 必须对单片机进行重新上电才可关闭。软件启动看门狗只需要对 EN\_WDT 位写 1 即可。若需要硬件启动看门狗, 则需要在 ISP 下载时进行如下图所示的设置:



CLR\_WDT: 看门狗定时器清零

0: 对单片机无影响

1: 清零看门狗定时器, 硬件自动将此位复位

IDL\_WDT: IDLE 模式时的看门狗控制位

0: IDLE 模式时看门狗停止计数

1: IDLE 模式时看门狗继续计数

WDT\_PS[2:0]: 看门狗定时器时钟分频系数

| WDT_PS[2:0] | 分频系数 | 12M 主频时的溢出时间 | 20M 主频时的溢出时间 |
|-------------|------|--------------|--------------|
| 000         | 2    | ≈ 65.5 毫秒    | ≈ 39.3 毫秒    |
| 001         | 4    | ≈ 131 毫秒     | ≈ 78.6 毫秒    |
| 010         | 8    | ≈ 262 毫秒     | ≈ 157 毫秒     |
| 011         | 16   | ≈ 524 毫秒     | ≈ 315 毫秒     |
| 100         | 32   | ≈ 1.05 秒     | ≈ 629 毫秒     |
| 101         | 64   | ≈ 2.10 秒     | ≈ 1.26 秒     |
| 110         | 128  | ≈ 4.20 秒     | ≈ 2.52 秒     |
| 111         | 256  | ≈ 8.39 秒     | ≈ 5.03 秒     |

看门狗溢出时间计算公式如下:

$$\text{看门狗溢出时间} = \frac{12 \times 32768 \times 2^{(\text{WDT\_PS}+1)}}{\text{SYSclk}}$$

### 6.3.2 软件复位 (IAP\_CONTR)

IAP\_CONTR (IAP 控制寄存器)

| 符号        | 地址  | B7    | B6   | B5    | B4       | B3 | B2 | B1 | B0 |
|-----------|-----|-------|------|-------|----------|----|----|----|----|
| IAP_CONTR | C7H | IAPEN | SWBS | SWRST | CMD_FAIL | -  | -  | -  | -  |

SWBS: 软件复位启动选择

- 0: 软件复位后从用户程序区开始执行代码。用户数据区的数据保持不变。
- 1: 软件复位后从系统 **ISP** 区开始执行代码。用户数据区的数据会被初始化。

SWRST: 软件复位触发位

- 0: 对单片机无影响
- 1: 触发软件复位

### 6.3.3 低压复位（RSTCFG）

#### RSTCFG（复位配置寄存器）

| 符号     | 地址  | B7 | B6    | B5 | B4     | B3 | B2 | B1        | B0 |
|--------|-----|----|-------|----|--------|----|----|-----------|----|
| RSTCFG | FFH | -  | ENLVR | -  | P54RST | -  | -  | LVDS[1:0] |    |

ENLVR: 低压复位控制位

0: 禁止低压复位。当系统检测到低压事件时, 会产生低压中断

1: 使能低压复位。当系统检测到低压事件时, 自动复位

P54RST: RST 管脚功能选择

0: RST 管脚用作普通 I/O 口 (P54)

1: RST 管脚用作复位脚 (**低电平复位**)

LVDS[1:0]: 低压检测门槛电压设置

| LVDS[1:0] | 低压检测门槛电压 |
|-----------|----------|
| 00        | 2.0V     |
| 01        | 2.4V     |
| 10        | 2.7V     |
| 11        | 3.0V     |

### 6.3.4 复位标志寄存器（RSTFLAG）

| 符号      | 地址    | B7 | B6 | B5 | B4      | B3      | B2     | B1     | B0     |
|---------|-------|----|----|----|---------|---------|--------|--------|--------|
| RSTFLAG | FE99H | -  |    |    | LVDRSTF | WDTRSTF | SWRSTF | ROMOVF | EXRSTF |

LVDRSTF: LVD 低压复位标志

读 0: 无意义

读 1: 当前的复位是由 LVD 低压复位所触发 (**上电复位默认值为 1**)

写 0: 无效果

写 1: 清除 LVDRST 标志位

WDTRSTF: 看门狗复位标志

读 0: 无意义 (**上电复位默认值为 0**)

读 1: 当前的复位是由看门狗溢出所触发

写 0: 无效果

写 1: 清除 WDTRST 标志位

SWRSTF: 软复位标志

读 0: 无意义

读 1: 当前的复位是由软件写 SWRST (IAP\_CONTR.5) 所触发 (**用户程序复位默认值为 1**)

写 0: 无效果

写 1: 清除 SWRST 标志位

ROMOVF: 代码区溢出标志

读 0: 无意义 (**上电复位默认值为 0**)

读 1: 当前的复位是由于 CPU 执行代码到非程序区导致的代码区溢出所触发

写 0: 无效果

写 1: 清除 ROMOV 标志位

EXRSTF: 外部复位标志

读 0: 无意义 (上电复位默认值为 0)

读 1: 当前的复位是外部复位脚 (P5.4/RST) 被拉低所触发

写 0: 无效果

写 1: 清除 EXRST 标志位

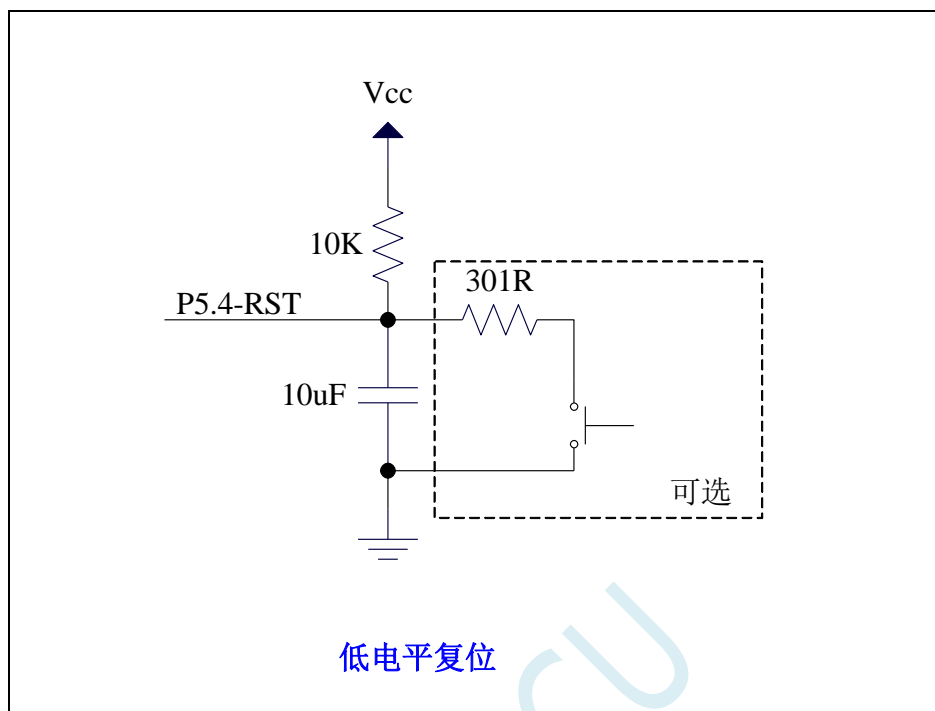
**关于用户程序软复位到系统区进行 USB-ISP 下载的说明:**

上电时需要 P3.2 同时接地才可进入 USB-ISP 下载模式, 为方便用户自主控制 ISP 下载, 特别增加: 当用户程序软复位到系统区时, 可不用 P3.2 接地就可进行 USB-ISP 下载。此功能的判断方式为进入 ISP 后判断 SWRSTF 寄存器位是否为 1, 若为 1 表示是用户软复位到系统区, 则不用 P3.2 接地, 否则需要 P3.2 接地。若用户需要软件软复位到系统区或者按键复位或者看门狗复位后需要进行 USB-ISP 下载, 可保持 SWRSTF 寄存器位为 1, 否则请在用户代码初始化时将 SWRSTF 寄存器位写 1, 以清零 SWRSTF。

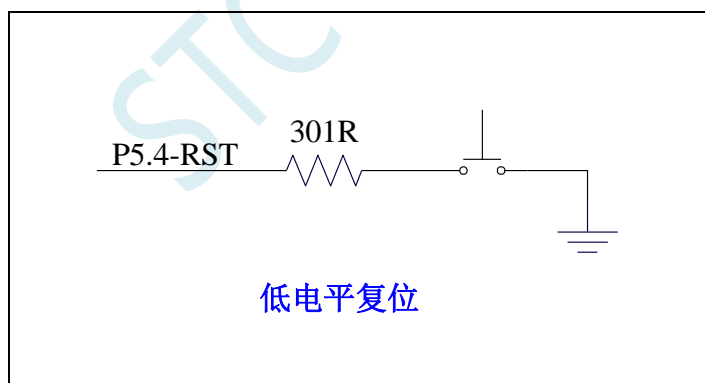
STC MCU



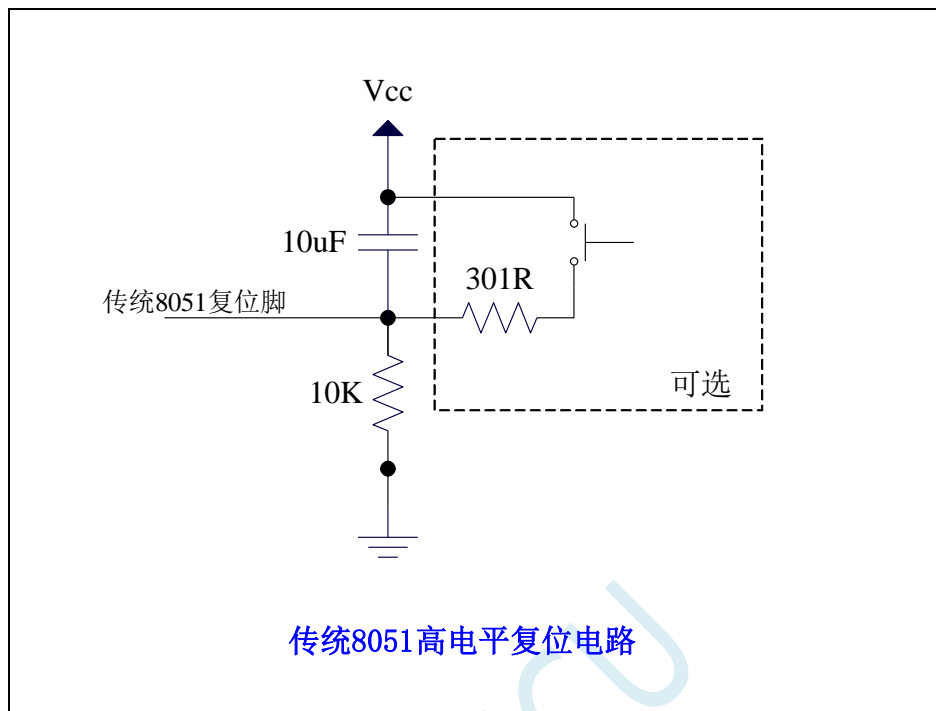
### 6.3.5 低电平上电复位参考电路（一般不需要）



### 6.3.6 低电平按键复位参考电路



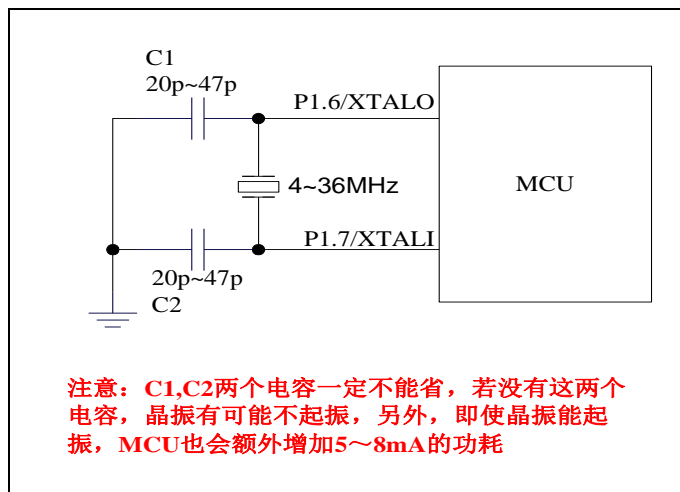
### 6.3.7 传统 8051 高电平上电复位参考电路



上图为传统 8051 的高电平复位电路，STC8H 的复位为低电平复位，与传统复位电路不同

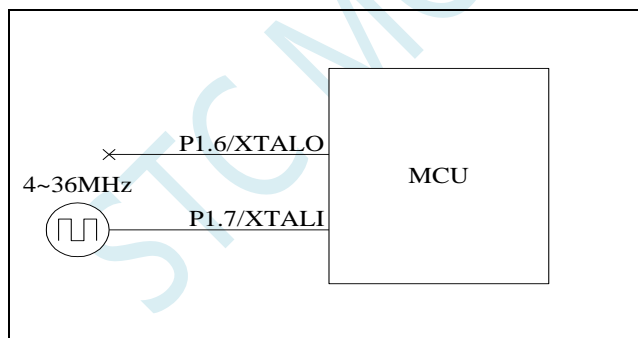
## 6.4 外部晶振及外部时钟电路

### 6.4.1 外部晶振输入电路



注意: 有 RTC 功能的 MCU, P1.6/P1.7 可以外部接 32768 的晶振或时钟

### 6.4.2 外部时钟输入电路 (P1.6 不可用作普通 I/O)



## 6.5 时钟停振/省电模式与系统电源管理

| 符号   | 描述      | 地址  | 位地址与符号 |       |      |     |     |     |    |     | 复位值       |
|------|---------|-----|--------|-------|------|-----|-----|-----|----|-----|-----------|
|      |         |     | B7     | B6    | B5   | B4  | B3  | B2  | B1 | B0  |           |
| PCON | 电源控制寄存器 | 87H | SMOD   | SMOD0 | LVDF | POF | GF1 | GF0 | PD | IDL | 0011,0000 |

### 6.5.1 电源控制寄存器（PCON）

| 符号   | 地址  | B7   | B6    | B5   | B4  | B3  | B2  | B1 | B0  |
|------|-----|------|-------|------|-----|-----|-----|----|-----|
| PCON | 87H | SMOD | SMOD0 | LVDF | POF | GF1 | GF0 | PD | IDL |

LVDF: 低压检测标志位。当系统检测到低压事件时, 硬件自动将此位置 1, 并向 CPU 提出中断请求。

此位需要用户软件清零。

POF: 上电标志位。当硬件自动将此位置 1。

PD: 时钟停振模式/掉电模式/停电模式控制位

0: 无影响

1: 单片机进入时钟停振模式/掉电模式/停电模式, CPU 以及全部外设均停止工作。唤醒后硬件自动清零。(注: 时钟停振模式下, CPU 和全部的外设均停止工作, 但 SRAM 和 XRAM 中的数据是一直维持不变的)

IDL: IDLE (空闲) 模式控制位

0: 无影响

1: 单片机进入 IDLE 模式, 只有 CPU 停止工作, 其他外设依然在运行。唤醒后硬件自动清零

注: 虽然 LVD 和比较器均可唤醒时钟停振模式化, 但时钟停振省电模式下, 不建议启动 LVD 和比较器, 否则硬件系统还会自动启动内部 1.19V 的高精准参考源, 这个高精准参考源有相应的抗温漂和调校线路, 大约会额外增加 300uA 的耗电, 而 MCU 进入时钟停振模式后, 3.3V 工作电压时只耗约 0.4uA 的电流, 所以进入时钟停振模式时不建议开 LVD 和比较器。如果确实需要用, 建议开启掉电唤醒定时器, 掉电唤醒定时器只会增加约 1.4uA 的耗电, 这个耗电一般系统是可以接受的。让掉电唤醒定时器每 5 秒唤醒一次 MCU, 唤醒后可用 LVD、比较器、ADC 检测外部电池电压, 检测工作约耗时 1mS 后再进入时钟停振/省电模式, 这样增加的平均电流小于 1uA, 则整体功耗大约为 2.8uA (0.4uA + 1.4uA + 1uA)。

掉电模式可以使用 INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、T3(P0.4)、T4(P0.6)、RXD(P3.0/P3.6/P1.6/P4.3)、RXD2(P1.0/P4.6)、RXD3(P0.0/P5.0)、RXD4(P0.2/P5.2)、I2C\_SDA(P1.4/P2.4/P3.3)以及比较器中断、低压检测中断、掉电唤醒定时器唤醒。

## 6.6 掉电唤醒定时器

内部掉电唤醒定时器是一个 15 位的计数器（由{WKTCH[6:0],WKTCL[7:0]}组成 15 位）。用于唤醒处于掉电模式的 MCU。

### 6.6.1 掉电唤醒定时器计数寄存器（WKTCL，WKTCH）

| 符号    | 地址  | B7    | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------|-----|-------|----|----|----|----|----|----|----|
| WKTCL | AAH |       |    |    |    |    |    |    |    |
| WKTCH | ABH | WKTEN |    |    |    |    |    |    |    |

WKTEN：掉电唤醒定时器的使能控制位

- 0：停用掉电唤醒定时器
- 1：启用掉电唤醒定时器

如果 STC8 系列单片机内置掉电唤醒专用定时器被允许（通过软件将 WKTCH 寄存器中的 WKTEN 位置 1），当 MCU 进入掉电模式/停机模式后，掉电唤醒专用定时器开始计数，当计数值与用户所设置的值相等时，掉电唤醒专用定时器将 MCU 唤醒。MCU 唤醒后，程序从上次设置单片机进入掉电模式语句的下一条语句开始往下执行。掉电唤醒之后，可以通过读 WKTCH 和 WKTCL 中的内容获取单片机在掉电模式中的睡眠时间。

这里请注意：用户在寄存器{WKTCH[6:0],WKTCL[7:0]}中写入的值必须比实际计数值少 1。如用户需计数 10 次，则将 9 写入寄存器{WKTCH[6:0],WKTCL[7:0]}中。同样，如果用户需计数 32767 次，则应对{WKTCH[6:0],WKTCL[7:0]}写入 7FFE<sub>H</sub>（即 32766）。（**计数值 0 和计数值 32767 为内部保留值，用户不能使用**）。内部掉电唤醒定时器有自己的内部时钟，掉电唤醒定时器计数一次的时间就是由该时钟决定的。内部掉电唤醒定时器的时钟频率约为 32KHz，误差较大。用户可以通过读 RAM 区 F8H 和 F9H 的内容（F8H 存放频率的高字节，F9H 存放低字节）来获取内部掉电唤醒专用定时器出厂时所记录的时钟频率。

掉电唤醒专用定时器计数时间的计算公式如下所示：（F<sub>wt</sub> 为我们从 RAM 区 F8H 和 F9H 获取到的内部掉电唤醒专用定时器的时钟频率）

掉电唤醒定时器定时时间 = 
$$\frac{10^6 \times 16 \times \text{计数次数}}{F_{wt}} \quad (\text{微秒})$$

假设 F<sub>wt</sub>=32KHz，则有：

| {WKTCH[6:0],WKTCL[7:0]} | 掉电唤醒专用定时器计数时间                          |
|-------------------------|--|
| <del>0（内部保留）</del>      |  |
| 1                       | 10 <sup>6</sup> ÷32K×16×(1+1)≈1 毫秒     |
| 9                       | 10 <sup>6</sup> ÷32K×16×(1+9)≈5 毫秒     |
| 99                      | 10 <sup>6</sup> ÷32K×16×(1+99)≈50 毫秒   |
| 999                     | 10 <sup>6</sup> ÷32K×16×(1+999)≈0.5 秒  |
| 4095                    | 10 <sup>6</sup> ÷32K×16×(1+4095)≈2 秒   |
| 32766                   | 10 <sup>6</sup> ÷32K×16×(1+32767)≈16 秒 |
| <del>32767（内部保留）</del>  |  |

## 6.7 范例程序

### 6.7.1 选择系统时钟源

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void main()
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    CLKSEL = 0x00;                //选择内部 IRC ( 默认 )

/*
    XOSCCR = 0xc0;                //启动外部晶振
    while (!(XOSCCR & 1));        //等待时钟稳定
    CLKDIV = 0x00;                //时钟不分频
    CLKSEL = 0x01;                //选择外部晶振
*/

/*
    IRC32KCR = 0x80;              //启动内部 32K IRC
    while (!(IRC32KCR & 1));      //等待时钟稳定
    CLKDIV = 0x00;                //时钟不分频
    CLKSEL = 0x03;                //选择内部 32K
*/

    while (1);
}
```

#### 汇编代码

```
;测试工作频率为 11.0592MHz

P_SW2      DATA      0BAH

CLKSEL      EQU        0FE00H
CLKDIV      EQU        0FE01H
HIRCCR      EQU        0FE02H
XOSCCR      EQU        0FE03H
```

```

IRC32KCR    EQU    0FE04H

P1M1        DATA    091H
P1M0        DATA    092H
P0M1        DATA    093H
P0M0        DATA    094H
P2M1        DATA    095H
P2M0        DATA    096H
P3M1        DATA    0B1H
P3M0        DATA    0B2H
P4M1        DATA    0B3H
P4M0        DATA    0B4H
P5M1        DATA    0C9H
P5M0        DATA    0CAH

                ORG    0000H
                LJMP   MAIN

MAIN:         ORG    0100H

                MOV     SP, #5FH
                ORL     P_SW2, #80H                ;使能访问 XFR

                MOV     P0M0, #00H
                MOV     P0M1, #00H
                MOV     P1M0, #00H
                MOV     P1M1, #00H
                MOV     P2M0, #00H
                MOV     P2M1, #00H
                MOV     P3M0, #00H
                MOV     P3M1, #00H
                MOV     P4M0, #00H
                MOV     P4M1, #00H
                MOV     P5M0, #00H
                MOV     P5M1, #00H

                MOV     A, #00H                ;选择内部 IRC ( 默认 )
                MOV     DPTR, #CLKSEL
                MOVX    @DPTR, A

;
;                MOV     A, #0C0H                ;启动外部晶振
;                MOV     DPTR, #XOSCCR
;                MOVX    @DPTR, A
;                MOVX    A, @DPTR
;                JNB     ACC.0, $-1                ;等待时钟稳定
;                CLR     A                ;时钟不分频
;                MOV     DPTR, #CLKDIV
;                MOVX    @DPTR, A
;                MOV     A, #01H                ;选择外部晶振
;                MOV     DPTR, #CLKSEL
;                MOVX    @DPTR, A

;
;                MOV     A, #80H                ;启动内部 32K IRC
;                MOV     DPTR, #IRC32KCR
;                MOVX    @DPTR, A
;                MOVX    A, @DPTR
;                JNB     ACC.0, $-1                ;等待时钟稳定
;                CLR     A                ;时钟不分频
;                MOV     DPTR, #CLKDIV

```

```
;      MOVX      @DPTR,A
;      MOV       A,#03H      ;选择内部32K
;      MOV       DPTR,#CLKSEL
;      MOVX      @DPTR,A

      JMP        $

      END
```

## 6.7.2 主时钟分频输出

### C 语言代码

```
//测试工作频率为11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void main()
{
    P_SW2 /= 0x80;      //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    // MCLKOCR = 0x01;      //主时钟输出到P5.4 口
    // MCLKOCR = 0x02;      //主时钟2分频输出到P5.4 口
    MCLKOCR = 0x04;      //主时钟4分频输出到P5.4 口
    // MCLKOCR = 0x84;      //主时钟4分频输出到P1.6 口

    while (1);
}
```

### 汇编代码

```
;测试工作频率为11.0592MHz

P_SW2      DATA      0BAH

MCLKOCR     EQU        0FE05H

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
```



```
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN

MAIN:      ORG          0100H

          MOV          SP, #5FH
          ORL          P_SW2, #80H          ;使能访问 XFR

          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

;          MOV          A, #01H          ;主时钟输出到 P5.4 口
;          MOV          A, #02H          ;主时钟 2 分频输出到 P5.4 口
          MOV          A, #04H          ;主时钟 4 分频输出到 P5.4 口
;          MOV          A, #84H          ;主时钟 4 分频输出到 P1.6 口
          MOV          DPTR, #MCLKOCR
          MOVX         @DPTR, A

          JMP          $

          END
```

### 6.7.3 看门狗定时器应用

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void main()
{
    P_SW2 /= 0x80;          //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
```

```
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

//  WDT_CONTR = 0x23;           //使能看门狗,溢出时间约为0.5s
//  WDT_CONTR = 0x24;           //使能看门狗,溢出时间约为1s
//  WDT_CONTR = 0x27;           //使能看门狗,溢出时间约为8s
//  P32 = 0;                     //测试端口

while (1)
{
//  WDT_CONTR = 0x33;           //清看门狗,否则系统复位
//  WDT_CONTR = 0x34;           //清看门狗,否则系统复位
//  WDT_CONTR = 0x37;           //清看门狗,否则系统复位

    Display();                  //显示模块
    Scankey();                  //按键扫描模块
    MotorDriver();              //电机驱动模块
}
}
```

汇编代码

;测试工作频率为11.0592MHz

|                  |             |                    |          |
|------------------|-------------|--------------------|----------|
| <i>WDT_CONTR</i> | <i>DATA</i> | <i>0C1H</i>        |          |
| <i>P_SW2</i>     | <i>DATA</i> | <i>0BAH</i>        |          |
| <i>P1M1</i>      | <i>DATA</i> | <i>091H</i>        |          |
| <i>P1M0</i>      | <i>DATA</i> | <i>092H</i>        |          |
| <i>P0M1</i>      | <i>DATA</i> | <i>093H</i>        |          |
| <i>P0M0</i>      | <i>DATA</i> | <i>094H</i>        |          |
| <i>P2M1</i>      | <i>DATA</i> | <i>095H</i>        |          |
| <i>P2M0</i>      | <i>DATA</i> | <i>096H</i>        |          |
| <i>P3M1</i>      | <i>DATA</i> | <i>0B1H</i>        |          |
| <i>P3M0</i>      | <i>DATA</i> | <i>0B2H</i>        |          |
| <i>P4M1</i>      | <i>DATA</i> | <i>0B3H</i>        |          |
| <i>P4M0</i>      | <i>DATA</i> | <i>0B4H</i>        |          |
| <i>P5M1</i>      | <i>DATA</i> | <i>0C9H</i>        |          |
| <i>P5M0</i>      | <i>DATA</i> | <i>0CAH</i>        |          |
|                  | <i>ORG</i>  | <i>0000H</i>       |          |
|                  | <i>LJMP</i> | <i>MAIN</i>        |          |
|                  | <i>ORG</i>  | <i>0100H</i>       |          |
| <i>MAIN:</i>     | <i>MOV</i>  | <i>SP, #5FH</i>    |          |
|                  | <i>ORL</i>  | <i>P_SW2, #80H</i> | ;使能访问XFR |
|                  | <i>MOV</i>  | <i>P0M0, #00H</i>  |          |
|                  | <i>MOV</i>  | <i>P0M1, #00H</i>  |          |
|                  | <i>MOV</i>  | <i>P1M0, #00H</i>  |          |
|                  | <i>MOV</i>  | <i>P1M1, #00H</i>  |          |

```

MOV      P2M0, #00H
MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

;        MOV      WDT_CONTR, #23H      ;使能看门狗,溢出时间约为0.5s
;        MOV      WDT_CONTR, #24H      ;使能看门狗,溢出时间约为1s
;        MOV      WDT_CONTR, #27H      ;使能看门狗,溢出时间约为8s
CLR      P3.2      ;测试端口

LOOP:
;        MOV      WDT_CONTR, #33H      ;清看门狗,否则系统复位
;        MOV      WDT_CONTR, #34H      ;清看门狗,否则系统复位
;        MOV      WDT_CONTR, #37H      ;清看门狗,否则系统复位

LCALL    DISPLAY      ;显示模块
LCALL    SCANKEY      ;按键扫描模块
LCALL    MOTORDRIVER   ;电机驱动模块
JMP      LOOP

END

```

## 6.7.4 软复位实现自定义下载

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```
void main()
```

```

{
    P_SW2 /= 0x80;      //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P32 = 1;      //测试端口
    P33 = 1;      //测试端口

    while (1)
    {

```

```

        if (!P32 && !P33)
        {
            IAP_CONTR /= 0x60;           //检查到P3.2 和P3.3 同时为0 时复位到ISP
        }
    }
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

```

IAP_CONTR    DATA    0C7H
P_SW2        DATA    0BAH

P1M1         DATA    091H
P1M0         DATA    092H
P0M1         DATA    093H
P0M0         DATA    094H
P2M1         DATA    095H
P2M0         DATA    096H
P3M1         DATA    0B1H
P3M0         DATA    0B2H
P4M1         DATA    0B3H
P4M0         DATA    0B4H
P5M1         DATA    0C9H
P5M0         DATA    0CAH

                ORG     0000H
                LJMP    MAIN

MAIN:          ORG     0100H

                MOV     SP, #5FH
                ORL     P_SW2, #80H           ;使能访问 XFR

                MOV     P0M0, #00H
                MOV     P0M1, #00H
                MOV     P1M0, #00H
                MOV     P1M1, #00H
                MOV     P2M0, #00H
                MOV     P2M1, #00H
                MOV     P3M0, #00H
                MOV     P3M1, #00H
                MOV     P4M0, #00H
                MOV     P4M1, #00H
                MOV     P5M0, #00H
                MOV     P5M1, #00H

                SETB    P3.2
                SETB    P3.3

LOOP:         JB      P3.2, LOOP
                JB      P3.3, LOOP
                MOV     IAP_CONTR, #60H      ;检查到P3.2 和P3.3 同时为0 时复位到ISP
                JMP     $

                END

```

## 6.7.5 低压检测

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void Lvd_Isr() interrupt 6
```

```
{
    PCON &= ~LVDF;           //清中断标志
    P32 = ~P32;              //测试端口
}
```

```
void main()
```

```
{
    P_SW2 /= 0x80;           //使能访问 XFR
```

```
    P0M0 = 0x00;
```

```
    P0M1 = 0x00;
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P2M0 = 0x00;
```

```
    P2M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P4M0 = 0x00;
```

```
    P4M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
    PCON &= ~LVDF;
```

```
// RSTCFG = ENLVR | LVD3V0;
```

```
RSTCFG = LVD3V0;
```

```
ELVD = 1;
```

```
EA = 1;
```

```
while (1);
```

```
}
```

### 汇编代码

;测试工作频率为 11.0592MHz

```
RSTCFG    DATA    0FFH
```

```
ENLVR     EQU      40H           ;RSTCFG.6
```

```
LVD2V0    EQU      00H           ;LVD@2.0V
```

```
LVD2V4    EQU      01H           ;LVD@2.4V
```

```
LVD2V7    EQU      02H           ;LVD@2.7V
```

```
LVD3V0    EQU      03H           ;LVD@3.0V
```

```
ELVD      BIT      IE.6
```

```
LVDF      EQU      20H           ;PCON.5
```

```
P_SW2     DATA    0BAH
```

```

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0033H
          LJMP         LVDISR

LVDISR:   ORG          0100H

          ANL          PCON,#NOT LVDF      ;清中断标志
          CPL          P3.2                ;测试端口
          RETI

MAIN:     MOV          SP,#5FH
          ORL          P_SW2,#80H          ;使能访问 XFR

          MOV          P0M0,#00H
          MOV          P0M1,#00H
          MOV          P1M0,#00H
          MOV          P1M1,#00H
          MOV          P2M0,#00H
          MOV          P2M1,#00H
          MOV          P3M0,#00H
          MOV          P3M1,#00H
          MOV          P4M0,#00H
          MOV          P4M1,#00H
          MOV          P5M0,#00H
          MOV          P5M1,#00H

          ANL          PCON,#NOT LVDF      ;上电后需要先清 LVDF 标志
;          MOV          RSTCFG,#ENLVR / LVD3V0 ;使能 3.0V 时低压复位,不产生 LVD 中断
          MOV          RSTCFG,#LVD3V0      ;使能 3.0V 时低压中断
          SETB         ELVD                 ;使能 LVD 中断
          SETB         EA
          JMP          $

          END

```

## 6.7.6 省电模式

### C 语言代码

```
//测试工作频率为 11.0592MHz
```

```
#include "stc8h.h"
```

```
#include "intrins.h"

void INT0_Isr() interrupt 0
{
    P34 = ~P34;                                     //测试端口
}

void main()
{
    P_SW2 /= 0x80;                                   //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    EX0 = 1;                                         //使能INT0 中断,用于唤醒MCU
    EA = 1;
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    PCON = IDL;                                     //MCU 进入 IDLE 模式
    // PCON = PD;                                   //MCU 进入掉电模式
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    P35 = 0;

    while (1);
}
```

汇编代码

;测试工作频率为11.0592MHz

|       |      |      |         |
|-------|------|------|---------|
| IDL   | EQU  | 01H  | ;PCON.0 |
| PD    | EQU  | 02H  | ;PCON.1 |
| P_SW2 | DATA | 0BAH |         |
| P1M1  | DATA | 091H |         |
| P1M0  | DATA | 092H |         |
| P0M1  | DATA | 093H |         |
| P0M0  | DATA | 094H |         |
| P2M1  | DATA | 095H |         |
| P2M0  | DATA | 096H |         |
| P3M1  | DATA | 0B1H |         |
| P3M0  | DATA | 0B2H |         |
| P4M1  | DATA | 0B3H |         |
| P4M0  | DATA | 0B4H |         |

```
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0003H
          LJMP         INT0ISR

INT0ISR:   ORG          0100H

          CPL          P3.4          ;测试端口
          RETI

MAIN:      MOV         SP, #5FH
          ORL          P_SW2, #80H   ;使能访问 XFR

          MOV         P0M0, #00H
          MOV         P0M1, #00H
          MOV         P1M0, #00H
          MOV         P1M1, #00H
          MOV         P2M0, #00H
          MOV         P2M1, #00H
          MOV         P3M0, #00H
          MOV         P3M1, #00H
          MOV         P4M0, #00H
          MOV         P4M1, #00H
          MOV         P5M0, #00H
          MOV         P5M1, #00H

          SETB        EX0          ;使能 INT0 中断,用于唤醒 MCU
          SETB        EA

          NOP
          NOP
;          MOV         PCON, #IDL     ;MCU 进入 IDLE 模式
          MOV         PCON, #PD     ;MCU 进入掉电模式
          NOP
          NOP
          NOP
          CLR         P3.5          ;测试端口
          JMP         $

          END
```

6.7.7 使用 INT0/INT1/INT2/INT3/INT4 管脚中断唤醒省电模式

C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void INT0_Isr() interrupt 0
{
    P10 = !P10;          //测试端口
```



```
}

void INT1_Isr() interrupt 2
{
    P10 = !P10;                                //测试端口
}

void INT2_Isr() interrupt 10
{
    P10 = !P10;                                //测试端口
}

void INT3_Isr() interrupt 11
{
    P10 = !P10;                                //测试端口
}

void INT4_Isr() interrupt 16
{
    P10 = !P10;                                //测试端口
}

void main()
{
    P_SW2 /= 0x80;                            //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    IT0 = 0;                                  //使能INT0 上升沿和下降沿中断
    // IT0 = 1;                               //使能INT0 下降沿中断
    EX0 = 1;                                  //使能INT0 中断

    IT1 = 0;                                  //使能INT1 上升沿和下降沿中断
    // IT1 = 1;                               //使能INT1 下降沿中断
    EX1 = 1;                                  //使能INT1 中断

    INTCLKO = EX2;                            //使能INT2 下降沿中断
    INTCLKO /= EX3;                           //使能INT3 下降沿中断
    INTCLKO /= EX4;                           //使能INT4 下降沿中断

    EA = 1;

    PCON = 0x02;                             //MCU 进入掉电模式
    _nop_();                                  //掉电模式被唤醒后,MCU 首先会执行此语句
                                              //然后再进入中断服务程序

    _nop_();
    _nop_();
    _nop_();
}
```

```

    while (1)
    {
        P11 = ~P11;
    }
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

```

INTCLKO    DATA    8FH
EX2         EQU      10H
EX3         EQU      20H
EX4         EQU      40H
P_SW2      DATA    0BAH

```

```

P1M1       DATA    091H
P1M0       DATA    092H
P0M1       DATA    093H
P0M0       DATA    094H
P2M1       DATA    095H
P2M0       DATA    096H
P3M1       DATA    0B1H
P3M0       DATA    0B2H
P4M1       DATA    0B3H
P4M0       DATA    0B4H
P5M1       DATA    0C9H
P5M0       DATA    0CAH

```

```

ORG        0000H
LJMP       MAIN

```

```

ORG        0003H
LJMP       INT0ISR
ORG        0013H
LJMP       INT1ISR
ORG        0053H
LJMP       INT2ISR
ORG        005BH
LJMP       INT3ISR
ORG        0083H
LJMP       INT4ISR

```

```

ORG        0100H

```

```

INT0ISR:
CPL        P1.0          ;测试端口
RETI

```

```

INT1ISR:
CPL        P1.0          ;测试端口
RETI

```

```

INT2ISR:
CPL        P1.0          ;测试端口
RETI

```

```

INT3ISR:
CPL        P1.0          ;测试端口
RETI

```

```

INT4ISR:
CPL        P1.0          ;测试端口

```

**RETI****MAIN:**

```

MOV    SP, #5FH
ORL    P_SW2, #80H           ;使能访问 XFR

MOV    P0M0, #00H
MOV    P0M1, #00H
MOV    P1M0, #00H
MOV    P1M1, #00H
MOV    P2M0, #00H
MOV    P2M1, #00H
MOV    P3M0, #00H
MOV    P3M1, #00H
MOV    P4M0, #00H
MOV    P4M1, #00H
MOV    P5M0, #00H
MOV    P5M1, #00H

CLR    IT0                   ;使能 INT0 上升沿和下降沿中断
; SETB  IT0                   ;使能 INT0 下降沿中断
SETB    EX0                   ;使能 INT0 中断

CLR    IT1                   ;使能 INT1 上升沿和下降沿中断
; SETB  IT1                   ;使能 INT1 下降沿中断
SETB    EX1                   ;使能 INT1 中断

MOV    INTCLKO, #EX2          ;使能 INT2 下降沿中断
ORL    INTCLKO, #EX3          ;使能 INT3 下降沿中断
ORL    INTCLKO, #EX4          ;使能 INT4 下降沿中断

SETB    EA

MOV    PCON, #02H             ;MCU 进入掉电模式
NOP                                     ;掉电模式被唤醒后, MCU 首先会执行此语句
NOP                                     ;然后再进入中断服务程序
NOP
NOP
NOP

LOOP:  CPL    P1.1
        JMP    LOOP

END

```

## 6.7.8 使用 T0/T1/T2/T3/T4 管脚中断唤醒省电模式

### C 语言代码

```
//测试工作频率为 11.0592MHz
```

```
#include "stc8h.h"
#include "intrins.h"
```

```
void TM0_Isr() interrupt 1
{
```

```
    P10 = !P10;                                     //测试端口
}

void TM1_Isr() interrupt 3
{
    P10 = !P10;                                     //测试端口
}

void TM2_Isr() interrupt 12
{
    P10 = !P10;                                     //测试端口
}

void TM3_Isr() interrupt 19
{
    P10 = !P10;                                     //测试端口
}

void TM4_Isr() interrupt 20
{
    P10 = !P10;                                     //测试端口
}

void main()
{
    P_SW2 /= 0x80;                                  //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x00;
    TL0 = 0x66;                                     //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1;                                        //启动定时器
    ET0 = 1;                                        //使能定时器中断

    TL1 = 0x66;                                     //65536-11.0592M/12/1000
    TH1 = 0xfc;
    TR1 = 1;                                        //启动定时器
    ET1 = 1;                                        //使能定时器中断

    T2L = 0x66;                                     //65536-11.0592M/12/1000
    T2H = 0xfc;
    AUXR = 0x10;                                    //启动定时器
    IE2 = ET2;                                       //使能定时器中断

    T3L = 0x66;                                     //65536-11.0592M/12/1000
    T3H = 0xfc;
    T4T3M = 0x08;                                   //启动定时器
```

```

IE2 /= ET3;                                     //使能定时器中断

T4L = 0x66;                                     //65536-11.0592M/12/1000
T4H = 0xfc;
T4T3M /= 0x80;                                 //启动定时器
IE2 /= ET4;                                     //使能定时器中断

EA = 1;

PCON = 0x02;                                   //MCU 进入掉电模式
_nop_();                                       //掉电唤醒后不会立即进入中断服务程序,
                                              //而是等到定时器溢出后才会进入中断服务程序

_nop_();
_nop_();
_nop_();

while (1)
{
    P1I = ~P1I;
}
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|          |      |      |
|----------|------|------|
| T2L      | DATA | 0D7H |
| T2H      | DATA | 0D6H |
| T3L      | DATA | 0D5H |
| T3H      | DATA | 0D4H |
| T4L      | DATA | 0D3H |
| T4H      | DATA | 0D2H |
| T4T3M    | DATA | 0D1H |
| AUXR     | DATA | 8EH  |
| P_SW2    | DATA | 0BAH |
|          |      |      |
| IE2      | DATA | 0AFH |
| ET2      | EQU  | 04H  |
| ET3      | EQU  | 20H  |
| ET4      | EQU  | 40H  |
|          |      |      |
| AUXINTIF | DATA | 0EFH |
| T2IF     | EQU  | 01H  |
| T3IF     | EQU  | 02H  |
| T4IF     | EQU  | 04H  |
|          |      |      |
| P1M1     | DATA | 091H |
| P1M0     | DATA | 092H |
| P0M1     | DATA | 093H |
| P0M0     | DATA | 094H |
| P2M1     | DATA | 095H |
| P2M0     | DATA | 096H |
| P3M1     | DATA | 0B1H |
| P3M0     | DATA | 0B2H |
| P4M1     | DATA | 0B3H |
| P4M0     | DATA | 0B4H |
| P5M1     | DATA | 0C9H |
| P5M0     | DATA | 0CAH |

```

    ORG      0000H
    LJMP     MAIN
    ORG      000BH
    LJMP     TM0ISR
    ORG      001BH
    LJMP     TM1ISR
    ORG      0063H
    LJMP     TM2ISR
    ORG      009BH
    LJMP     TM3ISR
    ORG      00A3H
    LJMP     TM4ISR

TM0ISR:    ORG      0100H

    CPL      P1.0          ;测试端口
    RETI

TM1ISR:    CPL      P1.0          ;测试端口
    RETI

TM2ISR:    CPL      P1.0          ;测试端口
    RETI

TM3ISR:    CPL      P1.0          ;测试端口
    RETI

TM4ISR:    CPL      P1.0          ;测试端口
    RETI

MAIN:

    MOV      SP, #5FH
    ORL      P_SW2, #80H    ;使能访问 XFR

    MOV      P0M0, #00H
    MOV      P0M1, #00H
    MOV      P1M0, #00H
    MOV      P1M1, #00H
    MOV      P2M0, #00H
    MOV      P2M1, #00H
    MOV      P3M0, #00H
    MOV      P3M1, #00H
    MOV      P4M0, #00H
    MOV      P4M1, #00H
    MOV      P5M0, #00H
    MOV      P5M1, #00H

    MOV      TMOD, #00H
    MOV      TL0, #66H      ;65536-11.0592M/12/1000
    MOV      TH0, #0FCH
    SETB     TR0            ;启动定时器
    SETB     ET0            ;使能定时器中断

    MOV      TL1, #66H      ;65536-11.0592M/12/1000
    MOV      TH1, #0FCH
    SETB     TR1            ;启动定时器
    SETB     ET1            ;使能定时器中断

    MOV      T2L, #66H      ;65536-11.0592M/12/1000

```

```

MOV    T2H,#0FCH
MOV    AUXR,#10H           ;启动定时器
MOV    IE2,#ET2           ;使能定时器中断

MOV    T3L,#66H           ;65536-11.0592M/12/1000
MOV    T3H,#0FCH
MOV    T4T3M,#08H        ;启动定时器
ORL    IE2,#ET3          ;使能定时器中断

MOV    T4L,#66H           ;65536-11.0592M/12/1000
MOV    T4H,#0FCH
ORL    T4T3M,#80H        ;启动定时器
ORL    IE2,#ET4          ;使能定时器中断

SETB   EA

MOV    PCON,#02H          ;MCU 进入掉电模式
NOP                    ;掉电唤醒后不会立即进入中断服务程序,
                        ;而是等到定时器溢出后才会进入中断服务程序

NOP
NOP
NOP
LOOP:  CPL    PL1
      JMP    LOOP

      END

```

## 6.7.9 使用 RxD/RxD2/RxD3/RxD4 管脚中断唤醒省电模式

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

void UART1_Isr() interrupt 4
{
}

```

```

void UART2_Isr() interrupt 8
{
}

```

```

void UART3_Isr() interrupt 17
{
}

```

```

void UART4_Isr() interrupt 18
{
}

```

```

void main()
{
    P_SW2 /= 0x80;           //使能访问 XFR
}

```

```

P0M0 = 0x00;
P0M1 = 0x00;
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

P_SW1 = 0x00; //RXD/P3.0 下降沿唤醒
// P_SW1 = 0x40; //RXD_2/P3.6 下降沿唤醒
// P_SW1 = 0x80; //RXD_3/P1.6 下降沿唤醒
// P_SW1 = 0xc0; //RXD_4/P4.3 下降沿唤醒

P_SW2 /= 0x00; //RXD2/P1.0 下降沿唤醒
// P_SW2 /= 0x01; //RXD2_2/P4.6 下降沿唤醒

P_SW2 /= 0x00; //RXD3/P0.0 下降沿唤醒
// P_SW2 /= 0x02; //RXD3_2/P5.0 下降沿唤醒

P_SW2 /= 0x00; //RXD4/P0.2 下降沿唤醒
// P_SW2 /= 0x04; //RXD4_2/P5.2 下降沿唤醒

ES = 1; //使能串口中断
IE2 = ES2; //使能串口中断
IE2 /= ES3; //使能串口中断
IE2 /= ES4; //使能串口中断
EA = 1;

PCON = 0x02; //MCU 进入掉电模式
_nop_(); //掉电唤醒后不会进入中断服务程序,
_nop_();
_nop_();
_nop_();

while (1)
{
    P11 = ~P11;
}
}

```

## 汇编代码

;测试工作频率为11.0592MHz

|       |      |      |
|-------|------|------|
| IE2   | DATA | 0AFH |
| ES2   | EQU  | 01H  |
| ES3   | EQU  | 08H  |
| ES4   | EQU  | 10H  |
| P_SW1 | DATA | 0A2H |
| P_SW2 | DATA | 0BAH |
| P0M1  | DATA | 093H |



```

P0M0    DATA    094H
P1M1    DATA    091H
P1M0    DATA    092H
P2M1    DATA    095H
P2M0    DATA    096H
P3M1    DATA    0B1H
P3M0    DATA    0B2H
P4M1    DATA    0B3H
P4M0    DATA    0B4H
P5M1    DATA    0C9H
P5M0    DATA    0CAH

```

```

ORG      0000H
LJMP     MAIN
ORG      0023H
LJMP     UART1ISR
ORG      0043H
LJMP     UART2ISR
ORG      008BH
LJMP     UART3ISR
ORG      0093H
LJMP     UART4ISR

```

```

ORG      0100H

```

```

UART1ISR:

```

```

    RETI

```

```

UART2ISR:

```

```

    RETI

```

```

UART3ISR:

```

```

    RETI

```

```

UART4ISR:

```

```

    RETI

```

```

MAIN:

```

```

    MOV    SP, #5FH

```

```

    ORL    P_SW2, #80H           ;使能访问 XFR

```

```

    MOV    P0M0, #00H

```

```

    MOV    P0M1, #00H

```

```

    MOV    P1M0, #00H

```

```

    MOV    P1M1, #00H

```

```

    MOV    P2M0, #00H

```

```

    MOV    P2M1, #00H

```

```

    MOV    P3M0, #00H

```

```

    MOV    P3M1, #00H

```

```

    MOV    P4M0, #00H

```

```

    MOV    P4M1, #00H

```

```

    MOV    P5M0, #00H

```

```

    MOV    P5M1, #00H

```

```

    MOV    P_SW1, #00H           ;RXD/P3.0 下降沿唤醒

```

```

;    MOV    P_SW1, #40H           ;RXD_2/P3.6 下降沿唤醒

```

```

;    MOV    P_SW1, #80H           ;RXD_3/P1.6 下降沿唤醒

```

```

;    MOV    P_SW1, #0C0H          ;RXD_4/P4.3 下降沿唤醒

```

```

    ORL    P_SW2, #00H           ;RXD2/P1.0 下降沿唤醒

```

```

;    ORL    P_SW2, #01H           ;RXD2_2/P4.6 下降沿唤醒

```

```

    ORL    P_SW2, #00H           ;RXD3/P0.0 下降沿唤醒

```

```
; ORL P_SW2,#02H ;RXD3_2/P5.0 下降沿唤醒

ORL P_SW2,#00H ;RXD4/P0.2 下降沿唤醒
; ORL P_SW2,#04H ;RXD4_2/P5.2 下降沿唤醒

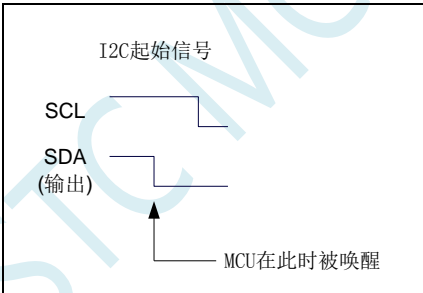
SETB ES ;使能串口中断
MOV IE2,#ES2 ;使能串口中断
ORL IE2,#ES3 ;使能串口中断
ORL IE2,#ES4 ;使能串口中断
SETB EA

MOV PCON,#02H ;MCU 进入掉电模式
NOP ;掉电唤醒后不会进入中断服务程序
NOP
NOP
NOP

LOOP: CPL PL1
      JMP LOOP

END
```

6.7.10 使用 I2C 的 SDA 脚唤醒 MCU 省电模式



C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void i2c_isr() interrupt 24
{
    I2CSLST &= ~0x40;
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
```

```
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

P_SW2 /= 0x00;
// P_SW2 /= 0x10;
// P_SW2 /= 0x30;
I2CCFG = 0x80;
I2CSLCR = 0x40;
EA = 1;

PCON = 0x02;
_nop_();
_nop_();
_nop_();
_nop_();

while (1)
{
    P1I = ~P1I;
}
}
```

汇编代码

;测试工作频率为 11.0592MHz

|         |       |        |
|---------|-------|--------|
| P_SW2   | DATA  | 0BAH   |
| I2CCFG  | XDATA | 0FE80H |
| I2CSLCR | XDATA | 0FE83H |
| I2CSLST | XDATA | 0FE84H |
| P0M1    | DATA  | 093H   |
| P0M0    | DATA  | 094H   |
| P1M1    | DATA  | 091H   |
| P1M0    | DATA  | 092H   |
| P2M1    | DATA  | 095H   |
| P2M0    | DATA  | 096H   |
| P3M1    | DATA  | 0B1H   |
| P3M0    | DATA  | 0B2H   |
| P4M1    | DATA  | 0B3H   |
| P4M0    | DATA  | 0B4H   |
| P5M1    | DATA  | 0C9H   |
| P5M0    | DATA  | 0CAH   |
|         | ORG   | 0000H  |
|         | LJMP  | MAIN   |
|         | ORG   | 00C3H  |
|         | LJMP  | I2CISR |
|         | ORG   | 0100H  |
| I2CISR: |       |        |
|         | PUSH  | ACC    |
|         | PUSH  | DPH    |
|         | PUSH  | DPL    |

```

    ORL        PSW2,#80H
    MOV        DPTR,#I2CSLST
    MOVX       A,@DPTR
    ANL        A,#NOT 40H
    MOVX       @DPTR,A
    POP        DPL
    POP        DPH
    POP        ACC
    RETI

MAIN:
    MOV        SP,#5FH
    ORL        P_SW2,#80H                ;使能访问 XFR

    MOV        P0M0,#00H
    MOV        P0M1,#00H
    MOV        P1M0,#00H
    MOV        P1M1,#00H
    MOV        P2M0,#00H
    MOV        P2M1,#00H
    MOV        P3M0,#00H
    MOV        P3M1,#00H
    MOV        P4M0,#00H
    MOV        P4M1,#00H
    MOV        P5M0,#00H
    MOV        P5M1,#00H

    ORL        P_SW2,#00H                ;SDA/P1.4 下降沿唤醒
//    ORL        P_SW2,#10H                ;SDA_2/P2.4 下降沿唤醒
//    ORL        P_SW2,#30H                ;SDA_4/P3.3 下降沿唤醒

    MOV        DPTR,#I2CCFG
    MOV        A,#80H
    MOVX       @DPTR,A                ;使能 I2C 模块的从机模式
    MOV        DPTR,#I2CSLCR
    MOV        A,#40H                ;使能起始信号中断
    SETB       EA

    MOV        PCON,#02H                ;MCU 进入掉电模式
    NOP                    ;掉电唤醒后不会进入中断服务程序,
    NOP
    NOP
    NOP

LOOP:
    CPL        P1.1
    JMP        LOOP

END

```

## 6.7.11 使用掉电唤醒定时器唤醒省电模式

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
```

```
#include "intrins.h"

void main()
{
    P_SW2 /= 0x80;                                     //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    WKTCL = 0xff;                                     //设定掉电唤醒时钟约为1 秒钟
    WKTCH = 0x87;

    while (1)
    {
        _nop_();
        _nop_();
        PCON = 0x02;                                //MCU 进入掉电模式
        _nop_();
        _nop_();
        _nop_();
        _nop_();
        P11 = ~P11;
    }
}
```

汇编代码

;测试工作频率为11.0592MHz

|       |      |       |
|-------|------|-------|
| WKTCL | DATA | 0AAH  |
| WKTCH | DATA | 0ABH  |
| P_SW2 | DATA | 0BAH  |
|       |      |       |
| P0M1  | DATA | 093H  |
| P0M0  | DATA | 094H  |
| P1M1  | DATA | 091H  |
| P1M0  | DATA | 092H  |
| P2M1  | DATA | 095H  |
| P2M0  | DATA | 096H  |
| P3M1  | DATA | 0B1H  |
| P3M0  | DATA | 0B2H  |
| P4M1  | DATA | 0B3H  |
| P4M0  | DATA | 0B4H  |
| P5M1  | DATA | 0C9H  |
| P5M0  | DATA | 0CAH  |
|       |      |       |
|       | ORG  | 0000H |
|       | LJMP | MAIN  |

```

        ORG            0100H

MAIN:

        MOV           SP, #5FH
        ORL           P_SW2, #80H           ;使能访问 XFR

        MOV           P0M0, #00H
        MOV           P0M1, #00H
        MOV           P1M0, #00H
        MOV           P1M1, #00H
        MOV           P2M0, #00H
        MOV           P2M1, #00H
        MOV           P3M0, #00H
        MOV           P3M1, #00H
        MOV           P4M0, #00H
        MOV           P4M1, #00H
        MOV           P5M0, #00H
        MOV           P5M1, #00H

        MOV           WKTCL, #0FFH         ;设定掉电唤醒时钟约为 1 秒钟
        MOV           WKTCH, #87H

LOOP:

        NOP
        NOP
        MOV           PCON, #02H           ;MCU 进入掉电模式
        NOP
        NOP
        NOP
        NOP
        CPL           P1.1
        JMP           LOOP

        END

```

## 6.7.12 LVD 中断唤醒省电模式，建议配合使用掉电唤醒定时器

时钟停振省电模式下，不建议启动 LVD 和比较器，否则硬件系统还会自动启动内部 1.19V 的高精度参考源，这个高精度参考源有相应的抗温漂和调校线路，大约会额外增加 300uA 的耗电，而 MCU 进入时钟停振模式后，3.3V 工作电压时只耗约 0.4uA 的电流，所以进入时钟停振模式时不建议开 LVD 和比较器。如果确实需要用，建议开启掉电唤醒定时器，掉电唤醒定时器只会增加约 1.4uA 的耗电，这个耗电一般系统是可以接受的。让掉电唤醒定时器每 5 秒唤醒一次 MCU，唤醒后可用 LVD、比较器、ADC 检测外部电池电压，检测工作约耗时 1mS 后再进入时钟停振/省电模式，这样增加的平均电流小于 1uA，则整体功耗大约为 2.8uA (0.4uA + 1.4uA + 1uA)。

### C 语言代码

```
//测试工作频率为 11.0592MHz
```

```
#include "stc8h.h"
#include "intrins.h"
```

```
void LVD_Isr() interrupt 6
{
```

```
    PCON &= ~LVDF;
```

```
    //清中断标志
```

```

    P10 = !P10;                                     //测试端口
}

void main()
{
    P_SW2 /= 0x80;                                   //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    PCON &= ~LVDF;                                   //上电需要清中断标志
    RSTCFG = LVD3V0;                                  //设置LVD 电压为3.0V
    ELVD = 1;                                         //使能LVD 中断
    EA = 1;

    PCON = 0x02;                                     //MCU 进入掉电模式
    _nop_();                                          //掉电唤醒后立即进入中断服务程序
    _nop_();
    _nop_();
    _nop_();

    while (1)
    {
        P11 = ~P11;
    }
}

```

## 汇编代码

;测试工作频率为11.0592MHz

|        |      |      |           |
|--------|------|------|-----------|
| P_SW2  | DATA | 0BAH |           |
| RSTCFG | DATA | 0FFH |           |
| ENLVR  | EQU  | 40H  | ;RSTCFG.6 |
| LVD2V0 | EQU  | 00H  | ;LVD@2.0V |
| LVD2V4 | EQU  | 01H  | ;LVD@2.4V |
| LVD2V7 | EQU  | 02H  | ;LVD@2.7V |
| LVD3V0 | EQU  | 03H  | ;LVD@3.0V |
| ELVD   | BIT  | 1E.6 |           |
| LVDF   | EQU  | 20H  | ;PCON.5   |
| P1M1   | DATA | 091H |           |
| P1M0   | DATA | 092H |           |
| P0M1   | DATA | 093H |           |
| P0M0   | DATA | 094H |           |
| P2M1   | DATA | 095H |           |
| P2M0   | DATA | 096H |           |
| P3M1   | DATA | 0B1H |           |

```

P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0033H
          LJMP         LVDISR

          ORG          0100H
LVDISR:
          ANL          PCON,#NOT LVDF      ;清中断标志
          CPL          P1.0                ;测试端口
          RETI

MAIN:
          MOV          SP,#5FH
          ORL          P_SW2,#80H          ;使能访问 XFR

          MOV          P0M0,#00H
          MOV          P0M1,#00H
          MOV          P1M0,#00H
          MOV          P1M1,#00H
          MOV          P2M0,#00H
          MOV          P2M1,#00H
          MOV          P3M0,#00H
          MOV          P3M1,#00H
          MOV          P4M0,#00H
          MOV          P4M1,#00H
          MOV          P5M0,#00H
          MOV          P5M1,#00H

          ANL          PCON,#NOT LVDF      ;上电需要清中断标志
          MOV          RSTCFG,#LVD3V0     ;设置 LVD 电压为 3.0V
          SETB         ELVD                ;使能 LVD 中断
          SETB         EA

          MOV          PCON,#02H           ;MCU 进入掉电模式
          NOP
          NOP
          NOP
          NOP

LOOP:
          CPL          P1.1
          JMP          LOOP

          END

```



### 6.7.13 比较器中断唤醒省电模式，建议配合使用掉电唤醒定时器

时钟停振省电模式下，不建议启动 LVD 和比较器，否则硬件系统还会自动启动内部 1.19V 的高精准参考源，这个高精度参考源有相应的抗温漂和调校线路，大约会额外增加 300uA 的耗电，而 MCU 进入时钟停振模式后，3.3V 工作电压时只耗约 0.4uA 的电流，所以进入时钟停振模式时不建议开 LVD 和比较器。如果确实需要用，建议开启掉电唤醒定时器，掉电唤醒定时器只会增加约 1.4uA 的耗电，这个耗电一般系统是可以接受的。让掉电唤醒定时器每 5 秒唤醒一次 MCU，唤醒后可用 LVD、比较器、ADC 检测外部电池电压，检测工作约耗时 1mS 后再进入时钟停振/省电模式，这样增加的平均电流小于 1uA，则整体功耗大约为 2.8uA (0.4uA + 1.4uA + 1uA)。

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void CMP_Isr() interrupt 21
{
    CMPCR1 &= ~0x40;
    P10 = !P10;
}
```

//清中断标志  
//测试端口

```
void main()
{
```

```
    P_SW2 /= 0x80;
```

//使能访问 XFR

```
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;
```

```
    CMPCR2 = 0x00;
    CMPCR1 = 0x80;
    CMPCR1 /= 0x30;
    CMPCR1 &= ~0x08;
    CMPCR1 /= 0x04;
    CMPCR1 /= 0x02;
    EA = 1;
```

//使能比较器模块  
//使能比较器边沿中断  
//P3.6 为 CMP+ 输入脚  
//P3.7 为 CMP- 输入脚  
//使能比较器输出

```
    PCON = 0x02;
    _nop_();
    _nop_();
    _nop_();
    _nop_();
```

//MCU 进入掉电模式  
//掉电唤醒后立即进入中断服务程序

```
    while (1)
```

```
{
    P1I = ~P1I;
}
```

汇编代码

;测试工作频率为 11.0592MHz

```
CMPCR1    DATA    0E6H
CMPCR2    DATA    0E7H
P_SW2     DATA    0BAH

P1M1      DATA    091H
P1M0      DATA    092H
P0M1      DATA    093H
P0M0      DATA    094H
P2M1      DATA    095H
P2M0      DATA    096H
P3M1      DATA    0B1H
P3M0      DATA    0B2H
P4M1      DATA    0B3H
P4M0      DATA    0B4H
P5M1      DATA    0C9H
P5M0      DATA    0CAH

ORG       0000H
LJMP      MAIN
ORG       00ABH
LJMP      CMPISR

CMPISR:    ORG       0100H

ANL       CMPCR1,#NOT 40H    ;清中断标志
CPL       P1.0              ;测试端口
RETI

MAIN:      MOV       SP,#5FH

ORL       P_SW2,#80H        ;使能访问 XFR

MOV       P0M0,#00H
MOV       P0M1,#00H
MOV       P1M0,#00H
MOV       P1M1,#00H
MOV       P2M0,#00H
MOV       P2M1,#00H
MOV       P3M0,#00H
MOV       P3M1,#00H
MOV       P4M0,#00H
MOV       P4M1,#00H
MOV       P5M0,#00H
MOV       P5M1,#00H

MOV       CMPCR2,#00H
MOV       CMPCR1,#80H        ;使能比较器模块
ORL       CMPCR1,#30H        ;使能比较器边沿中断
ANL       CMPCR1,#NOT 08H    ;P3.6 为 CMP+ 输入脚
ORL       CMPCR1,#04H        ;P3.7 为 CMP- 输入脚
```

```

    ORL      CMPCR1,#02H      ;使能比较器输出
    SETB     EA

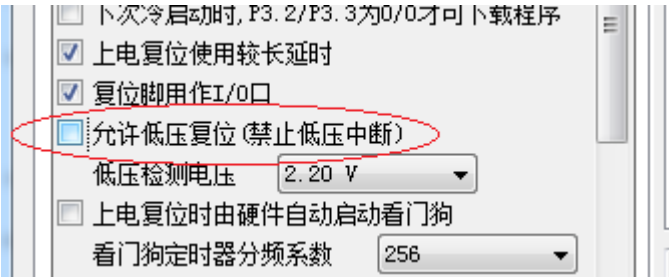
    MOV      PCON,#02H      ;MCU 进入掉电模式
    NOP      ;掉电唤醒后立即进入中断服务程序
    NOP
    NOP
    NOP

LOOP:
    CPL      PL1
    JMP      LOOP

END
```

### 6.7.14 使用 LVD 功能检测工作电压（电池电压）

若需要使用 LVD 功能检测电池电压，则在 ISP 下载时需要将低压复位功能去掉，如下图 “允许低压复位（禁止低压中断）” 的硬件选项的勾选项需要去掉



#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

#define FOSC      11059200UL
#define T1MS      (65536 - FOSC/4/100)

void delay()
{
    int i;

    for (i=0; i<100; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    unsigned char power;
```

```
P_SW2 /= 0x80;                                     //使能访问XFR

P0M0 = 0x00;
P0M1 = 0x00;
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

PCON &= ~LVDF;
RSTCFG = LVD3V0;

while (1)
{
    power = 0x0f;

    RSTCFG = LVD3V0;
    delay();
    PCON &= ~LVDF;
    delay();
    if (PCON & LVDF)
    {
        power >>= 1;
        RSTCFG = LVD2V7;
        delay();
        PCON &= ~LVDF;
        delay();
        if (PCON & LVDF)
        {
            power >>= 1;
            RSTCFG = LVD2V4;
            delay();
            PCON &= ~LVDF;
            delay();
            if (PCON & LVDF)
            {
                power >>= 1;
                RSTCFG = LVD2V2;
                delay();
                PCON &= ~LVDF;
                delay();
                if (PCON & LVDF)
                {
                    power >>= 1;
                }
            }
        }
    }
    RSTCFG = LVD3V0;
    P2 = ~power;    //P2.3~P2.0 显示电池电量
}
}
```

## 汇编代码

;测试工作频率为 11.0592MHz

```

RSTCFG      DATA      0FFH
LVD2V0      EQU         00H           ;LVD@2.0V
LVD2V4      EQU         01H           ;LVD@2.4V
LVD2V7      EQU         02H           ;LVD@2.7V
LVD3V0      EQU         03H           ;LVD@3.0V
P_SW2       DATA      0BAH

LVDF        EQU         20H           ;PCON.5

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

ORG         0000H
JMP         MAIN

MAIN:        ORG         0100H

MOV         SP, #5FH
ORL         P_SW2, #80H           ;使能访问 XFR

MOV         P0M0, #00H
MOV         P0M1, #00H
MOV         P1M0, #00H
MOV         P1M1, #00H
MOV         P2M0, #00H
MOV         P2M1, #00H
MOV         P3M0, #00H
MOV         P3M1, #00H
MOV         P4M0, #00H
MOV         P4M1, #00H
MOV         P5M0, #00H
MOV         P5M1, #00H

ANL         PCON, #NOT LVDF
MOV         RSTCFG, #LVD3V0

LOOP:       MOV         B, #0FH

MOV         RSTCFG, #LVD3V0
CALL        DELAY
ANL         PCON, #NOT LVDF
CALL        DELAY
MOV         A, PCON

```

```

ANL      A,#LVDF
JZ        SKIP
MOV      A,B
CLR      C
RRC      A
MOV      B,A

MOV      RSTCFG,#LVD2V7
CALL     DELAY
ANL      PCON,#NOT LVDF
CALL     DELAY
MOV      A,PCON
ANL      A,#LVDF
JZ        SKIP
MOV      A,B
CLR      C
RRC      A
MOV      B,A

MOV      RSTCFG,#LVD2V4
CALL     DELAY
ANL      PCON,#NOT LVDF
CALL     DELAY
MOV      A,PCON
ANL      A,#LVDF
JZ        SKIP
MOV      A,B
CLR      C
RRC      A
MOV      B,A

MOV      RSTCFG,#LVD2V2
CALL     DELAY
ANL      PCON,#NOT LVDF
CALL     DELAY
MOV      A,PCON
ANL      A,#LVDF
JZ        SKIP
MOV      A,B
CLR      C
RRC      A
MOV      B,A

```

**SKIP:**

```

MOV      A,B
CPL      A
MOV      P2,A          ;P2.3~P2.0 显示电池电量
JMP      LOOP

```

**DELAY:**

```

MOV      R0,#100

```

**NEXT:**

```

NOP
NOP
NOP
NOP
DJNZ     R0,NEXT
RET

```

*END*

---

---

STC MCU

# 7 自动频率校准，自动追频（CRE）

| 产品线                    | CRE |
|------------------------|-----|
| STC8H1K08 系列           |     |
| STC8H1K28 系列           |     |
| STC8H3K64S4 系列         |     |
| STC8H3K64S2 系列         |     |
| STC8H8K64U 系列          |     |
| STC8H4K64TL 系列         | ●   |
| STC8H4K64TLCD 系列 A 系列  |     |
| STC8H4K64TLCD 系列 A+ 系列 | ●   |
| STC8H1K08T 系列          | ●   |

STC8H 部分单片机系列内建一个频率自动校准模块（CRE），CRE 模块是使用外部的 32.768KHz 晶振对内部高速 IRC（HIRC）的 IRTRIM 寄存器进行自动调整，以达到自动频率校准的功能。需要使用自动校准时，只需要根据给定的公式设置好目标频率的计数值和误差范围，然启动 CRE 模块，硬件便会进行自动频率校准，当 HIRC 的频率达到用户所设置误差范围内时，校准完成标志会被置位。

## 7.1 相关寄存器

| 符号      | 描述           | 地址    | 位地址与符号    |      |          |    |       |        |        |        | 复位值       |
|---------|--------------|-------|-----------|------|----------|----|-------|--------|--------|--------|-----------|
|         |              |       | B7        | B6   | B5       | B4 | B3    | B2     | B1     | B0     |           |
| CRECR   | CRE 控制寄存器    | FDA8H | ENCRE     | MONO | UPT[1:0] |    | CREHF | CREINC | CREDEC | CRERDY | 0000,0000 |
| CRECNTH | CRE 校准目标寄存器  | FDA9H | CNT[15:8] |      |          |    |       |        |        |        | 0000,0000 |
| CRECNTL | CRE 校准目标寄存器  | FDAAH | CNT[7:0]  |      |          |    |       |        |        |        | 0000,0000 |
| CRERES  | CRE 分辨率控制寄存器 | FDABH | RES[7:0]  |      |          |    |       |        |        |        | 0000,0000 |

### 7.1.1 CRE 控制寄存器（CRECR）

| 符号    | 地址    | B7    | B6   | B5       | B4 | B3    | B2     | B1     | B0     |
|-------|-------|-------|------|----------|----|-------|--------|--------|--------|
| CRECR | FDA8H | ENCRE | MONO | UPT[1:0] |    | CREHF | CREINC | CREDEC | CRERDY |

ENCRE：CRE模块控制位

- 0：关闭 CRE 模块。
- 1：使能 CRE 模块。

MONO：自动校准步幅控制

- 0：单步模式。每个校准周期，硬件自动将 IRTRIM 递增或递减 1。
  - 1：双步模式。每个校准周期，硬件自动将 IRTRIM 递增或递减 2。
- 单步模式比双步模式校准后的 IRC 精度更高，但自动校准的时间比双步模式长。

UPT[1:0]：CRE 校准周期选择

| UPT[1:0] | 校准周期 |
|----------|------|
| 00       | 1ms  |



|    |      |
|----|------|
| 01 | 4ms  |
| 10 | 32ms |
| 11 | 64ms |

CREHF: 高频模式选择

0: 低频模式 (目标频率小于或等于 50MHz)。

1: 高频模式 (目标频率大于 50MHz)。

CREINC: CRE校准正处于上调状态。只读位。

CREDEC: CRE校准正处于下调状态。只读位。

CRERDY: CRE校准完成状态。只读位。

0: CRE 校准功能未启动或者未校准完成。

1: CRE 校准已完成。

## 7.1.2 CRE 校准计数值寄存器 (CRECNT)

| 符号      | 地址    | B7           | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------|-------|--------------|----|----|----|----|----|----|----|
| CRECNTH | FDA9H | CRECNT[15:8] |    |    |    |    |    |    |    |
| CRECNTL | FDAAH | CRECNT[7:0]  |    |    |    |    |    |    |    |

CRECNT[15:0]: 16位校准计数值。

目标校准值计算公式:

低频模式 (CREHF=0):  $CRECNT = (16 * \text{目标频率(Hz)}) / 32768$

高频模式 (CREHF=1):  $CRECNT = (8 * \text{目标频率(Hz)}) / 32768$

(详细设置见范例程序)

## 7.1.3 CRE 校准误差值寄存器 (CRERES)

| 符号     | 地址    | B7          | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-------|-------------|----|----|----|----|----|----|----|
| CRERES | FDABH | CRERES[7:0] |    |    |    |    |    |    |    |

CRERES[7:0]: 8位校准误差值 (解析度控制)。

由于内部高速IRC的解析度远低于外部的32.768K晶振, 最终的校准值无法与CRECNT所设置的目标值完全一致, 所以必须通过CRERES寄存器设定一个误差范围。

校准误差计算公式:

$CRERES = \text{误差范围(\%)} * \text{目标校准值}$

(误差范围一般控制在 1%~0.3%即可, 不建议超出此范围)

(详细设置见范例程序)

## 7.2 范例程序

### 7.2.1 自动校准内部高速 IRC (HIRC)

例如: 校准的目标频率为22.1184MHz, 校准误差范围为 $\pm 0.5\%$

则需要将CREHF设置为0, CRECNT设置为 $(16 * 22118400) / 32768 = 10800$  (2A30H),

即将CRECNTH设置为2AH, CRECNTL设置为30H, CRERES设置为 $10800 * 0.5\% = 54$  (36H)

---



---

//测试工作频率为11.0592MHz

```
#include "stc8h.h"                                     //头文件见下载软件

#define CNT22M      (16 * 22118400L) / 32768           //校准目标频率为 22.1184M
#define RES22M      (CNT22M * 5 / 1000)               //设置校准误差为 0.5%

void main()
{
    P_SW2 /= 0x80;                                     //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    X32KCR = 0xc0;                                     //启动外部 32K 晶振
    while (!(X32KCR & 1));                             //等待时钟稳定

    IRCBAND &= ~0x03;
    IRCBAND /= 0x02;                                   //选择 27M 频段
    CLKSEL = 0x00;                                     //选择内部高速 HIRC 为系统时钟

    CRECNTH = CNT22M >> 8;                             //设置目标校准值
    CRECNTL = CNT22M;
    CRERES = RES22M;                                   //设置校准误差
    CRECR = 0x90;                                       //使能 CRE 功能, 并设置校准周期为 4ms

    while (1)
    {
        if (CRECR & 0x01)
        {
            //频率自动校准完成
        }
    }
}
```

---



---

STC MCU

## 8 存储器

STC8H 系列单片机的程序存储器和数据存储器是各自独立编址的。由于没有提供访问外部程序存储器的总线，所有单片机的所有程序存储器都是片上 Flash 存储器，不能访问外部程序存储器。

STC8H 系列单片机内部集成了大容量的数据存储器。STC8H 系列单片机内部的数据存储器在物理和逻辑上都分为两个地址空间:内部 RAM(256 字节)和内部扩展 RAM。其中内部 RAM 的高 128 字节的数据存储器与特殊功能寄存器(SFRs)地址重叠，实际使用时通过不同的寻址方式加以区分。

### 8.1 程序存储器

程序存储器用于存放用户程序、数据以及表格等信息。

单片机复位后，程序计数器(PC)的内容为 0000H，从 0000H 单元开始执行程序。另外中断服务程序的入口地址(又称中断向量)也位于程序存储器单元。在程序存储器中，每个中断都有一个固定的入口地址，当中断发生并得到响应后，单片机就会自动跳转到相应的中断入口地址去执行程序。外部中断 0 (INT0) 的中断服务程序的入口地址是 0003H，定时器/计数器 0 (TIMER0) 中断服务程序的入口地址是 000BH，外部中断 1 (INT1) 的中断服务程序的入口地址是 0013H，定时器/计数器 1 (TIMER1) 的中断服务程序的入口地址是 001BH 等。更多的中断服务程序的入口地址(中断向量)请参考中断介绍章节。

由于相邻中断入口地址的间隔区间仅仅有 8 个字节，一般情况下无法保存完整的中断服务程序，因此在中断响应的地址区域存放一条无条件转移指令，指向真正存放中断服务程序的空间去执行。

STC8 系列单片机中都包含有 Flash 数据存储器(EEPROM)。以字节为单位进行读/写数据，以 512 字节为页单位进行擦除，可在线反复编程擦写 10 万次以上，提高了使用的灵活性和方便性。

## 8.2 数据存储器

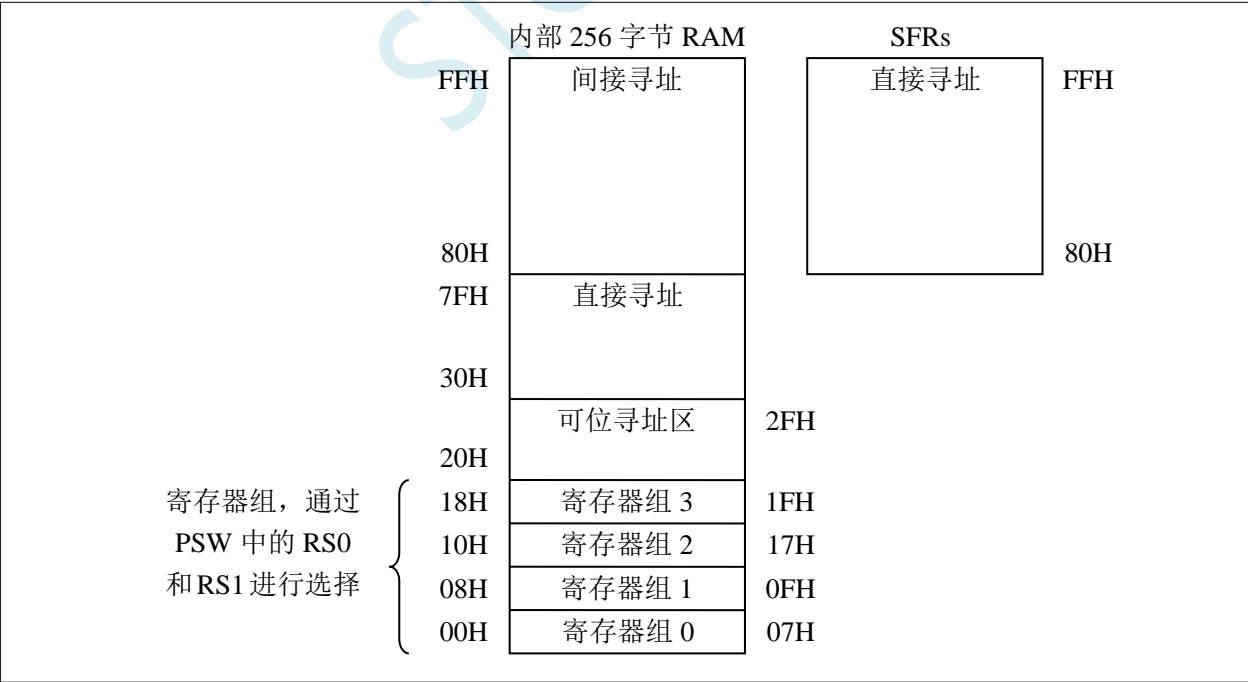
STC8H 系列单片机内部集成的 RAM 可用于存放程序执行的中间结果和过程数据。

| 单片机系列            | 内部直接访问 RAM<br>(DATA) | 内部间接访问 RAM<br>(IDATA) | 内部扩展 RAM<br>(XDATA) |
|------------------|----------------------|-----------------------|---------------------|
| STC8H1K08 系列     | 128 字节               | 128 字节                | 1024 字节             |
| STC8H1K28 系列     | 128 字节               | 128 字节                | 1024 字节             |
| STC8H3K64S4 系列   | 128 字节               | 128 字节                | 3072 字节             |
| STC8H3K64S2 系列   | 128 字节               | 128 字节                | 3072 字节             |
| STC8H8K64U 系列    | 128 字节               | 128 字节                | 8192 字节             |
| STC8H4K64TL 系列   | 128 字节               | 128 字节                | 4096 字节             |
| STC8H4K64TLCD 系列 | 128 字节               | 128 字节                | 4096 字节             |
| STC8H1K08T 系列    | 128 字节               | 128 字节                | 1024 字节             |

### 8.2.1 内部 RAM

内部 RAM 共 256 字节，可分为 2 个部分：低 128 字节 RAM 和高 128 字节 RAM。低 128 字节的数据存储器与传统 8051 兼容，既可直接寻址也可间接寻址。高 128 字节 RAM（在 8052 中扩展了高 128 字节 RAM）与特殊功能寄存器区共用相同的逻辑地址，都使用 80H~FFH，但在物理上是分别独立的，使用时通过不同的寻址方式加以区分。高 128 字节 RAM 只能间接寻址，特殊功能寄存器区只可直接寻址。

内部 RAM 的结构如下图所示：



低 128 字节 RAM 也称通用 RAM 区。通用 RAM 区又可分为工作寄存器组区，可位寻址区，用户 RAM 区和堆栈区。工作寄存器组区地址从 00H~1FH 共 32 字节单元，分为 4 组，每一组称为一个寄存器组，每组包含 8 个 8 位的工作寄存器，编号均为 R0 ~ R7，但属于不同的物理空间。通过使用工作寄存器组，

可以提高运算速度。R0~R7 是常用的寄存器，提供 4 组是因为 1 组往往不够用。程序状态字 PSW 寄存器中的 RS1 和 RS0 组合决定当前使用的工作寄存器组，见下面 PSW 寄存器的介绍。

8.2.2 程序状态寄存器（PSW）

| 符号  | 地址  | B7 | B6 | B5 | B4  | B3  | B2 | B1 | B0 |
|-----|-----|----|----|----|-----|-----|----|----|----|
| PSW | D0H | CY | AC | F0 | RS1 | RS0 | OV | F1 | P  |

CY：进/借位标志位。

AC：辅组进/借位标志位。

F0：用户标志位 0。

RS1，RS0：工作寄存器选择位

| RS1 | RS0 | 工作寄存器组（R0~R7）  |
|-----|-----|----------------|
| 0   | 0   | 第 0 组（00H~07H） |
| 0   | 1   | 第 1 组（08H~0FH） |
| 1   | 0   | 第 2 组（10H~17H） |
| 1   | 1   | 第 3 组（18H~1FH） |

OV：溢出标志位。

F1：用户标志位 1。

P：奇偶校验标志位。

可位寻址区的地址从 20H~2FH 共 16 个字节单元。20H~2FH 单元既可像普通 RAM 单元一样按字节存取，也可以对单元中的任何一位单独存取，共 128 位，所对应的逻辑位地址范围是 00H~7FH。位地址范围是 00H~7FH，内部 RAM 低 128 字节的地址也是 00H~7FH，从外表看，二者地址是一样的，实际上二者具有本质的区别：位地址指向的是一个位，而字节地址指向的是一个字节单元，在程序中使用不同的指令区分。

内部 RAM 中的 30H~FFH 单元是用户 RAM 和堆栈区。一个 8 位的堆栈指针(SP)，用于指向堆栈区。单片机复位后，堆栈指针 SP 为 07H，指向了工作寄存器组 0 中的 R7，因此，用户初始化程序都应对 SP 设置初值，一般设置在 80H 以后的单元为宜。

堆栈指针是一个 8 位专用寄存器。它指示出堆栈顶部在内部 RAM 块中的位置。系统复位后，SP 初始化位 07H，使得堆栈事实上由 08H 单元开始，考虑 08H~1FH 单元分别属于工作寄存器组 1~3，若在程序设计中用到这些区，则最好把 SP 值改变为 80H 或更大的值为宜。STC8 系列单片机的堆栈是向上生长的，即将数据压入堆栈后，SP 内容增大。

## 8.2.3 内部扩展 RAM, XRAM, XDATA

STC8H 系列单片机片内除了集成 256 字节的内部 RAM 外, 还集成了内部的扩展 RAM。访问内部扩展 RAM 的方法和传统 8051 单片机访问外部扩展 RAM 的方法相同, 但是不影响 P0 口、P2 口、以及 RD、WR 和 ALE 等端口上的信号。

在汇编语言中, 内部扩展 RAM 通过 MOVX 指令访问,

```
MOVX    A,@DPTR
MOVX    @DPTR,A
MOVX    A,@Ri
MOVX    @Ri,A
```

在 C 语言中, 可使用 xdata 声明存储类型即可。如:

```
unsigned char xdata i;
```

注: pdata 即为 xdata 的低 256 字节, 用 MOVX @Ri,A 和 MOVX A,@Ri 进行访问。但读写 pdata 类型的变量比 xdata 类型要慢, 所有建议用户代码中均统一使用 xdata 在扩展 RAM 中声明变量。

单片机内部扩展 RAM 是否可以访问, 受辅助寄存器 AUXR 中的 EXTRAM 位控制。

## 8.2.4 辅助寄存器 (AUXR)

| 符号   | 地址  | B7    | B6    | B5        | B4  | B3     | B2    | B1            | B0    |
|------|-----|-------|-------|-----------|-----|--------|-------|---------------|-------|
| AUXR | 8EH | T0x12 | T1x12 | UART_M0x6 | T2R | T2_C/T | T2x12 | <b>EXTRAM</b> | S1ST2 |

EXTRAM: 扩展 RAM 访问控制

0: 访问内部扩展 RAM。

1: 内部扩展 RAM 被禁用。

### 8.2.5 外部扩展 RAM, XRAM, XDATA

STC8H 系列封装管脚数为 40 及其以上的单片机具有扩展 64KB 外部数据存储器的能力。访问外部数据存储器期间，WR/RD/ALE 信号要有效。STC8H 系列单片机新增了一个控制外部 64K 字节数据总线速度的特殊功能寄存器 BUS\_SPEED，说明如下：

### 8.2.6 总线速度控制寄存器（BUS\_SPEED）

| 符号        | 地址  | B7        | B6 | B5 | B4 | B3 | B2         | B1 | B0 |
|-----------|-----|-----------|----|----|----|----|------------|----|----|
| BUS_SPEED | A1H | RW_S[1:0] |    |    |    |    | SPEED[2:0] |    |    |

RW\_S[1:0]: RD/WR 控制线选择位

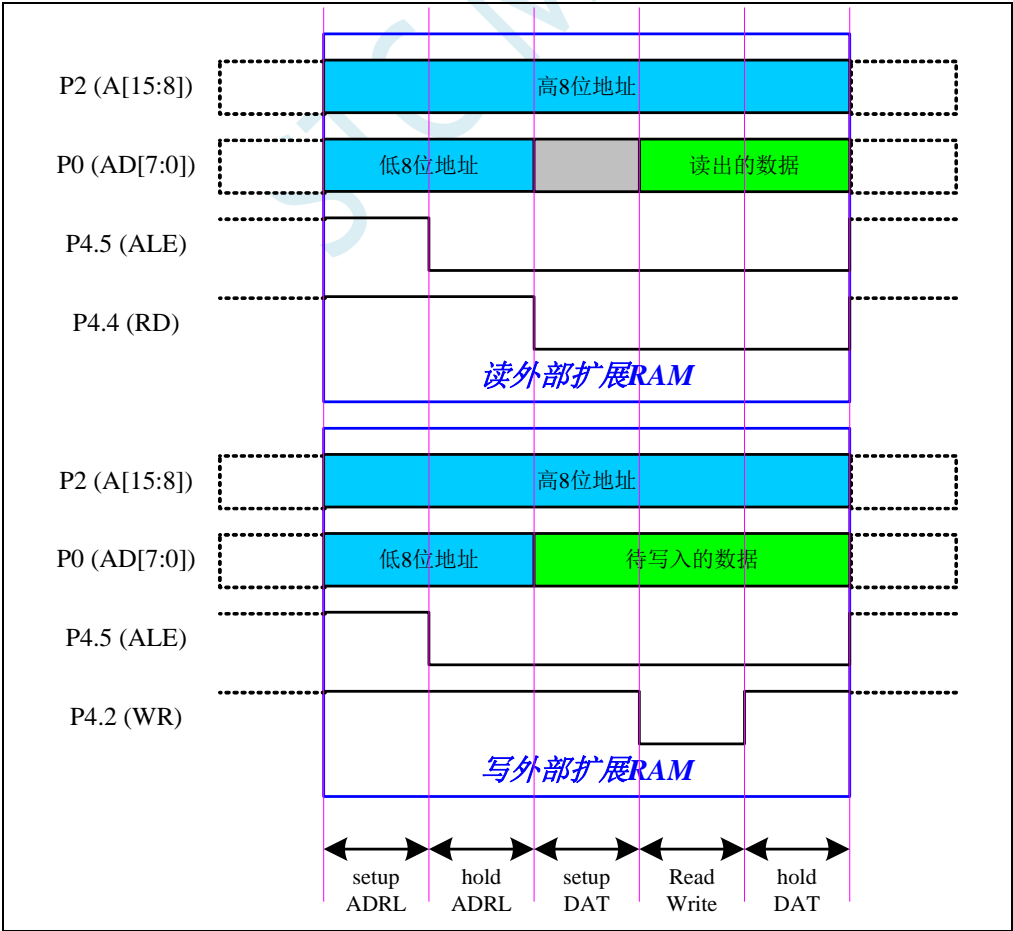
00: P4.4 为 RD, P4.2 为 WR

x1: 保留

SPEED[2:0]: 总线读写速度控制（读写数据时控制信号和数据信号的准备时间和保持时间）

| 指令           | 时钟数        |                |
|--------------|------------|----------------|
|              | 访问内部扩展 RAM | 访问外部扩展 RAM     |
| MOVX A,@Ri   | 3          | 3+5* (SPEED+1) |
| MOVX @Ri,A   | 3          | 3+5* (SPEED+1) |
| MOVX A,@DPTR | 2          | 2+5* (SPEED+1) |
| MOVX @DPTR,A | 2          | 2+5* (SPEED+1) |

读写外部扩展 RAM 时序如下图所示：





## 8.2.7 8051 中可位寻址的数据存储器

8051 单片机内部可位寻址的数据存储器包括两部分：第一部分的地址范围为 00H~7FH，第二部分的地址范围是 80H~FFH。00H~7FH 的位寻址区域是数据区 20H~2FH 这 16 个字节的映射；而 80H~FFH 的位寻址区域则是所有的特殊功能寄存器中地址能被 8 整除的 16 个特殊功能寄存器（包括 80H、88H、90H、98H、A0H、A8H、B0H、B8H、C0H、C8H、D0H、D8H、E0H、E8H、F0H、F8H）的映射。

| 数据存储器地址    | 位寻址地址         |               |               |               |               |               |               |               |
|------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
|            | B7            | B6            | B5            | B4            | B3            | B2            | B1            | B0            |
| F8H (P7)   | FFH<br>F8H. 7 | FEH<br>F8H. 6 | FDH<br>F8H. 5 | FCH<br>F8H. 4 | FBH<br>F8H. 3 | FAH<br>F8H. 2 | F9H<br>F8H. 1 | F8H<br>F8H. 0 |
| F0H (B)    | F7H<br>F0H. 7 | F6H<br>F0H. 6 | F5H<br>F0H. 5 | F4H<br>F0H. 4 | F3H<br>F0H. 3 | F2H<br>F0H. 2 | F1H<br>F0H. 1 | F0H<br>F0H. 0 |
| E8H (P6)   | EFH<br>E8H. 7 | EEH<br>E8H. 6 | EDH<br>E8H. 5 | ECH<br>E8H. 4 | EBH<br>E8H. 3 | EAH<br>E8H. 2 | E9H<br>E8H. 1 | E8H<br>E8H. 0 |
| E0H (ACC)  | E7H<br>E0H. 7 | E6H<br>E0H. 6 | E5H<br>E0H. 5 | E4H<br>E0H. 4 | E3H<br>E0H. 3 | E2H<br>E0H. 2 | E1H<br>E0H. 1 | E0H<br>E0H. 0 |
| D8H (CCON) | DFH<br>D8H. 7 | DEH<br>D8H. 6 | DDH<br>D8H. 5 | DCH<br>D8H. 4 | DBH<br>D8H. 3 | DAH<br>D8H. 2 | D9H<br>D8H. 1 | D8H<br>D8H. 0 |
| D0H (PSW)  | D7H<br>D0H. 7 | D6H<br>D0H. 6 | D5H<br>D0H. 5 | D4H<br>D0H. 4 | D3H<br>D0H. 3 | D2H<br>D0H. 2 | D1H<br>D0H. 1 | D0H<br>D0H. 0 |
| C8H (P5)   | CFH<br>C8H. 7 | CEH<br>C8H. 6 | CDH<br>C8H. 5 | CCH<br>C8H. 4 | CBH<br>C8H. 3 | CAH<br>C8H. 2 | C9H<br>C8H. 1 | C8H<br>C8H. 0 |
| C0H (P4)   | C7H<br>C0H. 7 | C6H<br>C0H. 6 | C5H<br>C0H. 5 | C4H<br>C0H. 4 | C3H<br>C0H. 3 | C2H<br>C0H. 2 | C1H<br>C0H. 1 | C0H<br>C0H. 0 |
| B8H (IP)   | BFH<br>B8H. 7 | BEH<br>B8H. 6 | BDH<br>B8H. 5 | BCH<br>B8H. 4 | BBH<br>B8H. 3 | BAH<br>B8H. 2 | B9H<br>B8H. 1 | B8H<br>B8H. 0 |
| B0H (P3)   | B7H<br>B0H. 7 | B6H<br>B0H. 6 | B5H<br>B0H. 5 | B4H<br>B0H. 4 | B3H<br>B0H. 3 | B2H<br>B0H. 2 | B1H<br>B0H. 1 | B0H<br>B0H. 0 |
| A8H (IE)   | AFH<br>A8H. 7 | AEH<br>A8H. 6 | ADH<br>A8H. 5 | ACH<br>A8H. 4 | ABH<br>A8H. 3 | AAH<br>A8H. 2 | A9H<br>A8H. 1 | A8H<br>A8H. 0 |
| A0H (P2)   | A7H<br>A0H. 7 | A6H<br>A0H. 6 | A5H<br>A0H. 5 | A4H<br>A0H. 4 | A3H<br>A0H. 3 | A2H<br>A0H. 2 | A1H<br>A0H. 1 | A0H<br>A0H. 0 |
| 98H (SCON) | 9FH<br>98H. 7 | 9EH<br>98H. 6 | 9DH<br>98H. 5 | 9CH<br>98H. 4 | 9BH<br>98H. 3 | 9AH<br>98H. 2 | 99H<br>98H. 1 | 98H<br>98H. 0 |
| 90H (P1)   | 97H<br>90H. 7 | 96H<br>90H. 6 | 95H<br>90H. 5 | 94H<br>90H. 4 | 93H<br>90H. 3 | 92H<br>90H. 2 | 91H<br>90H. 1 | 90H<br>90H. 0 |
| 88H (TCON) | 8FH<br>88H. 7 | 8EH<br>88H. 6 | 8DH<br>88H. 5 | 8CH<br>88H. 4 | 8BH<br>88H. 3 | 8AH<br>88H. 2 | 89H<br>88H. 1 | 88H<br>88H. 0 |
| 80H (P0)   | 87H<br>80H. 7 | 86H<br>80H. 6 | 85H<br>80H. 5 | 84H<br>80H. 4 | 83H<br>80H. 3 | 82H<br>80H. 2 | 81H<br>80H. 1 | 80H<br>80H. 0 |
| 2FH        | 7FH<br>2FH. 7 | 7EH<br>2FH. 6 | 7DH<br>2FH. 5 | 7CH<br>2FH. 4 | 7BH<br>2FH. 3 | 7AH<br>2FH. 2 | 79H<br>2FH. 1 | 78H<br>2FH. 0 |
| 2EH        | 77H<br>2EH. 7 | 76H<br>2EH. 6 | 75H<br>2EH. 5 | 74H<br>2EH. 4 | 73H<br>2EH. 3 | 72H<br>2EH. 2 | 71H<br>2EH. 1 | 70H<br>2EH. 0 |
| 2DH        | 6FH<br>2DH. 7 | 6EH<br>2DH. 6 | 6DH<br>2DH. 5 | 6CH<br>2DH. 4 | 6BH<br>2DH. 3 | 6AH<br>2DH. 2 | 69H<br>2DH. 1 | 68H<br>2DH. 0 |
| 2CH        | 67H<br>2CH. 7 | 66H<br>2CH. 6 | 65H<br>2CH. 5 | 64H<br>2CH. 4 | 63H<br>2CH. 3 | 62H<br>2CH. 2 | 61H<br>2CH. 1 | 60H<br>2CH. 0 |
| 2BH        | 5FH<br>2BH. 7 | 5EH<br>2BH. 6 | 5DH<br>2BH. 5 | 5CH<br>2BH. 4 | 5BH<br>2BH. 3 | 5AH<br>2BH. 2 | 59H<br>2BH. 1 | 58H<br>2BH. 0 |
| 2AH        | 57H<br>2AH. 7 | 56H<br>2AH. 6 | 55H<br>2AH. 5 | 54H<br>2AH. 4 | 53H<br>2AH. 3 | 52H<br>2AH. 2 | 51H<br>2AH. 1 | 50H<br>2AH. 0 |

|     |               |               |               |               |               |               |               |               |
|-----|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| 29H | 4FH<br>29H. 7 | 4EH<br>29H. 6 | 4DH<br>29H. 5 | 4CH<br>29H. 4 | 4BH<br>29H. 3 | 4AH<br>29H. 2 | 49H<br>29H. 1 | 48H<br>29H. 0 |
| 28H | 47H<br>28H. 7 | 46H<br>28H. 6 | 45H<br>28H. 5 | 44H<br>28H. 4 | 43H<br>28H. 3 | 42H<br>28H. 2 | 41H<br>28H. 1 | 40H<br>28H. 0 |
| 27H | 3FH<br>27H. 7 | 3EH<br>27H. 6 | 3DH<br>27H. 5 | 3CH<br>27H. 4 | 3BH<br>27H. 3 | 3AH<br>27H. 2 | 39H<br>27H. 1 | 38H<br>27H. 0 |
| 26H | 37H<br>26H. 7 | 36H<br>26H. 6 | 35H<br>26H. 5 | 34H<br>26H. 4 | 33H<br>26H. 3 | 32H<br>26H. 2 | 31H<br>26H. 1 | 30H<br>26H. 0 |
| 25H | 2FH<br>25H. 7 | 2EH<br>25H. 6 | 2DH<br>25H. 5 | 2CH<br>25H. 4 | 2BH<br>25H. 3 | 2AH<br>25H. 2 | 29H<br>25H. 1 | 28H<br>25H. 0 |
| 24H | 27H<br>24H. 7 | 26H<br>24H. 6 | 25H<br>24H. 5 | 24H<br>24H. 4 | 23H<br>24H. 3 | 22H<br>24H. 2 | 21H<br>24H. 1 | 20H<br>24H. 0 |
| 23H | 1FH<br>23H. 7 | 1EH<br>23H. 6 | 1DH<br>23H. 5 | 1CH<br>23H. 4 | 1BH<br>23H. 3 | 1AH<br>23H. 2 | 19H<br>23H. 1 | 18H<br>23H. 0 |
| 22H | 17H<br>22H. 7 | 16H<br>22H. 6 | 15H<br>22H. 5 | 14H<br>22H. 4 | 13H<br>22H. 3 | 12H<br>22H. 2 | 11H<br>22H. 1 | 10H<br>22H. 0 |
| 21H | 0FH<br>21H. 7 | 0EH<br>21H. 6 | 0DH<br>21H. 5 | 0CH<br>21H. 4 | 0BH<br>21H. 3 | 0AH<br>21H. 2 | 09H<br>21H. 1 | 08H<br>21H. 0 |
| 20H | 07H<br>20H. 7 | 06H<br>20H. 6 | 05H<br>20H. 5 | 04H<br>20H. 4 | 03H<br>20H. 3 | 02H<br>20H. 2 | 01H<br>20H. 1 | 00H<br>20H. 0 |

### 8.2.8 扩展 SFR 使能寄存器 EAXFR 的使用说明

STC8G/8H 的扩展 SFR 地址范围为 0FA00H~0FFFFH, 如需访问 XFR 区域的扩展 SFR, 需要先将 EAXFR (P\_SW2.7) 置 1, 并使用 MOVX A,@DPTR 和 MOVX @DPTR,A 这两条指令进行读写操作。XFR 的地址范围与外部扩展 RAM 地址的 0FA00H~0FFFFH 区域是重叠的。

1、若用户不使用外部扩展 RAM 或者外部扩展 RAM 的最大地址不超过 0FA00H (例如只外扩 32K RAM), 这种情况下不会有不同区域的访问地址冲突, 可以在**上电系统初始化时将 EAXFR 寄存器设置为 1 (例如: P\_SW2 |= 0x80;)**, 后续一直保持为 1 不用再修改, 即可正常访问 XFR 区域。

2、若用户有外扩 64K 的扩展 RAM, 则在访问 XFR 和外部扩展 RAM 时需要注意:

- 访问 XFR 时需要将 EAXFR 寄存器位设置为 1;
- 访问地址范围在 0FA00H~0FFFFH 的外部扩展 RAM 时需要将 EAXFR 设置为 0;
- 访问地址范围在 0000H~0F9FFH 的外部扩展 RAM 时, 与 EAXFR 设置的值无关

| 符号    | 地址  | B7    | B6 | B5         | B4 | B3     | B2   | B1   | B0   |
|-------|-----|-------|----|------------|----|--------|------|------|------|
| P_SW2 | BAH | EAXFR | -  | I2C_S[1:0] |    | CMPO_S | S4_S | S3_S | S2_S |

EAXFR: 扩展 RAM 区特殊功能寄存器 (XFR) 访问控制寄存器

- 0: 禁止访问 XFR
- 1: 使能访问 XFR。

**当需要访问 XFR 时, 必须先将 EAXFR 置 1, 才能对 XFR 进行正常的读写。**

## 8.3 存储器中的特殊参数，在 ISP 下载时可烧录进程序 FLASH

STC8H 系列单片机内部的数据存储器和程序存储器中保存有与芯片相关的一些特殊参数，包括：全球唯一 ID 号、32K 掉电唤醒定时器的频率、内部 1.19V 参考信号源值以及 IRC 参数。

**注意：对于固件版本为 7.4.4 以及后续版本的芯片，所有的特殊参数均只能从只读特殊功能寄存器中存储（CHIPID）中读取**

这些参数在程序存储器（ROM）中的存放地址分别如下：

| 参数名称                    | 保存地址                    |                         |                         | 参数说明                     |
|-------------------------|-------------------------|-------------------------|-------------------------|--------------------------|
|                         | STC8H1K08<br>STC8H1K08T | STC8H1K12<br>STC8H1K12T | STC8H1K17<br>STC8H1K17T |                          |
| 全球唯一ID号                 | 1FF9H~1FFFH             | 2FF9H~2FFFH             | 43F9H~43FFH             | 7字节                      |
| 内部1.19V参考信号源（BGV）       | 1FF7H~1FF8H             | 2FF7H~2FF8H             | 43F7H~43F8H             | 毫伏（高字节在前）                |
| 32K掉电唤醒定时器的频率           | 1FF5H~1FF6H             | 2FF5H~2FF6H             | 43F5H~43F6H             | Hz（高字节在前）                |
| 22.1184MHz的IRC参数（20M频段） | 1FF4H                   | 2FF4H                   | 43F4H                   | —                        |
| 24MHz的IRC参数（20M频段）      | 1FF3H                   | 2FF3H                   | 43F3H                   | —                        |
| 20MHz的IRC参数（20M频段）      | 1FF2H                   | 2FF2H                   | 43F2H                   | 固件版本为7.3.12U<br>以及后续版本有效 |
| 27MHz的IRC参数（35M频段）      | 1FF1H                   | 2FF1H                   | 43F1H                   |                          |
| 30MHz的IRC参数（35M频段）      | 1FF0H                   | 2FF0H                   | 43F0H                   |                          |
| 33.1776MHz的IRC参数（35M频段） | 1FEFH                   | 2FEFH                   | 43EFH                   |                          |
| 35MHz的IRC参数（35M频段）      | 1FEEH                   | 2FEEH                   | 43EEH                   |                          |
| 36.864MHz的IRC参数（35M频段）  | 1FEDH                   | 2FEDH                   | 43EDH                   |                          |
| 保留                      | 1FECH                   | 2FECH                   | 43ECH                   |                          |
| 保留                      | 1FEBH                   | 2FEBH                   | 43EBH                   |                          |
| 20M频段的VTRIM参数           | 1FEAH                   | 2FEAH                   | 43EAH                   |                          |
| 35M频段的VTRIM参数           | 1FE9H                   | 2FE9H                   | 43E9H                   |                          |

| 参数名称                    | 保存地址        |             |             |              | 参数说明                     |
|-------------------------|-------------|-------------|-------------|--------------|--------------------------|
|                         | STC8H1K16   | STC8H1K24   | STC8H1K28   | STC8H1K33    |                          |
| 全球唯一ID号                 | 3FF9H~3FFFH | 5FF9H~5FFFH | 6FF9H~6FFFH | 83F9H~83FFFH | 7字节                      |
| 内部1.19V参考信号源（BGV）       | 3FF7H~3FF8H | 5FF7H~5FF8H | 6FF7H~6FF8H | 83F7H~83F8H  | 毫伏（高字节在前）                |
| 32K掉电唤醒定时器的频率           | 3FF5H~3FF6H | 5FF5H~5FF6H | 6FF5H~6FF6H | 83F5H~83F6H  | Hz（高字节在前）                |
| 22.1184MHz的IRC参数（20M频段） | 3FF4H       | 5FF4H       | 6FF4H       | 83F4H        | —                        |
| 24MHz的IRC参数（20M频段）      | 3FF3H       | 5FF3H       | 6FF3H       | 83F3H        | —                        |
| 20MHz的IRC参数（20M频段）      | 3FF2H       | 5FF2H       | 6FF2H       | 83F2H        | 固件版本为7.3.12U<br>以及后续版本有效 |
| 27MHz的IRC参数（35M频段）      | 3FF1H       | 5FF1H       | 6FF1H       | 83F1H        |                          |
| 30MHz的IRC参数（35M频段）      | 3FF0H       | 5FF0H       | 6FF0H       | 83F0H        |                          |
| 33.1776MHz的IRC参数（35M频段） | 3FEFH       | 5FEFH       | 6FEFH       | 83EFH        |                          |
| 35MHz的IRC参数（35M频段）      | 3FEEH       | 5FEEH       | 6FEEH       | 83EEH        |                          |

|                        |       |       |       |       |  |
|------------------------|-------|-------|-------|-------|--|
| 36.864MHz的IRC参数（35M频段） | 3FEDH | 5FEDH | 6FEDH | 83EDH |  |
| 保留                     | 3FECH | 5FECH | 6FECH | 83ECH |  |
| 保留                     | 3FEBH | 5FEBH | 6FEBH | 83EBH |  |
| 20M频段的VRTRIM参数         | 3FEAH | 5FEAH | 6FEAH | 83EAH |  |
| 35M频段的VRTRIM参数         | 3FE9H | 5FE9H | 6FE9H | 83E9H |  |

| 参数名称                    | 保存地址   |  |  |  | 参数说明                     |
|-------------------------|--|--|--|--|--------------------------|
|                         | STC8H3K32S4<br>STC8H3K32S2<br>STC8H4K32TL<br>STC8H4K32TLCD | STC8H3K48S4<br>STC8H3K48S2<br>STC8H4K48TL<br>STC8H4K48TLCD | STC8H3K60S4<br>STC8H3K60S2<br>STC8H4K60TL<br>STC8H4K60TLCD | STC8H3K64S4<br>STC8H3K64S2<br>STC8H4K64TL<br>STC8H4K64TLCD |                          |
| 全球唯一ID号                 | 7FF9H~7FFFH  | BFF9H~BFFFH  | EFF9H~EFFFH  | FDF9H~FDFFH  | 7字节                      |
| 内部1.19V参考信号源（BGV）       | 7FF7H~7FF8H  | BFF7H~BFF8H  | EFF7H~EFF8H  | FDF7H~FDF8H  | 毫伏（高字节在前）                |
| 32K掉电唤醒定时器的频率           | 7FF5H~7FF6H  | BFF5H~BFF6H  | EFF5H~EFF6H  | FDF5H~FDF6H  | Hz（高字节在前）                |
| 22.1184MHz的IRC参数（27M频段） | 7FF4H  | BFF4H  | EFF4H  | FDF4H  | —                        |
| 24MHz的IRC参数（27M频段）      | 7FF3H  | BFF3H  | EFF3H  | FDF3H  | —                        |
| 20MHz的IRC参数（27M频段）      | 7FF2H  | BFF2H  | EFF2H  | FDF2H  | 固件版本为7.3.12U<br>以及后续版本有效 |
| 27MHz的IRC参数（27M频段）      | 7FF1H  | BFF1H  | EFF1H  | FDF1H  |                          |
| 30MHz的IRC参数（27M频段）      | 7FF0H  | BFF0H  | EFF0H  | FDF0H  |                          |
| 33.1776MHz的IRC参数（27M频段） | 7FEFH  | BFEFH  | EFEFH  | FDEFH  |                          |
| 35MHz的IRC参数（44M频段）      | 7FEEH  | BFEEH  | EFEEH  | FDEEH  |                          |
| 36.864MHz的IRC参数（44M频段）  | 7FEDH  | BFEDH  | EFEDH  | FDEDH  |                          |
| 40MHz的IRC参数（44M频段）      | 7FECH  | BFECH  | EFECH  | FDECH  |                          |
| 45MHz的IRC参数（44M频段）      | 7FEBH  | BFEBH  | EFEBH  | FDEBH  |                          |
| 6M频段的VRTRIM参数           | 7FEAH  | BFEAH  | EFEAH  | FDEAH  |                          |
| 10M频段的VRTRIM参数          | 7FE9H  | BFE9H  | EFE9H  | FDE9H  |                          |
| 27M频段的VRTRIM参数          | 7FE8H  | BFE8H  | EFE8H  | FDE8H  |                          |
| 44M频段的VRTRIM参数          | 7FE7H  | BFE7H  | EFE7H  | FDE7H  |                          |

| 参数名称                    | 保存地址        |             |             |             | 参数说明         |
|-------------------------|-------------|-------------|-------------|-------------|--------------|
|                         | STC8H8K32U  | STC8H8K48U  | STC8H8K60U  | STC8H8K64U  |              |
| 全球唯一ID号                 | 7FF9H~7FFFH | BFF9H~BFFFH | EFF9H~EFFFH | FDF9H~FDFFH | 7字节          |
| 内部1.19V参考信号源（BGV）       | 7FF7H~7FF8H | BFF7H~BFF8H | EFF7H~EFF8H | FDF7H~FDF8H | 毫伏（高字节在前）    |
| 32K掉电唤醒定时器的频率           | 7FF5H~7FF6H | BFF5H~BFF6H | EFF5H~EFF6H | FDF5H~FDF6H | Hz（高字节在前）    |
| 22.1184MHz的IRC参数（27M频段） | 7FF4H       | BFF4H       | EFF4H       | FDF4H       | —            |
| 24MHz的IRC参数（27M频段）      | 7FF3H       | BFF3H       | EFF3H       | FDF3H       | —            |
| 27MHz的IRC参数（27M频段）      | 7FF2H       | BFF2H       | EFF2H       | FDF2H       | 固件版本为7.3.12U |

|                          |       |        |        |       |          |
|--------------------------|-------|--------|--------|-------|----------|
| 30MHz的IRC参数 (27M频段)      | 7FF1H | BFF1H  | EFF1H  | FDF1H | 以及后续版本有效 |
| 33.1776MHz的IRC参数 (27M频段) | 7FF0H | BFF0H  | EFF0H  | FDF0H |          |
| 35MHz的IRC参数 (44M频段)      | 7FEFH | BFEFH  | EFEFH  | FDEFH |          |
| 36.864MHz的IRC参数 (44M频段)  | 7FEEH | BFEEH  | EFEEH  | FDEEH |          |
| 40MHz的IRC参数 (44M频段)      | 7FEDH | BFEDH  | EFEDH  | FDEDH |          |
| 44.2368MHz的IRC参数 (44M频段) | 7FECH | BFECH  | EFECH  | FDECH |          |
| 48MHz的IRC参数 (44M频段)      | 7FEBH | BFEBH  | EFEBH  | FDEBH |          |
| 6M频段的VRTRIM参数            | 7FEAH | BFEAH  | EFEAH  | FDEAH |          |
| 10M频段的VRTRIM参数           | 7FE9H | BFEB9H | EFEB9H | FDE9H |          |
| 27M频段的VRTRIM参数           | 7FE8H | BFEB8H | EFEB8H | FDE8H |          |
| 44M频段的VRTRIM参数           | 7FE7H | BFEB7H | EFEB7H | FDE7H |          |

注意: STC8H8K64U 系列扩展的 IRC 参数列表于其他系列并不兼容

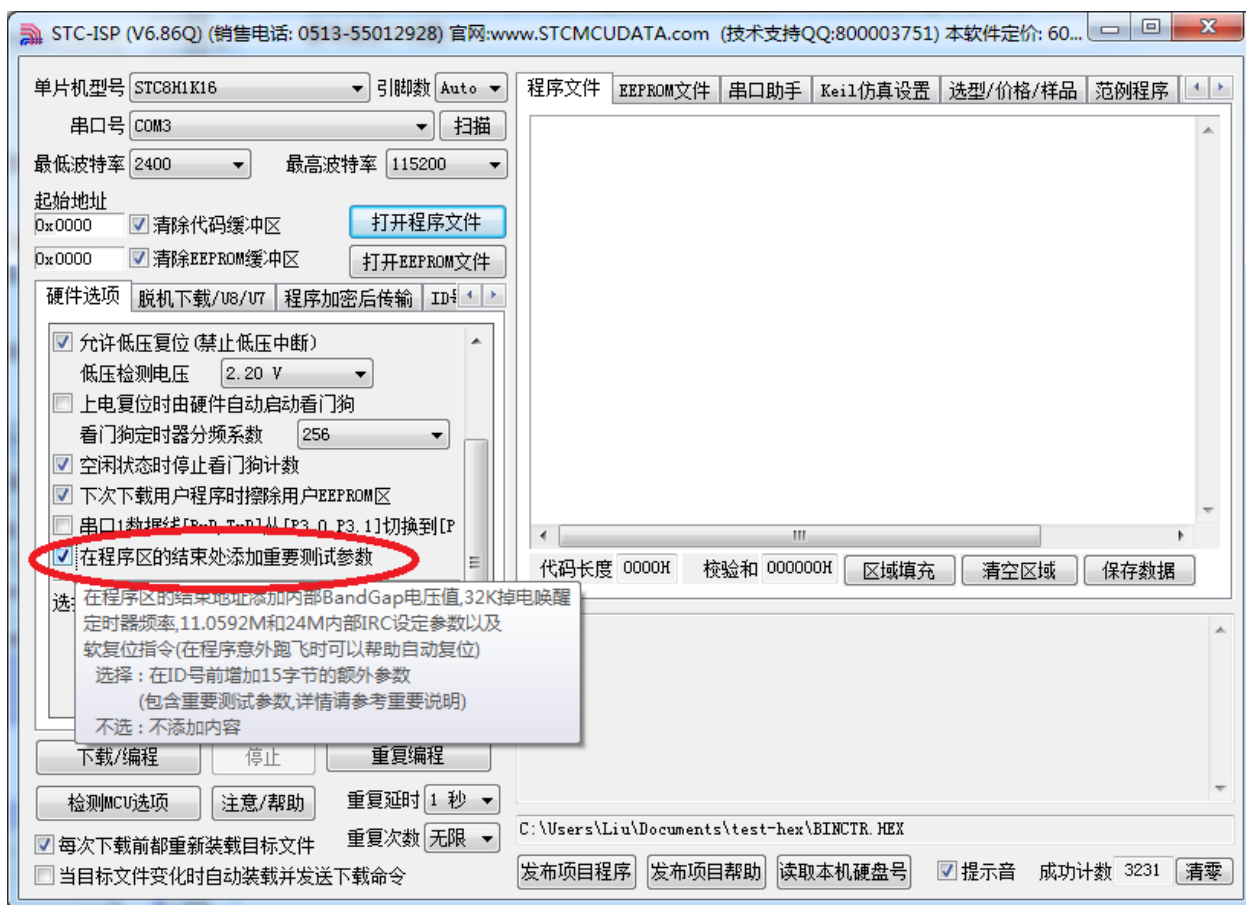
**注意：对于固件版本为 7.4.4 以及后续版本的芯片，所有的特殊参数均只能从只读特殊功能寄存器中存储（CHIPID）中读取**

这些参数在数据存储器（RAM）中的存放地址分别如下：

| 参数名称              | 保存地址             | 参数说明      |
|-------------------|------------------|-----------|
| 内部1.19V参考信号源（BGV） | idata: 0EFH~0F0H | 毫伏（高字节在前） |
| 全球唯一ID号           | idata: 0F1H~0F7H | 7字节       |
| 32K掉电唤醒定时器的频率     | idata: 0F8H~0F9H | Hz（高字节在前） |
| 22.1184MHz的IRC参数  | idata: 0FAH      | —         |
| 24MHz的IRC参数       | idata: 0FBH      | —         |

## 特别说明

- 1、由于 RAM 中的参数可能被修改，所以一般不建议用户使用，特别是用户使用 ID 号进行加密时，强烈建议用于读取 ROM 中的 ID 数据。
- 2、由于 STC8H1K28、STC8H1K12、STC8H1K12T、STC8H3K64S4、STC8H3K64S2、STC8H8K64U、STC8H4K64TL、STC8H4K64TLCD 这几个型号的 EEPROM 的大小用户是可以自己设置的，有可能将保存重要参数的 ROM 空间设置为 EEPROM 而人为的将重要参数擦除或修改，所以使用这个型号进行 ID 号进行加密时可能需要考虑这个问题。
- 3、默认情况下，程序存储器中只有全球唯一 ID 号的数据，而内部 1.19V 参考信号源值、32K 掉电唤醒定时器的频率以及 IRC 参数都是没有的，需要在 ISP 下载时选择如下图所示的选项才可用。





## 8.4 只读特殊功能寄存器中存储的唯一 ID 号和重要参数 (CHIPID)

**注意:** 对于固件版本为 7.4.4 以及后续版本的芯片, 所有的特殊参数均只能从只读特殊功能寄存器中存储 (CHIPID) 中读取

| 产品线                    | CHIPID |
|------------------------|--------|
| STC8H1K08 系列           |        |
| STC8H1K28 系列           |        |
| STC8H3K64S4 系列         |        |
| STC8H3K64S2 系列         |        |
| STC8H8K64U 系列 A 版本     |        |
| STC8H8K64U 系列 B/C/D 版本 | ●      |
| STC8H4K64TL 系列         |        |
| STC8H4K64TLCD 系列       | ●      |
| STC8H1K08T 系列          | ●      |

STC8H 系列部分单片机内置有 32 字节的只读特殊功能寄存器 CHIPID。CHIPID 中内容, 用户程序只能读取, 不可修改。使用 CHIPID 中的数据对用户程序进行加密是 STC 官方推荐的最优方案。

### 相关寄存器

| 符号       | 描述        | 地址    | 位地址与符号                      |    |    |    |                             |    |    |    | 复位值       |
|----------|-----------|-------|-----------------------------|----|----|----|-----------------------------|----|----|----|-----------|
|          |           |       | B7                          | B6 | B5 | B4 | B3                          | B2 | B1 | B0 |           |
| CHIPID0  | 硬件数字 ID00 | FDE0H | 全球唯一 ID 号（第 0 字节）           |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID1  | 硬件数字 ID01 | FDE1H | 全球唯一 ID 号（第 1 字节）           |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID2  | 硬件数字 ID02 | FDE2H | 全球唯一 ID 号（第 2 字节）           |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID3  | 硬件数字 ID03 | FDE3H | 全球唯一 ID 号（第 3 字节）           |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID4  | 硬件数字 ID04 | FDE4H | 全球唯一 ID 号（第 4 字节）           |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID5  | 硬件数字 ID05 | FDE5H | 全球唯一 ID 号（第 5 字节）           |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID6  | 硬件数字 ID06 | FDE6H | 全球唯一 ID 号（第 6 字节）           |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID7  | 硬件数字 ID07 | FDE7H | 内部 1.19V 参考信号源—BGV（高字节）     |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID8  | 硬件数字 ID08 | FDE8H | 内部 1.19V 参考信号源—BGV（低字节）     |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID9  | 硬件数字 ID09 | FDE9H | 32K 掉电唤醒定时器的频率（高字节）         |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID10 | 硬件数字 ID10 | FDEAH | 32K 掉电唤醒定时器的频率（低字节）         |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID11 | 硬件数字 ID11 | FDEBH | USB 系列（STC8H8K64U 系列）       |    |    |    | 其它系列                        |    |    |    | nnnn,nnnn |
|          |           |       | 22.1184MHz 的 IRC 参数（27M 频段） |    |    |    | 22.1184MHz 的 IRC 参数（27M 频段） |    |    |    |           |
| CHIPID12 | 硬件数字 ID12 | FDECH | 24MHz 的 IRC 参数（27M 频段）      |    |    |    | 24MHz 的 IRC 参数（27M 频段）      |    |    |    | nnnn,nnnn |
| CHIPID13 | 硬件数字 ID13 | FDEDH | 27MHz 的 IRC 参数（27M 频段）      |    |    |    | 20MHz 的 IRC 参数（27M 频段）      |    |    |    | nnnn,nnnn |
| CHIPID14 | 硬件数字 ID14 | FDEEH | 30MHz 的 IRC 参数（27M 频段）      |    |    |    | 27MHz 的 IRC 参数（27M 频段）      |    |    |    | nnnn,nnnn |
| CHIPID15 | 硬件数字 ID15 | FDEFH | 33.1776MHz 的 IRC 参数（27M 频段） |    |    |    | 30MHz 的 IRC 参数（27M 频段）      |    |    |    | nnnn,nnnn |
| CHIPID16 | 硬件数字 ID16 | FDF0H | 35MHz 的 IRC 参数（44M 频段）      |    |    |    | 33.1776MHz 的 IRC 参数（27M 频段） |    |    |    | nnnn,nnnn |

|          |           |       |                              |                             |           |
|----------|-----------|-------|------------------------------|-----------------------------|-----------|
| CHIPID17 | 硬件数字 ID17 | FDF1H | 36.864MHz 的 IRC 参数 (44M 频段)  | 35MHz 的 IRC 参数 (44M 频段)     | nnnn,nnnn |
| CHIPID18 | 硬件数字 ID18 | FDF2H | 40MHz 的 IRC 参数 (44M 频段)      | 36.864MHz 的 IRC 参数 (44M 频段) | nnnn,nnnn |
| CHIPID19 | 硬件数字 ID19 | FDF3H | 44.2368MHz 的 IRC 参数 (44M 频段) | 40MHz 的 IRC 参数 (44M 频段)     | nnnn,nnnn |
| CHIPID20 | 硬件数字 ID20 | FDF4H | 48MHz 的 IRC 参数 (44M 频段)      | 45MHz 的 IRC 参数 (44M 频段)     | nnnn,nnnn |
| CHIPID21 | 硬件数字 ID21 | FDF5H | 6M 频段的 VRTRIM 参数             |                             | nnnn,nnnn |
| CHIPID22 | 硬件数字 ID22 | FDF6H | 10M 频段的 VRTRIM 参数            |                             | nnnn,nnnn |
| CHIPID23 | 硬件数字 ID23 | FDF7H | 27M 频段的 VRTRIM 参数            |                             | nnnn,nnnn |
| CHIPID24 | 硬件数字 ID24 | FDF8H | 44M 频段的 VRTRIM 参数            |                             | nnnn,nnnn |
| CHIPID25 | 硬件数字 ID25 | FDF9H | 00H                          |                             | nnnn,nnnn |
| CHIPID26 | 硬件数字 ID26 | FDFAH | 用户程序空间结束地址 (高字节)             |                             | nnnn,nnnn |
| CHIPID27 | 硬件数字 ID27 | FDFBH | 芯片测试时间 (年)                   |                             | nnnn,nnnn |
| CHIPID28 | 硬件数字 ID28 | FDFCH | 芯片测试时间 (月)                   |                             | nnnn,nnnn |
| CHIPID29 | 硬件数字 ID29 | FDFDH | 芯片测试时间 (日)                   |                             | nnnn,nnnn |
| CHIPID30 | 硬件数字 ID30 | FDFEH | 芯片封装形式编号                     |                             | nnnn,nnnn |
| CHIPID31 | 硬件数字 ID31 | FDFFH | 5AH                          |                             | nnnn,nnnn |

## 8.4.1 CHIP 之全球唯一 ID 号解读

| 符号      | 描述        | 地址    | 位地址与符号             |    |    |    |    |    |    |    | 复位值       |
|---------|-----------|-------|--------------------|----|----|----|----|----|----|----|-----------|
|         |           |       | B7                 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |           |
| CHIPID0 | 硬件数字 ID00 | FDE0H | 全球唯一 ID 号 (第 0 字节) |    |    |    |    |    |    |    | nnnn,nnnn |
| CHIPID1 | 硬件数字 ID01 | FDE1H | 全球唯一 ID 号 (第 1 字节) |    |    |    |    |    |    |    | nnnn,nnnn |
| CHIPID2 | 硬件数字 ID02 | FDE2H | 全球唯一 ID 号 (第 2 字节) |    |    |    |    |    |    |    | nnnn,nnnn |
| CHIPID3 | 硬件数字 ID03 | FDE3H | 全球唯一 ID 号 (第 3 字节) |    |    |    |    |    |    |    | nnnn,nnnn |
| CHIPID4 | 硬件数字 ID04 | FDE4H | 全球唯一 ID 号 (第 4 字节) |    |    |    |    |    |    |    | nnnn,nnnn |
| CHIPID5 | 硬件数字 ID05 | FDE5H | 全球唯一 ID 号 (第 5 字节) |    |    |    |    |    |    |    | nnnn,nnnn |
| CHIPID6 | 硬件数字 ID06 | FDE6H | 全球唯一 ID 号 (第 6 字节) |    |    |    |    |    |    |    | nnnn,nnnn |

[CHIPID0, CHIPID1]: 16 位 MCU ID, 用于区别不同的单片机型号 (高位在前)。

STC8 系列常用的 MCU ID 如下表所示:

|                       |                     |                     |                    |                        |                       |
|-----------------------|---------------------|---------------------|--------------------|------------------------|-----------------------|
| STC8H4K16TLCDC (F811) | STC8H4K16LCD (F821) | STC8H4K16TLR (F7C9) | STC8H3K16S4 (F741) | STC8H3K16S2 (F749)     | STC8H8K16U (F781)     |
| STC8H4K32TLCDC (F812) | STC8H4K32LCD (F822) | STC8H4K32TLR (F7CA) | STC8H3K32S4 (F742) | STC8H3K32S2 (F74A)     | STC8H8K32U (F782)     |
| STC8H4K48TLCDC (F815) | STC8H4K48LCD (F825) | STC8H4K48TLR (F7CD) | STC8H3K48S4 (F745) | STC8H3K48S2 (F74D)     | STC8H8K48U (F785)     |
| STC8H4K60TLCDC (F813) | STC8H4K60LCD (F823) | STC8H4K60TLR (F7CB) | STC8H3K60S4 (F743) | STC8H3K60S2 (F74B)     | STC8H8K60U (F783)     |
| STC8H4K64TLCDC (F814) | STC8H4K64LCD (F824) | STC8H4K64TLR (F7CC) | STC8H3K64S4 (F744) | STC8H3K64S2 (F74C)     | STC8H8K64U (F784)     |
| STC8H1K06 (F733)      | STC8H1K16 (F721)    | STC8G2K16S4 (F761)  | STC8G2K16S2 (F769) | STC8G1K06A-8PIN (F793) | STC8G1K06-8PIN (F7A3) |
| STC8H1K08 (F734)      | STC8H1K20 (F722)    | STC8G2K32S4 (F762)  | STC8G2K32S2 (F76A) | STC8G1K08A-8PIN (F794) | STC8G1K08-8PIN (F7A4) |
| STC8H1K10 (F735)      | STC8H1K24 (F723)    | STC8G2K48S4 (F765)  | STC8G2K48S2 (F76D) | STC8G1K10A-8PIN (F795) | STC8G1K10-8PIN (F7A5) |
| STC8H1K12 (F736)      | STC8H1K28 (F724)    | STC8G2K60S4 (F763)  | STC8G2K60S2 (F76B) | STC8G1K12A-8PIN (F796) | STC8G1K12-8PIN (F7A6) |
| STC8H1K17 (F737)      | STC8H1K33 (F725)    | STC8G2K64S4 (F764)  | STC8G2K64S2 (F76C) | STC8G1K17A-8PIN (F797) | STC8G1K17-8PIN (F7A7) |
| STC8G1K06 (F753)      | STC8C2K16S4 (F7D1)  | STC8C2K16S2 (F7D9)  | STC8A8K16D4 (F7F1) | STC8H1K06T (F843)      |                       |
| STC8G1K08 (F754)      | STC8C2K32S4 (F7D2)  | STC8C2K32S2 (F7DA)  | STC8A8K32D4 (F7F2) | STC8H1K08T (F844)      |                       |
| STC8G1K10 (F755)      | STC8C2K48S4 (F7D5)  | STC8C2K48S2 (F7DD)  | STC8A8K48D4 (F7F5) | STC8H1K10T (F845)      |                       |
| STC8G1K12 (F756)      | STC8C2K60S4 (F7D3)  | STC8C2K60S2 (F7DB)  | STC8A8K60D4 (F7F3) | STC8H1K12T (F846)      |                       |
| STC8G1K17 (F757)      | STC8C2K64S4 (F7D4)  | STC8C2K64S2 (F7DC)  | STC8A8K64D4 (F7F4) | STC8H1K17T (F847)      |                       |

[CHIPID2, CHIPID3]: 16 位测试机台编号 (高位在前)。

[CHIPID4, CHIPID5, CHIPID6]: 24 位测试流水编号 (高位在前)。

## 8.4.2 CHIP 之内部参考信号源解读

| 符号      | 描述        | 地址    | 位地址与符号                   |    |    |    |    |    |    |    | 复位值       |
|---------|-----------|-------|--------------------------|----|----|----|----|----|----|----|-----------|
|         |           |       | B7                       | B6 | B5 | B4 | B3 | B2 | B1 | B0 |           |
| CHIPID7 | 硬件数字 ID07 | FDE7H | 内部 1.19V 参考信号源-BGV (高字节) |    |    |    |    |    |    |    | nnnn,nnnn |
| CHIPID8 | 硬件数字 ID08 | FDE8H | 内部 1.19V 参考信号源-BGV (低字节) |    |    |    |    |    |    |    | nnnn,nnnn |

[CHIPID7, CHIPID8]: 16 位内部参考信号源电压值 (高位在前)。

标准值为 1190 (04A6H), 单位为 mV, 即 1.19V。但实际的芯片由于存在制造误差。内部参考信号源的电压值并不会受工作电压 VCC 的影响, 所以内部参考信号源可以和 ADC 结合用于校准 ADC, 也可和比较器结合用于侦测工作电压。

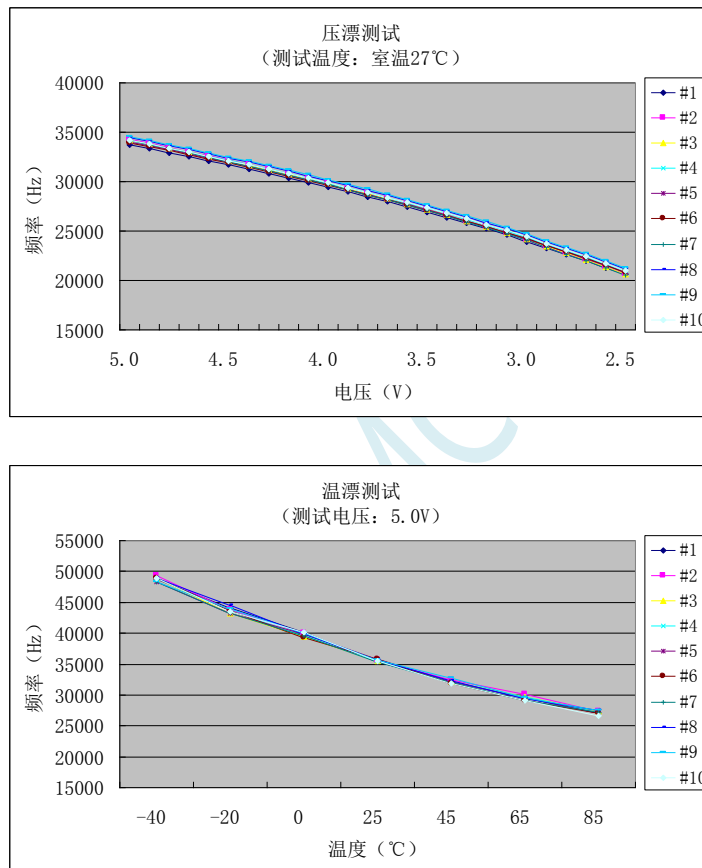
# 8.4.3 CHIP 之内部 32K 的 IRC 振荡频率解读

| 符号       | 描述        | 地址    | 位地址与符号               |    |    |    |    |    |    | 复位值       |
|----------|-----------|-------|----------------------|----|----|----|----|----|----|-----------|
|          |           |       | B7                   | B6 | B5 | B4 | B3 | B2 | B1 | B0        |
| CHIPID9  | 硬件数字 ID09 | FDE9H | 32K 掉电唤醒定时器的频率 (高字节) |    |    |    |    |    |    | nnnn,nnnn |
| CHIPID10 | 硬件数字 ID10 | FDEAH | 32K 掉电唤醒定时器的频率 (低字节) |    |    |    |    |    |    | nnnn,nnnn |

[CHIPID9, CHIPID10]: 16 位 32K IRC 振荡器频率值 (高位在前)。

标准值为 32768 (8000H), 单位为 Hz, 即 32.768KHz。但实际的芯片由于存在制造误差, 而且温漂和压漂均比较大。

内部 32K 振荡器的压漂测试线性图和温漂线性图如下:



# 8.4.4 CHIP 之高精度 IRC 参数解读

| 符号       | 描述        | 地址    | 位地址与符号                      |    |    |    |                             |    |    |    | 复位值       |
|----------|-----------|-------|-----------------------------|----|----|----|-----------------------------|----|----|----|-----------|
|          |           |       | B7                          | B6 | B5 | B4 | B3                          | B2 | B1 | B0 |           |
| CHIPID11 | 硬件数字 ID11 | FDEBH | USB 系列（STC8H8K64U 系列）       |    |    |    | 其它系列                        |    |    |    | nnnn,nnnn |
|          |           |       | 22.1184MHz 的 IRC 参数（27M 频段） |    |    |    | 22.1184MHz 的 IRC 参数（27M 频段） |    |    |    |           |
| CHIPID12 | 硬件数字 ID12 | FDECH | 24MHz 的 IRC 参数（27M 频段）      |    |    |    | 24MHz 的 IRC 参数（27M 频段）      |    |    |    | nnnn,nnnn |
| CHIPID13 | 硬件数字 ID13 | FDEDH | 27MHz 的 IRC 参数（27M 频段）      |    |    |    | 20MHz 的 IRC 参数（27M 频段）      |    |    |    | nnnn,nnnn |
| CHIPID14 | 硬件数字 ID14 | FDEEH | 30MHz 的 IRC 参数（27M 频段）      |    |    |    | 27MHz 的 IRC 参数（27M 频段）      |    |    |    | nnnn,nnnn |
| CHIPID15 | 硬件数字 ID15 | FDEFH | 33.1776MHz 的 IRC 参数（27M 频段） |    |    |    | 30MHz 的 IRC 参数（27M 频段）      |    |    |    | nnnn,nnnn |
| CHIPID16 | 硬件数字 ID16 | FDF0H | 35MHz 的 IRC 参数（44M 频段）      |    |    |    | 33.1776MHz 的 IRC 参数（27M 频段） |    |    |    | nnnn,nnnn |
| CHIPID17 | 硬件数字 ID17 | FDF1H | 36.864MHz 的 IRC 参数（44M 频段）  |    |    |    | 35MHz 的 IRC 参数（44M 频段）      |    |    |    | nnnn,nnnn |
| CHIPID18 | 硬件数字 ID18 | FDF2H | 40MHz 的 IRC 参数（44M 频段）      |    |    |    | 36.864MHz 的 IRC 参数（44M 频段）  |    |    |    | nnnn,nnnn |
| CHIPID19 | 硬件数字 ID19 | FDF3H | 44.2368MHz 的 IRC 参数（44M 频段） |    |    |    | 40MHz 的 IRC 参数（44M 频段）      |    |    |    | nnnn,nnnn |
| CHIPID20 | 硬件数字 ID20 | FDF4H | 48MHz 的 IRC 参数（44M 频段）      |    |    |    | 45MHz 的 IRC 参数（44M 频段）      |    |    |    | nnnn,nnnn |
| CHIPID21 | 硬件数字 ID21 | FDF5H | 6M 频段的 VRTRIM 参数            |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID22 | 硬件数字 ID22 | FDF6H | 10M 频段的 VRTRIM 参数           |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID23 | 硬件数字 ID23 | FDF7H | 27M 频段的 VRTRIM 参数           |    |    |    |                             |    |    |    | nnnn,nnnn |
| CHIPID24 | 硬件数字 ID24 | FDF8H | 44M 频段的 VRTRIM 参数           |    |    |    |                             |    |    |    | nnnn,nnnn |

支持 CHIPID 功能的 STC8H 系列单片机，内部集成的高精度 IRC 分 4 个频段，每个频段对应的参考电压值在出厂时已进行了校准，当选择不同的频段时，只需要将相应频段的电压校准值填入 VRTRIM 寄存器即可。4 个频段的中心频率分别为 6MHz、10MHz、27MHz 和 44MHz，由于制造误差，中心频率一般可能有±5%的偏差，为了得到精确的用户频率，可使用 IRTRIM 对频率进行微调校准。使用 STC 官方提供的下载软件下载用户程序时，系统会根据用户所设定频率自动设置 VRTRIM 和 IRTRIM 寄存器。同时，在 CHIPID 也内部预置了 10 个常用频率的 IRTRIM 值以及 4 个频段的参考电压值校准值，让用户可以在程序运行过程中动态的修改工作频率。

[CHIPID11 : CHIPID20]: 10 个常用频率的 IRTRIM 值。括号里面的注解即为对应的频段

[CHIPID21 : CHIPID24]: 4 个频段的参考电压值校准值。

用户动态修改频率时，只需要将[CHIPID11 : CHIPID20]中的某个频率校准值读出并写入 IRTRIM 寄存器，同时根据该频率所对应的频段将[CHIPID21 : CHIPID24]中的某个电压校准值读出并写入 VRTRIM 寄存器即可。详细操作请参考后续章节的范例程序。

### 8.4.5 CHIP 之测试时间参数解读

| 符号       | 描述        | 地址    | 位地址与符号    |    |    |    |    |    |    |    | 复位值       |
|----------|-----------|-------|-----------|----|----|----|----|----|----|----|-----------|
|          |           |       | B7        | B6 | B5 | B4 | B3 | B2 | B1 | B0 |           |
| CHIPID27 | 硬件数字 ID27 | FDFBH | 芯片测试时间（年） |    |    |    |    |    |    |    | nnnn,nnnn |
| CHIPID28 | 硬件数字 ID28 | FDFCH | 芯片测试时间（月） |    |    |    |    |    |    |    | nnnn,nnnn |
| CHIPID29 | 硬件数字 ID29 | FDFDH | 芯片测试时间（日） |    |    |    |    |    |    |    | nnnn,nnnn |

测试时间的年、月、日参数均为 BCD 码。（例如：CHIPID27=0x21，CHIPID28=0x11，CHIPID29=0x18，则目标芯片的生产测试日期为 2021 年 11 月 18 日）

### 8.4.6 CHIP 之芯片封装形式编号解读

| 符号       | 描述        | 地址    | 位地址与符号   |    |    |    |    |    |    |    | 复位值       |
|----------|-----------|-------|----------|----|----|----|----|----|----|----|-----------|
|          |           |       | B7       | B6 | B5 | B4 | B3 | B2 | B1 | B0 |           |
| CHIPID30 | 硬件数字 ID30 | FDFEH | 芯片封装形式编号 |    |    |    |    |    |    |    | nnnn,nnnn |

| 封装编号 | 封装形式    |  | 封装编号 | 封装形式    |
|------|---------|--|------|---------|
| 0x00 | DIP8    |  | 0x50 | SOP32   |
| 0x01 | SOP8    |  | 0x51 | LQFP32  |
| 0x02 | DFN8    |  | 0x52 | QFN32   |
| 0x10 | DIP16   |  | 0x53 | PLCC32  |
| 0x11 | SOP16   |  | 0x54 | QFN32S  |
| 0x20 | DIP18   |  | 0x60 | PDIP40  |
| 0x21 | SOP18   |  | 0x70 | LQFP44  |
| 0x30 | DIP20   |  | 0x71 | PLCC44  |
| 0x31 | SOP20   |  | 0x72 | PQFP44  |
| 0x32 | TSSOP20 |  | 0x80 | LQFP48  |
| 0x33 | LSSOP20 |  | 0x81 | QFN48   |
| 0x34 | QFN20   |  | 0x90 | LQFP64  |
| 0x40 | SKDIP28 |  | 0x91 | LQFP64S |
| 0x41 | SOP28   |  | 0x92 | LQFP64L |
| 0x42 | TSSOP28 |  | 0x93 | LQFP64M |
| 0x43 | QFN28   |  | 0x94 | QFN64   |

## 8.5 范例程序

### 8.5.1 读取内部 1.19V 参考信号源-BGV 值 (从 CHIPID 中读取)

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT      (65536 - (FOSC / 115200+2) / 4)
```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
#define USBCHIPID
#define CPUIDBASE      0xfde0
#define VREF_ADDR      (*(unsigned int volatile xdata*)(CPUIDBASE + 0x07))
```

```
bit      busy;
```

```
void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}
```

```
void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}
```

```
void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}
```

```
void main()
{
```

```
P_SW2 /= 0x80; //使能访问 XFR

P0M0 = 0x00;
P0M1 = 0x00;
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

UartInit();
ES = 1;
EA = 1;
UartSend(VREF_ADDR >> 8); //读取内部 1.19V 参考信号源的高字节
UartSend(VREF_ADDR);      //读取内部 1.19V 参考信号源的低字节

while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

```
CPUIDBASE EQU 0FDE0H
VREF_ADDR EQU CPUIDBASE + 07H

AUXR DATA 8EH
P_SW2 DATA 0BAH

BUSY BIT 20H.0

P0M1 DATA 093H
P0M0 DATA 094H
P1M1 DATA 091H
P1M0 DATA 092H
P2M1 DATA 095H
P2M0 DATA 096H
P3M1 DATA 0B1H
P3M0 DATA 0B2H
P4M1 DATA 0B3H
P4M0 DATA 0B4H
P5M1 DATA 0C9H
P5M0 DATA 0CAH

ORG 0000H
LJMP MAIN
ORG 0023H
LJMP UART_ISR

ORG 0100H

UART_ISR:
JNB TI,CHKRI
CLR TI
```



```

        CLR        BUSY

CHKRI:
        JNB        RI,UARTISR_EXIT
        CLR        RI
UARTISR_EXIT:
        RETI

UART_INIT:
        MOV        SCON,#50H
        MOV        TMOD,#00H
        MOV        TL1,#0E8H                ;65536-11059200/115200/4=0FFE8H
        MOV        TH1,#0FFH
        SETB       TRI
        MOV        AUXR,#40H
        CLR        BUSY
        RET

UART_SEND:
        JB         BUSY,$
        SETB       BUSY
        MOV        SBUF,A
        RET

MAIN:
        MOV        SP,#5FH
        ORL        P_SW2,#80H                ;使能访问 XFR

        MOV        P0M0,#00H
        MOV        P0M1,#00H
        MOV        P1M0,#00H
        MOV        P1M1,#00H
        MOV        P2M0,#00H
        MOV        P2M1,#00H
        MOV        P3M0,#00H
        MOV        P3M1,#00H
        MOV        P4M0,#00H
        MOV        P4M1,#00H
        MOV        P5M0,#00H
        MOV        P5M1,#00H

        LCALL      UART_INIT
        SETB       ES
        SETB       EA

        MOV        DPTR,#VREF_ADDR
        CLR        A
        MOVX        A,@DPTR                ;读取内部 1.19V 参考信号源的高字节
        LCALL      UART_SEND
        INC        DPTR
        MOVX        A,@DPTR                ;读取内部 1.19V 参考信号源的低字节
        LCALL      UART_SEND

LOOP:
        JMP        LOOP

        END

```

## 8.5.2 读取内部 1.19V 参考信号源-BGV 值 (从 Flash 程序存储器 (ROM) 中读取)

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT      (65536 - (FOSC / 115200+2) / 4)
```

//加2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
bit      busy;
int      *BGV;
```

```
void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}
```

```
void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}
```

```
void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}
```

```
void main()
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
```

```

P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

BGV = (int code *)0x3ff7;           // STC8H1K16
UartInit();
ES = 1;
EA = 1;
UartSend(*BGV >> 8);               // 读取内部 1.19V 参考信号源的高字节
UartSend(*BGV);                     // 读取内部 1.19V 参考信号源的低字节

while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|               |      |                 |            |
|---------------|------|-----------------|------------|
| AUXR          | DATA | 8EH             |            |
| BGV           | EQU  | 03FF7H          | ;STC8H1K16 |
| P_SW2         | DATA | 0BAH            |            |
| BUSY          | BIT  | 20H.0           |            |
| P1M1          | DATA | 091H            |            |
| P1M0          | DATA | 092H            |            |
| P0M1          | DATA | 093H            |            |
| P0M0          | DATA | 094H            |            |
| P2M1          | DATA | 095H            |            |
| P2M0          | DATA | 096H            |            |
| P3M1          | DATA | 0B1H            |            |
| P3M0          | DATA | 0B2H            |            |
| P4M1          | DATA | 0B3H            |            |
| P4M0          | DATA | 0B4H            |            |
| P5M1          | DATA | 0C9H            |            |
| P5M0          | DATA | 0CAH            |            |
|               | ORG  | 0000H           |            |
|               | LJMP | MAIN            |            |
|               | ORG  | 0023H           |            |
|               | LJMP | UART_ISR        |            |
|               | ORG  | 0100H           |            |
| UART_ISR:     |      |                 |            |
|               | JNB  | TI,CHKRI        |            |
|               | CLR  | TI              |            |
|               | CLR  | BUSY            |            |
| CHKRI:        |      |                 |            |
|               | JNB  | RI,UARTISR_EXIT |            |
|               | CLR  | RI              |            |
| UARTISR_EXIT: |      |                 |            |
|               | RETI |                 |            |

**UART\_INIT:**

```

MOV     SCON,#50H
MOV     TMOD,#00H
MOV     TL1,#0E8H           ;65536-11059200/115200/4=0FFE8H
MOV     TH1,#0FFH
SETB    TRI
MOV     AUXR,#40H
CLR     BUSY
RET

```

**UART\_SEND:**

```

JB      BUSY,$
SETB    BUSY
MOV     SBUF,A
RET

```

**MAIN:**

```

MOV     SP,#5FH
ORL     P_SW2,#80H           ;使能访问 XFR

MOV     P0M0,#00H
MOV     P0M1,#00H
MOV     P1M0,#00H
MOV     P1M1,#00H
MOV     P2M0,#00H
MOV     P2M1,#00H
MOV     P3M0,#00H
MOV     P3M1,#00H
MOV     P4M0,#00H
MOV     P4M1,#00H
MOV     P5M0,#00H
MOV     P5M1,#00H

LCALL   UART_INIT
SETB    ES
SETB    EA

MOV     DPTR,#BGV
CLR     A
MOVC    A,@A+DPTR           ;读取内部 1.19V 参考信号源的高字节
LCALL   UART_SEND
MOV     A,#1
MOVC    A,@A+DPTR           ;读取内部 1.19V 参考信号源的低字节
LCALL   UART_SEND

```

**LOOP:**

```

JMP     LOOP

```

```

END

```

### ~~8.5.3 读取内部 1.19V 参考信号源 BGV 值 (从 RAM 中读取)~~

#### C 语言代码

```
//测试工作频率为 11.0592MHz
```

```
#include "stc8h.h"
#include "intrins.h"

#define FOSC      11059200UL
#define BRT      (65536 - (FOSC / 115200+2) / 4)

//加2 操作是为了让 Keil 编译器
//自动实现四舍五入运算

bit      busy;
int      *BGV;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;
}
```

```

    BGV = (int idata *)0xef;
    UartInit();
    ES = 1;
    EA = 1;
    UartSend(*BGV >> 8);           //读取内部 1.19V 参考信号源的高字节
    UartSend(*BGV);               //读取内部 1.19V 参考信号源的低字节

    while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

```

AUXR      DATA      8EH
BGV       DATA      0EFH
P_SW2     DATA      0BAH

BUSY      BIT        20H.0

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG         0000H
          LJMP        MAIN
          ORG         0023H
          LJMP        UART_ISR

          ORG         0100H

UART_ISR:

          JNB         TI,CHKRI
          CLR         TI
          CLR         BUSY

CHKRI:

          JNB         RI,UARTISR_EXIT
          CLR         RI

UARTISR_EXIT:
          RETI

UART_INIT:
          MOV         SCON,#50H
          MOV         TMOD,#00H
          MOV         TL1,#0E8H           ;65536-11059200/115200/4=0FFE8H
          MOV         TH1,#0FFH
          SETB        TRI
          MOV         AUXR,#40H
          CLR         BUSY
```

```

    RET

UART_SEND:
    JB     BUSY,$
    SETB   BUSY
    MOV    SBUF,A
    RET

MAIN:
    MOV    SP,#5FH
    ORL    P_SW2,#80H           ;使能访问 XFR

    MOV    P0M0,#00H
    MOV    P0M1,#00H
    MOV    P1M0,#00H
    MOV    P1M1,#00H
    MOV    P2M0,#00H
    MOV    P2M1,#00H
    MOV    P3M0,#00H
    MOV    P3M1,#00H
    MOV    P4M0,#00H
    MOV    P4M1,#00H
    MOV    P5M0,#00H
    MOV    P5M1,#00H

    LCALL  UART_INIT
    SETB   ES
    SETB   EA

    MOV    R0,#BGV
    MOV    A,@R0                ;读取内部 1.19V 参考信号源的高字节
    LCALL  UART_SEND
    INC    R0
    MOV    A,@R0                ;读取内部 1.19V 参考信号源的低字节
    LCALL  UART_SEND

LOOP:
    JMP    LOOP

END

```

## 8.5.4 读取全球唯一 ID 号 (从 CHIPID 中读取)

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT       (65536 - (FOSC / 115200+2) / 4)
```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
#define CPUIDBASE    0xfde0
```

```
#define ID_ADDR ((unsigned char volatile xdata *)(CPUIDBASE + 0x00))

bit busy;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void main()
{
    char i;

    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;

    for (i=0; i<7; i++)
```



```

    {
        UartSend(ID_ADDR[i]);
    }

    while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

```

CPUIDBASE EQU 0FDE0H
ID_ADDR EQU CPUIDBASE + 00H

```

```

AUXR DATA 8EH
P_SW2 DATA 0BAH

```

```

BUSY BIT 20H.0

```

```

P0M1 DATA 093H

```

```

P0M0 DATA 094H

```

```

P1M1 DATA 091H

```

```

P1M0 DATA 092H

```

```

P2M1 DATA 095H

```

```

P2M0 DATA 096H

```

```

P3M1 DATA 0B1H

```

```

P3M0 DATA 0B2H

```

```

P4M1 DATA 0B3H

```

```

P4M0 DATA 0B4H

```

```

P5M1 DATA 0C9H

```

```

P5M0 DATA 0CAH

```

```

ORG 0000H

```

```

LJMP MAIN

```

```

ORG 0023H

```

```

LJMP UART_ISR

```

```

ORG 0100H

```

UART\_ISR:

```

JNB TI,CHKRI

```

```

CLR TI

```

```

CLR BUSY

```

CHKRI:

```

JNB RI,UARTISR_EXIT

```

```

CLR RI

```

UARTISR\_EXIT:

```

RETI

```

UART\_INIT:

```

MOV SCON,#50H

```

```

MOV TMOD,#00H

```

```

MOV TL1,#0E8H

```

```

MOV TH1,#0FFH

```

```

SETB TRI

```

```

MOV AUXR,#40H

```

```

CLR BUSY

```

```

RET

```

;65536-11059200/115200/4=0FFE8H

**UART\_SEND:**

```
JB     BUSY,$
SETB   BUSY
MOV     SBUF,A
RET
```

**MAIN:**

```
MOV     SP, #5FH
ORL     P_SW2, #80H      ;使能访问 XFR
```

```
MOV     P0M0, #00H
MOV     P0M1, #00H
MOV     P1M0, #00H
MOV     P1M1, #00H
MOV     P2M0, #00H
MOV     P2M1, #00H
MOV     P3M0, #00H
MOV     P3M1, #00H
MOV     P4M0, #00H
MOV     P4M1, #00H
MOV     P5M0, #00H
MOV     P5M1, #00H
```

```
LCALL   UART_INIT
SETB    ES
SETB    EA
```

```
MOV     DPTR, #ID_ADDR
MOV     R1, #7
```

**NEXT:**

```
CLR     A
MOVX    A, @DPTR
LCALL   UART_SEND
INC     DPTR
DJNZ    R1, NEXT
```

**LOOP:**

```
JMP     LOOP
```

```
END
```

## 8.5.5 读取全球唯一 ID 号 (从 Flash 程序存储器 (ROM) 中读取)

C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT       (65536 - (FOSC / 115200 + 2) / 4)
```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
bit     busy;
char     *ID;
```

```
void UartIsr() interrupt 4
```

```
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}
```

```
void UartInit()
```

```
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}
```

```
void UartSend(char dat)
```

```
{
    while (busy);
    busy = 1;
    SBUF = dat;
}
```

```
void main()
```

```
{
    char i;

    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    ID = (char code *)0x3ff9; // STC8H1K16
    UartInit();
    ES = 1;
    EA = 1;

    for (i=0; i<7; i++)
    {
        UartSend(ID[i]);
    }
}
```

```

    }

    while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

```

AUXR      DATA      8EH
ID         EQU        03FF9H          ; STC8H1K16
P_SW2     DATA      0BAH

BUSY      BIT        20H.0

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

        ORG          0000H
        LJMP         MAIN
        ORG          0023H
        LJMP         UART_ISR

        ORG          0100H

UART_ISR:
        JNB          TI,CHKRI
        CLR          TI
        CLR          BUSY

CHKRI:
        JNB          RI,UARTISR_EXIT
        CLR          RI

UARTISR_EXIT:
        RETI

UART_INIT:
        MOV          SCON,#50H
        MOV          TMOD,#00H
        MOV          TL1,#0E8H          ;65536-11059200/115200/4=0FFE8H
        MOV          TH1,#0FFH
        SETB         TRI
        MOV          AUXR,#40H
        CLR          BUSY
        RET

UART_SEND:
        JB           BUSY,$
        SETB         BUSY
        MOV          SBUF,A

```

```

        RET

MAIN:
        MOV     SP, #5FH
        ORL     P_SW2, #80H           ;使能访问 XFR

        MOV     P0M0, #00H
        MOV     P0M1, #00H
        MOV     P1M0, #00H
        MOV     P1M1, #00H
        MOV     P2M0, #00H
        MOV     P2M1, #00H
        MOV     P3M0, #00H
        MOV     P3M1, #00H
        MOV     P4M0, #00H
        MOV     P4M1, #00H
        MOV     P5M0, #00H
        MOV     P5M1, #00H

        LCALL   UART_INIT
        SETB    ES
        SETB    EA

        MOV     DPTR, #ID
        MOV     RI, #7
NEXT:   CLR     A
        MOVC    A, @A+DPTR
        LCALL   UART_SEND
        INC     DPTR
        DJNZ    RI, NEXT

LOOP:   JMP     LOOP

        END

```

### 8.5.6 ~~读取全球唯一 ID 号 (从 RAM 中读取)~~

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT       (65536 - (FOSC / 115200+2) / 4)
```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
bit     busy;
char    *ID;
```

```
void UartIsr() interrupt 4
{
    if (TI)
```

```
{
    TI = 0;
    busy = 0;
}
if (RI)
{
    RI = 0;
}
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void main()
{
    char i;

    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    ID = (char idata *)0xf1;
    UartInit();
    ES = 1;
    EA = 1;

    for (i=0; i<7; i++)
    {
        UartSend(ID[i]);
    }

    while (1);
}
```

## 汇编代码

;测试工作频率为 11.0592MHz

```

AUXR      DATA      8EH
ID         DATA      0F1H
P_SW2     DATA      0BAH

BUSY      BIT         20H.0

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0023H
          LJMP         UART_ISR

          ORG          0100H

UART_ISR:
          JNB          TI,CHKRI
          CLR          TI
          CLR          BUSY

CHKRI:
          JNB          RI,UARTISR_EXIT
          CLR          RI

UARTISR_EXIT:
          RETI

UART_INIT:
          MOV          SCON,#50H
          MOV          TMOD,#00H
          MOV          TL1,#0E8H
          MOV          TH1,#0FFH
          SETB         TR1
          MOV          AUXR,#40H
          CLR          BUSY
          RET

UART_SEND:
          JB           BUSY,$
          SETB         BUSY
          MOV          SBUF,A
          RET

MAIN:
          MOV          SP, #5FH

```

;65536-11059200/115200/4=0FFE8H

```

        ORL        P_SW2,#80H           ;使能访问 XFR

        MOV        P0M0, #00H
        MOV        P0M1, #00H
        MOV        P1M0, #00H
        MOV        P1M1, #00H
        MOV        P2M0, #00H
        MOV        P2M1, #00H
        MOV        P3M0, #00H
        MOV        P3M1, #00H
        MOV        P4M0, #00H
        MOV        P4M1, #00H
        MOV        P5M0, #00H
        MOV        P5M1, #00H

        LCALL      UART_INIT
        SETB       ES
        SETB       EA

        MOV        R0,#ID
        MOV        R1,#7
NEXT:    MOV        A,@R0
        LCALL      UART_SEND
        INC        R0
        DJNZ       R1,NEXT

LOOP:

        JMP        LOOP

        END
    
```

## 8.5.7 读取 32K 掉电唤醒定时器的频率 (从 CHIPID 中读取)

### C 语言代码

```

//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

#define FOSC      11059200UL
#define BRT       (65536 - (FOSC / 115200+2) / 4)

//加 2 操作是为了让 Keil 编译器
//自动实现四舍五入运算

#define CPUIDBASE 0xfde0
#define F32K_ADDR (*(unsigned int volatile xdata*)(CPUIDBASE + 0x09))

bit        busy;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
}
    
```



```
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;

    UartSend(F32K_ADDR >> 8); //读取 32K 频率的高字节
    UartSend(F32K_ADDR);      //读取 32K 频率的低字节

    while (1);
}
```

---

## 汇编代码

---

;测试工作频率为 11.0592MHz

**CPUIDBASE EQU 0FDE0H**

*F32K\_ADDR EQU CPUIDBASE + 09H*

*AUXR DATA 8EH*  
*P\_SW2 DATA 0BAH*

*BUSY BIT 20H.0*

*P0M1 DATA 093H*  
*P0M0 DATA 094H*  
*P1M1 DATA 091H*  
*P1M0 DATA 092H*  
*P2M1 DATA 095H*  
*P2M0 DATA 096H*  
*P3M1 DATA 0B1H*  
*P3M0 DATA 0B2H*  
*P4M1 DATA 0B3H*  
*P4M0 DATA 0B4H*  
*P5M1 DATA 0C9H*  
*P5M0 DATA 0CAH*

*ORG 0000H*  
*LJMP MAIN*  
*ORG 0023H*  
*LJMP UART\_ISR*

*ORG 0100H*

*UART\_ISR:*

*JNB TI,CHKRI*  
*CLR TI*  
*CLR BUSY*

*CHKRI:*

*JNB RI,UARTISR\_EXIT*  
*CLR RI*

*UARTISR\_EXIT:*

*RETI*

*UART\_INIT:*

*MOV SCON,#50H*  
*MOV TMOD,#00H*  
*MOV TL1,#0E8H ;65536-11059200/115200/4=0FFE8H*  
*MOV TH1,#0FFH*  
*SETB TRI*  
*MOV AUXR,#40H*  
*CLR BUSY*  
*RET*

*UART\_SEND:*

*JB BUSY,\$*  
*SETB BUSY*  
*MOV SBUF,A*  
*RET*

*MAIN:*

*MOV SP, #5FH*  
*ORL P\_SW2,#80H ;使能访问 XFR*  
  
*MOV P0M0, #00H*  
*MOV P0M1, #00H*

```

MOV      P1M0, #00H
MOV      P1M1, #00H
MOV      P2M0, #00H
MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

LCALL    UART_INIT
SETB     ES
SETB     EA

MOV      DPTR, #F32K_ADDR
CLR      A
MOVX     A, @DPTR          ; 读取 32K 频率的高字节
LCALL    UART_SEND
INC      DPTR
CLR      A
MOVX     A, @DPTR          ; 读取 32K 频率的低字节
LCALL    UART_SEND

LOOP:
        JMP      LOOP

END

```

## 8.5.8 读取 32K 掉电唤醒定时器的频率（从 Flash 程序存储器（ROM）中读取）

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define FOSC      11059200UL
#define BRT       (65536 - (FOSC / 115200 + 2) / 4)

```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```

bit      busy;
int      *F32K;

```

```

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
}

```

```

    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    F32K = (int code *)0x3ff5; // STC8H1K16
    UartInit();
    ES = 1;
    EA = 1;

    UartSend(*F32K >> 8); //读取 32K 频率的高字节
    UartSend(*F32K);      //读取 32K 频率的低字节

    while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|       |      |        |             |
|-------|------|--------|-------------|
| AUXR  | DATA | 8EH    |             |
| F32K  | EQU  | 03FF5H | ; STC8H1K16 |
| P_SW2 | DATA | 0BAH   |             |

```

BUSY      BIT      20H.0

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

      ORG      0000H
      LJMP     MAIN
      ORG      0023H
      LJMP     UART_ISR

      ORG      0100H

UART_ISR:
      JNB      TI,CHKRI
      CLR      TI
      CLR      BUSY

CHKRI:
      JNB      RI,UARTISR_EXIT
      CLR      RI

UARTISR_EXIT:
      RETI

UART_INIT:
      MOV      SCON,#50H
      MOV      TMOD,#00H
      MOV      TL1,#0E8H ;65536-11059200/115200/4=0FFE8H
      MOV      TH1,#0FFH
      SETB     TRI
      MOV      AUXR,#40H
      CLR      BUSY
      RET

UART_SEND:
      JB       BUSY,$
      SETB     BUSY
      MOV      SBUF,A
      RET

MAIN:
      MOV      SP,#5FH
      ORL      P_SW2,#80H ;使能访问XFR

      MOV      P0M0,#00H
      MOV      P0M1,#00H
      MOV      P1M0,#00H
      MOV      P1M1,#00H
      MOV      P2M0,#00H
      MOV      P2M1,#00H

```

```

MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

LCALL    UART_INIT
SETB     ES
SETB     EA

MOV      DPTR, #F32K
CLR      A
MOVC     A, @A+DPTR          ;读取 32K 频率的高字节
LCALL    UART_SEND
INC      DPTR
CLR      A
MOVC     A, @A+DPTR          ;读取 32K 频率的低字节
LCALL    UART_SEND

LOOP:
        JMP      LOOP

```

END

## 8.5.9 读取 32K 掉电唤醒定时器的频率 (从 RAM 中读取)

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define FOSC      11059200UL
#define BRT       (65536 - (FOSC / 115200+2) / 4)

```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```

bit      busy;
int      *F32K;

```

```

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

```

void UartInit()

```

{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void main()
{
    P_SW2 /= 0x80;                                     //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    F32K = (int idata *)0xf8;
    UartInit();
    ES = 1;
    EA = 1;

    UartSend(*F32K >> 8);                             //读取 32K 频率的高字节
    UartSend(*F32K);                                   //读取 32K 频率的低字节

    while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|       |      |       |
|-------|------|-------|
| AUXR  | DATA | 8EH   |
| F32K  | DATA | 0F8H  |
| P_SW2 | DATA | 0BAH  |
| BUSY  | BIT  | 20H.0 |
| P1M1  | DATA | 091H  |
| P1M0  | DATA | 092H  |
| P0M1  | DATA | 093H  |
| P0M0  | DATA | 094H  |

```

P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0023H
          LJMP         UART_ISR

          ORG          0100H

UART_ISR:
          JNB          TI,CHKRI
          CLR          TI
          CLR          BUSY

CHKRI:
          JNB          RI,UARTISR_EXIT
          CLR          RI

UARTISR_EXIT:
          RETI

UART_INIT:
          MOV          SCON,#50H
          MOV          TMOD,#00H
          MOV          TL1,#0E8H          ;65536-11059200/115200/4=0FFE8H
          MOV          TH1,#0FFH
          SETB         TRI
          MOV          AUXR,#40H
          CLR          BUSY
          RET

UART_SEND:
          JB           BUSY,$
          SETB         BUSY
          MOV          SBUF,A
          RET

MAIN:
          MOV          SP,#5FH
          ORL          P_SW2,#80H          ;使能访问 XFR

          MOV          P0M0,#00H
          MOV          P0M1,#00H
          MOV          P1M0,#00H
          MOV          P1M1,#00H
          MOV          P2M0,#00H
          MOV          P2M1,#00H
          MOV          P3M0,#00H
          MOV          P3M1,#00H
          MOV          P4M0,#00H
          MOV          P4M1,#00H
          MOV          P5M0,#00H
          MOV          P5M1,#00H

```



```

        LCALL    UART_INIT
        SETB     ES
        SETB     EA

        MOV      R0,#F32K
        MOV      A,@R0                ;读取 32K 频率的高字节
        LCALL    UART_SEND
        INC      R0
        MOV      A,@R0                ;读取 32K 频率的低字节
        LCALL    UART_SEND

LOOP:
        JMP      LOOP

END

```

## 8.5.10 用户自定义内部 IRC 频率 (从 CHIPID 中读取)

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

sfr      VRTRIM      =    0xA6;

#define    CLKSEL      (*(unsigned char volatile xdata *)0xfe00)
#define    CLKDIV      (*(unsigned char volatile xdata *)0xfe01)

#define    USBCHIPID
#define    CPUIDBASE    0xfde0

#define    T22M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0b))    //22.1184MHz
#define    T24M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0c))    //24MHz
#ifdef    USBCHIPID
#define    T27M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0d))    //27MHz
#define    T30M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0e))    //30MHz
#define    T33M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0f))    //33.1776MHz
#define    T35M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x10))    //35MHz
#define    T36M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x11))    //36.864MHz
#define    T40M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x12))    //40MHz
#define    T44M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x13))    //44.2368MHz
#define    T48M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x14))    //48MHz
#else
#define    T20M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0d))    //20MHz
#define    T27M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0e))    //27MHz
#define    T30M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0f))    //30MHz
#define    T33M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x10))    //33.1776MHz
#define    T35M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x11))    //35MHz
#define    T36M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x12))    //36.864MHz
#define    T40M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x13))    //40MHz
#define    T45M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x14))    //45MHz
#endif
#define    VRT6M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x15))    //VRTRIM_6M
#define    VRT10M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x16))    //VRTRIM_10M

```

```
#define VRT27M_ADDR (*(unsigned char volatile xdata *)(CPUIDBASE + 0x17)) //VRTRIM_27M
#define VRT44M_ADDR (*(unsigned char volatile xdata *)(CPUIDBASE + 0x18)) //VRTRIM_44M
```

```
void main()
```

```
{
    P_SW2 /= 0x80; //使能访问XFR
```

```
    P0M0 = 0x00;
```

```
    P0M1 = 0x00;
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P2M0 = 0x00;
```

```
    P2M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P4M0 = 0x00;
```

```
    P4M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
//#ifndef USBCHIPID
```

```
// //选择20MHz
```

```
// CLKDIV = 0x04;
```

```
// IRTRIM = T20M_ADDR;
```

```
// VRTRIM = VRT27M_ADDR;
```

```
// IRCBAND = 0x02;
```

```
// CLKDIV = 0x00;
```

```
//#endif
```

```
// //选择22.1184MHz
```

```
// CLKDIV = 0x04;
```

```
// IRTRIM = T22M_ADDR;
```

```
// VRTRIM = VRT27M_ADDR;
```

```
// IRCBAND = 0x02;
```

```
// CLKDIV = 0x00;
```

```
//选择24MHz
```

```
CLKDIV = 0x04;
```

```
IRTRIM = T24M_ADDR;
```

```
VRTRIM = VRT27M_ADDR;
```

```
IRCBAND = 0x02;
```

```
CLKDIV = 0x00;
```

```
// //选择27MHz
```

```
// CLKDIV = 0x04;
```

```
// IRTRIM = T27M_ADDR;
```

```
// VRTRIM = VRT27M_ADDR;
```

```
// IRCBAND = 0x02;
```

```
// CLKDIV = 0x00;
```

```
// //选择30MHz
```

```
// CLKDIV = 0x04;
```

```
// IRTRIM = T30M_ADDR;
```

```
// VRTRIM = VRT27M_ADDR;
```

```
// IRCBAND = 0x02;
```

```
// CLKDIV = 0x00;
```

```
// //选择33.1776MHz
```

```
// CLKDIV = 0x04;
```

```

//  IRTRIM = T33M_ADDR;
//  VRTRIM = VRT27M_ADDR;
//  IRCBAND = 0x02;
//  CLKDIV = 0x00;

//  //选择35MHz
//  CLKDIV = 0x04;
//  IRTRIM = T35M_ADDR;
//  VRTRIM = VRT44M_ADDR;
//  IRCBAND = 0x03;
//  CLKDIV = 0x00;

//#ifdef  USBCHIPID
//  //选择44.2368MHz
//  CLKDIV = 0x04;
//  IRTRIM = T44M_ADDR;
//  VRTRIM = VRT44M_ADDR;
//  IRCBAND = 0x03;
//  CLKDIV = 0x00;

//  //选择48MHz
//  CLKDIV = 0x04;
//  IRTRIM = T48M_ADDR;
//  VRTRIM = VRT44M_ADDR;
//  IRCBAND = 0x03;
//  CLKDIV = 0x00;
//#else
//  //选择40MHz
//  CLKDIV = 0x04;
//  IRTRIM = T40M_ADDR;
//  VRTRIM = VRT44M_ADDR;
//  IRCBAND = 0x03;
//  CLKDIV = 0x00;

//  //选择45MHz
//  CLKDIV = 0x04;
//  IRTRIM = T45M_ADDR;
//  VRTRIM = VRT44M_ADDR;
//  IRCBAND = 0x03;
//  CLKDIV = 0x00;
//#endif

```

```

    while (1);
}

```

## 汇编代码

;测试工作频率为11.0592MHz

\$SET(USBCHIPID = 1)

CPUIDBASE EQU 0FDE0H

T22M\_ADDR EQU CPUIDBASE + 0BH ;22.1184MHz

T24M\_ADDR EQU CPUIDBASE + 0CH ;24MHz

\$IF(USBCHIPID == 1)

T27M\_ADDR EQU CPUIDBASE + 0DH ;27MHz

T30M\_ADDR EQU CPUIDBASE + 0EH ;30MHz

T33M\_ADDR EQU CPUIDBASE + 0FH ;33.1776MHz

T35M\_ADDR EQU CPUIDBASE + 10H ;35MHz

```

T36M_ADDR    EQU    CPUIDBASE + 11H    ;36.864MHz
T40M_ADDR    EQU    CPUIDBASE + 12H    ;40MHz
T44M_ADDR    EQU    CPUIDBASE + 13H    ;44.2368MHz
T48M_ADDR    EQU    CPUIDBASE + 14H    ;20MHz
$ELSE
T20M_ADDR    EQU    CPUIDBASE + 0DH    ;20MHz
T27M_ADDR    EQU    CPUIDBASE + 0EH    ;27MHz
T30M_ADDR    EQU    CPUIDBASE + 0FH    ;30MHz
T33M_ADDR    EQU    CPUIDBASE + 10H    ;33.1776MHz
T35M_ADDR    EQU    CPUIDBASE + 11H    ;35MHz
T36M_ADDR    EQU    CPUIDBASE + 12H    ;36.864MHz
T40M_ADDR    EQU    CPUIDBASE + 13H    ;40MHz
T45M_ADDR    EQU    CPUIDBASE + 14H    ;45MHz
$ENDIF
VRT6M_ADDR    EQU    CPUIDBASE + 15H    ;VRTRIM_6M
VRT10M_ADDR    EQU    CPUIDBASE + 16H    ;VRTRIM_10M
VRT27M_ADDR    EQU    CPUIDBASE + 17H    ;VRTRIM_27M
VRT44M_ADDR    EQU    CPUIDBASE + 18H    ;VRTRIM_44M

P_SW2        DATA    0BAH
CLKSEL        EQU    0FE00H
CLKDIV        EQU    0FE01H

IRCBAND       DATA    09DH
IRTRIM        DATA    09FH
VRTRIM        DATA    0A6H

P1M1         DATA    091H
P1M0         DATA    092H
P0M1         DATA    093H
P0M0         DATA    094H
P2M1         DATA    095H
P2M0         DATA    096H
P3M1         DATA    0B1H
P3M0         DATA    0B2H
P4M1         DATA    0B3H
P4M0         DATA    0B4H
P5M1         DATA    0C9H
P5M0         DATA    0CAH

ORG          0000H
LJMP        MAIN

ORG          0100H
MAIN:
MOV         SP, #5FH
ORL         P_SW2, #80H    ;使能访问 XFR

MOV         P0M0, #00H
MOV         P0M1, #00H
MOV         P1M0, #00H
MOV         P1M1, #00H
MOV         P2M0, #00H
MOV         P2M1, #00H
MOV         P3M0, #00H
MOV         P3M1, #00H
MOV         P4M0, #00H
MOV         P4M1, #00H
MOV         P5M0, #00H

```

**MOV P5M1, #00H**

```

; $IF(USBCHIPID == 0)
;      ;选择 20MHz
;      MOV      A, #4
;      MOV      DPTR, #CLKDIV
;      MOVX     @DPTR, A
;      MOV      DPTR, #T20M_ADDR
;      CLR      A
;      MOVX     A, @DPTR
;      MOV      IRTRIM, A
;      MOV      DPTR, #VRT27M_ADDR
;      CLR      A
;      MOVX     A, @DPTR
;      MOV      VRTRIM, A
;      MOV      IRCBAND, #02H
;      MOV      A, #0
;      MOV      DPTR, #CLKDIV
;      MOVX     @DPTR, A
; $ENDIF

```

```

;      ;选择 22.1184MHz
;      MOV      A, #4
;      MOV      DPTR, #CLKDIV
;      MOVX     @DPTR, A
;      MOV      DPTR, #T22M_ADDR
;      CLR      A
;      MOVX     A, @DPTR
;      MOV      IRTRIM, A
;      MOV      DPTR, #VRT27M_ADDR
;      CLR      A
;      MOVX     A, @DPTR
;      MOV      VRTRIM, A
;      MOV      IRCBAND, #02H
;      MOV      A, #0
;      MOV      DPTR, #CLKDIV
;      MOVX     @DPTR, A

```

```

;      ;选择 24MHz
;      MOV      A, #4
;      MOV      DPTR, #CLKDIV
;      MOVX     @DPTR, A
;      MOV      DPTR, #T24M_ADDR
;      CLR      A
;      MOVX     A, @DPTR
;      MOV      IRTRIM, A
;      MOV      DPTR, #VRT27M_ADDR
;      CLR      A
;      MOVX     A, @DPTR
;      MOV      VRTRIM, A
;      MOV      IRCBAND, #02H
;      MOV      A, #0
;      MOV      DPTR, #CLKDIV
;      MOVX     @DPTR, A

```

```

;      ;选择 27MHz
;      MOV      A, #4
;      MOV      DPTR, #CLKDIV
;      MOVX     @DPTR, A

```

```

;      MOV      DPTR,#T27M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT27M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#02H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A

;      ;选择 30MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T30M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT27M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#02H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A

;      ;选择 33.1776MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T33M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT27M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#02H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A

;      ;选择 35MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T35M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A

```

```

;      MOV      IRCBAND,#03H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A

;      ;选择 36.864MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T36M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#03H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A

;IF(USBCHIPID == 1)
;      ;选择 44.2368MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#03H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A

;      ;选择 48MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T48M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#03H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;ELSE
;      ;选择 40MHz
;      MOV      A,#4

```

```

;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T40M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#03H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A

;      ;选择45MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T45M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#03H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
; $ENDIF

      JMP      $

      END

```

### 8.5.11 用户自定义内部 IRC 频率-两频段（从 Flash 程序存储器（ROM）中读取）

#### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define CLKSEL      (*(unsigned char volatile xdata *)0xfe00)
#define CLKDIV      (*(unsigned char volatile xdata *)0xfe01)

```

//下表为 STC8H1K08-20Pin 的参数列表

```

#define ID_ROMADDR      ((unsigned char code *)0x1ff9)
#define VREF_ROMADDR    (*(unsigned int code *)0x1ff7)
#define F32K_ROMADDR    (*(unsigned int code *)0x1ff5)

```



```

#define T22M_ROMADDR (*(unsigned char code *)0x1ff4) //22.1184MHz
#define T24M_ROMADDR (*(unsigned char code *)0x1ff3) //24MHz
#define T20M_ROMADDR (*(unsigned char code *)0x1ff2) //20MHz
#define T27M_ROMADDR (*(unsigned char code *)0x1ff1) //27MHz
#define T30M_ROMADDR (*(unsigned char code *)0x1ff0) //30MHz
#define T33M_ROMADDR (*(unsigned char code *)0x1fef) //33.1776MHz
#define T35M_ROMADDR (*(unsigned char code *)0x1fee) //35MHz
#define T36M_ROMADDR (*(unsigned char code *)0x1fed) //36.864MHz
#define VRT20M_ROMADDR (*(unsigned char code *)0x1fea) //VRTRIM_20M
#define VRT35M_ROMADDR (*(unsigned char code *)0x1fe9) //VRTRIM_35M

```

```
void main()
```

```

{
    P_SW2 /= 0x80; //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    // //选择20MHz
    // CLKDIV = 0x04;
    // IRTRIM = T20M_ROMADDR;
    // VRTRIM = VRT20M_ROMADDR;
    // IRCBAND = 0x00;
    // CLKDIV = 0x00;

    // //选择22.1184MHz
    // CLKDIV = 0x04;
    // IRTRIM = T22M_ROMADDR;
    // VRTRIM = VRT20M_ROMADDR;
    // IRCBAND = 0x00;
    // CLKDIV = 0x00;

    // //选择24MHz
    CLKDIV = 0x04;
    IRTRIM = T24M_ROMADDR;
    VRTRIM = VRT20M_ROMADDR;
    IRCBAND = 0x00;
    CLKDIV = 0x00;

    // //选择27MHz
    // CLKDIV = 0x04;
    // IRTRIM = T27M_ROMADDR;
    // VRTRIM = VRT35M_ROMADDR;
    // IRCBAND = 0x01;
    // CLKDIV = 0x00;

    // //选择30MHz
    // CLKDIV = 0x04;
    // IRTRIM = T30M_ROMADDR;

```

```
// VRTRIM = VRT35M_ROMADDR;
// IRCBAND = 0x01;
// CLKDIV = 0x00;

// //选择 33.1776MHz
// CLKDIV = 0x04;
// IRTRIM = T33M_ROMADDR;
// VRTRIM = VRT35M_ROMADDR;
// IRCBAND = 0x01;
// CLKDIV = 0x00;

// //选择 35MHz
// CLKDIV = 0x04;
// IRTRIM = T35M_ROMADDR;
// VRTRIM = VRT35M_ROMADDR;
// IRCBAND = 0x01;
// CLKDIV = 0x00;

while (1);
}
```

## 汇编代码

;测试工作频率为 11.0592MHz

;下表为 STC8HIK08-20Pin 的参数列表

|                       |             |               |              |
|-----------------------|-------------|---------------|--------------|
| <b>ID_ROMADDR</b>     | <b>EQU</b>  | <b>01FF9H</b> |              |
| <b>VREF_ROMADDR</b>   | <b>EQU</b>  | <b>01FF7H</b> |              |
| <b>F32K_ROMADDR</b>   | <b>EQU</b>  | <b>01FF5H</b> |              |
| <b>T22M_ROMADDR</b>   | <b>EQU</b>  | <b>01FF4H</b> | //22.1184MHz |
| <b>T24M_ROMADDR</b>   | <b>EQU</b>  | <b>01FF3H</b> | //24MHz      |
| <b>T20M_ROMADDR</b>   | <b>EQU</b>  | <b>01FF2H</b> | //20MHz      |
| <b>T27M_ROMADDR</b>   | <b>EQU</b>  | <b>01FF1H</b> | //27MHz      |
| <b>T30M_ROMADDR</b>   | <b>EQU</b>  | <b>01FF0H</b> | //30MHz      |
| <b>T33M_ROMADDR</b>   | <b>EQU</b>  | <b>01FEFH</b> | //33.1776MHz |
| <b>T35M_ROMADDR</b>   | <b>EQU</b>  | <b>01FEEH</b> | //35MHz      |
| <b>T36M_ROMADDR</b>   | <b>EQU</b>  | <b>01FEDH</b> | //36.864MHz  |
| <b>VRT20M_ROMADDR</b> | <b>EQU</b>  | <b>01FEAH</b> | //VRTRIM_20M |
| <b>VRT35M_ROMADDR</b> | <b>EQU</b>  | <b>01FE9H</b> | //VRTRIM_35M |
| <b>P_SW2</b>          | <b>DATA</b> | <b>0BAH</b>   |              |
| <b>CLKSEL</b>         | <b>EQU</b>  | <b>0FE00H</b> |              |
| <b>CLKDIV</b>         | <b>EQU</b>  | <b>0FE01H</b> |              |
| <b>IRCBAND</b>        | <b>DATA</b> | <b>09DH</b>   |              |
| <b>IRCTRIM</b>        | <b>DATA</b> | <b>09FH</b>   |              |
| <b>VRTRIM</b>         | <b>DATA</b> | <b>0A6H</b>   |              |
| <b>P1M1</b>           | <b>DATA</b> | <b>091H</b>   |              |
| <b>P1M0</b>           | <b>DATA</b> | <b>092H</b>   |              |
| <b>P0M1</b>           | <b>DATA</b> | <b>093H</b>   |              |
| <b>P0M0</b>           | <b>DATA</b> | <b>094H</b>   |              |
| <b>P2M1</b>           | <b>DATA</b> | <b>095H</b>   |              |
| <b>P2M0</b>           | <b>DATA</b> | <b>096H</b>   |              |
| <b>P3M1</b>           | <b>DATA</b> | <b>0B1H</b>   |              |
| <b>P3M0</b>           | <b>DATA</b> | <b>0B2H</b>   |              |
| <b>P4M1</b>           | <b>DATA</b> | <b>0B3H</b>   |              |
| <b>P4M0</b>           | <b>DATA</b> | <b>0B4H</b>   |              |
| <b>P5M1</b>           | <b>DATA</b> | <b>0C9H</b>   |              |

```

P5M0      DATA      0CAH

           ORG         0000H
           LJMP        MAIN

MAIN:      ORG         0100H

           MOV         SP, #5FH
           ORL         P_SW2, #80H           ;使能访问 XFR

           MOV         P0M0, #00H
           MOV         P0M1, #00H
           MOV         P1M0, #00H
           MOV         P1M1, #00H
           MOV         P2M0, #00H
           MOV         P2M1, #00H
           MOV         P3M0, #00H
           MOV         P3M1, #00H
           MOV         P4M0, #00H
           MOV         P4M1, #00H
           MOV         P5M0, #00H
           MOV         P5M1, #00H

;          ;选择 20MHz
;          MOV         A, #4
;          MOV         DPTR, #CLKDIV
;          MOV         DPTR, #T20M_ROMADDR
;          CLR         A
;          MOVC        A, @A+DPTR
;          MOV         IRTRIM, A
;          MOV         DPTR, #VRT20M_ROMADDR
;          CLR         A
;          MOVC        A, @A+DPTR
;          MOV         VRTRIM, A
;          MOV         IRCBAND, #00H
;          MOV         A, #0
;          MOV         DPTR, #CLKDIV

;          ;选择 22.1184MHz
;          MOV         A, #4
;          MOV         DPTR, #CLKDIV
;          MOV         DPTR, #T22M_ROMADDR
;          CLR         A
;          MOVC        A, @A+DPTR
;          MOV         IRTRIM, A
;          MOV         DPTR, #VRT20M_ROMADDR
;          CLR         A
;          MOVC        A, @A+DPTR
;          MOV         VRTRIM, A
;          MOV         IRCBAND, #00H
;          MOV         A, #0
;          MOV         DPTR, #CLKDIV

;          ;选择 24MHz
           MOV         A, #4
           MOV         DPTR, #CLKDIV
           MOV         DPTR, #T24M_ROMADDR
           CLR         A
           MOVC        A, @A+DPTR

```

```

MOV      IRTRIM,A
MOV      DPTR,#VRT20M_ROMADDR
CLR      A
MOVC     A,@A+DPTR
MOV      VRTRIM,A
MOV      IRCBAND,#00H
MOV      A,#0
MOV      DPTR,#CLKDIV

;      ;选择 27MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T27M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT35M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#01H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV

;      ;选择 30MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T30M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT35M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#01H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV

;      ;选择 33.1776MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T33M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT35M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#01H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV

;      ;选择 35MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T35M_ROMADDR
;      CLR      A

```

```

;      MOV      A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT35M_ROMADDR
;      CLR      A
;      MOV      A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#01H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV

;      ;选择 36.864MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T36M_ROMADDR
;      CLR      A
;      MOV      A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT35M_ROMADDR
;      CLR      A
;      MOV      A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#01H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV

      JMP      $

      END

```

## 8.5.12 用户自定义内部 IRC 频率-四频段（从 Flash 程序存储器（ROM）中读取）

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define CLKSEL      (*(unsigned char volatile xdata *)0xfe00)
#define CLKDIV      (*(unsigned char volatile xdata *)0xfe01)

```

// 下表为 STC8H3K64S4 的参数列表

```

#define ID_ROMADDR      ((unsigned char code *)0xfd9)
#define VREF_ROMADDR    (*(unsigned int code *)0xfd7)
#define F32K_ROMADDR    (*(unsigned int code *)0xfd5)
#define T22M_ROMADDR    (*(unsigned char code *)0xfd4)           //22.1184MHz
#define T24M_ROMADDR    (*(unsigned char code *)0xfd3)           //24MHz
#define T20M_ROMADDR    (*(unsigned char code *)0xfd2)           //20MHz
#define T27M_ROMADDR    (*(unsigned char code *)0xfd1)           //27MHz
#define T30M_ROMADDR    (*(unsigned char code *)0xfd0)           //30MHz
#define T33M_ROMADDR    (*(unsigned char code *)0xfdef)          //33.1776MHz
#define T35M_ROMADDR    (*(unsigned char code *)0xfdee)          //35MHz
#define T36M_ROMADDR    (*(unsigned char code *)0xfded)          //36.864MHz

```

```
#define T40M_ROMADDR (*(unsigned char code *)0xfdec) //40MHz
#define T45M_ROMADDR (*(unsigned char code *)0xfdeb) //45MHz
#define VRT6M_ROMADDR (*(unsigned char code *)0xfdea) //VRTRIM_6M
#define VRT10M_ROMADDR (*(unsigned char code *)0xfe9) //VRTRIM_10M
#define VRT27M_ROMADDR (*(unsigned char code *)0xfde8) //VRTRIM_27M
#define VRT44M_ROMADDR (*(unsigned char code *)0xfe7) //VRTRIM_44M
```

```
void main()
```

```
{
    P_SW2 /= 0x80; //使能访问 XFR
```

```
    P0M0 = 0x00;
```

```
    P0M1 = 0x00;
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P2M0 = 0x00;
```

```
    P2M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P4M0 = 0x00;
```

```
    P4M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
// //选择20MHz
```

```
// CLKDIV = 0x04;
```

```
// IRTRIM = T20M_ROMADDR;
```

```
// VRTRIM = VRT27M_ROMADDR;
```

```
// IRCBAND = 0x02;
```

```
// CLKDIV = 0x00;
```

```
// //选择22.1184MHz
```

```
// CLKDIV = 0x04;
```

```
// IRTRIM = T22M_ROMADDR;
```

```
// VRTRIM = VRT27M_ROMADDR;
```

```
// IRCBAND = 0x02;
```

```
// CLKDIV = 0x00;
```

```
//选择24MHz
```

```
CLKDIV = 0x04;
```

```
IRTRIM = T24M_ROMADDR;
```

```
VRTRIM = VRT27M_ROMADDR;
```

```
IRCBAND = 0x02;
```

```
CLKDIV = 0x00;
```

```
// //选择27MHz
```

```
// CLKDIV = 0x04;
```

```
// IRTRIM = T27M_ROMADDR;
```

```
// VRTRIM = VRT27M_ROMADDR;
```

```
// IRCBAND = 0x02;
```

```
// CLKDIV = 0x00;
```

```
// //选择30MHz
```

```
// CLKDIV = 0x04;
```

```
// IRTRIM = T30M_ROMADDR;
```

```
// VRTRIM = VRT27M_ROMADDR;
```

```
// IRCBAND = 0x02;
```

```
// CLKDIV = 0x00;
```

```
// //选择 33.1776MHz
// CLKDIV = 0x04;
// IRTRIM = T33M_ROMADDR;
// VRTRIM = VRT27M_ROMADDR;
// IRCBAND = 0x02;
// CLKDIV = 0x00;

// //选择 35MHz
// CLKDIV = 0x04;
// IRTRIM = T35M_ROMADDR;
// VRTRIM = VRT44M_ROMADDR;
// IRCBAND = 0x03;
// CLKDIV = 0x00;

// //选择 36.864MHz
// CLKDIV = 0x04;
// IRTRIM = T36M_ROMADDR;
// VRTRIM = VRT44M_ROMADDR;
// IRCBAND = 0x03;
// CLKDIV = 0x00;

// //选择 40MHz
// CLKDIV = 0x04;
// IRTRIM = T40M_ROMADDR;
// VRTRIM = VRT44M_ROMADDR;
// IRCBAND = 0x03;
// CLKDIV = 0x00;

// //选择 45MHz
// CLKDIV = 0x04;
// IRTRIM = T45M_ROMADDR;
// VRTRIM = VRT44M_ROMADDR;
// IRCBAND = 0x03;
// CLKDIV = 0x00;
```

```
while (1);
```

```
}
```

## 汇编代码

;测试工作频率为 11.0592MHz

;下表为 STC8H3K64S4 的参数列表

|                |     |        |              |
|----------------|-----|--------|--------------|
| ID_ROMADDR     | EQU | 0FDF9H |              |
| VREF_ROMADDR   | EQU | 0FDF7H |              |
| F32K_ROMADDR   | EQU | 0FDF5H |              |
| T22M_ROMADDR   | EQU | 0FDF4H | //22.1184MHz |
| T24M_ROMADDR   | EQU | 0FDF3H | //24MHz      |
| T20M_ROMADDR   | EQU | 0FDF2H | //20MHz      |
| T27M_ROMADDR   | EQU | 0FDF1H | //27MHz      |
| T30M_ROMADDR   | EQU | 0FDF0H | //30MHz      |
| T33M_ROMADDR   | EQU | 0FDEFH | //33.1776MHz |
| T35M_ROMADDR   | EQU | 0FDEEH | //35MHz      |
| T36M_ROMADDR   | EQU | 0FDEDH | //36.864MHz  |
| T40M_ROMADDR   | EQU | 0FDECH | //40MHz      |
| T44M_ROMADDR   | EQU | 0FDEBH | //45MHz      |
| VRT6M_ROMADDR  | EQU | 0FDEAH | //VRTRIM_6M  |
| VRT10M_ROMADDR | EQU | 0FDE9H | //VRTRIM_10M |
| VRT27M_ROMADDR | EQU | 0FDE8H | //VRTRIM_27M |

```

VRT44M_ROMADDR    EQU    0FDE7H                //VRTRIM_44M

P_SW2              DATA    0BAH
CLKSEL             EQU      0FE00H
CLKDIV             EQU      0FE01H

IRCBAND            DATA    09DH
IRCTRIM            DATA    09FH
VRTRIM             DATA    0A6H

P1M1              DATA    091H
P1M0              DATA    092H
P0M1              DATA    093H
P0M0              DATA    094H
P2M1              DATA    095H
P2M0              DATA    096H
P3M1              DATA    0B1H
P3M0              DATA    0B2H
P4M1              DATA    0B3H
P4M0              DATA    0B4H
P5M1              DATA    0C9H
P5M0              DATA    0CAH

                ORG      0000H
                LJMP     MAIN

MAIN:            ORG      0100H

                MOV      SP, #5FH
                ORL       P_SW2, #80H           ;使能访问 XFR

                MOV      P0M0, #00H
                MOV      P0M1, #00H
                MOV      P1M0, #00H
                MOV      P1M1, #00H
                MOV      P2M0, #00H
                MOV      P2M1, #00H
                MOV      P3M0, #00H
                MOV      P3M1, #00H
                MOV      P4M0, #00H
                MOV      P4M1, #00H
                MOV      P5M0, #00H
                MOV      P5M1, #00H

;                ;选择 20MHz
;                MOV      A, #4
;                MOV      DPTR, #CLKDIV
;                MOV      DPTR, #T20M_ROMADDR
;                CLR      A
;                MOVC     A, @A+DPTR
;                MOV      IRTRIM, A
;                MOV      DPTR, #VRT27M_ROMADDR
;                CLR      A
;                MOVC     A, @A+DPTR
;                MOV      VRTRIM, A
;                MOV      IRCBAND, #02H
;                MOV      A, #0
;                MOV      DPTR, #CLKDIV

```



```

;      ;选择 22.1184MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T22M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT27M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#02H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV

```

;选择 24MHz

```

MOV      A,#4
MOV      DPTR,#CLKDIV
MOV      DPTR,#T24M_ROMADDR
CLR      A
MOVC     A,@A+DPTR
MOV      IRTRIM,A
MOV      DPTR,#VRT27M_ROMADDR
CLR      A
MOVC     A,@A+DPTR
MOV      VRTRIM,A
MOV      IRCBAND,#02H
MOV      A,#0
MOV      DPTR,#CLKDIV

```

;选择 27MHz

```

;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T27M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT27M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#02H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV

```

;选择 30MHz

```

;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T30M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT27M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#02H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV

```

```
; ;选择 33.1776MHz
; MOV A,#4
; MOV DPTR,#CLKDIV
; MOV DPTR,#T33M_ROMADDR
; CLR A
; MOVC A,@A+DPTR
; MOV IRTRIM,A
; MOV DPTR,#VRT27M_ROMADDR
; CLR A
; MOVC A,@A+DPTR
; MOV VRTRIM,A
; MOV IRCBAND,#02H
; MOV A,#0
; MOV DPTR,#CLKDIV
```

```
; ;选择 35MHz
; MOV A,#4
; MOV DPTR,#CLKDIV
; MOV DPTR,#T35M_ROMADDR
; CLR A
; MOVC A,@A+DPTR
; MOV IRTRIM,A
; MOV DPTR,#VRT44M_ROMADDR
; CLR A
; MOVC A,@A+DPTR
; MOV VRTRIM,A
; MOV IRCBAND,#03H
; MOV A,#0
; MOV DPTR,#CLKDIV
```

```
; ;选择 36.864MHz
; MOV A,#4
; MOV DPTR,#CLKDIV
; MOV DPTR,#T36M_ROMADDR
; CLR A
; MOVC A,@A+DPTR
; MOV IRTRIM,A
; MOV DPTR,#VRT44M_ROMADDR
; CLR A
; MOVC A,@A+DPTR
; MOV VRTRIM,A
; MOV IRCBAND,#03H
; MOV A,#0
; MOV DPTR,#CLKDIV
```

```
; ;选择 40MHz
; MOV A,#4
; MOV DPTR,#CLKDIV
; MOV DPTR,#T40M_ROMADDR
; CLR A
; MOVC A,@A+DPTR
; MOV IRTRIM,A
; MOV DPTR,#VRT44M_ROMADDR
; CLR A
; MOVC A,@A+DPTR
; MOV VRTRIM,A
; MOV IRCBAND,#03H
; MOV A,#0
```

```
;      MOV      DPTR,#CLKDIV
;
;      ;选择 45MHz
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T45M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT44M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#03H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV

      JMP      $

      END
```

### 8.5.13 用户自定义内部 IRC 频率 (从 RAM 中读取)

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
char    *IRC22M;
char    *IRC24M;
```

```
void main()
```

```
{
    P_SW2 /= 0x80;                                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    IRC22M = (char idata *)0xfa;
    IRC24M = (char idata *) 0xfb;
//    IRTRIM = *IRC22M;                            //装载 22.1184MHz 的 IRC 参数
    IRTRIM = *IRC24M;                            //装载 24MHz 的 IRC 参数

    CLKDIV = 0;                                    //主时钟不预分频
```

```

    while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

```

P_SW2      DATA      0BAH
CLKDIV      EQU        0FE01H

IRTRIM      DATA      09FH

IRC22M      DATA      0FAH
IRC24M      DATA      0FBH

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

            ORG          0000H
            LJMP         MAIN

MAIN:        ORG          0100H

            MOV          SP, #5FH
            ORL          P_SW2, #80H          ;使能访问 XFR

            MOV          P0M0, #00H
            MOV          P0M1, #00H
            MOV          P1M0, #00H
            MOV          P1M1, #00H
            MOV          P2M0, #00H
            MOV          P2M1, #00H
            MOV          P3M0, #00H
            MOV          P3M1, #00H
            MOV          P4M0, #00H
            MOV          P4M1, #00H
            MOV          P5M0, #00H
            MOV          P5M1, #00H

;            MOV          R0, #IRC22M          ;装载 22.1184MHz 的 IRC 参数
;            MOV          IRTRIM, @R0
;            MOV          R0, #IRC24M          ;装载 24MHz 的 IRC 参数
;            MOV          IRTRIM, @R0

            MOV          A, #0                ;主时钟不预分频
            MOV          DPTR, #CLKDIV
            MOVX          @DPTR, A

```

*JMP*

\$

*END*

---

---

STC MCU

## 9 特殊功能寄存器

### 9.1 STC8H1K08 系列

|     | 0/8  | 1/9       | 2/A      | 3/B       | 4/C       | 5/D     | 6/E      | 7/F       |
|-----|------|-----------|----------|-----------|-----------|---------|----------|-----------|
| F8H |      |           |          |           |           |         |          | RSTCFG    |
| F0H | B    |           |          |           |           | IAP_TPS |          |           |
| E8H |      |           |          |           |           |         |          | AUXINTIF  |
| E0H | ACC  |           |          | DPS       | DPL1      | DPH1    | CMPCR1   | CMPCR2    |
| D8H |      |           |          |           |           |         | ADCCFG   |           |
| D0H | PSW  |           |          |           |           |         | T2H      | T2L       |
| C8H | P5   | P5M1      | P5M0     |           |           | SPSTAT  | SPCTL    | SPDAT     |
| C0H |      | WDT_CONTR | IAP_DATA | IAP_ADDRH | IAP_ADDRL | IAP_CMD | IAP_TRIG | IAP_CONTR |
| B8H | IP   | SADEN     | P_SW2    |           | ADC_CONTR | ADC_RES | ADC_RESL |           |
| B0H | P3   | P3M1      | P3M0     |           |           | IP2     | IP2H     | IPH       |
| A8H | IE   | SADDR     | WKTCL    | WKTCH     |           |         | TA       | IE2       |
| A0H |      |           | P_SW1    |           |           |         |          |           |
| 98H | SCON | SBUF      | S2CON    | S2BUF     |           | IRCBAND | LIRTRIM  | IRTRIM    |
| 90H | P1   | P1M1      | P1M0     |           |           |         |          |           |
| 88H | TCON | TMOD      | TL0      | TL1       | TH0       | TH1     | AUXR     | INTCLKO   |
| 80H |      | SP        | DPL      | DPH       |           |         |          | PCON      |

↑  
可位寻址

不可位寻址

注意: 寄存器地址能够被 8 整除的才可进行位寻址, 不能被 8 整除的则不可位寻址

|       | 0/8        | 1/9        | 2/A        | 3/B        | 4/C        | 5/D        | 6/E        | 7/F        |
|-------|------------|------------|------------|------------|------------|------------|------------|------------|
| FEF8H | PWMB_CCR6L | PWMB_CCR7H | PWMB_CCR7L | PWMB_CCR8H | PWMB_CCR8L | PWMB_BKR   | PWMB_DTR   | PWMB_OISR  |
| FEF0H | PWMB_PSCRH | PWMB_PSCRL | PWMB_ARRH  | PWMB_ARRL  | PWMB_RCR   | PWMB_CCR5H | PWMB_CCR5L | PWMB_CCR6H |
| FEE8H | PWMB_CCMR1 | PWMB_CCMR2 | PWMB_CCMR3 | PWMB_CCMR4 | PWMB_CCER1 | PWMB_CCER2 | PWMB_CNTRH | PWMB_CNTRL |
| FEE0H | PWMB_CR1   | PWMB_CR2   | PWMB_SMCR  | PWMB_ETR   | PWMB_IER   | PWMB_SR1   | PWMB_SR2   | PWMB_EGR   |
| FED8H | PWMA_CCR2L | PWMA_CCR3H | PWMA_CCR3L | PWMA_CCR4H | PWMA_CCR4L | PWMA_BKR   | PWMA_DTR   | PWMA_OISR  |
| FED0H | PWMA_PSCRH | PWMA_PSCRL | PWMA_ARRH  | PWMA_ARRL  | PWMA_RCR   | PWMA_CCR1H | PWMA_CCR1L | PWMA_CCR2H |
| FEC8H | PWMA_CCMR1 | PWMA_CCMR2 | PWMA_CCMR3 | PWMA_CCMR4 | PWMA_CCER1 | PWMA_CCER2 | PWMA_CNTRH | PWMA_CNTRL |
| FEC0H | PWMA_CR1   | PWMA_CR2   | PWMA_SMCR  | PWMA_ETR   | PWMA_IER   | PWMA_SR1   | PWMA_SR2   | PWMA_EGR   |
| FEB0H | PWMA_ETRPS | PWMA_ENO   | PWMA_PS    | PWMA_IOAUX | PWMB_ETRPS | PWMB_ENO   | PWMB_PS    | PWMB_IOAUX |
| FEA8H | ADCTIM     |            |            |            |            |            |            |            |
| FEA0H |            |            | TM2PS      |            |            |            |            |            |
| FE88H | I2CMSAUX   |            |            |            |            |            |            |            |
| FE80H | I2CCFG     | I2CMSCR    | I2CMSST    | I2CSLCR    | I2CSLST    | I2CSLADR   | I2CTxD     | I2CRxD     |
| FE30H |            | P1IE       |            | P3IE       |            |            |            |            |
| FE28H |            | P1DR       |            | P3DR       |            | P5DR       |            |            |
| FE20H |            | P1SR       |            | P3SR       |            | P5SR       |            |            |
| FE18H |            | P1NCS      |            | P3NCS      |            | P5NCS      |            |            |

|       |        |        |        |        |          |         |       |
|-------|--------|--------|--------|--------|----------|---------|-------|
| FE10H |        | P1PU   |        | P3PU   |          | P5PU    |       |
| FE00H | CLKSEL | CLKDIV | HIRCCR | XOSCCR | IRC32KCR | MCLKOCR | IRCDB |

STC MCU

## 9.2 STC8H1K28 系列

|     | 0/8  | 1/9       | 2/A      | 3/B       | 4/C       | 5/D     | 6/E      | 7/F       |
|-----|------|-----------|----------|-----------|-----------|---------|----------|-----------|
| F8H |      |           |          |           |           |         |          | RSTCFG    |
| F0H | B    |           |          |           |           | IAP_TPS |          |           |
| E8H |      |           |          |           |           |         |          | AUXINTIF  |
| E0H | ACC  |           |          | DPS       | DPL1      | DPH1    | CMPCR1   | CMPCR2    |
| D8H |      |           |          |           |           |         | ADCCFG   |           |
| D0H | PSW  | T4T3M     | T4H      | T4L       | T3H       | T3L     | T2H      | T2L       |
| C8H | P5   | P5M1      | P5M0     |           |           | SPSTAT  | SPCTL    | SPDAT     |
| C0H |      | WDT_CONTR | IAP_DATA | IAP_ADDRH | IAP_ADDRL | IAP_CMD | IAP_TRIG | IAP_CONTR |
| B8H | IP   | SADEN     | P_SW2    |           | ADC_CONTR | ADC_RES | ADC_RESL |           |
| B0H | P3   | P3M1      | P3M0     |           |           | IP2     | IP2H     | IPH       |
| A8H | IE   | SADDR     | WKTCL    | WKTCH     |           |         | TA       | IE2       |
| A0H | P2   |           | P_SW1    |           |           |         |          |           |
| 98H | SCON | SBUF      | S2CON    | S2BUF     |           | IRCBAND | LIRTRIM  | IRTRIM    |
| 90H | P1   | P1M1      | P1M0     | P0M1      | P0M0      | P2M1    | P2M0     |           |
| 88H | TCON | TMOD      | TL0      | TL1       | TH0       | TH1     | AUXR     | INTCLKO   |
| 80H | P0   | SP        | DPL      | DPH       |           |         |          | PCON      |

↑  
可位寻址

不可位寻址

注意: 寄存器地址能够被 8 整除的才可进行位寻址, 不能被 8 整除的则不可位寻址

|       | 0/8        | 1/9        | 2/A        | 3/B        | 4/C        | 5/D        | 6/E        | 7/F        |
|-------|------------|------------|------------|------------|------------|------------|------------|------------|
| FEF8H | PWMB_CCR6L | PWMB_CCR7H | PWMB_CCR7L | PWMB_CCR8H | PWMB_CCR8L | PWMB_BKR   | PWMB_DTR   | PWMB_OISR  |
| FEF0H | PWMB_PSCRH | PWMB_PSCRL | PWMB_ARRH  | PWMB_ARRL  | PWMB_RCR   | PWMB_CCR5H | PWMB_CCR5L | PWMB_CCR6H |
| FEE8H | PWMB_CCMR1 | PWMB_CCMR2 | PWMB_CCMR3 | PWMB_CCMR4 | PWMB_CCER1 | PWMB_CCER2 | PWMB_CNTRH | PWMB_CNTRL |
| FEE0H | PWMB_CR1   | PWMB_CR2   | PWMB_SMCR  | PWMB_ETR   | PWMB_IER   | PWMB_SR1   | PWMB_SR2   | PWMB_EGR   |
| FED8H | PWMA_CCR2L | PWMA_CCR3H | PWMA_CCR3L | PWMA_CCR4H | PWMA_CCR4L | PWMA_BKR   | PWMA_DTR   | PWMA_OISR  |
| FED0H | PWMA_PSCRH | PWMA_PSCRL | PWMA_ARRH  | PWMA_ARRL  | PWMA_RCR   | PWMA_CCR1H | PWMA_CCR1L | PWMA_CCR2H |
| FEC8H | PWMA_CCMR1 | PWMA_CCMR2 | PWMA_CCMR3 | PWMA_CCMR4 | PWMA_CCER1 | PWMA_CCER2 | PWMA_CNTRH | PWMA_CNTRL |
| FEC0H | PWMA_CR1   | PWMA_CR2   | PWMA_SMCR  | PWMA_ETR   | PWMA_IER   | PWMA_SR1   | PWMA_SR2   | PWMA_EGR   |
| FEB0H | PWMA_ETRPS | PWMA_ENO   | PWMA_PS    | PWMA_IOAUX | PWMB_ETRPS | PWMB_ENO   | PWMB_PS    | PWMB_IOAUX |
| FEA8H | ADCTIM     |            |            |            |            |            |            |            |
| FEA0H |            |            | TM2PS      | TM3PS      | TM4PS      |            |            |            |
| FE88H | I2CMSAUX   |            |            |            |            |            |            |            |
| FE80H | I2CCFG     | I2CMSCR    | I2CMSST    | I2CSLCR    | I2CSLST    | I2CSLADR   | I2CTxD     | I2CRxD     |
| FE30H | P0IE       | P1IE       |            |            |            |            |            |            |
| FE28H | P0DR       | P1DR       | P2DR       | P3DR       |            | P5DR       |            |            |
| FE20H | P0SR       | P1SR       | P2SR       | P3SR       |            | P5SR       |            |            |
| FE18H | P0NCS      | P1NCS      | P2NCS      | P3NCS      |            | P5NCS      |            |            |
| FE10H | P0PU       | P1PU       | P2PU       | P3PU       |            | P5PU       |            |            |
| FE00H | CLKSEL     | CLKDIV     | HIRCCR     | XOSCCR     | IRC32KCR   | MCLKOCR    | IRCDDB     |            |



## 9.3 STC8H3K64S4 系列

|     | 0/8  | 1/9       | 2/A      | 3/B       | 4/C       | 5/D     | 6/E      | 7/F       |
|-----|------|-----------|----------|-----------|-----------|---------|----------|-----------|
| F8H | P7   |           |          |           |           |         |          | RSTCFG    |
| F0H | B    |           |          |           |           | IAP_TPS |          |           |
| E8H | P6   |           |          |           |           |         | IP3H     | AUXINTIF  |
| E0H | ACC  | P7M1      | P7M0     | DPS       | DPL1      | DPH1    | CMPCR1   | CMPCR2    |
| D8H |      |           |          |           |           |         | ADCCFG   | IP3       |
| D0H | PSW  | T4T3M     | T4H      | T4L       | T3H       | T3L     | T2H      | T2L       |
| C8H | P5   | P5M1      | P5M0     | P6M1      | P6M0      | SPSTAT  | SPCTL    | SPDAT     |
| C0H | P4   | WDT_CONTR | IAP_DATA | IAP_ADDRH | IAP_ADDRL | IAP_CMD | IAP_TRIG | IAP_CONTR |
| B8H | IP   | SADEN     | P_SW2    |           | ADC_CONTR | ADC_RES | ADC_RESL |           |
| B0H | P3   | P3M1      | P3M0     | P4M1      | P4M0      | IP2     | IP2H     | IPH       |
| A8H | IE   | SADDR     | WKTCL    | WKTCH     | S3CON     | S3BUF   | TA       | IE2       |
| A0H | P2   | BUS_SPEED | P_SW1    |           |           |         |          |           |
| 98H | SCON | SBUF      | S2CON    | S2BUF     |           | IRCBAND | LIRTRIM  | IRTRIM    |
| 90H | P1   | P1M1      | P1M0     | P0M1      | P0M0      | P2M1    | P2M0     |           |
| 88H | TCON | TMOD      | TL0      | TL1       | TH0       | TH1     | AUXR     | INTCLKO   |
| 80H | P0   | SP        | DPL      | DPH       | S4CON     | S4BUF   |          | PCON      |

↑  
可位寻址

不可位寻址

注意: 寄存器地址能够被 8 整除的才可进行位寻址, 不能被 8 整除的则不可位寻址

|       | 0/8        | 1/9        | 2/A        | 3/B        | 4/C        | 5/D        | 6/E        | 7/F        |
|-------|------------|------------|------------|------------|------------|------------|------------|------------|
| FEF8H | PWMB_CCR6L | PWMB_CCR7H | PWMB_CCR7L | PWMB_CCR8H | PWMB_CCR8L | PWMB_BKR   | PWMB_DTR   | PWMB_OISR  |
| FEF0H | PWMB_PSCRH | PWMB_PSCRL | PWMB_ARRH  | PWMB_ARRL  | PWMB_RCR   | PWMB_CCR5H | PWMB_CCR5L | PWMB_CCR6H |
| FEE8H | PWMB_CCMR1 | PWMB_CCMR2 | PWMB_CCMR3 | PWMB_CCMR4 | PWMB_CCER1 | PWMB_CCER2 | PWMB_CNTRH | PWMB_CNTRL |
| FEE0H | PWMB_CR1   | PWMB_CR2   | PWMB_SMCR  | PWMB_ETR   | PWMB_JER   | PWMB_SR1   | PWMB_SR2   | PWMB_EGR   |
| FED8H | PWMA_CCR2L | PWMA_CCR3H | PWMA_CCR3L | PWMA_CCR4H | PWMA_CCR4L | PWMA_BKR   | PWMA_DTR   | PWMA_OISR  |
| FED0H | PWMA_PSCRH | PWMA_PSCRL | PWMA_ARRH  | PWMA_ARRL  | PWMA_RCR   | PWMA_CCR1H | PWMA_CCR1L | PWMA_CCR2H |
| FEC8H | PWMA_CCMR1 | PWMA_CCMR2 | PWMA_CCMR3 | PWMA_CCMR4 | PWMA_CCER1 | PWMA_CCER2 | PWMA_CNTRH | PWMA_CNTRL |
| FEC0H | PWMA_CR1   | PWMA_CR2   | PWMA_SMCR  | PWMA_ETR   | PWMA_JER   | PWMA_SR1   | PWMA_SR2   | PWMA_EGR   |
| FEB0H | PWMA_ETRPS | PWMA_ENO   | PWMA_PS    | PWMA_IOAUX | PWMB_ETRPS | PWMB_ENO   | PWMB_PS    | PWMB_IOAUX |
| FEA8H | ADCTIM     |            |            |            | T3T4PIN    |            |            |            |
| FEA0H |            |            | TM2PS      | TM3PS      | TM4PS      |            |            |            |
| FE88H | I2CMSAUX   |            |            |            |            |            |            |            |
| FE80H | I2CCFG     | I2CMSCR    | I2CMSST    | I2CSLCR    | I2CSLST    | I2CSLADR   | I2CTxD     | I2CRxD     |
| FE30H | P0IE       | P1IE       |            |            |            |            |            |            |
| FE28H | P0DR       | P1DR       | P2DR       | P3DR       | P4DR       | P5DR       | P6DR       | P7DR       |
| FE20H | P0SR       | P1SR       | P2SR       | P3SR       | P4SR       | P5SR       | P6SR       | P7SR       |
| FE18H | P0NCS      | P1NCS      | P2NCS      | P3NCS      | P4NCS      | P5NCS      | P6NCS      | P7NCS      |
| FE10H | P0PU       | P1PU       | P2PU       | P3PU       | P4PU       | P5PU       | P6PU       | P7PU       |
| FE08H | SPFUNC     | RSTFLAG    |            |            |            |            |            |            |

|       |        |        |        |        |          |         |        |          |
|-------|--------|--------|--------|--------|----------|---------|--------|----------|
| FE00H | CLKSEL | CLKDIV | HIRCCR | XOSCCR | IRC32KCR | MCLKOCR | IRCDB  | IRC48MCR |
| FD40H | P0WKUE | P1WKUE | P2WKUE | P3WKUE | P4WKUE   | P5WKUE  | P6WKUE | P7WKUE   |
| FD30H | P0IM1  | P1IM1  | P2IM1  | P3IM1  | P4IM1    | P5IM1   | P6IM1  | P7IM1    |
| FD20H | P0IM0  | P1IM0  | P2IM0  | P3IM0  | P4IM0    | P5IM0   | P6IM0  | P7IM0    |
| FD10H | P0INTF | P1INTF | P2INTF | P3INTF | P4INTF   | P5INTF  | P6INTF | P7INTF   |
| FD00H | P0INTE | P1INTE | P2INTE | P3INTE | P4INTE   | P5INTE  | P6INTE | P7INTE   |
| FCF0H | MD3    | MD2    | MD1    | MD0    | MD5      | MD4     | ARCON  | OPCON    |

## 9.4 STC8H3K64S2 系列

|     | 0/8  | 1/9       | 2/A      | 3/B       | 4/C       | 5/D     | 6/E      | 7/F       |
|-----|------|-----------|----------|-----------|-----------|---------|----------|-----------|
| F8H | P7   |           |          |           |           |         |          | RSTCFG    |
| F0H | B    |           |          |           |           | IAP_TPS |          |           |
| E8H | P6   |           |          |           |           |         | IP3H     | AUXINTIF  |
| E0H | ACC  | P7M1      | P7M0     | DPS       | DPL1      | DPH1    | CMPCR1   | CMPCR2    |
| D8H |      |           |          |           |           |         | ADCCFG   | IP3       |
| D0H | PSW  | T4T3M     | T4H      | T4L       | T3H       | T3L     | T2H      | T2L       |
| C8H | P5   | P5M1      | P5M0     | P6M1      | P6M0      | SPSTAT  | SPCTL    | SPDAT     |
| C0H | P4   | WDT_CONTR | IAP_DATA | IAP_ADDRH | IAP_ADDRL | IAP_CMD | IAP_TRIG | IAP_CONTR |
| B8H | IP   | SADEN     | P_SW2    |           | ADC_CONTR | ADC_RES | ADC_RESL |           |
| B0H | P3   | P3M1      | P3M0     | P4M1      | P4M0      | IP2     | IP2H     | IPH       |
| A8H | IE   | SADDR     | WKTCL    | WKTCH     |           |         | TA       | IE2       |
| A0H | P2   | BUS_SPEED | P_SW1    |           |           |         |          |           |
| 98H | SCON | SBUF      | S2CON    | S2BUF     |           | IRCBAND | LIRTRIM  | IRTRIM    |
| 90H | P1   | P1M1      | P1M0     | P0M1      | P0M0      | P2M1    | P2M0     |           |
| 88H | TCON | TMOD      | TL0      | TL1       | TH0       | TH1     | AUXR     | INTCLKO   |
| 80H | P0   | SP        | DPL      | DPH       |           |         |          | PCON      |

↑  
可位寻址

不可位寻址

注意: 寄存器地址能够被 8 整除的才可进行位寻址, 不能被 8 整除的则不可位寻址

|       | 0/8        | 1/9        | 2/A        | 3/B        | 4/C        | 5/D        | 6/E        | 7/F        |
|-------|------------|------------|------------|------------|------------|------------|------------|------------|
| FEF8H | PWMB_CCR6L | PWMB_CCR7H | PWMB_CCR7L | PWMB_CCR8H | PWMB_CCR8L | PWMB_BKR   | PWMB_DTR   | PWMB_OISR  |
| FEF0H | PWMB_PSCRH | PWMB_PSCRL | PWMB_ARRH  | PWMB_ARRL  | PWMB_RCR   | PWMB_CCR5H | PWMB_CCR5L | PWMB_CCR6H |
| FEE8H | PWMB_CCMR1 | PWMB_CCMR2 | PWMB_CCMR3 | PWMB_CCMR4 | PWMB_CCER1 | PWMB_CCER2 | PWMB_CNTRH | PWMB_CNTRL |
| FEE0H | PWMB_CR1   | PWMB_CR2   | PWMB_SMCR  | PWMB_ETR   | PWMB_IER   | PWMB_SR1   | PWMB_SR2   | PWMB_EGR   |
| FED8H | PWMA_CCR2L | PWMA_CCR3H | PWMA_CCR3L | PWMA_CCR4H | PWMA_CCR4L | PWMA_BKR   | PWMA_DTR   | PWMA_OISR  |
| FED0H | PWMA_PSCRH | PWMA_PSCRL | PWMA_ARRH  | PWMA_ARRL  | PWMA_RCR   | PWMA_CCR1H | PWMA_CCR1L | PWMA_CCR2H |
| FEC8H | PWMA_CCMR1 | PWMA_CCMR2 | PWMA_CCMR3 | PWMA_CCMR4 | PWMA_CCER1 | PWMA_CCER2 | PWMA_CNTRH | PWMA_CNTRL |
| FEC0H | PWMA_CR1   | PWMA_CR2   | PWMA_SMCR  | PWMA_ETR   | PWMA_IER   | PWMA_SR1   | PWMA_SR2   | PWMA_EGR   |
| FEB0H | PWMA_ETRPS | PWMA_ENO   | PWMA_PS    | PWMA_IOAUX | PWMB_ETRPS | PWMB_ENO   | PWMB_PS    | PWMB_IOAUX |
| FEA8H | ADCTIM     |            |            |            | T3T4PIN    |            |            |            |
| FEA0H |            |            | TM2PS      | TM3PS      | TM4PS      |            |            |            |
| FE88H | I2CMSAUX   |            |            |            |            |            |            |            |
| FE80H | I2CCFG     | I2CMSCR    | I2CMSST    | I2CSLCR    | I2CSLST    | I2CSLADR   | I2CTxD     | I2CRxD     |
| FE30H | P0IE       | P1IE       |            |            |            |            |            |            |
| FE28H | P0DR       | P1DR       | P2DR       | P3DR       | P4DR       | P5DR       | P6DR       | P7DR       |
| FE20H | P0SR       | P1SR       | P2SR       | P3SR       | P4SR       | P5SR       | P6SR       | P7SR       |
| FE18H | P0NCS      | P1NCS      | P2NCS      | P3NCS      | P4NCS      | P5NCS      | P6NCS      | P7NCS      |
| FE10H | P0PU       | P1PU       | P2PU       | P3PU       | P4PU       | P5PU       | P6PU       | P7PU       |
| FE08H | SPFUNC     | RSTFLAG    |            |            |            |            |            |            |

|       |        |        |        |        |          |         |        |          |
|-------|--------|--------|--------|--------|----------|---------|--------|----------|
| FE00H | CLKSEL | CLKDIV | HIRCCR | XOSCCR | IRC32KCR | MCLKOCR | IRCDB  | IRC48MCR |
| FD40H | P0WKUE | P1WKUE | P2WKUE | P3WKUE | P4WKUE   | P5WKUE  | P6WKUE | P7WKUE   |
| FD30H | P0IM1  | P1IM1  | P2IM1  | P3IM1  | P4IM1    | P5IM1   | P6IM1  | P7IM1    |
| FD20H | P0IM0  | P1IM0  | P2IM0  | P3IM0  | P4IM0    | P5IM0   | P6IM0  | P7IM0    |
| FD10H | P0INTF | P1INTF | P2INTF | P3INTF | P4INTF   | P5INTF  | P6INTF | P7INTF   |
| FD00H | P0INTE | P1INTE | P2INTE | P3INTE | P4INTE   | P5INTE  | P6INTE | P7INTE   |
| FCF0H | MD3    | MD2    | MD1    | MD0    | MD5      | MD4     | ARCON  | OPCON    |

## 9.5 STC8H8K64U-64Pin/48Pin USB 系列

|     | 0/8  | 1/9       | 2/A      | 3/B       | 4/C       | 5/D     | 6/E      | 7/F       |
|-----|------|-----------|----------|-----------|-----------|---------|----------|-----------|
| F8H | P7   |           |          |           | USBADR    |         |          | RSTCFG    |
| F0H | B    |           |          |           | USBCON    | IAP_TPS |          |           |
| E8H | P6   |           |          |           | USBDAT    |         | IP3H     | AUXINTIF  |
| E0H | ACC  | P7M1      | P7M0     | DPS       | DPL1      | DPH1    | CMPCR1   | CMPCR2    |
| D8H |      |           |          |           | USBCCLK   |         | ADCCFG   | IP3       |
| D0H | PSW  | T4T3M     | T4H      | T4L       | T3H       | T3L     | T2H      | T2L       |
| C8H | P5   | P5M1      | P5M0     | P6M1      | P6M0      | SPSTAT  | SPCTL    | SPDAT     |
| C0H | P4   | WDT_CONTR | IAP_DATA | IAP_ADDRH | IAP_ADDRL | IAP_CMD | IAP_TRIG | IAP_CONTR |
| B8H | IP   | SADEN     | P_SW2    |           | ADC_CONTR | ADC_RES | ADC_RESL |           |
| B0H | P3   | P3M1      | P3M0     | P4M1      | P4M0      | IP2     | IP2H     | IPH       |
| A8H | IE   | SADDR     | WKTCL    | WKTCH     | S3CON     | S3BUF   | TA       | IE2       |
| A0H | P2   | BUS_SPEED | P_SW1    |           |           |         |          |           |
| 98H | SCON | SBUF      | S2CON    | S2BUF     |           | IRCBAND | LIRTRIM  | IRTRIM    |
| 90H | P1   | P1M1      | P1M0     | P0M1      | P0M0      | P2M1    | P2M0     |           |
| 88H | TCON | TMOD      | TL0      | TL1       | TH0       | TH1     | AUXR     | INTCLKO   |
| 80H | P0   | SP        | DPL      | DPH       | S4CON     | S4BUF   |          | PCON      |

可位寻址

不可位寻址

注意: 寄存器地址能够被 8 整除的才可进行位寻址, 不能被 8 整除的则不可位寻址

|       | 0/8        | 1/9        | 2/A        | 3/B        | 4/C        | 5/D        | 6/E        | 7/F        |
|-------|------------|------------|------------|------------|------------|------------|------------|------------|
| FEF8H | PWMB_CCR6L | PWMB_CCR7H | PWMB_CCR7L | PWMB_CCR8H | PWMB_CCR8L | PWMB_BKR   | PWMB_DTR   | PWMB_OISR  |
| FEF0H | PWMB_PSCRH | PWMB_PSCRL | PWMB_ARRH  | PWMB_ARRL  | PWMB_RCR   | PWMB_CCR5H | PWMB_CCR5L | PWMB_CCR6H |
| FEE8H | PWMB_CCMR1 | PWMB_CCMR2 | PWMB_CCMR3 | PWMB_CCMR4 | PWMB_CCER1 | PWMB_CCER2 | PWMB_CNTRH | PWMB_CNTRL |
| FEE0H | PWMB_CR1   | PWMB_CR2   | PWMB_SMCR  | PWMB_ETR   | PWMB_JER   | PWMB_SR1   | PWMB_SR2   | PWMB_EGR   |
| FED8H | PWMA_CCR2L | PWMA_CCR3H | PWMA_CCR3L | PWMA_CCR4H | PWMA_CCR4L | PWMA_BKR   | PWMA_DTR   | PWMA_OISR  |
| FED0H | PWMA_PSCRH | PWMA_PSCRL | PWMA_ARRH  | PWMA_ARRL  | PWMA_RCR   | PWMA_CCR1H | PWMA_CCR1L | PWMA_CCR2H |
| FEC8H | PWMA_CCMR1 | PWMA_CCMR2 | PWMA_CCMR3 | PWMA_CCMR4 | PWMA_CCER1 | PWMA_CCER2 | PWMA_CNTRH | PWMA_CNTRL |
| FEC0H | PWMA_CR1   | PWMA_CR2   | PWMA_SMCR  | PWMA_ETR   | PWMA_JER   | PWMA_SR1   | PWMA_SR2   | PWMA_EGR   |
| FEB0H | PWMA_ETRPS | PWMA_ENO   | PWMA_PS    | PWMA_IOAUX | PWMB_ETRPS | PWMB_ENO   | PWMB_PS    | PWMB_IOAUX |
| FEA8H | ADCTIM     |            |            |            | T3T4PIN    | ADCEXCFG   | CMPEXCFG   |            |
| FEA0H |            |            | TM2PS      | TM3PS      | TM4PS      |            |            |            |
| FE98H | SPFUNC     | RSTFLAG    |            |            |            |            |            |            |
| FE88H | I2CMSAUX   |            |            |            |            |            |            |            |
| FE80H | I2CCFG     | I2CMSCR    | I2CMSST    | I2CSLCR    | I2CSLST    | I2CSLADR   | I2CTxD     | I2CRxD     |
| FE30H | P0IE       | P1IE       |            |            |            |            |            |            |
| FE28H | P0DR       | P1DR       | P2DR       | P3DR       | P4DR       | P5DR       | P6DR       | P7DR       |
| FE20H | P0SR       | P1SR       | P2SR       | P3SR       | P4SR       | P5SR       | P6SR       | P7SR       |
| FE18H | P0NCS      | P1NCS      | P2NCS      | P3NCS      | P4NCS      | P5NCS      | P6NCS      | P7NCS      |
| FE10H | P0PU       | P1PU       | P2PU       | P3PU       | P4PU       | P5PU       | P6PU       | P7PU       |

|       |        |        |        |        |          |         |       |          |
|-------|--------|--------|--------|--------|----------|---------|-------|----------|
| FE00H | CLKSEL | CLKDIV | HIRCCR | XOSCCR | IRC32KCR | MCLKOCR | IRCDB | IRC48MCR |
| FCF0H | MD3    | MD2    | MD1    | MD0    | MD5      | MD4     | ARCON | OPCON    |

STC MCU

## 9.6 STC8H4K64TL 系列

|     | 0/8  | 1/9       | 2/A      | 3/B       | 4/C       | 5/D     | 6/E      | 7/F       |
|-----|------|-----------|----------|-----------|-----------|---------|----------|-----------|
| F8H |      |           |          |           |           |         |          | RSTCFG    |
| F0H | B    |           |          |           |           | IAP_TPS |          |           |
| E8H |      |           |          |           |           |         | IP3H     | AUXINTIF  |
| E0H | ACC  |           |          | DPS       | DPL1      | DPH1    | CMPCR1   | CMPCR2    |
| D8H |      |           |          |           |           |         | ADCCFG   | IP3       |
| D0H | PSW  | T4T3M     | T4H      | T4L       | T3H       | T3L     | T2H      | T2L       |
| C8H | P5   | P5M1      | P5M0     |           |           | SPSTAT  | SPCTL    | SPDAT     |
| C0H | P4   | WDT_CONTR | IAP_DATA | IAP_ADDRH | IAP_ADDRL | IAP_CMD | IAP_TRIG | IAP_CONTR |
| B8H | IP   | SADEN     | P_SW2    |           | ADC_CONTR | ADC_RES | ADC_RESL |           |
| B0H | P3   | P3M1      | P3M0     | P4M1      | P4M0      | IP2     | IP2H     | IPH       |
| A8H | IE   | SADDR     | WKTCL    | WKTCH     | S3CON     | S3BUF   | TA       | IE2       |
| A0H | P2   | BUS_SPEED | P_SW1    |           |           |         |          |           |
| 98H | SCON | SBUF      | S2CON    | S2BUF     |           | IRCBAND | LIRTRIM  | IRTRIM    |
| 90H | P1   | P1M1      | P1M0     | P0M1      | P0M0      | P2M1    | P2M0     |           |
| 88H | TCON | TMOD      | TL0      | TL1       | TH0       | TH1     | AUXR     | INTCLKO   |
| 80H | P0   | SP        | DPL      | DPH       | S4CON     | S4BUF   |          | PCON      |

↑  
可位寻址

不可位寻址

注意: 寄存器地址能够被 8 整除的才可进行位寻址, 不能被 8 整除的则不可位寻址

|       | 0/8        | 1/9        | 2/A        | 3/B        | 4/C        | 5/D        | 6/E        | 7/F        |
|-------|------------|------------|------------|------------|------------|------------|------------|------------|
| FEF8H | PWMB_CCR6L | PWMB_CCR7H | PWMB_CCR7L | PWMB_CCR8H | PWMB_CCR8L | PWMB_BKR   | PWMB_DTR   | PWMB_OISR  |
| FEF0H | PWMB_PSCRH | PWMB_PSCRL | PWMB_ARRH  | PWMB_ARRL  | PWMB_RCR   | PWMB_CCR5H | PWMB_CCR5L | PWMB_CCR6H |
| FEE8H | PWMB_CCMR1 | PWMB_CCMR2 | PWMB_CCMR3 | PWMB_CCMR4 | PWMB_CCER1 | PWMB_CCER2 | PWMB_CNTRH | PWMB_CNTRL |
| FEE0H | PWMB_CR1   | PWMB_CR2   | PWMB_SMCR  | PWMB_ETR   | PWMB_IER   | PWMB_SR1   | PWMB_SR2   | PWMB_EGR   |
| FED8H | PWMA_CCR2L | PWMA_CCR3H | PWMA_CCR3L | PWMA_CCR4H | PWMA_CCR4L | PWMA_BKR   | PWMA_DTR   | PWMA_OISR  |
| FED0H | PWMA_PSCRH | PWMA_PSCRL | PWMA_ARRH  | PWMA_ARRL  | PWMA_RCR   | PWMA_CCR1H | PWMA_CCR1L | PWMA_CCR2H |
| FEC8H | PWMA_CCMR1 | PWMA_CCMR2 | PWMA_CCMR3 | PWMA_CCMR4 | PWMA_CCER1 | PWMA_CCER2 | PWMA_CNTRH | PWMA_CNTRL |
| FEC0H | PWMA_CR1   | PWMA_CR2   | PWMA_SMCR  | PWMA_ETR   | PWMA_IER   | PWMA_SR1   | PWMA_SR2   | PWMA_EGR   |
| FEB0H | PWMA_ETRPS | PWMA_ENO   | PWMA_PS    | PWMA_IOAUX | PWMB_ETRPS | PWMB_ENO   | PWMB_PS    | PWMB_IOAUX |
| FEA8H | ADCTIM     |            |            |            | T3T4PIN    | ADCEXCFG   | CMPEXCFG   |            |
| FEA0H |            |            | TM2PS      | TM3PS      | TM4PS      |            |            |            |
| FE88H | I2CMSAUX   |            |            |            |            |            |            |            |
| FE80H | I2CCFG     | I2CMSCR    | I2CMSST    | I2CSLCR    | I2CSLST    | I2CSLADR   | I2CTxD     | I2CRxD     |
| FE70H | YEAR       | MONTH      | DAY        | HOURL      | MIN        | SEC        | SSEC       |            |
| FE68H | INIYEAR    | INIMONTH   | INIDAY     | INIHOUR    | INIMIN     | INISEC     | INISSEC    |            |
| FE60H | RTCCR      | RTCCFG     | RTCIEH     | RTCIF      | ALAHOUR    | ALAMIN     | ALASEC     | ALASSEC    |
| FE50H | LCMIFCFG   | LCMIFCFG2  | LCMIFCR    | LCMIFSTA   | LCMIFDATL  | LCMIFDATH  |            |            |
| FE30H | P0IE       | P1IE       |            |            |            | P5IE       |            |            |
| FE28H | P0DR       | P1DR       | P2DR       | P3DR       | P4DR       | P5DR       |            |            |

|       |              |              |               |               |               |               |               |              |
|-------|--------------|--------------|---------------|---------------|---------------|---------------|---------------|--------------|
| FE20H | P0SR         | P1SR         | P2SR          | P3SR          | P4SR          | P5SR          |               |              |
| FE18H | P0NCS        | P1NCS        | P2NCS         | P3NCS         | P4NCS         | P5NCS         |               |              |
| FE10H | P0PU         | P1PU         | P2PU          | P3PU          | P4PU          | P5PU          |               |              |
| FE08H | X32KCR       |              |               |               |               |               |               |              |
| FE00H | CLKSEL       | CLKDIV       | HIRCCR        | XOSCCR        | IRC32KCR      | MCLKOCR       | IRCDB         |              |
| FD60H | PINIPL       | PINIPH       |               |               |               |               |               |              |
| FD40H | P0WKUE       | P1WKUE       | P2WKUE        | P3WKUE        | P4WKUE        | P5WKUE        |               |              |
| FD30H | P0IM1        | P1IM1        | P2IM1         | P3IM1         | P4IM1         | P5IM1         |               |              |
| FD20H | P0IM0        | P1IM0        | P2IM0         | P3IM0         | P4IM0         | P5IM0         |               |              |
| FD10H | P0INTF       | P1INTF       | P2INTF        | P3INTF        | P4INTF        | P5INTF        |               |              |
| FD00H | P0INTE       | P1INTE       | P2INTE        | P3INTE        | P4INTE        | P5INTE        |               |              |
| FCF0H | MD3          | MD2          | MD1           | MD0           | MD5           | MD4           | ARCON         | OPCON        |
| FB68H | TSTH12H      | TSTH12L      | TSTH13H       | TSTH13L       | TSTH14H       | TSTH14L       | TSTH15H       | TSTH15L      |
| FB60H | TSTH08H      | TSTH08L      | TSTH09H       | TSTH09L       | TSTH10H       | TSTH10L       | TSTH11H       | TSTH11L      |
| FB58H | TSTH04H      | TSTH04L      | TSTH05H       | TSTH05L       | TSTH06H       | TSTH06L       | TSTH07H       | TSTH07L      |
| FB50H | TSTH00H      | TSTH00L      | TSTH01H       | TSTH01L       | TSTH02H       | TSTH02L       | TSTH03H       | TSTH03L      |
| FB48H | TSRT         | TSDATH       | TSDATL        |               |               |               |               |              |
| FB40H | TSCHEN1      | TSCHEN2      | TSCFG1        | TSCFG2        | TSWUTC        | TSCTRL        | TSSTA1        | TSSTA2       |
| FB28H | COM0_DC_H    | COM1_DC_H    | COM2_DC_H     | COM3_DC_H     | COM4_DC_H     | COM5_DC_H     | COM6_DC_H     | COM7_DC_H    |
| FB20H | COM0_DC_L    | COM1_DC_L    | COM2_DC_L     | COM3_DC_L     | COM4_DC_L     | COM5_DC_L     | COM6_DC_L     | COM7_DC_L    |
| FB18H | COM0_DA_H    | COM1_DA_H    | COM2_DA_H     | COM3DA_H      | COM4_DA_H     | COM5_DA_H     | COM6_DA_H     | COM7_DA_H    |
| FB10H | COM0_DA_L    | COM1_DA_L    | COM2_DA_L     | COM3DA_L      | COM4_DA_L     | COM5_DA_L     | COM6_DA_L     | COM7_DA_L    |
| FB00H | COMEN        | SEGENL       | SEGENH        | LEDCTRL       | LEDCKS        |               |               |              |
| FA78H | DMA_LCM_RXAL |              |               |               |               |               |               |              |
| FA70H | DMA_LCM_CFG  | DMA_LCM_CR   | DMA_LCM_STA   | DMA_LCM_AMT   | DMA_LCM_DONE  | DMA_LCM_TXAH  | DMA_LCM_TXAL  | DMA_LCM_RXAH |
| FA68H | DMA_UR4R_CFG | DMA_UR4R_CR  | DMA_UR4R_STA  | DMA_UR4R_AMT  | DMA_UR4R_DONE | DMA_UR4R_RXAH | DMA_UR4R_RXAL |              |
| FA60H | DMA_UR4T_CFG | DMA_UR4T_CR  | DMA_UR4T_STA  | DMA_UR4T_AMT  | DMA_UR4T_DONE | DMA_UR4T_TXAH | DMA_UR4T_TXAL |              |
| FA58H | DMA_UR3R_CFG | DMA_UR3R_CR  | DMA_UR3R_STA  | DMA_UR3R_AMT  | DMA_UR3R_DONE | DMA_UR3R_RXAH | DMA_UR3R_RXAL |              |
| FA50H | DMA_UR3T_CFG | DMA_UR3T_CR  | DMA_UR3T_STA  | DMA_UR3T_AMT  | DMA_UR3T_DONE | DMA_UR3T_TXAH | DMA_UR3T_TXAL |              |
| FA48H | DMA_UR2R_CFG | DMA_UR2R_CR  | DMA_UR2R_STA  | DMA_UR2R_AMT  | DMA_UR2R_DONE | DMA_UR2R_RXAH | DMA_UR2R_RXAL |              |
| FA40H | DMA_UR2T_CFG | DMA_UR2T_CR  | DMA_UR2T_STA  | DMA_UR2T_AMT  | DMA_UR2T_DONE | DMA_UR2T_TXAH | DMA_UR2T_TXAL |              |
| FA38H | DMA_UR1R_CFG | DMA_UR1R_CR  | DMA_UR1R_STA  | DMA_UR1R_AMT  | DMA_UR1R_DONE | DMA_UR1R_RXAH | DMA_UR1R_RXAL |              |
| FA30H | DMA_UR1T_CFG | DMA_UR1T_CR  | DMA_UR1T_STA  | DMA_UR1T_AMT  | DMA_UR1T_DONE | DMA_UR1T_TXAH | DMA_UR1T_TXAL |              |
| FA28H | DMA_SPI_RXAL | DMA_SPI_CFG2 |               |               |               |               |               |              |
| FA20H | DMA_SPI_CFG  | DMA_SPI_CR   | DMA_SPI_STA   | DMA_SPI_AMT   | DMA_SPI_DONE  | DMA_SPI_TXAH  | DMA_SPI_TXAL  | DMA_SPI_RXAH |
| FA18H | DMA_ADC_RXAL | DMA_ADC_CFG2 | DMA_ADC_CHSW0 | DMA_ADC_CHSW1 |               |               |               |              |
| FA10H | DMA_ADC_CFG  | DMA_ADC_CR   | DMA_ADC_STA   |               |               |               |               | DMA_ADC_RXAH |
| FA08H | DMA_M2M_RXAL |              |               |               |               |               |               |              |
| FA00H | DMA_M2M_CFG  | DMA_M2M_CR   | DMA_M2M_STA   | DMA_M2M_AMT   | DMA_M2M_DONE  | DMA_M2M_TXAH  | DMA_M2M_TXAL  | DMA_M2M_RXAH |



## 9.7 STC8H4K64TLCD 系列

|     | 0/8  | 1/9       | 2/A      | 3/B       | 4/C       | 5/D     | 6/E      | 7/F       |
|-----|------|-----------|----------|-----------|-----------|---------|----------|-----------|
| F8H | P7   |           |          |           |           |         |          | RSTCFG    |
| F0H | B    |           |          |           |           | IAP_TPS |          |           |
| E8H | P6   |           |          |           |           |         | IP3H     | AUXINTIF  |
| E0H | ACC  | P7M1      | P7M0     | DPS       | DPL1      | DPH1    | CMPCR1   | CMPCR2    |
| D8H |      |           |          |           |           |         | ADCCFG   | IP3       |
| D0H | PSW  | T4T3M     | T4H      | T4L       | T3H       | T3L     | T2H      | T2L       |
| C8H | P5   | P5M1      | P5M0     | P6M1      | P6M0      | SPSTAT  | SPCTL    | SPDAT     |
| C0H | P4   | WDT_CONTR | IAP_DATA | IAP_ADDRH | IAP_ADDRL | IAP_CMD | IAP_TRIG | IAP_CONTR |
| B8H | IP   | SADEN     | P_SW2    |           | ADC_CONTR | ADC_RES | ADC_RESL |           |
| B0H | P3   | P3M1      | P3M0     | P4M1      | P4M0      | IP2     | IP2H     | IPH       |
| A8H | IE   | SADDR     | WKTCL    | WKTCH     | S3CON     | S3BUF   | TA       | IE2       |
| A0H | P2   | BUS_SPEED | P_SW1    |           |           |         |          |           |
| 98H | SCON | SBUF      | S2CON    | S2BUF     |           | IRCBAND | LIRTRIM  | IRTRIM    |
| 90H | P1   | P1M1      | P1M0     | P0M1      | P0M0      | P2M1    | P2M0     |           |
| 88H | TCON | TMOD      | TL0      | TL1       | TH0       | TH1     | AUXR     | INTCLKO   |
| 80H | P0   | SP        | DPL      | DPH       | S4CON     | S4BUF   |          | PCON      |

↑  
可位寻址

不可位寻址

注意: 寄存器地址能够被 8 整除的才可进行位寻址, 不能被 8 整除的则不可位寻址

|       | 0/8        | 1/9        | 2/A        | 3/B        | 4/C        | 5/D        | 6/E        | 7/F        |
|-------|------------|------------|------------|------------|------------|------------|------------|------------|
| FEF8H | PWMB_CCR6L | PWMB_CCR7H | PWMB_CCR7L | PWMB_CCR8H | PWMB_CCR8L | PWMB_BKR   | PWMB_DTR   | PWMB_OISR  |
| FEF0H | PWMB_PSCRH | PWMB_PSCRL | PWMB_ARRH  | PWMB_ARRL  | PWMB_RCR   | PWMB_CCR5H | PWMB_CCR5L | PWMB_CCR6H |
| FEE8H | PWMB_CCMR1 | PWMB_CCMR2 | PWMB_CCMR3 | PWMB_CCMR4 | PWMB_CCER1 | PWMB_CCER2 | PWMB_CNTRH | PWMB_CNTRL |
| FEE0H | PWMB_CR1   | PWMB_CR2   | PWMB_SMC   | PWMB_ETR   | PWMB_IER   | PWMB_SR1   | PWMB_SR2   | PWMB_EGR   |
| FED8H | PWMA_CCR2L | PWMA_CCR3H | PWMA_CCR3L | PWMA_CCR4H | PWMA_CCR4L | PWMA_BKR   | PWMA_DTR   | PWMA_OISR  |
| FED0H | PWMA_PSCRH | PWMA_PSCRL | PWMA_ARRH  | PWMA_ARRL  | PWMA_RCR   | PWMA_CCR1H | PWMA_CCR1L | PWMA_CCR2H |
| FEC8H | PWMA_CCMR1 | PWMA_CCMR2 | PWMA_CCMR3 | PWMA_CCMR4 | PWMA_CCER1 | PWMA_CCER2 | PWMA_CNTRH | PWMA_CNTRL |
| FEC0H | PWMA_CR1   | PWMA_CR2   | PWMA_SMC   | PWMA_ETR   | PWMA_IER   | PWMA_SR1   | PWMA_SR2   | PWMA_EGR   |
| FEB0H | PWMA_ETRPS | PWMA_ENO   | PWMA_PS    | PWMA_IOAUX | PWMB_ETRPS | PWMB_ENO   | PWMB_PS    | PWMB_IOAUX |
| FEA8H | ADCTIM     |            |            |            | T3T4PIN    | ADCEXCFG   | CMPEXCFG   |            |
| FEA0H |            |            | TM2PS      | TM3PS      | TM4PS      |            |            |            |
| FE88H | I2CMSAUX   |            |            |            |            |            |            |            |
| FE80H | I2CCFG     | I2CMSCR    | I2CMSST    | I2CSLCR    | I2CSLST    | I2CSLADR   | I2CTxD     | I2CRxD     |
| FE70H | YEAR       | MONTH      | DAY        | HOURL      | MIN        | SEC        | SSEC       |            |
| FE68H | INIYEAR    | INIMONTH   | INIDAY     | INIHOUR    | INIMIN     | INISEC     | INISSEC    |            |
| FE60H | RTCCR      | RTCCFG     | RTCEN      | RTCIF      | ALAHOUR    | ALAMIN     | ALASEC     | ALASSEC    |
| FE50H | LCMIFCFG   | LCMIFCFG2  | LCMIFCR    | LCMIFSTA   | LCMIFDATH  | LCMIFDATH  |            |            |
| FE30H | P0IE       | P1IE       | P2IE       | P3IE       | P4IE       | P5IE       | P6IE       | P7IE       |
| FE28H | P0DR       | P1DR       | P2DR       | P3DR       | P4DR       | P5DR       | P6DR       | P7DR       |

|       |              |              |               |               |               |               |               |              |
|-------|--------------|--------------|---------------|---------------|---------------|---------------|---------------|--------------|
| FE20H | P0SR         | P1SR         | P2SR          | P3SR          | P4SR          | P5SR          | P6SR          | P7SR         |
| FE18H | P0NCS        | P1NCS        | P2NCS         | P3NCS         | P4NCS         | P5NCS         | P6NCS         | P7NCS        |
| FE10H | P0PU         | P1PU         | P2PU          | P3PU          | P4PU          | P5PU          | P6PU          | P7PU         |
| FE08H | X32KCR       |              |               |               |               |               |               |              |
| FE00H | CLKSEL       | CLKDIV       | HIRCCR        | XOSCCR        | IRC32KCR      | MCLKOCR       | IRCDDB        |              |
| FD60H | PINIPL       | PINIPH       |               |               |               |               |               |              |
| FD40H | P0WKUE       | P1WKUE       | P2WKUE        | P3WKUE        | P4WKUE        | P5WKUE        | P6WKUE        | P7WKUE       |
| FD30H | P0IM1        | P1IM1        | P2IM1         | P3IM1         | P4IM1         | P5IM1         | P6IM1         | P7IM1        |
| FD20H | P0IM0        | P1IM0        | P2IM0         | P3IM0         | P4IM0         | P5IM0         | P6IM0         | P7IM0        |
| FD10H | P0INTF       | P1INTF       | P2INTF        | P3INTF        | P4INTF        | P5INTF        | P6INTF        | P7INTF       |
| FD00H | P0INTE       | P1INTE       | P2INTE        | P3INTE        | P4INTE        | P5INTE        | P6INTE        | P7INTE       |
| FCF0H | MD3          | MD2          | MD1           | MD0           | MD5           | MD4           | ARCON         | OPCON        |
| FBA8H | C3SEGV0      | C3SEGV1      | C3SEGV2       | C3SEGV3       | C3SEGV4       |               |               |              |
| FBA0H | C2SEGV0      | C2SEGV1      | C2SEGV2       | C2SEGV3       | C2SEGV4       |               |               |              |
| FB98H | C1SEGV0      | C1SEGV1      | C1SEGV2       | C1SEGV3       | C1SEGV4       |               |               |              |
| FB90H | C0SEGV0      | C0SEGV1      | C0SEGV2       | C0SEGV3       | C0SEGV4       |               |               |              |
| FB88H | COMON        |              | SEGON1        | SEGON2        | SEGON3        | SEGON4        | SEGON5        |              |
| FB80H | LCDCFG       | LCDCFG2      | DBLEN         | COMLENL       | COMLENM       | COMLENH       | BLINKRATE     | LCDCR        |
| FB68H | TSTH12H      | TSTH12L      | TSTH13H       | TSTH13L       | TSTH14H       | TSTH14L       | TSTH15H       | TSTH15L      |
| FB60H | TSTH08H      | TSTH08L      | TSTH09H       | TSTH09L       | TSTH10H       | TSTH10L       | TSTH11H       | TSTH11L      |
| FB58H | TSTH04H      | TSTH04L      | TSTH05H       | TSTH05L       | TSTH06H       | TSTH06L       | TSTH07H       | TSTH07L      |
| FB50H | TSTH00H      | TSTH00L      | TSTH01H       | TSTH01L       | TSTH02H       | TSTH02L       | TSTH03H       | TSTH03L      |
| FB48H | TSRT         | TSDATH       | TSDATL        |               |               |               |               |              |
| FB40H | TSCHEN1      | TSCHEN2      | TSCFG1        | TSCFG2        | TSWUTC        | TSCTRL        | TSSTA1        | TSSTA2       |
| FA78H | DMA_LCM_RXAL |              |               |               |               |               |               |              |
| FA70H | DMA_LCM_CFG  | DMA_LCM_CR   | DMA_LCM_STA   | DMA_LCM_AMT   | DMA_LCM_DONE  | DMA_LCM_TXAH  | DMA_LCM_TXAL  | DMA_LCM_RXAH |
| FA68H | DMA_UR4R_CFG | DMA_UR4R_CR  | DMA_UR4R_STA  | DMA_UR4R_AMT  | DMA_UR4R_DONE | DMA_UR4R_RXAH | DMA_UR4R_RXAL |              |
| FA60H | DMA_UR4T_CFG | DMA_UR4T_CR  | DMA_UR4T_STA  | DMA_UR4T_AMT  | DMA_UR4T_DONE | DMA_UR4T_TXAH | DMA_UR4T_TXAL |              |
| FA58H | DMA_UR3R_CFG | DMA_UR3R_CR  | DMA_UR3R_STA  | DMA_UR3R_AMT  | DMA_UR3R_DONE | DMA_UR3R_RXAH | DMA_UR3R_RXAL |              |
| FA50H | DMA_UR3T_CFG | DMA_UR3T_CR  | DMA_UR3T_STA  | DMA_UR3T_AMT  | DMA_UR3T_DONE | DMA_UR3T_TXAH | DMA_UR3T_TXAL |              |
| FA48H | DMA_UR2R_CFG | DMA_UR2R_CR  | DMA_UR2R_STA  | DMA_UR2R_AMT  | DMA_UR2R_DONE | DMA_UR2R_RXAH | DMA_UR2R_RXAL |              |
| FA40H | DMA_UR2T_CFG | DMA_UR2T_CR  | DMA_UR2T_STA  | DMA_UR2T_AMT  | DMA_UR2T_DONE | DMA_UR2T_TXAH | DMA_UR2T_TXAL |              |
| FA38H | DMA_UR1R_CFG | DMA_UR1R_CR  | DMA_UR1R_STA  | DMA_UR1R_AMT  | DMA_UR1R_DONE | DMA_UR1R_RXAH | DMA_UR1R_RXAL |              |
| FA30H | DMA_UR1T_CFG | DMA_UR1T_CR  | DMA_UR1T_STA  | DMA_UR1T_AMT  | DMA_UR1T_DONE | DMA_UR1T_TXAH | DMA_UR1T_TXAL |              |
| FA28H | DMA_SPI_RXAL | DMA_SPI_CFG2 |               |               |               |               |               |              |
| FA20H | DMA_SPI_CFG  | DMA_SPI_CR   | DMA_SPI_STA   | DMA_SPI_AMT   | DMA_SPI_DONE  | DMA_SPI_TXAH  | DMA_SPI_TXAL  | DMA_SPI_RXAH |
| FA18H | DMA_ADC_RXAL | DMA_ADC_CFG2 | DMA_ADC_CHSW0 | DMA_ADC_CHSW1 |               |               |               |              |
| FA10H | DMA_ADC_CFG  | DMA_ADC_CR   | DMA_ADC_STA   |               |               |               |               | DMA_ADC_RXAH |
| FA08H | DMA_M2M_RXAL |              |               |               |               |               |               |              |
| FA00H | DMA_M2M_CFG  | DMA_M2M_CR   | DMA_M2M_STA   | DMA_M2M_AMT   | DMA_M2M_DONE  | DMA_M2M_TXAH  | DMA_M2M_TXAL  | DMA_M2M_RXAH |

## 9.8 STC8H1K08T 系列

|     | 0/8  | 1/9       | 2/A      | 3/B       | 4/C       | 5/D     | 6/E      | 7/F       |
|-----|------|-----------|----------|-----------|-----------|---------|----------|-----------|
| F8H |      |           |          |           |           |         |          | RSTCFG    |
| F0H | B    |           |          |           |           | IAP_TPS |          |           |
| E8H |      |           |          |           |           |         | IP3H     | AUXINTIF  |
| E0H | ACC  |           |          | DPS       | DPL1      | DPH1    | CMPCR1   | CMPCR2    |
| D8H |      |           |          |           | PLLCR     |         | ADCCFG   | IP3       |
| D0H | PSW  |           |          |           |           |         | T2H      | T2L       |
| C8H | P5   | P5M1      | P5M0     |           |           | SPSTAT  | SPCTL    | SPDAT     |
| C0H |      | WDT_CONTR | IAP_DATA | IAP_ADDRH | IAP_ADDRL | IAP_CMD | IAP_TRIG | IAP_CONTR |
| B8H | IP   | SADEN     | P_SW2    |           | ADC_CONTR | ADC_RES | ADC_RESL |           |
| B0H | P3   | P3M1      | P3M0     |           |           | IP2     | IP2H     | IPH       |
| A8H | IE   | SADDR     | WKTCL    | WKTCH     |           |         | TA       | IE2       |
| A0H |      | BUS_SPEED | P_SW1    |           |           |         |          |           |
| 98H | SCON | SBUF      | S2CON    | S2BUF     |           | IRCBAND | LIRTRIM  | IRTRIM    |
| 90H | P1   | P1M1      | P1M0     |           |           |         |          |           |
| 88H | TCON | TMOD      | TL0      | TL1       | TH0       | TH1     | AUXR     | INTCLKO   |
| 80H |      | SP        | DPL      | DPH       |           |         |          | PCON      |

↑  
可位寻址

不可位寻址

注意: 寄存器地址能够被 8 整除的才可进行位寻址, 不能被 8 整除的则不可位寻址

|       | 0/8        | 1/9        | 2/A        | 3/B        | 4/C        | 5/D        | 6/E        | 7/F        |
|-------|------------|------------|------------|------------|------------|------------|------------|------------|
| FEF8H | PWMB_CCR6L | PWMB_CCR7H | PWMB_CCR7L | PWMB_CCR8H | PWMB_CCR8L | PWMB_BKR   | PWMB_DTR   | PWMB_OISR  |
| FEF0H | PWMB_PSCRH | PWMB_PSCRL | PWMB_ARRH  | PWMB_ARRL  | PWMB_RCR   | PWMB_CCR5H | PWMB_CCR5L | PWMB_CCR6H |
| FEE8H | PWMB_CCMR1 | PWMB_CCMR2 | PWMB_CCMR3 | PWMB_CCMR4 | PWMB_CCER1 | PWMB_CCER2 | PWMB_CNTRH | PWMB_CNTRL |
| FEE0H | PWMB_CR1   | PWMB_CR2   | PWMB_SMCRL | PWMB_ETR   | PWMB_IER   | PWMB_SR1   | PWMB_SR2   | PWMB_EGR   |
| FED8H | PWMA_CCR2L | PWMA_CCR3H | PWMA_CCR3L | PWMA_CCR4H | PWMA_CCR4L | PWMA_BKR   | PWMA_DTR   | PWMA_OISR  |
| FED0H | PWMA_PSCRH | PWMA_PSCRL | PWMA_ARRH  | PWMA_ARRL  | PWMA_RCR   | PWMA_CCR1H | PWMA_CCR1L | PWMA_CCR2H |
| FEC8H | PWMA_CCMR1 | PWMA_CCMR2 | PWMA_CCMR3 | PWMA_CCMR4 | PWMA_CCER1 | PWMA_CCER2 | PWMA_CNTRH | PWMA_CNTRL |
| FEC0H | PWMA_CR1   | PWMA_CR2   | PWMA_SMCRL | PWMA_ETR   | PWMA_IER   | PWMA_SR1   | PWMA_SR2   | PWMA_EGR   |
| FEB0H | PWMA_ETRPS | PWMA_ENO   | PWMA_PS    | PWMA_IOAUX | PWMB_ETRPS | PWMB_ENO   | PWMB_PS    | PWMB_IOAUX |
| FEA8H | ADCTIM     |            |            |            | T3T4PIN    | ADCEXCFG   | CMPEXCFG   |            |
| FEA0H |            |            | TM2PS      |            |            |            |            |            |
| FE88H | I2CMSAUX   |            |            |            |            |            |            |            |
| FE80H | I2CCFG     | I2CMSCR    | I2CMSST    | I2CSLCR    | I2CSLST    | I2CSLADR   | I2CTxD     | I2CRxD     |
| FE70H | YEAR       | MONTH      | DAY        | HOURL      | MIN        | SEC        | SSEC       |            |
| FE68H | INIYEAR    | INIMONTH   | INIDAY     | INIHOUR    | INIMIN     | INISEC     | INISSEC    |            |
| FE60H | RTCCR      | RTCCFG     | RTCEN      | RTCIF      | ALAHOUR    | ALAMIN     | ALASEC     | ALASSEC    |
| FE30H |            | P1IE       |            | P3IE       |            | P5IE       |            |            |
| FE28H |            | P1DR       |            | P3DR       |            | P5DR       |            |            |
| FE20H |            | P1SR       |            | P3SR       |            | P5SR       |            |            |

|       |              |              |               |               |               |               |               |              |
|-------|--------------|--------------|---------------|---------------|---------------|---------------|---------------|--------------|
| FE18H |              | P1NCS        |               | P3NCS         |               | P5NCS         |               |              |
| FE10H |              | P1PU         |               | P3PU          |               | P5PU          |               |              |
| FE08H | X32KCR       |              |               |               |               |               |               |              |
| FE00H | CLKSEL       | CLKDIV       | HIRCCR        | XOSCCR        | IRC32KCR      | MCLKOCR       | IRCDDB        |              |
| FD60H | PINIPL       | PINIPH       |               |               |               |               |               |              |
| FD40H |              | P1WKUE       |               | P3WKUE        |               | P5WKUE        |               |              |
| FD30H |              | P1IM1        |               | P3IM1         |               | P5IM1         |               |              |
| FD20H |              | P1IM0        |               | P3IM0         |               | P5IM0         |               |              |
| FD10H |              | P1INTF       |               | P3INTF        |               | P5INTF        |               |              |
| FD00H |              | P1INTE       |               | P3INTE        |               | P5INTE        |               |              |
| FCF0H | MD3          | MD2          | MD1           | MD0           | MD5           | MD4           | ARCON         | OPCON        |
| FBF0H | HSPWMA_CFG   | HSPWMA_ADR   | HSPWMA_DAT    |               | HSPWMB_CFG    | HSPWMB_ADR    | HSPWMB_DAT    |              |
| FBA8H |              |              |               |               |               |               |               |              |
| FBA0H |              |              |               |               |               |               |               |              |
| FB98H |              |              |               |               |               |               |               |              |
| FB90H |              |              |               |               |               |               |               |              |
| FB88H |              |              |               |               |               |               |               |              |
| FB80H |              |              |               |               |               |               |               |              |
| FB68H | TSTH12H      | TSTH12L      | TSTH13H       | TSTH13L       | TSTH14H       | TSTH14L       | TSTH15H       | TSTH15L      |
| FB60H | TSTH08H      | TSTH08L      | TSTH09H       | TSTH09L       | TSTH10H       | TSTH10L       | TSTH11H       | TSTH11L      |
| FB58H | TSTH04H      | TSTH04L      | TSTH05H       | TSTH05L       | TSTH06H       | TSTH06L       | TSTH07H       | TSTH07L      |
| FB50H | TSTH00H      | TSTH00L      | TSTH01H       | TSTH01L       | TSTH02H       | TSTH02L       | TSTH03H       | TSTH03L      |
| FB48H | TSRT         | TSDATH       | TSDATL        |               |               |               |               |              |
| FB40H | TSCHEN1      | TSCHEN2      | TSCFG1        | TSCFG2        | TSWUTC        | TSCTRL        | TSSTA1        | TSSTA2       |
| FA78H | DMA_LCM_RXAL |              |               |               |               |               |               |              |
| FA70H | DMA_LCM_CFG  | DMA_LCM_CR   | DMA_LCM_STA   | DMA_LCM_AMT   | DMA_LCM_DONE  | DMA_LCM_TXAH  | DMA_LCM_TXAL  | DMA_LCM_RXAH |
| FA68H | DMA_UR4R_CFG | DMA_UR4R_CR  | DMA_UR4R_STA  | DMA_UR4R_AMT  | DMA_UR4R_DONE | DMA_UR4R_RXAH | DMA_UR4R_RXAL |              |
| FA60H | DMA_UR4T_CFG | DMA_UR4T_CR  | DMA_UR4T_STA  | DMA_UR4T_AMT  | DMA_UR4T_DONE | DMA_UR4T_TXAH | DMA_UR4T_TXAL |              |
| FA58H | DMA_UR3R_CFG | DMA_UR3R_CR  | DMA_UR3R_STA  | DMA_UR3R_AMT  | DMA_UR3R_DONE | DMA_UR3R_RXAH | DMA_UR3R_RXAL |              |
| FA50H | DMA_UR3T_CFG | DMA_UR3T_CR  | DMA_UR3T_STA  | DMA_UR3T_AMT  | DMA_UR3T_DONE | DMA_UR3T_TXAH | DMA_UR3T_TXAL |              |
| FA48H | DMA_UR2R_CFG | DMA_UR2R_CR  | DMA_UR2R_STA  | DMA_UR2R_AMT  | DMA_UR2R_DONE | DMA_UR2R_RXAH | DMA_UR2R_RXAL |              |
| FA40H | DMA_UR2T_CFG | DMA_UR2T_CR  | DMA_UR2T_STA  | DMA_UR2T_AMT  | DMA_UR2T_DONE | DMA_UR2T_TXAH | DMA_UR2T_TXAL |              |
| FA38H | DMA_UR1R_CFG | DMA_UR1R_CR  | DMA_UR1R_STA  | DMA_UR1R_AMT  | DMA_UR1R_DONE | DMA_UR1R_RXAH | DMA_UR1R_RXAL |              |
| FA30H | DMA_UR1T_CFG | DMA_UR1T_CR  | DMA_UR1T_STA  | DMA_UR1T_AMT  | DMA_UR1T_DONE | DMA_UR1T_TXAH | DMA_UR1T_TXAL |              |
| FA28H | DMA_SPI_RXAL | DMA_SPI_CFG2 |               |               |               |               |               |              |
| FA20H | DMA_SPI_CFG  | DMA_SPI_CR   | DMA_SPI_STA   | DMA_SPI_AMT   | DMA_SPI_DONE  | DMA_SPI_TXAH  | DMA_SPI_TXAL  | DMA_SPI_RXAH |
| FA18H | DMA_ADC_RXAL | DMA_ADC_CFG2 | DMA_ADC_CHSW0 | DMA_ADC_CHSW1 |               |               |               |              |
| FA10H | DMA_ADC_CFG  | DMA_ADC_CR   | DMA_ADC_STA   |               |               |               |               | DMA_ADC_RXAH |
| FA08H | DMA_M2M_RXAL |              |               |               |               |               |               |              |
| FA00H | DMA_M2M_CFG  | DMA_M2M_CR   | DMA_M2M_STA   | DMA_M2M_AMT   | DMA_M2M_DONE  | DMA_M2M_TXAH  | DMA_M2M_TXAL  | DMA_M2M_RXAH |

## 9.9 特殊功能寄存器列表

注意：寄存器地址能够被 8 整除的才可进行位寻址，不能被 8 整除的则不可位寻址。

STC8H 能进行位寻址的寄存器：P0（80H）、TCON（88H）、P1（90H）、SCON（98H）、P2（A0H）、IE（A8H）、P3（B0H）、IP（B8H）、P4（C0H）、P5（C8H）、PSW（D0H）、ACC（E0H）、P6（E8H）、B（F0H）、P7（F8H）

| 符号        | 描述             | 地址  | 位地址与符号      |       |           |       |            |            |              |        | 复位值       |
|-----------|----------------|-----|-------------|-------|-----------|-------|------------|------------|--------------|--------|-----------|
|           |                |     | B7          | B6    | B5        | B4    | B3         | B2         | B1           | B0     |           |
| P0        | P0 端口          | 80H | P07         | P06   | P05       | P04   | P03        | P02        | P01          | P00    | 1111,1111 |
| SP        | 堆栈指针           | 81H |             |       |           |       |            |            |              |        | 0000,0111 |
| DPL       | 数据指针（低字节）      | 82H |             |       |           |       |            |            |              |        | 0000,0000 |
| DPH       | 数据指针（高字节）      | 83H |             |       |           |       |            |            |              |        | 0000,0000 |
| S4CON     | 串口 4 控制寄存器     | 84H | S4SM0       | S4ST4 | S4SM2     | S4REN | S4TB8      | S4RB8      | S4TI         | S4RI   | 0000,0000 |
| S4BUF     | 串口 4 数据寄存器     | 85H |             |       |           |       |            |            |              |        | 0000,0000 |
| PCON      | 电源控制寄存器        | 87H | SMOD        | SMOD0 | LVDF      | POF   | GF1        | GF0        | PD           | IDL    | 0011,0000 |
| TCON      | 定时器控制寄存器       | 88H | TF1         | TR1   | TF0       | TR0   | IE1        | IT1        | IE0          | IT0    | 0000,0000 |
| TMOD      | 定时器模式寄存器       | 89H | GATE        | C/T   | M1        | M0    | GATE       | C/T        | M1           | M0     | 0000,0000 |
| TL0       | 定时器 0 低 8 位寄存器 | 8AH |             |       |           |       |            |            |              |        | 0000,0000 |
| TL1       | 定时器 1 低 8 位寄存器 | 8BH |             |       |           |       |            |            |              |        | 0000,0000 |
| TH0       | 定时器 0 高 8 位寄存器 | 8CH |             |       |           |       |            |            |              |        | 0000,0000 |
| TH1       | 定时器 1 高 8 位寄存器 | 8DH |             |       |           |       |            |            |              |        | 0000,0000 |
| AUXR      | 辅助寄存器 1        | 8EH | T0x12       | T1x12 | UART_M0x6 | T2R   | T2_C/T     | T2x12      | EXTRAM       | S1ST2  | 0000,0001 |
| INTCLKO   | 中断与时钟输出控制寄存器   | 8FH | -           | EX4   | EX3       | EX2   | -          | T2CLKO     | T1CLKO       | T0CLKO | x000,x000 |
| P1        | P1 端口          | 90H | P17         | P16   | P15       | P14   | P13        | P12        | P11          | P10    | 1111,1111 |
| P1M1      | P1 口配置寄存器 1    | 91H | P17M1       | P16M1 | P15M1     | P14M1 | P13M1      | P12M1      | P11M1        | P10M1  | 1111,1111 |
| P1M0      | P1 口配置寄存器 0    | 92H | P17M0       | P16M0 | P15M0     | P14M0 | P13M0      | P12M0      | P11M0        | P10M0  | 0000,0000 |
| P0M1      | P0 口配置寄存器 1    | 93H | P07M1       | P06M1 | P05M1     | P04M1 | P03M1      | P02M1      | P01M1        | P00M1  | 1111,1111 |
| P0M0      | P0 口配置寄存器 0    | 94H | P07M0       | P06M0 | P05M0     | P04M0 | P03M0      | P02M0      | P01M0        | P00M0  | 0000,0000 |
| P2M1      | P2 口配置寄存器 1    | 95H | P27M1       | P26M1 | P25M1     | P24M1 | P23M1      | P22M1      | P21M1        | P20M1  | 1111,1111 |
| P2M0      | P2 口配置寄存器 0    | 96H | P27M0       | P26M0 | P25M0     | P24M0 | P23M0      | P22M0      | P21M0        | P20M0  | 0000,0000 |
| SCON      | 串口 1 控制寄存器     | 98H | SM0/FE      | SM1   | SM2       | REN   | TB8        | RB8        | TI           | RI     | 0000,0000 |
| SBUF      | 串口 1 数据寄存器     | 99H |             |       |           |       |            |            |              |        | 0000,0000 |
| S2CON     | 串口 2 控制寄存器     | 9AH | S2SM0       | -     | S2SM2     | S2REN | S2TB8      | S2RB8      | S2TI         | S2RI   | 0100,0000 |
| S2BUF     | 串口 2 数据寄存器     | 9BH |             |       |           |       |            |            |              |        | 0000,0000 |
| IRCBAND   | IRC 频段选择检测     | 9DH | -           | -     | -         | -     | -          | -          | SEL[1:0]     |        | xxxx,xxnn |
| LIRTRIM   | IRC 频率微调寄存器    | 9EH | -           | -     | -         | -     | -          | -          | LIRTRIM[1:0] |        | xxxx,xxnn |
| IRTRIM    | IRC 频率调整寄存器    | 9FH | IRTRIM[7:0] |       |           |       |            |            |              |        | nnnn,nnnn |
| P2        | P2 端口          | A0H | P27         | P26   | P25       | P24   | P23        | P22        | P21          | P20    | 1111,1111 |
| BUS_SPEED | 总线速度控制寄存器      | A1H | RW_S[1:0]   |       |           |       |            | SPEED[2:0] |              |        | 00xx,x000 |
| P_SW1     | 外设端口切换寄存器 1    | A2H | S1_S[1:0]   |       |           |       | SPI_S[1:0] |            | 0            | -      | nn00,000x |
| IE        | 中断允许寄存器        | A8H | EA          | ELVD  | EADC      | ES    | ET1        | EX1        | ET0          | EX0    | 0000,0000 |
| SADDR     | 串口 1 从机地址寄存器   | A9H |             |       |           |       |            |            |              |        | 0000,0000 |
| WKTCL     | 掉电唤醒定时器低字节     | AAH |             |       |           |       |            |            |              |        | 1111,1111 |

|           |                |     |                 |           |            |           |              |             |          |        |           |
|-----------|----------------|-----|-----------------|-----------|------------|-----------|--------------|-------------|----------|--------|-----------|
| WKTCH     | 掉电唤醒定时器高字节     | ABH | WKTEN           |           |            |           |              |             |          |        | 0111,1111 |
| S3CON     | 串口 3 控制寄存器     | ACH | S3SM0           | S3ST4     | S3SM2      | S3REN     | S3TB8        | S3RB8       | S3TI     | S3RI   | 0000,0000 |
| S3BUF     | 串口 3 数据寄存器     | ADH |                 |           |            |           |              |             |          |        | 0000,0000 |
| TA        | DPTR 时序控制寄存器   | AEH |                 |           |            |           |              |             |          |        | 0000,0000 |
| IE2       | 中断允许寄存器 2      | AFH | EUSB<br>ETKSU   | ET4       | ET3        | ES4       | ES3          | ET2         | ESPI     | ES2    | 0000,0000 |
| P3        | P3 端口          | B0H | P37             | P36       | P35        | P34       | P33          | P32         | P31      | P30    | 1111,1111 |
| P3M1      | P3 口配置寄存器 1    | B1H | P37M1           | P36M1     | P35M1      | P34M1     | P33M1        | P32M1       | P31M1    | P30M1  | 1111,1100 |
| P3M0      | P3 口配置寄存器 0    | B2H | P37M0           | P36M0     | P35M0      | P34M0     | P33M0        | P32M0       | P31M0    | P30M0  | 0000,0000 |
| P4M1      | P4 口配置寄存器 1    | B3H | P47M1           | P46M1     | P45M1      | P44M1     | P43M1        | P42M1       | P41M1    | P40M1  | 1111,1111 |
| P4M0      | P4 口配置寄存器 0    | B4H | P47M0           | P46M0     | P45M0      | P44M0     | P43M0        | P42M0       | P41M0    | P40M0  | 0000,0000 |
| IP2       | 中断优先级控制寄存器 2   | B5H | PUSB<br>PTKSU   | PI2C      | PCMP       | PX4       | PPWMB        | PPWMA       | PSPI     | PS2    | 0000,0000 |
| IP2H      | 高中断优先级控制寄存器 2  | B6H | PUSBH<br>PTKSUH | PI2CH     | PCMPH      | PX4H      | PPWMBH       | PPWMAH      | PSPIH    | PS2H   | 0000,0000 |
| IPH       | 高中断优先级控制寄存器    | B7H | -               | PLVDH     | PADCH      | PSH       | PT1H         | PX1H        | PT0H     | PX0H   | x000,0000 |
| IP        | 中断优先级控制寄存器     | B8H | -               | PLVD      | PADC       | PS        | PT1          | PX1         | PT0      | PX0    | x000,0000 |
| SADEN     | 串口 1 从机地址屏蔽寄存器 | B9H |                 |           |            |           |              |             |          |        | 0000,0000 |
| P_SW2     | 外设端口切换寄存器 2    | BAH | EAXFR           | -         | I2C_S[1:0] |           | CMPO_S       | S4_S        | S3_S     | S2_S   | 0x00,0000 |
| ADC_CONTR | ADC 控制寄存器      | BCH | ADC_POWER       | ADC_START | ADC_FLAG   | ADC_EPWMT | ADC_CHS[3:0] |             |          |        | 0000,0000 |
| ADC_RES   | ADC 转换结果高位寄存器  | BDH |                 |           |            |           |              |             |          |        | 0000,0000 |
| ADC_RESL  | ADC 转换结果低位寄存器  | BEH |                 |           |            |           |              |             |          |        | 0000,0000 |
| P4        | P4 端口          | C0H | P47             | P46       | P45        | P44       | P43          | P42         | P41      | P40    | 1111,1111 |
| WDT_CONTR | 看门狗控制寄存器       | C1H | WDT_FLAG        | -         | EN_WDT     | CLR_WDT   | IDL_WDT      | WDT_PS[2:0] |          |        | 0x00,0000 |
| IAP_DATA  | IAP 数据寄存器      | C2H |                 |           |            |           |              |             |          |        | 1111,1111 |
| IAP_ADDRH | IAP 高地址寄存器     | C3H |                 |           |            |           |              |             |          |        | 0000,0000 |
| IAP_ADDRL | IAP 低地址寄存器     | C4H |                 |           |            |           |              |             |          |        | 0000,0000 |
| IAP_CMD   | IAP 命令寄存器      | C5H | -               | -         | -          | -         | -            | -           | CMD[1:0] |        | xxxx,xx00 |
| IAP_TRIG  | IAP 触发寄存器      | C6H |                 |           |            |           |              |             |          |        | 0000,0000 |
| IAP_CONTR | IAP 控制寄存器      | C7H | IAPEN           | SWBS      | SWRST      | CMD_FAIL  | -            | -           | -        | -      | 0000,xxxx |
| P5        | P5 端口          | C8H | -               | -         | P55        | P54       | P53          | P52         | P51      | P50    | xx11,1111 |
| P5M1      | P5 口配置寄存器 1    | C9H | -               | -         | P55M1      | P54M1     | P53M1        | P52M1       | P51M1    | P50M1  | xx11,1111 |
| P5M0      | P5 口配置寄存器 0    | CAH | -               | -         | P55M0      | P54M0     | P53M0        | P52M0       | P51M0    | P50M0  | xx00,0000 |
| P6M1      | P6 口配置寄存器 1    | CBH | P67M1           | P66M1     | P65M1      | P64M1     | P63M1        | P62M1       | P61M1    | P60M1  | 1111,1111 |
| P6M0      | P6 口配置寄存器 0    | CCH | P67M0           | P66M0     | P65M0      | P64M0     | P63M0        | P62M0       | P61M0    | P60M0  | 0000,0000 |
| SPSTAT    | SPI 状态寄存器      | CDH | SPIF            | WCOL      | -          | -         | -            | -           | -        | -      | 00xx,xxxx |
| SPCTL     | SPI 控制寄存器      | CEH | SSIG            | SPEN      | DORD       | MSTR      | CPOL         | CPHA        | SPR[1:0] |        | 0000,0100 |
| SPDAT     | SPI 数据寄存器      | CFH |                 |           |            |           |              |             |          |        | 0000,0000 |
| PSW       | 程序状态字寄存器       | D0H | CY              | AC        | F0         | RS1       | RS0          | OV          | F1       | P      | 0000,0000 |
| T4T3M     | 定时器 4/3 控制寄存器  | D1H | T4R             | T4_C/T    | T4x12      | T4CLKO    | T3R          | T3_C/T      | T3x12    | T3CLKO | 0000,0000 |
| T4H       | 定时器 4 高字节      | D2H |                 |           |            |           |              |             |          |        | 0000,0000 |
| T4L       | 定时器 4 低字节      | D3H |                 |           |            |           |              |             |          |        | 0000,0000 |
| T3H       | 定时器 3 高字节      | D4H |                 |           |            |           |              |             |          |        | 0000,0000 |
| T3L       | 定时器 3 低字节      | D5H |                 |           |            |           |              |             |          |        | 0000,0000 |

|          |               |     |         |           |             |        |            |         |             |        |           |
|----------|---------------|-----|---------|-----------|-------------|--------|------------|---------|-------------|--------|-----------|
| T2H      | 定时器 2 高字节     | D6H |         |           |             |        |            |         |             |        | 0000,0000 |
| T2L      | 定时器 2 低字节     | D7H |         |           |             |        |            |         |             |        | 0000,0000 |
| USBCCLK  | USB 时钟控制寄存器   | DCH | ENCKM   | PCKI[1:0] |             | CRE    | TST_USB    | TST_PHY | PHYTST[1:0] |        | 0010,0000 |
| PLLCR    | PLL 时钟控制寄存器   | DCH | ENCKM   | PCKI[1:0] |             | -      | -          | -       | -           | -      | 000x,xxxx |
| ADCCFG   | ADC 配置寄存器     | DEH | -       | -         | RESFMT      | -      | SPEED[3:0] |         |             |        | xx0x,0000 |
| IP3      | 中断优先级控制寄存器 3  | DFH | -       | -         | -           | -      | -          | PRTC    | PS4         | PS3    | xxxx,x000 |
| ACC      | 累加器           | E0H |         |           |             |        |            |         |             |        | 0000,0000 |
| P7M1     | P7 口配置寄存器 1   | E1H | P77M1   | P76M1     | P75M1       | P74M1  | P73M1      | P72M1   | P71M1       | P70M1  | 1111,1111 |
| P7M0     | P7 口配置寄存器 0   | E2H | P77M0   | P76M0     | P75M0       | P74M0  | P73M0      | P72M0   | P71M0       | P70M0  | 0000,0000 |
| DPS      | DPTR 指针选择器    | E3H | ID1     | ID0       | TSL         | AU1    | AU0        | -       | -           | SEL    | 0000,0xx0 |
| DPL1     | 第二组数据指针 (低字节) | E4H |         |           |             |        |            |         |             |        | 0000,0000 |
| DPH1     | 第二组数据指针 (高字节) | E5H |         |           |             |        |            |         |             |        | 0000,0000 |
| CMPCR1   | 比较器控制寄存器 1    | E6H | CMPEN   | CMPIF     | PIE         | NIE    | PIS        | NIS     | CMPOE       | CMPRES | 0000,0000 |
| CMPCR2   | 比较器控制寄存器 2    | E7H | INVCMP0 | DISFLT    | LCDTY[5:0]  |        |            |         |             |        | 0000,0000 |
| P6       | P6 端口         | E8H | P67     | P66       | P65         | P64    | P63        | P62     | P61         | P60    | 1111,1111 |
| USBDAT   | USB 数据寄存器     | ECH |         |           |             |        |            |         |             |        | 0000,0000 |
| IP3H     | 高中断优先级控制寄存器 3 | EEH | -       | -         | -           | -      | -          | PRTCH   | PS4H        | PS3H   | xxxx,x000 |
| AUXINTIF | 扩展外部中断标志寄存器   | EFH | -       | INT4IF    | INT3IF      | INT2IF | -          | T4IF    | T3IF        | T2IF   | x000,x000 |
| B        | B 寄存器         | F0H |         |           |             |        |            |         |             |        | 0000,0000 |
| USBCON   | USB 控制寄存器     | F4H | ENUSB   | USBRST    | PS2M        | PUEN   | PDEN       | DFREC   | DP          | DM     | 0000,0000 |
| IAP_TPS  | IAP 等待时间控制寄存器 | F5H | -       | -         | IAPTPS[5:0] |        |            |         |             |        | xx00,0000 |
| P7       | P7 端口         | F8H | P77     | P76       | P75         | P74    | P73        | P72     | P71         | P70    | 1111,1111 |
| USBADR   | USB 地址寄存器     | FCH | BUSY    | AUTORD    | UADR[5:0]   |        |            |         |             |        | 0000,0000 |
| RSTCFG   | 复位配置寄存器       | FFH | -       | ENLVR     | -           | P54RST | -          | -       | LVDS[1:0]   |        | x0x0,xx00 |

下列特殊功能寄存器为扩展 SFR，逻辑地址位于 XDATA 区域，访问前需要将 P\_SW2 (BAH) 寄存器的最高位 (EAXFR) 置 1，然后使用 MOVX A,@DPTR 和 MOVX @DPTR,A 指令进行访问

| 符号       | 描述              | 地址    | 位地址与符号         |               |       |       |       |       |             |          | 复位值       |
|----------|-----------------|-------|----------------|---------------|-------|-------|-------|-------|-------------|----------|-----------|
|          |                 |       | B7             | B6            | B5    | B4    | B3    | B2    | B1          | B0       |           |
| CLKSEL   | 时钟选择寄存器         | FE00H | -              | -             | -     | -     | -     | -     | MCKSEL[1:0] |          | xxxx,xx00 |
| CLKDIV   | 时钟分频寄存器         | FE01H |                |               |       |       |       |       |             |          | nnnn,nnnn |
| HIRCCR   | 内部高速振荡器控制寄存器    | FE02H | ENHIRC         | -             | -     | -     | -     | -     | -           | HIRCST   | 1xxx,xxx0 |
| XOSCCR   | 外部晶振控制寄存器       | FE03H | ENXOSC         | XITYPE        | -     | -     | -     | -     | -           | XOSCST   | 00xx,xxx0 |
| IRC32KCR | 内部 32K 振荡器控制寄存器 | FE04H | ENIRC32K       | -             | -     | -     | -     | -     | -           | IRC32KST | 0xxx,xxx0 |
| MCLKOCR  | 主时钟输出控制寄存器      | FE05H | MCLKO_S        | MCLKODIV[6:0] |       |       |       |       |             |          | 0000,0000 |
| IRCDB    | 内部高速振荡器去抖控制     | FE06H | IRCDB_PAR[7:0] |               |       |       |       |       |             |          | 1000,0000 |
| IRC48MCR | 内部 48M 振荡器控制寄存器 | FE07H | ENIRC48M       | -             | -     | -     | -     | -     | -           | IRC48MST | 1xxx,xxx0 |
| P0PU     | P0 口上拉电阻控制寄存器   | FE10H | P07PU          | P06PU         | P05PU | P04PU | P03PU | P02PU | P01PU       | P00PU    | 0000,0000 |
| P1PU     | P1 口上拉电阻控制寄存器   | FE11H | P17PU          | P16PU         | P15PU | P14PU | P13PU | P12PU | P11PU       | P10PU    | 0000,0000 |
| P2PU     | P2 口上拉电阻控制寄存器   | FE12H | P27PU          | P26PU         | P25PU | P24PU | P23PU | P22PU | P21PU       | P20PU    | 0000,0000 |
| P3PU     | P3 口上拉电阻控制寄存器   | FE13H | P37PU          | P36PU         | P35PU | P34PU | P33PU | P32PU | P31PU       | P30PU    | 0000,0000 |
| P4PU     | P4 口上拉电阻控制寄存器   | FE14H | P47PU          | P46PU         | P45PU | P44PU | P43PU | P42PU | P41PU       | P40PU    | 0000,0000 |
| P5PU     | P5 口上拉电阻控制寄存器   | FE15H | -              | -             | P55PU | P54PU | P53PU | P52PU | P51PU       | P50PU    | xx00,0000 |

|           |                |       |                |               |              |             |               |          |            |         |           |  |
|-----------|----------------|-------|----------------|---------------|--------------|-------------|---------------|----------|------------|---------|-----------|--|
| P6PU      | P6 口上拉电阻控制寄存器  | FE16H | P67PU          | P66PU         | P65PU        | P64PU       | P63PU         | P62PU    | P61PU      | P60PU   | 0000,0000 |  |
| P7PU      | P7 口上拉电阻控制寄存器  | FE17H | P77PU          | P76PU         | P75PU        | P74PU       | P73PU         | P72PU    | P71PU      | P70PU   | 0000,0000 |  |
| P0NCS     | P0 口施密特触发控制寄存器 | FE18H | P07NCS         | P06NCS        | P05NCS       | P04NCS      | P03NCS        | P02NCS   | P01NCS     | P00NCS  | 0000,0000 |  |
| P1NCS     | P1 口施密特触发控制寄存器 | FE19H | P17NCS         | P16NCS        | P15NCS       | P14NCS      | P13NCS        | P12NCS   | P11NCS     | P10NCS  | 0000,0000 |  |
| P2NCS     | P2 口施密特触发控制寄存器 | FE1AH | P27NCS         | P26NCS        | P25NCS       | P24NCS      | P23NCS        | P22NCS   | P21NCS     | P20NCS  | 0000,0000 |  |
| P3NCS     | P3 口施密特触发控制寄存器 | FE1BH | P37NCS         | P36NCS        | P35NCS       | P34NCS      | P33NCS        | P32NCS   | P31NCS     | P30NCS  | 0000,0000 |  |
| P4NCS     | P4 口施密特触发控制寄存器 | FE1CH | P47NCS         | P46NCS        | P45NCS       | P44NCS      | P43NCS        | P42NCS   | P41NCS     | P40NCS  | 0000,0000 |  |
| P5NCS     | P5 口施密特触发控制寄存器 | FE1DH | -              | -             | P55NCS       | P54NCS      | P53NCS        | P52NCS   | P51NCS     | P50NCS  | xx00,0000 |  |
| P6NCS     | P6 口施密特触发控制寄存器 | FE1EH | P67NCS         | P66NCS        | P65NCS       | P64NCS      | P63NCS        | P62NCS   | P61NCS     | P60NCS  | 0000,0000 |  |
| P7NCS     | P7 口施密特触发控制寄存器 | FE1FH | P77NCS         | P76NCS        | P75NCS       | P74NCS      | P73NCS        | P72NCS   | P71NCS     | P70NCS  | 0000,0000 |  |
| P0SR      | P0 口电平转换速率寄存器  | FE20H | P07SR          | P06SR         | P05SR        | P04SR       | P03SR         | P02SR    | P01SR      | P00SR   | 1111,1111 |  |
| P1SR      | P1 口电平转换速率寄存器  | FE21H | P17SR          | P16SR         | P15SR        | P14SR       | P13SR         | P12SR    | P11SR      | P10SR   | 1111,1111 |  |
| P2SR      | P2 口电平转换速率寄存器  | FE22H | P27SR          | P26SR         | P25SR        | P24SR       | P23SR         | P22SR    | P21SR      | P20SR   | 1111,1111 |  |
| P3SR      | P3 口电平转换速率寄存器  | FE23H | P37SR          | P36SR         | P35SR        | P34SR       | P33SR         | P32SR    | P31SR      | P30SR   | 1111,1111 |  |
| P4SR      | P4 口电平转换速率寄存器  | FE24H | P47SR          | P46SR         | P45SR        | P44SR       | P43SR         | P42SR    | P41SR      | P40SR   | 1111,1111 |  |
| P5SR      | P5 口电平转换速率寄存器  | FE25H | -              | -             | P55SR        | P54SR       | P53SR         | P52SR    | P51SR      | P50SR   | xx11,1111 |  |
| P6SR      | P6 口电平转换速率寄存器  | FE26H | P57SR          | P66SR         | P65SR        | P64SR       | P63SR         | P62SR    | P61SR      | P60SR   | 1111,1111 |  |
| P7SR      | P7 口电平转换速率寄存器  | FE27H | P77SR          | P76SR         | P75SR        | P74SR       | P73SR         | P72SR    | P71SR      | P70SR   | 1111,1111 |  |
| P0DR      | P0 口驱动电流控制寄存器  | FE28H | P07DR          | P06DR         | P05DR        | P04DR       | P03DR         | P02DR    | P01DR      | P00DR   | 1111,1111 |  |
| P1DR      | P1 口驱动电流控制寄存器  | FE29H | P17DR          | P16DR         | P15DR        | P14DR       | P13DR         | P12DR    | P11DR      | P10DR   | 1111,1111 |  |
| P2DR      | P2 口驱动电流控制寄存器  | FE2AH | P27DR          | P26DR         | P25DR        | P24DR       | P23DR         | P22DR    | P21DR      | P20DR   | 1111,1111 |  |
| P3DR      | P3 口驱动电流控制寄存器  | FE2BH | P37DR          | P36DR         | P35DR        | P34DR       | P33DR         | P32DR    | P31DR      | P30DR   | 1111,1111 |  |
| P4DR      | P4 口驱动电流控制寄存器  | FE2CH | P47DR          | P46DR         | P45DR        | P44DR       | P43DR         | P42DR    | P41DR      | P40DR   | 1111,1111 |  |
| P5DR      | P5 口驱动电流控制寄存器  | FE2DH | -              | -             | P55DR        | P54DR       | P53DR         | P52DR    | P51DR      | P50DR   | xx11,1111 |  |
| P6DR      | P6 口驱动电流控制寄存器  | FE2EH | P67DR          | P66DR         | P65DR        | P64DR       | P63DR         | P62DR    | P61DR      | P60DR   | 1111,1111 |  |
| P7DR      | P7 口驱动电流控制寄存器  | FE2FH | P77DR          | P76DR         | P75DR        | P74DR       | P73DR         | P72DR    | P71DR      | P70DR   | 1111,1111 |  |
| P0IE      | P0 口输入使能控制寄存器  | FE30H | P07IE          | P06IE         | P05IE        | P04IE       | P03IE         | P02IE    | P01IE      | P00IE   | 1111,1111 |  |
| P1IE      | P1 口输入使能控制寄存器  | FE31H | P17IE          | P16IE         | P15IE        | P14IE       | P13IE         | P12IE    | P11IE      | P10IE   | 1111,1111 |  |
| P2IE      | P2 口输入使能控制寄存器  | FE32H | P27IE          | P26IE         | P25IE        | P24IE       | P23IE         | P22IE    | P21IE      | P20IE   | 1111,1111 |  |
| P3IE      | P3 口输入使能控制寄存器  | FE33H | P37IE          | P36IE         | P35IE        | P34IE       | P33IE         | P32IE    | P31IE      | P30IE   | 1111,1111 |  |
| P4IE      | P4 口输入使能控制寄存器  | FE34H | P47IE          | P46IE         | P45IE        | P44IE       | P43IE         | P42IE    | P41IE      | P40IE   | 1111,1111 |  |
| P5IE      | P5 口输入使能控制寄存器  | FE35H | -              | -             | P55IE        | P54IE       | P53IE         | P52IE    | P51IE      | P50IE   | xx11,1111 |  |
| P6IE      | P6 口输入使能控制寄存器  | FE36H | P67IE          | P66IE         | P65IE        | P64IE       | P63IE         | P62IE    | P61IE      | P60IE   | 1111,1111 |  |
| P7IE      | P7 口输入使能控制寄存器  | FE37H | P77IE          | P76IE         | P75IE        | P74IE       | P73IE         | P72IE    | P71IE      | P70IE   | 1111,1111 |  |
| LCMIFCFG  | LCM 接口配置寄存器    | FE50H | LCMIFIE        | -             | LCMIFIP[1:0] |             | LCMIFDPS[1:0] |          | D16_D8     | M68_I80 | 0x00,0000 |  |
| LCMIFCFG2 | LCM 接口配置寄存器 2  | FE51H | -              | LCMIFCPS[1:0] |              | SETUPT[2:0] |               |          | HOLDT[1:0] |         | x000,0000 |  |
| LCMIFCR   | LCM 接口控制寄存器    | FE52H | ENLCMIF        | -             | -            | -           | -             | CMD[2:0] |            |         | 0xxx,x000 |  |
| LCMIFSTA  | LCM 接口状态寄存器    | FE53H | -              | -             | -            | -           | -             | -        | -          | LCMIFIF | xxxx,xxx0 |  |
| LCMIDDATL | LCM 接口低字节数据    | FE54H | LCMIFDAT[7:0]  |               |              |             |               |          |            |         | 0000,0000 |  |
| LCMIDDATH | LCM 接口高字节数据    | FE55H | LCMIFDAT[15:8] |               |              |             |               |          |            |         | 0000,0000 |  |
| RTCCR     | RTC 控制寄存器      | FE60H | -              | -             | -            | -           | -             | -        | -          | RUNRTC  | xxxx,xxx0 |  |
| RTCCFG    | RTC 配置寄存器      | FE61H | -              | -             | -            | -           | -             | -        | RTCKS      | SETRTC  | xxxx,xx00 |  |
| RTCIEN    | RTC 中断使能寄存器    | FE62H | EALAI          | EDAYI         | EHOURI       | EMINI       | ESECI         | ESEC2I   | ESEC8I     | ESEC32I | 0000,0000 |  |
| RTCIF     | RTC 中断请求寄存器    | FE63H | ALAIF          | DAYIF         | HOURIF       | MINIF       | SECIF         | SEC2IF   | SEC8IF     | SEC32IF | 0000,0000 |  |



|            |                            |       |               |             |               |              |            |                |             |           |           |           |
|------------|----------------------------|-------|---------------|-------------|---------------|--------------|------------|----------------|-------------|-----------|-----------|-----------|
| ALAHOUR    | RTC 闹钟的小时值                 | FE64H | -             | -           | -             |              |            |                |             |           | xxx0,0000 |           |
| ALAMIN     | RTC 闹钟的分钟值                 | FE65H | -             | -           |               |              |            |                |             |           |           | xx00,0000 |
| ALASEC     | RTC 闹钟的秒值                  | FE66H | -             | -           |               |              |            |                |             |           |           | xx00,0000 |
| ALASSEC    | RTC 闹钟的 1/128 秒值           | FE67H | -             |             |               |              |            |                |             |           | x000,0000 |           |
| INIYEAR    | RTC 年初初始化                  | FE68H | -             |             |               |              |            |                |             |           | x000,0000 |           |
| INIMONTH   | RTC 月初初始化                  | FE69H | -             | -           | -             | -            |            |                |             |           | xxxx,0000 |           |
| INIDAY     | RTC 日初始化                   | FE6AH | -             | -           | -             |              |            |                |             |           | xxx0,0000 |           |
| INIHOUR    | RTC 小时初始化                  | FE6BH | -             | -           | -             |              |            |                |             |           | xxx0,0000 |           |
| INIMIN     | RTC 分钟初始化                  | FE6CH | -             | -           |               |              |            |                |             |           | xx00,0000 |           |
| INISEC     | RTC 秒初始化                   | FE6DH | -             | -           |               |              |            |                |             |           | xx00,0000 |           |
| INISSEC    | RTC1/128 秒初始化              | FE6EH | -             |             |               |              |            |                |             | x000,0000 |           |           |
| YEAR       | RTC 的年计数值                  | FE70H | -             |             |               |              |            |                |             | x000,0000 |           |           |
| MONTH      | RTC 的月计数值                  | FE71H | -             | -           | -             | -            |            |                |             |           | xxxx,0000 |           |
| DAY        | RTC 的日计数值                  | FE72H | -             | -           | -             |              |            |                |             |           | xxx0,0000 |           |
| HOURL      | RTC 的小时计数值                 | FE73H | -             | -           | -             |              |            |                |             |           | xxx0,0000 |           |
| MIN        | RTC 的分钟计数值                 | FE74H | -             | -           |               |              |            |                |             |           | xx00,0000 |           |
| SEC        | RTC 的秒计数值                  | FE75H | -             | -           |               |              |            |                |             |           | xx00,0000 |           |
| SSEC       | RTC 的 1/128 秒计数值           | FE76H | -             |             |               |              |            |                |             | x000,0000 |           |           |
| I2CCFG     | I <sup>2</sup> C 配置寄存器     | FE80H | ENI2C         | MSSL        | MSSPEED[6:1]  |              |            |                |             |           | 0000,0000 |           |
| I2CMSCR    | I <sup>2</sup> C 主机控制寄存器   | FE81H | EMSI          | -           | -             | -            | MSCMD[3:0] |                |             |           | 0xxx,0000 |           |
| I2CMSST    | I <sup>2</sup> C 主机状态寄存器   | FE82H | MSBUSY        | MSIF        | -             | -            | -          | -              | MSACKI      | MSACKO    | 00xx,xx10 |           |
| I2CSLCR    | I <sup>2</sup> C 从机控制寄存器   | FE83H | -             | ESTAI       | ERXI          | ETXI         | ESTOI      | -              | -           | SLRST     | x000,0xx0 |           |
| I2CSLST    | I <sup>2</sup> C 从机状态寄存器   | FE84H | SLBUSY        | STAIF       | RXIF          | TXIF         | STOIF      | TXING          | SLACKI      | SLACKO    | 0000,0000 |           |
| I2CSLADR   | I <sup>2</sup> C 从机地址寄存器   | FE85H | I2CSLADR[7:1] |             |               |              |            |                |             | MA        | 0000,0000 |           |
| I2CTXD     | I <sup>2</sup> C 数据发送寄存器   | FE86H |               |             |               |              |            |                |             |           | 0000,0000 |           |
| I2CRXD     | I <sup>2</sup> C 数据接收寄存器   | FE87H |               |             |               |              |            |                |             |           | 0000,0000 |           |
| I2CMSAUX   | I <sup>2</sup> C 主机辅助控制寄存器 | FE88H | -             | -           | -             | -            | -          | -              | -           | WDTA      | xxxx,xxx0 |           |
| SPFUNC     | 特殊功能控制寄存器                  | FE98H |               |             |               |              |            |                |             | BKSWR     | xxxx,xxx0 |           |
| RSTFLAG    | 复位标志寄存器                    | FE99H |               |             |               | LVDRSTF      | WDTRSTF    | SWRSTF         | ROMOVF      | EXRSTF    | xxx1,0100 |           |
| TM2PS      | 定时器 2 时钟预分频寄存器             | FEA2H |               |             |               |              |            |                |             |           | 0000,0000 |           |
| TM3PS      | 定时器 3 时钟预分频寄存器             | FEA3H |               |             |               |              |            |                |             |           | 0000,0000 |           |
| TM4PS      | 定时器 4 时钟预分频寄存器             | FEA4H |               |             |               |              |            |                |             |           | 0000,0000 |           |
| ADCTIM     | ADC 时序控制寄存器                | FEA8H | CSSETUP       | CSHOLD[1:0] |               | SMPDUTY[4:0] |            |                |             |           | 0010,1010 |           |
| ADCEXCFG   | ADC 扩展配置寄存器                | FEADH | -             | -           | ADCETRS [1:0] |              | -          | CVTIMESEL[2:0] |             |           | xx00,x000 |           |
| CMPEXCFG   | 比较器扩展配置寄存器                 | FEAEH | CHYS[1:0]     |             | -             | -            | -          | CMPNS          | CMPPS[1:0]  |           | 00xx,x000 |           |
| T3T4PIN    | T3/T4 选择寄存器                | FEACH | -             | -           | -             | -            | -          | -              | -           | T3T4SEL   | xxxx,xxx0 |           |
| PWMA_ETRPS | PWMA 的 ETR 选择寄存器           | FEB0H |               |             |               |              |            | BRKAPS         | ETRAPS[1:0] |           | xxxx,x000 |           |
| PWMA_ENO   | PWMA 输出使能控制                | FEB1H | ENO4N         | ENO4P       | ENO3N         | ENO3P        | ENO2N      | ENO2P          | ENO1N       | ENO1P     | 0000,0000 |           |
| PWMA_PS    | PWMA 输出脚选择寄存器              | FEB2H | C4PS[1:0]     |             | C3PS[1:0]     |              | C2PS[1:0]  |                | C1PS[1:0]   |           | 0000,0000 |           |
| PWMA_IOAUX | PWMA 辅助寄存器                 | FEB3H | AUX4N         | AUX4P       | AUX3N         | AUX3P        | AUX2N      | AUX2P          | AUX1N       | AUX1P     | 0000,0000 |           |
| PWMB_ETRPS | PWMB 的 ETR 选择寄存器           | FEB4H |               |             |               |              |            | BRKBPS         | ETRBPS[1:0] |           | xxxx,x000 |           |
| PWMB_ENO   | PWMB 输出使能控制                | FEB5H | -             | ENO8P       | -             | ENO7P        | -          | ENO6P          | -           | ENO5P     | x0x0,x0x0 |           |
| PWMB_PS    | PWMB 输出脚选择寄存器              | FEB6H | C8PS[1:0]     |             | C7PS[1:0]     |              | C6PS[1:0]  |                | C5PS[1:0]   |           | 0000,0000 |           |
| PWMB_IOAUX | PWMB 辅助寄存器                 | FEB7H | -             | AUX8P       | -             | AUX7P        | -          | AUX6P          | -           | AUX5P     | x0x0,x0x0 |           |

|            |                   |       |           |           |           |             |          |           |           |           |           |           |
|------------|-------------------|-------|-----------|-----------|-----------|-------------|----------|-----------|-----------|-----------|-----------|-----------|
| PWMA_CR1   | PWMA 控制寄存器 1      | FEC0H | ARPE      | CMS[1:0]  |           | DIR         | OPM      | URS       | UDIS      | CEN       | 0000,0000 |           |
| PWMA_CR2   | PWMA 控制寄存器 2      | FEC1H | -         | MMS[2:0]  |           |             | -        | COMS      | -         | CCPC      | x000,x0x0 |           |
| PWMA_SMCRR | PWMA 从模式控制寄存器     | FEC2H | MSM       | TS[2:0]   |           |             | -        | SMS[2:0]  |           |           | 0000,x000 |           |
| PWMA_ETR   | PWMA 外部触发寄存器      | FEC3H | ETP       | ECE       | ETPS[1:0] |             | ETF[3:0] |           |           | 0000,0000 |           |           |
| PWMA_IER   | PWMA 中断使能寄存器      | FEC4H | BIE       | TIE       | COMIE     | CC4IE       | CC3IE    | CC2IE     | CC1IE     | UIE       | 0000,0000 |           |
| PWMA_SR1   | PWMA 状态寄存器 1      | FEC5H | BIF       | TIF       | COMIF     | CC4IF       | CC3IF    | CC2IF     | CC1IF     | UIF       | 0000,0000 |           |
| PWMA_SR2   | PWMA 状态寄存器 2      | FEC6H | -         | -         | -         | CC4OF       | CC3OF    | CC2OF     | CC1OF     | -         | xxx0,000x |           |
| PWMA_EGR   | PWMA 事件发生寄存器      | FEC7H | BG        | TG        | COMG      | CC4G        | CC3G     | CC2G      | CC1G      | UG        | 0000,0000 |           |
| PWMA_CCMR1 | PWMA 捕获模式寄存器 1    | FEC8H | OC1CE     | OC1M[2:0] |           |             | OC1PE    | OC1FE     | CC1S[1:0] |           | 0000,0000 |           |
|            | PWMA 比较模式寄存器 1    |       | IC1F[3:0] |           |           | IC1PSC[1:0] |          | CC1S[1:0] |           | 0000,0000 |           |           |
| PWMA_CCMR2 | PWMA 捕获模式寄存器 2    | FEC9H | OC2CE     | OC2M[2:0] |           |             | OC2PE    | OC2FE     | CC2S[1:0] |           | 0000,0000 |           |
|            | PWMA 比较模式寄存器 2    |       | IC2F[3:0] |           |           | IC2PSC[1:0] |          | CC2S[1:0] |           | 0000,0000 |           |           |
| PWMA_CCMR3 | PWMA 捕获模式寄存器 3    | FECAH | OC3CE     | OC3M[2:0] |           |             | OC3PE    | OC3FE     | CC3S[1:0] |           | 0000,0000 |           |
|            | PWMA 比较模式寄存器 3    |       | IC3F[3:0] |           |           | IC3PSC[1:0] |          | CC3S[1:0] |           | 0000,0000 |           |           |
| PWMA_CCMR4 | PWMA 捕获模式寄存器 4    | FECBH | OC4CE     | OC4M[2:0] |           |             | OC4PE    | OC4FE     | CC4S[1:0] |           | 0000,0000 |           |
|            | PWMA 比较模式寄存器 4    |       | IC4F[3:0] |           |           | IC4PSC[1:0] |          | CC4S[1:0] |           | 0000,0000 |           |           |
| PWMA_CCER1 | PWMA 捕获比较使能寄存器 1  | FECCH | CC2NP     | CC2NE     | CC2P      | CC2E        | CC1NP    | CC1NE     | CC1P      | CC1E      | 0000,0000 |           |
| PWMA_CCER2 | PWMA 捕获比较使能寄存器 2  | FECDH | CC4NP     | CC4NE     | CC4P      | CC4E        | CC3NP    | CC3NE     | CC3P      | CC3E      | 0000,0000 |           |
| PWMA_CNTRH | PWMA 计数器高字节       | FECEH |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_CNTRL | PWMA 计数器低字节       | FECFH |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_PSCRH | PWMA 预分频高字节       | FED0H |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_PSCRL | PWMA 预分频低字节       | FED1H |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_ARRH  | PWMA 自动重装寄存器高字节   | FED2H |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_ARRL  | PWMA 自动重装寄存器低字节   | FED3H |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_RCR   | PWMA 重复计数器寄存器     | FED4H |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_CCR1H | PWMA 比较捕获寄存器 1 高位 | FED5H |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_CCR1L | PWMA 比较捕获寄存器 1 低位 | FED6H |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_CCR2H | PWMA 比较捕获寄存器 2 高位 | FED7H |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_CCR2L | PWMA 比较捕获寄存器 2 低位 | FED8H |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_CCR3H | PWMA 比较捕获寄存器 3 高位 | FED9H |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_CCR3L | PWMA 比较捕获寄存器 3 低位 | FEDAH |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_CCR4H | PWMA 比较捕获寄存器 4 高位 | FEDBH |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_CCR4L | PWMA 比较捕获寄存器 4 低位 | FEDCH |           |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_BKR   | PWMA 刹车寄存器        | FEDDH | MOE       | AOE       | BKP       | BKE         | OSSR     | OSSI      | LOCK[1:0] |           | 0000,0000 |           |
| PWMA_DTR   | PWMA 死区控制寄存器      | FEDEH | DTG[7:0]  |           |           |             |          |           |           |           |           | 0000,0000 |
| PWMA_OISR  | PWMA 输出空闲状态寄存器    | FEDFH | OIS4N     | OIS4      | OIS3N     | OIS3        | OIS2N    | OIS2      | OIS1N     | OIS1      | 0000,0000 |           |
| PWMB_CR1   | PWMB 控制寄存器 1      | FEE0H | ARPE      | CMS[1:0]  |           | DIR         | OPM      | URS       | UDIS      | CEN       | 0000,0000 |           |
| PWMB_CR2   | PWMB 控制寄存器 2      | FEE1H | -         | MMS[2:0]  |           |             | -        | COMS      | -         | CCPC      | x000,x0x0 |           |
| PWMB_SMCRR | PWMB 从模式控制寄存器     | FEE2H | MSM       | TS[2:0]   |           |             | -        | SMS[2:0]  |           |           | 0000,x000 |           |
| PWMB_ETR   | PWMB 外部触发寄存器      | FEE3H | ETP       | ECE       | ETPS[1:0] |             | ETF[3:0] |           |           | 0000,0000 |           |           |
| PWMB_IER   | PWMB 中断使能寄存器      | FEE4H | BIE       | TIE       | COMIE     | CC8IE       | CC7IE    | CC6IE     | CC5IE     | UIE       | 0000,0000 |           |
| PWMB_SR1   | PWMB 状态寄存器 1      | FEE5H | BIF       | TIF       | COMIF     | CC8IF       | CC7IF    | CC6IF     | CC5IF     | UIF       | 0000,0000 |           |
| PWMB_SR2   | PWMB 状态寄存器 2      | FEE6H | -         | -         | -         | CC8OF       | CC7OF    | CC6OF     | CC5OF     | -         | xxx0,000x |           |
| PWMB_EGR   | PWMB 事件发生寄存器      | FEE7H | BG        | TG        | COMG      | CC8G        | CC7G     | CC6G      | CC5G      | UG        | 0000,0000 |           |

|            |                   |       |           |           |         |         |             |         |           |         |           |           |
|------------|-------------------|-------|-----------|-----------|---------|---------|-------------|---------|-----------|---------|-----------|-----------|
| PWMB_CCMR1 | PWMB 捕获模式寄存器 1    | FEE8H | OC5CE     | OC5M[2:0] |         |         | OC5PE       | OC5FE   | CC5S[1:0] |         | 0000,0000 |           |
|            | PWMB 比较模式寄存器 1    |       | IC5F[3:0] |           |         |         | IC5PSC[1:0] |         | CC5S[1:0] |         | 0000,0000 |           |
| PWMB_CCMR2 | PWMB 捕获模式寄存器 2    | FEE9H | OC6CE     | OC6M[2:0] |         |         | OC6PE       | OC6FE   | CC6S[1:0] |         | 0000,0000 |           |
|            | PWMB 比较模式寄存器 2    |       | IC6F[3:0] |           |         |         | IC6PSC[1:0] |         | CC6S[1:0] |         | 0000,0000 |           |
| PWMB_CCMR3 | PWMB 捕获模式寄存器 3    | FEEAH | OC7CE     | OC7M[2:0] |         |         | OC7PE       | OC7FE   | CC7S[1:0] |         | 0000,0000 |           |
|            | PWMB 比较模式寄存器 3    |       | IC7F[3:0] |           |         |         | IC7PSC[1:0] |         | CC7S[1:0] |         | 0000,0000 |           |
| PWMB_CCMR4 | PWMB 捕获模式寄存器 4    | FEEBH | OC8CE     | OC8M[2:0] |         |         | OC8PE       | OC8FE   | CC8S[1:0] |         | 0000,0000 |           |
|            | PWMB 比较模式寄存器 4    |       | IC8F[3:0] |           |         |         | IC8PSC[1:0] |         | CC8S[1:0] |         | 0000,0000 |           |
| PWMB_CCER1 | PWMB 捕获比较使能寄存器 1  | FEECH | -         | -         | CC6P    | CC6E    | -           | -       | CC5P      | CC5E    | xx00,xx00 |           |
| PWMB_CCER2 | PWMB 捕获比较使能寄存器 2  | FEEDH | -         | -         | CC8P    | CC8E    | -           | -       | CC7P      | CC7E    | xx00,xx00 |           |
| PWMB_CNTRH | PWMB 计数器高字节       | FEEEH |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_CNTRL | PWMB 计数器低字节       | FEFEH |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_PSCRH | PWMB 预分频高字节       | FEF0H |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_PSCRL | PWMB 预分频低字节       | FEF1H |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_ARRH  | PWMB 自动重装寄存器高字节   | FEF2H |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_ARRL  | PWMB 自动重装寄存器低字节   | FEF3H |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_RCR   | PWMB 重复计数器寄存器     | FEF4H |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_CCR5H | PWMB 比较捕获寄存器 1 高位 | FEF5H |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_CCR5L | PWMB 比较捕获寄存器 1 低位 | FEF6H |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_CCR6H | PWMB 比较捕获寄存器 2 高位 | FEF7H |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_CCR6L | PWMB 比较捕获寄存器 2 低位 | FEF8H |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_CCR7H | PWMB 比较捕获寄存器 3 高位 | FEF9H |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_CCR7L | PWMB 比较捕获寄存器 3 低位 | FEFAH |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_CCR8H | PWMB 比较捕获寄存器 4 高位 | FEFBH |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_CCR8L | PWMB 比较捕获寄存器 4 低位 | FEFCH |           |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_BKR   | PWMB 刹车寄存器        | FEFDH | MOE       | AOE       | BKP     | BKE     | OSSR        | OSSI    | LOCK[1:0] |         | 0000,0000 |           |
| PWMB_DTR   | PWMB 死区控制寄存器      | FEFEH | DTG[7:0]  |           |         |         |             |         |           |         |           | 0000,0000 |
| PWMB_OISR  | PWMB 输出空闲状态寄存器    | FEFFH | -         | OIS8      | -       | OIS7    | -           | OIS6    | -         | OIS5    | x0x0,x0x0 |           |
| MD3        | MDU 数据寄存器         | FCF0H | MD3[7:0]  |           |         |         |             |         |           |         |           | 0000,0000 |
| MD2        | MDU 数据寄存器         | FCF1H | MD2[7:0]  |           |         |         |             |         |           |         |           | 0000,0000 |
| MD1        | MDU 数据寄存器         | FCF2H | MD1[7:0]  |           |         |         |             |         |           |         |           | 0000,0000 |
| MD0        | MDU 数据寄存器         | FCF3H | MD0[7:0]  |           |         |         |             |         |           |         |           | 0000,0000 |
| MD5        | MDU 数据寄存器         | FCF4H | MD5[7:0]  |           |         |         |             |         |           |         |           | 0000,0000 |
| MD4        | MDU 数据寄存器         | FCF5H | MD4[7:0]  |           |         |         |             |         |           |         |           | 0000,0000 |
| ARCON      | MDU 模式控制寄存器       | FCF6H | MODE[2:0] |           |         | SC[4:0] |             |         |           |         |           | 0000,0000 |
| OPCON      | MDU 操作控制寄存器       | FCF7H | -         | MDOV      | -       | -       | -           | -       | RST       | ENOP    | x0xx,xx00 |           |
| P0INTE     | P0 口中断使能寄存器       | FD00H | P07INTE   | P06INTE   | P05INTE | P04INTE | P03INTE     | P02INTE | P01INTE   | P00INTE | 0000,0000 |           |
| P1INTE     | P1 口中断使能寄存器       | FD01H | P17INTE   | P16INTE   | P15INTE | P14INTE | P13INTE     | P12INTE | P11INTE   | P10INTE | 0000,0000 |           |
| P2INTE     | P2 口中断使能寄存器       | FD02H | P27INTE   | P26INTE   | P25INTE | P24INTE | P23INTE     | P22INTE | P21INTE   | P20INTE | 0000,0000 |           |
| P3INTE     | P3 口中断使能寄存器       | FD03H | P37INTE   | P36INTE   | P35INTE | P34INTE | P33INTE     | P32INTE | P31INTE   | P30INTE | 0000,0000 |           |
| P4INTE     | P4 口中断使能寄存器       | FD04H | P47INTE   | P46INTE   | P45INTE | P44INTE | P43INTE     | P42INTE | P41INTE   | P40INTE | 0000,0000 |           |
| P5INTE     | P5 口中断使能寄存器       | FD05H | -         | -         | P55INTE | P54INTE | P53INTE     | P52INTE | P51INTE   | P50INTE | xx00,0000 |           |
| P6INTE     | P6 口中断使能寄存器       | FD06H | P67INTE   | P66INTE   | P65INTE | P64INTE | P63INTE     | P62INTE | P61INTE   | P60INTE | 0000,0000 |           |
| P7INTE     | P7 口中断使能寄存器       | FD07H | P77INTE   | P76INTE   | P75INTE | P74INTE | P73INTE     | P72INTE | P71INTE   | P70INTE | 0000,0000 |           |

|           |                |       |         |         |              |         |         |              |         |         |           |
|-----------|----------------|-------|---------|---------|--------------|---------|---------|--------------|---------|---------|-----------|
| P0INTF    | P0 口中断标志寄存器    | FD10H | P07INTF | P06INTF | P05INTF      | P04INTF | P03INTF | P02INTF      | P01INTF | P00INTF | 0000,0000 |
| P1INTF    | P1 口中断标志寄存器    | FD11H | P17INTF | P16INTF | P15INTF      | P14INTF | P13INTF | P12INTF      | P11INTF | P10INTF | 0000,0000 |
| P2INTF    | P2 口中断标志寄存器    | FD12H | P27INTF | P26INTF | P25INTF      | P24INTF | P23INTF | P22INTF      | P21INTF | P20INTF | 0000,0000 |
| P3INTF    | P3 口中断标志寄存器    | FD13H | P37INTF | P36INTF | P35INTF      | P34INTF | P33INTF | P32INTF      | P31INTF | P30INTF | 0000,0000 |
| P4INTF    | P4 口中断标志寄存器    | FD14H | P47INTF | P46INTF | P45INTF      | P44INTF | P43INTF | P42INTF      | P41INTF | P40INTF | 0000,0000 |
| P5INTF    | P5 口中断标志寄存器    | FD15H | -       | -       | P55INTF      | P54INTF | P53INTF | P52INTF      | P51INTF | P50INTF | xx00,0000 |
| P6INTF    | P6 口中断标志寄存器    | FD16H | P67INTF | P66INTF | P65INTF      | P64INTF | P63INTF | P62INTF      | P61INTF | P60INTF | 0000,0000 |
| P7INTF    | P7 口中断标志寄存器    | FD17H | P77INTF | P76INTF | P75INTF      | P74INTF | P73INTF | P72INTF      | P71INTF | P70INTF | 0000,0000 |
| P0IM0     | P0 口中断模式寄存器 0  | FD20H | P07IM0  | P06IM0  | P05IM0       | P04IM0  | P03IM0  | P02IM0       | P01IM0  | P00IM0  | 0000,0000 |
| P1IM0     | P1 口中断模式寄存器 0  | FD21H | P17IM0  | P16IM0  | P15IM0       | P14IM0  | P13IM0  | P12IM0       | P11IM0  | P10IM0  | 0000,0000 |
| P2IM0     | P2 口中断模式寄存器 0  | FD22H | P27IM0  | P26IM0  | P25IM0       | P24IM0  | P23IM0  | P22IM0       | P21IM0  | P20IM0  | 0000,0000 |
| P3IM0     | P3 口中断模式寄存器 0  | FD23H | P37IM0  | P36IM0  | P35IM0       | P34IM0  | P33IM0  | P32IM0       | P31IM0  | P30IM0  | 0000,0000 |
| P4IM0     | P4 口中断模式寄存器 0  | FD24H | P47IM0  | P46IM0  | P45IM0       | P44IM0  | P43IM0  | P42IM0       | P41IM0  | P40IM0  | 0000,0000 |
| P5IM0     | P5 口中断模式寄存器 0  | FD25H | -       | -       | P55IM0       | P54IM0  | P53IM0  | P52IM0       | P51IM0  | P50IM0  | xx00,0000 |
| P6IM0     | P6 口中断模式寄存器 0  | FD26H | P67IM0  | P66IM0  | P65IM0       | P64IM0  | P63IM0  | P62IM0       | P61IM0  | P60IM0  | 0000,0000 |
| P7IM0     | P7 口中断模式寄存器 0  | FD27H | P77IM0  | P76IM0  | P75IM0       | P74IM0  | P73IM0  | P72IM0       | P71IM0  | P70IM0  | 0000,0000 |
| P0IM1     | P0 口中断模式寄存器 1  | FD30H | P07IM1  | P06IM1  | P05IM1       | P04IM1  | P03IM1  | P02IM1       | P01IM1  | P00IM1  | 0000,0000 |
| P1IM1     | P1 口中断模式寄存器 1  | FD31H | P17IM1  | P16IM1  | P15IM1       | P14IM1  | P13IM1  | P12IM1       | P11IM1  | P10IM1  | 0000,0000 |
| P2IM1     | P2 口中断模式寄存器 1  | FD32H | P27IM1  | P26IM1  | P25IM1       | P24IM1  | P23IM1  | P22IM1       | P21IM1  | P20IM1  | 0000,0000 |
| P3IM1     | P3 口中断模式寄存器 1  | FD33H | P37IM1  | P36IM1  | P35IM1       | P34IM1  | P33IM1  | P32IM1       | P31IM1  | P30IM1  | 0000,0000 |
| P4IM1     | P4 口中断模式寄存器 1  | FD34H | P47IM1  | P46IM1  | P45IM1       | P44IM1  | P43IM1  | P42IM1       | P41IM1  | P40IM1  | 0000,0000 |
| P5IM1     | P5 口中断模式寄存器 1  | FD35H | -       | -       | P55IM1       | P54IM1  | P53IM1  | P52IM1       | P51IM1  | P50IM1  | xx00,0000 |
| P6IM1     | P6 口中断模式寄存器 1  | FD36H | P67IM1  | P66IM1  | P65IM1       | P64IM1  | P63IM1  | P62IM1       | P61IM1  | P60IM1  | 0000,0000 |
| P7IM1     | P7 口中断模式寄存器 1  | FD37H | P77IM1  | P76IM1  | P75IM1       | P74IM1  | P73IM1  | P72IM1       | P71IM1  | P70IM1  | 0000,0000 |
| P0WKUE    | P0 口中断唤醒使能寄存器  | FD40H | P07WKUE | P06WKUE | P05WKUE      | P04WKUE | P03WKUE | P02WKUE      | P01WKUE | P00WKUE | 0000,0000 |
| P1WKUE    | P1 口中断唤醒使能寄存器  | FD41H | P17WKUE | P16WKUE | P15WKUE      | P14WKUE | P13WKUE | P12WKUE      | P11WKUE | P10WKUE | 0000,0000 |
| P2WKUE    | P2 口中断唤醒使能寄存器  | FD42H | P27WKUE | P26WKUE | P25WKUE      | P24WKUE | P23WKUE | P22WKUE      | P21WKUE | P20WKUE | 0000,0000 |
| P3WKUE    | P3 口中断唤醒使能寄存器  | FD43H | P37WKUE | P36WKUE | P35WKUE      | P34WKUE | P33WKUE | P32WKUE      | P31WKUE | P30WKUE | 0000,0000 |
| P4WKUE    | P4 口中断唤醒使能寄存器  | FD44H | P47WKUE | P46WKUE | P45WKUE      | P44WKUE | P43WKUE | P42WKUE      | P41WKUE | P40WKUE | 0000,0000 |
| P5WKUE    | P5 口中断唤醒使能寄存器  | FD45H | -       | -       | P55WKUE      | P54WKUE | P53WKUE | P52WKUE      | P51WKUE | P50WKUE | xx00,0000 |
| P6WKUE    | P6 口中断唤醒使能寄存器  | FD46H | P67WKUE | P66WKUE | P65WKUE      | P64WKUE | P63WKUE | P62WKUE      | P61WKUE | P60WKUE | 0000,0000 |
| P7WKUE    | P7 口中断唤醒使能寄存器  | FD47H | P77WKUE | P76WKUE | P75WKUE      | P74WKUE | P73WKUE | P72WKUE      | P71WKUE | P70WKUE | 0000,0000 |
| PINIPL    | I/O 口中断优先级低寄存器 | FD60H | P7IP    | P6IP    | P5IP         | P4IP    | P3IP    | P2IP         | P1IP    | P0IP    | 0000,0000 |
| PINIPH    | I/O 口中断优先级高寄存器 | FD61H | P7IPH   | P6IPH   | P5IPH        | P4IPH   | P3IPH   | P2IPH        | P1IPH   | P0IPH   | 0000,0000 |
| COMEN     | COM 使能寄存器      | FB00H | C7EN    | C6EN    | C5EN         | C4EN    | C3EN    | C2EN         | C1EN    | C0EN    | 0000,0000 |
| SEGENL    | SEG 使能寄存器      | FB01H | S7EN    | S6EN    | S5EN         | S4EN    | S3EN    | S2EN         | S1EN    | S0EN    | 0000,0000 |
| SEGENH    | SEG 使能寄存器      | FB02H | S15EN   | S14EN   | S13EN        | S12EN   | S11EN   | S10EN        | S9EN    | S8EN    | 0000,0000 |
| LEDCTRL   | LED 控制寄存器      | FB03H | LEDON   | -       | LEDMODE[1:0] |         | -       | LEDDUTY[2:0] |         |         | 0x00,x000 |
| LEDCKS    | LED 时钟分频寄存器    | FB04H |         |         |              |         |         |              |         |         | 0000,0001 |
| COM0_DA_L | 共阳模式显示数据       | FB10H |         |         |              |         |         |              |         |         | 0000,0000 |
| COM1_DA_L | 共阳模式显示数据       | FB11H |         |         |              |         |         |              |         |         | 0000,0000 |
| COM2_DA_L | 共阳模式显示数据       | FB12H |         |         |              |         |         |              |         |         | 0000,0000 |
| COM3_DA_L | 共阳模式显示数据       | FB13H |         |         |              |         |         |              |         |         | 0000,0000 |
| COM4_DA_L | 共阳模式显示数据       | FB14H |         |         |              |         |         |              |         |         | 0000,0000 |

|           |               |       |        |          |        |        |              |         |             |       |           |           |
|-----------|---------------|-------|--------|----------|--------|--------|--------------|---------|-------------|-------|-----------|-----------|
| COM5_DA_L | 共阳模式显示数据      | FB15H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM6_DA_L | 共阳模式显示数据      | FB16H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM7_DA_L | 共阳模式显示数据      | FB17H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM0_DA_H | 共阳模式显示数据      | FB18H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM1_DA_H | 共阳模式显示数据      | FB19H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM2_DA_H | 共阳模式显示数据      | FB1AH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM3_DA_H | 共阳模式显示数据      | FB1BH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM4_DA_H | 共阳模式显示数据      | FB1CH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM5_DA_H | 共阳模式显示数据      | FB1DH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM6_DA_H | 共阳模式显示数据      | FB1EH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM7_DA_H | 共阳模式显示数据      | FB1FH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM0_DC_L | 共阴模式显示数据      | FB20H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM1_DC_L | 共阴模式显示数据      | FB21H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM2_DC_L | 共阴模式显示数据      | FB22H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM3_DC_L | 共阴模式显示数据      | FB23H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM4_DC_L | 共阴模式显示数据      | FB24H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM5_DC_L | 共阴模式显示数据      | FB25H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM6_DC_L | 共阴模式显示数据      | FB26H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM7_DC_L | 共阴模式显示数据      | FB27H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM0_DC_H | 共阴模式显示数据      | FB28H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM1_DC_H | 共阴模式显示数据      | FB29H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM2_DC_H | 共阴模式显示数据      | FB2AH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM3_DC_H | 共阴模式显示数据      | FB2BH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM4_DC_H | 共阴模式显示数据      | FB2CH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM5_DC_H | 共阴模式显示数据      | FB2DH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM6_DC_H | 共阴模式显示数据      | FB2EH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| COM7_DC_H | 共阴模式显示数据      | FB2FH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| TSCHEN1   | 触摸按键使能寄存器 1   | FB40H | TKEN7  | TKEN6    | TKEN5  | TKEN4  | TKEN3        | TKEN2   | TKEN1       | TKEN0 | 0000,0000 |           |
| TSCHEN2   | 触摸按键使能寄存器 2   | FB41H | TKEN15 | TKEN14   | TKEN13 | TKEN12 | TKEN11       | TKEN10  | TKEN9       | TKEN8 | 0000,0000 |           |
| TSCFG1    | 触摸按键配置寄存器 1   | FB42H | -      | SCR[2:0] |        |        | -            | DT[2:0] |             |       | x000,x000 |           |
| TSCFG2    | 触摸按键配置寄存器 2   | FB43H | -      | -        | -      | -      | -            | -       | TSVR[1:0]   |       | xxxx,xx00 |           |
| TSWUTC    | 触摸按键唤醒控制寄存器   | FB44H |        |          |        |        |              |         |             |       |           | 0000,0001 |
| TSCTRL    | 触摸按键控制寄存器     | FB45H | TSGO   | SINGLE   | TSWAIT | TSWUCS | TSDCEN       | TSWUEN  | TSSAMP[1:0] |       | 0000,0000 |           |
| TSSTA1    | 触摸按键状态寄存器 1   | FB46H | LEDWK  | -        | -      | -      | TSWKCHN[3:0] |         |             |       | 0xxx,0000 |           |
| TSSTA2    | 触摸按键状态寄存器 2   | FB47H | TSIF   | TSDOV    | -      | -      | TSDNCHN[3:0] |         |             |       | 00xx,0000 |           |
| TSRT      | 触摸按键时间控制寄存器   | FB48H |        |          |        |        |              |         |             |       |           | 0000,0001 |
| TSDATH    | 触摸按键数据高字节     | FB49H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| TSDATL    | 触摸按键数据低字节     | FB4AH |        |          |        |        |              |         |             |       |           | 0000,0000 |
| TSTH00H   | 触摸按键 0 门槛值高字节 | FB50H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| TSTH00L   | 触摸按键 0 门槛值低字节 | FB51H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| TSTH01H   | 触摸按键 1 门槛值高字节 | FB52H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| TSTH01L   | 触摸按键 1 门槛值低字节 | FB53H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| TSTH02H   | 触摸按键 2 门槛值高字节 | FB54H |        |          |        |        |              |         |             |       |           | 0000,0000 |
| TSTH02L   | 触摸按键 2 门槛值低字节 | FB55H |        |          |        |        |              |         |             |       |           | 0000,0000 |

|           |                  |       |                |       |       |       |               |              |        |           |           |
|-----------|------------------|-------|----------------|-------|-------|-------|---------------|--------------|--------|-----------|-----------|
| TSTH03H   | 触摸按键 3 门檻值高字节    | FB56H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH03L   | 触摸按键 3 门檻值低字节    | FB57H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH04H   | 触摸按键 4 门檻值高字节    | FB58H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH04L   | 触摸按键 4 门檻值低字节    | FB59H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH05H   | 触摸按键 5 门檻值高字节    | FB5AH |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH05L   | 触摸按键 5 门檻值低字节    | FB5BH |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH06H   | 触摸按键 6 门檻值高字节    | FB5CH |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH06L   | 触摸按键 6 门檻值低字节    | FB5DH |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH07H   | 触摸按键 7 门檻值高字节    | FB5EH |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH07L   | 触摸按键 7 门檻值低字节    | FB5FH |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH08H   | 触摸按键 8 门檻值高字节    | FB60H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH08L   | 触摸按键 8 门檻值低字节    | FB61H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH09H   | 触摸按键 9 门檻值高字节    | FB62H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH09L   | 触摸按键 9 门檻值低字节    | FB63H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH10H   | 触摸按键 10 门檻值高字节   | FB64H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH10L   | 触摸按键 10 门檻值低字节   | FB65H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH11H   | 触摸按键 11 门檻值高字节   | FB66H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH11L   | 触摸按键 11 门檻值低字节   | FB67H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH12H   | 触摸按键 12 门檻值高字节   | FB68H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH12L   | 触摸按键 12 门檻值低字节   | FB69H |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH13H   | 触摸按键 13 门檻值高字节   | FB6AH |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH13L   | 触摸按键 13 门檻值低字节   | FB6BH |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH14H   | 触摸按键 14 门檻值高字节   | FB6CH |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH14L   | 触摸按键 14 门檻值低字节   | FB6DH |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH15H   | 触摸按键 15 门檻值高字节   | FB6EH |                |       |       |       |               |              |        |           | 0000,0000 |
| TSTH15L   | 触摸按键 15 门檻值低字节   | FB6FH |                |       |       |       |               |              |        |           | 0000,0000 |
| LCDCFG    | LCD 配置寄存器        | FB80H | CLKSEL         | -     | -     | -     | VRLPSEL       | VLCDSEL[2:0] |        |           | 0xxx,0000 |
| LCDCFG2   | LCD 配置寄存器 2      | FB81H | -              | -     | -     | -     | SEG3PS        | SEG2PS       | SEG1PS | SEG0PS    | xxxx,0000 |
| DBLEN     | 死区时间长度配置         | FB82H | -              | -     | -     | -     | -             | DBLEN[2:0]   |        |           | xxxx,x000 |
| COMLENL   | COM 时间长度配置低位     | FB83H | COMLEN[7:0]    |       |       |       |               |              |        |           | 0000,0000 |
| COMLENM   | COM 时间长度配置中位     | FB84H | COMLEN[15:8]   |       |       |       |               |              |        |           | 0000,0000 |
| COMLENH   | COM 时间长度配置高位     | FB85H | -              | -     | -     | -     | COMLEN[19:16] |              |        | xxxx,0000 |           |
| BLINKRATE | 闪烁率配置寄存器         | FB86H | BLANKRATE[7:0] |       |       |       |               |              |        |           | 1000,0000 |
| LCDCR     | LCD 控制寄存器        | FB87H | -              | -     | -     | -     | -             | ACTMODE[1:0] |        | ENLCD     | xxxx,x000 |
| COMON     | COM 线使能寄存器       | FB88H | -              | -     | -     | -     | COM3          | COM2         | COM1   | COM0      | xxxx,0000 |
| SEGON1    | SEG 线使能寄存器 1     | FB8AH | SEG7           | SEG6  | SEG5  | SEG4  | SEG3          | SEG2         | SEG1   | SEG0      | 0000,0000 |
| SEGON2    | SEG 线使能寄存器 2     | FB8BH | SEG15          | SEG14 | SEG13 | SEG12 | SEG11         | SEG10        | SEG9   | SEG8      | 0000,0000 |
| SEGON3    | SEG 线使能寄存器 3     | FB8CH | SEG23          | SEG22 | SEG21 | SEG20 | SEG19         | SEG18        | SEG17  | SEG16     | 0000,0000 |
| SEGON4    | SEG 线使能寄存器 4     | FB8DH | SEG31          | SEG30 | SEG29 | SEG28 | SEG27         | SEG26        | SEG25  | SEG24     | 0000,0000 |
| SEGON5    | SEG 线使能寄存器 5     | FB8EH | SEG39          | SEG38 | SEG37 | SEG36 | SEG35         | SEG34        | SEG33  | SEG32     | 0000,0000 |
| C0SEGV0   | C0SEG7_0 数据寄存器   | FB90H | C0S7           | C0S6  | C0S5  | C0S4  | C0S3          | C0S2         | C0S1   | C0S0      | 0000,0000 |
| C0SEGV1   | C0SEG15_8 数据寄存器  | FB91H | C0S15          | C0S14 | C0S13 | C0S12 | C0S11         | C0S10        | C0S9   | C0S8      | 0000,0000 |
| C0SEGV2   | C0SEG23_16 数据寄存器 | FB92H | C0S23          | C0S22 | C0S21 | C0S20 | C0S19         | C0S18        | C0S17  | C0S16     | 0000,0000 |
| C0SEGV3   | C0SEG31_24 数据寄存器 | FB93H | C0S31          | C0S30 | C0S29 | C0S28 | C0S27         | C0S26        | C0S25  | C0S24     | 0000,0000 |

|               |                  |       |           |           |        |       |                |       |             |        |           |  |
|---------------|------------------|-------|-----------|-----------|--------|-------|----------------|-------|-------------|--------|-----------|--|
| C0SEGV4       | C0SEG39_32 数据寄存器 | FB94H | C0S39     | C0S38     | C0S37  | C0S36 | C0S35          | C0S34 | C0S33       | C0S32  | 0000,0000 |  |
| C1SEGV0       | C1SEG7_0 数据寄存器   | FB98H | C1S7      | C1S6      | C1S5   | C1S4  | C1S3           | C1S2  | C1S1        | C1S0   | 0000,0000 |  |
| C1SEGV1       | C1SEG15_8 数据寄存器  | FB99H | C1S15     | C1S14     | C1S13  | C1S12 | C1S11          | C1S10 | C1S9        | C1S8   | 0000,0000 |  |
| C1SEGV2       | C1SEG23_16 数据寄存器 | FB9AH | C1S23     | C1S22     | C1S21  | C1S20 | C1S19          | C1S18 | C1S17       | C1S16  | 0000,0000 |  |
| C1SEGV3       | C1SEG31_24 数据寄存器 | FB9BH | C1S31     | C1S30     | C1S29  | C1S28 | C1S27          | C1S26 | C1S25       | C1S24  | 0000,0000 |  |
| C1SEGV4       | C1SEG39_32 数据寄存器 | FB9CH | C1S39     | C1S38     | C1S37  | C1S36 | C1S35          | C1S34 | C1S33       | C1S32  | 0000,0000 |  |
| C2SEGV0       | C2SEG7_0 数据寄存器   | FBA0H | C2S7      | C2S6      | C2S5   | C2S4  | C2S3           | C2S2  | C2S1        | C2S0   | 0000,0000 |  |
| C2SEGV1       | C2SEG15_8 数据寄存器  | FBA1H | C2S15     | C2S14     | C2S13  | C2S12 | C2S11          | C2S10 | C2S9        | C2S8   | 0000,0000 |  |
| C2SEGV2       | C2SEG23_16 数据寄存器 | FBA2H | C2S23     | C2S22     | C2S21  | C2S20 | C2S19          | C2S18 | C2S17       | C2S16  | 0000,0000 |  |
| C2SEGV3       | C2SEG31_24 数据寄存器 | FBA3H | C2S31     | C2S30     | C2S29  | C2S28 | C2S27          | C2S26 | C2S25       | C2S24  | 0000,0000 |  |
| C2SEGV4       | C2SEG39_32 数据寄存器 | FBA4H | C2S39     | C2S38     | C2S37  | C2S36 | C2S35          | C2S34 | C2S33       | C2S32  | 0000,0000 |  |
| C3SEGV0       | C3SEG7_0 数据寄存器   | FBA8H | C3S7      | C3S6      | C3S5   | C3S4  | C3S3           | C3S2  | C3S1        | C3S0   | 0000,0000 |  |
| C3SEGV1       | C3SEG15_8 数据寄存器  | FBA9H | C3S15     | C3S14     | C3S13  | C3S12 | C3S11          | C3S10 | C3S9        | C3S8   | 0000,0000 |  |
| C3SEGV2       | C3SEG23_16 数据寄存器 | FBAAH | C3S23     | C3S22     | C3S21  | C3S20 | C3S19          | C3S18 | C3S17       | C3S16  | 0000,0000 |  |
| C3SEGV3       | C3SEG31_24 数据寄存器 | FBABH | C3S31     | C3S30     | C3S29  | C3S28 | C3S27          | C3S26 | C3S25       | C3S24  | 0000,0000 |  |
| C3SEGV4       | C3SEG39_32 数据寄存器 | FBACH | C3S39     | C3S38     | C3S37  | C3S36 | C3S35          | C3S34 | C3S33       | C3S32  | 0000,0000 |  |
| HSPWMA_CFG    | 高速 PWMA 配置寄存器    | FBF0H | -         | -         | -      | -     | AUTORD         | INTEN | ASYNCEN     | 1      | xxxx,0001 |  |
| HSPWMA_ADR    | 高速 PWMA 地址寄存器    | FBF1H | RW/BUSY   | ADDR[6:0] |        |       |                |       |             |        | 0000,0000 |  |
| HSPWMA_DAT    | 高速 PWMA 数据寄存器    | FBF2H | DATA[7:0] |           |        |       |                |       |             |        | 0000,0000 |  |
| HSPWMB_CFG    | 高速 PWMB 配置寄存器    | FBF4H | -         | -         | -      | -     | AUTORD         | INTEN | ASYNCEN     | 1      | xxxx,0001 |  |
| HSPWMB_ADR    | 高速 PWMB 地址寄存器    | FBF5H | RW/BUSY   | ADDR[6:0] |        |       |                |       |             |        | 0000,0000 |  |
| HSPWMB_DAT    | 高速 PWMB 数据寄存器    | FBF6H | DATA[7:0] |           |        |       |                |       |             |        | 0000,0000 |  |
| DMA_M2M_CFG   | M2M_DMA 配置寄存器    | FA00H | M2MIE     | -         | TXACO  | RXACO | M2MIP[1:0]     |       | M2MPTY[1:0] |        | 0x00,0000 |  |
| DMA_M2M_CR    | M2M_DMA 控制寄存器    | FA01H | ENM2M     | TRIG      | -      | -     | -              | -     | -           | -      | 00xx,xxxx |  |
| DMA_M2M_STA   | M2M_DMA 状态寄存器    | FA02H | -         | -         | -      | -     | -              | -     | -           | M2MIF  | xxxx,xxx0 |  |
| DMA_M2M_AMT   | M2M_DMA 传输总字节数   | FA03H |           |           |        |       |                |       |             |        | 0000,0000 |  |
| DMA_M2M_DONE  | M2M_DMA 传输完成字节数  | FA04H |           |           |        |       |                |       |             |        | 0000,0000 |  |
| DMA_M2M_TXAH  | M2M_DMA 发送高地址    | FA05H |           |           |        |       |                |       |             |        | 0000,0000 |  |
| DMA_M2M_TXAL  | M2M_DMA 发送低地址    | FA06H |           |           |        |       |                |       |             |        | 0000,0000 |  |
| DMA_M2M_RXAH  | M2M_DMA 接收高地址    | FA07H |           |           |        |       |                |       |             |        | 0000,0000 |  |
| DMA_M2M_RXAL  | M2M_DMA 接收低地址    | FA08H |           |           |        |       |                |       |             |        | 0000,0000 |  |
| DMA_ADC_CFG   | ADC_DMA 配置寄存器    | FA10H | ADCIE     | -         | -      | -     | ADCMIP[1:0]    |       | ADCPTY[1:0] |        | 0xxx,0000 |  |
| DMA_ADC_CR    | ADC_DMA 控制寄存器    | FA11H | ENADC     | TRIG      | -      | -     | -              | -     | -           | -      | 00xx,xxxx |  |
| DMA_ADC_STA   | ADC_DMA 状态寄存器    | FA12H | -         | -         | -      | -     | -              | -     | -           | ADCIF  | xxxx,xxx0 |  |
| DMA_ADC_RXAH  | ADC_DMA 接收高地址    | FA17H |           |           |        |       |                |       |             |        | 0000,0000 |  |
| DMA_ADC_RXAL  | ADC_DMA 接收低地址    | FA18H |           |           |        |       |                |       |             |        | 0000,0000 |  |
| DMA_ADC_CFG2  | ADC_DMA 配置寄存器 2  | FA19H | -         | -         | -      | -     | CVTIMESEL[3:0] |       |             |        | xxxx,0000 |  |
| DMA_ADC_CHSW0 | ADC_DMA 通道使能     | FA1AH | CH7       | CH6       | CH5    | CH4   | CH3            | CH2   | CH1         | CH0    | 0000,0001 |  |
| DMA_ADC_CHSW1 | ADC_DMA 通道使能     | FA1BH | CH15      | CH14      | CH13   | CH12  | CH11           | CH10  | CH9         | CH8    | 1000,0000 |  |
| DMA_SPI_CFG   | SPI_DMA 配置寄存器    | FA20H | SPIIE     | ACT_TX    | ACT_RX | -     | SPIIP[1:0]     |       | SPIPTY[1:0] |        | 000x,0000 |  |
| DMA_SPI_CR    | SPI_DMA 控制寄存器    | FA21H | ENSPI     | TRIG_M    | TRIG_S | -     | -              | -     | -           | CLRIFO | 000x,xxx0 |  |
| DMA_SPI_STA   | SPI_DMA 状态寄存器    | FA22H | -         | -         | -      | -     | -              | TXOVW | RXLOSS      | SPIIF  | xxxx,x000 |  |
| DMA_SPI_AMT   | SPI_DMA 传输总字节数   | FA23H |           |           |        |       |                |       |             |        | 0000,0000 |  |
| DMA_SPI_DONE  | SPI_DMA 传输完成字节数  | FA24H |           |           |        |       |                |       |             |        | 0000,0000 |  |



|               |                  |       |        |      |      |   |             |       |              |         |           |
|---------------|------------------|-------|--------|------|------|---|-------------|-------|--------------|---------|-----------|
| DMA_SPI_TXAH  | SPI_DMA 发送高地址    | FA25H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_SPI_TXAL  | SPI_DMA 发送低地址    | FA26H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_SPI_RXAH  | SPI_DMA 接收高地址    | FA27H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_SPI_RXAL  | SPI_DMA 接收低地址    | FA28H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_SPI_CFG2  | SPI_DMA 配置寄存器 2  | FA29H | -      | -    | -    | - | -           | WRPSS | SSS[1:0]     |         | xxxx,x000 |
| DMA_UR1T_CFG  | UR1T_DMA 配置寄存器   | FA30H | UR1TIE | -    | -    | - | UR1TIP[1:0] |       | UR1TPTY[1:0] |         | 0xxx,0000 |
| DMA_UR1T_CR   | UR1T_DMA 控制寄存器   | FA31H | ENUR1T | TRIG | -    | - | -           | -     | -            | -       | 00xx,xxxx |
| DMA_UR1T_STA  | UR1T_DMA 状态寄存器   | FA32H | -      | -    | -    | - | -           | TXOVW | -            | UR1TIF  | xxxx,x0x0 |
| DMA_UR1T_AMT  | UR1T_DMA 传输总字节数  | FA33H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR1T_DONE | UR1T_DMA 传输完成字节数 | FA34H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR1T_TXAH | UR1T_DMA 发送高地址   | FA35H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR1T_TXAL | UR1T_DMA 发送低地址   | FA36H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR1R_CFG  | UR1R_DMA 配置寄存器   | FA38H | UR1RIE | -    | -    | - | UR1RIP[1:0] |       | UR1RPTY[1:0] |         | 0xxx,0000 |
| DMA_UR1R_CR   | UR1R_DMA 控制寄存器   | FA39H | ENUR1R | -    | TRIG | - | -           | -     | -            | CLRFIFO | 0x0x,xxx0 |
| DMA_UR1R_STA  | UR1R_DMA 状态寄存器   | FA3AH | -      | -    | -    | - | -           | -     | RXLOSS       | UR1RIF  | xxxx,xx00 |
| DMA_UR1R_AMT  | UR1R_DMA 传输总字节数  | FA3BH |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR1R_DONE | UR1R_DMA 传输完成字节数 | FA3CH |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR1R_RXAH | UR1R_DMA 接收高地址   | FA3DH |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR1R_RXAL | UR1R_DMA 接收低地址   | FA3EH |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR2T_CFG  | UR2T_DMA 配置寄存器   | FA40H | UR2TIE | -    | -    | - | UR2TIP[1:0] |       | UR2TPTY[1:0] |         | 0xxx,0000 |
| DMA_UR2T_CR   | UR2T_DMA 控制寄存器   | FA41H | ENUR2T | TRIG | -    | - | -           | -     | -            | -       | 00xx,xxxx |
| DMA_UR2T_STA  | UR2T_DMA 状态寄存器   | FA42H | -      | -    | -    | - | -           | TXOVW | -            | UR2TIF  | xxxx,x0x0 |
| DMA_UR2T_AMT  | UR2T_DMA 传输总字节数  | FA43H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR2T_DONE | UR2T_DMA 传输完成字节数 | FA44H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR2T_TXAH | UR2T_DMA 发送高地址   | FA45H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR2T_TXAL | UR2T_DMA 发送低地址   | FA46H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR2R_CFG  | UR2R_DMA 配置寄存器   | FA48H | UR2RIE | -    | -    | - | UR2RIP[1:0] |       | UR2RPTY[1:0] |         | 0xxx,0000 |
| DMA_UR2R_CR   | UR2R_DMA 控制寄存器   | FA49H | ENUR2R | -    | TRIG | - | -           | -     | -            | CLRFIFO | 0x0x,xxx0 |
| DMA_UR2R_STA  | UR2R_DMA 状态寄存器   | FA4AH | -      | -    | -    | - | -           | -     | RXLOSS       | UR2RIF  | xxxx,xx00 |
| DMA_UR2R_AMT  | UR2R_DMA 传输总字节数  | FA4BH |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR2R_DONE | UR2R_DMA 传输完成字节数 | FA4CH |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR2R_RXAH | UR2R_DMA 接收高地址   | FA4DH |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR2R_RXAL | UR2R_DMA 接收低地址   | FA4EH |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR3T_CFG  | UR3T_DMA 配置寄存器   | FA50H | UR3TIE | -    | -    | - | UR3TIP[1:0] |       | UR3TPTY[1:0] |         | 0xxx,0000 |
| DMA_UR3T_CR   | UR3T_DMA 控制寄存器   | FA51H | ENUR3T | TRIG | -    | - | -           | -     | -            | -       | 00xx,xxxx |
| DMA_UR3T_STA  | UR3T_DMA 状态寄存器   | FA52H | -      | -    | -    | - | -           | TXOVW | -            | UR3TIF  | xxxx,x0x0 |
| DMA_UR3T_AMT  | UR3T_DMA 传输总字节数  | FA53H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR3T_DONE | UR3T_DMA 传输完成字节数 | FA54H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR3T_TXAH | UR3T_DMA 发送高地址   | FA55H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR3T_TXAL | UR3T_DMA 发送低地址   | FA56H |        |      |      |   |             |       |              |         | 0000,0000 |
| DMA_UR3R_CFG  | UR3R_DMA 配置寄存器   | FA58H | UR3RIE | -    | -    | - | UR3RIP[1:0] |       | UR3RPTY[1:0] |         | 0xxx,0000 |
| DMA_UR3R_CR   | UR3R_DMA 控制寄存器   | FA59H | ENUR3R | -    | TRIG | - | -           | -     | -            | CLRFIFO | 0x0x,xxx0 |
| DMA_UR3R_STA  | UR3R_DMA 状态寄存器   | FA5AH | -      | -    | -    | - | -           | -     | RXLOSS       | UR3RIF  | xxxx,xx00 |
| DMA_UR3R_AMT  | UR3R_DMA 传输总字节数  | FA5BH |        |      |      |   |             |       |              |         | 0000,0000 |



|               |                  |       |        |        |        |        |             |       |              |        |           |
|---------------|------------------|-------|--------|--------|--------|--------|-------------|-------|--------------|--------|-----------|
| DMA_UR3R_DONE | UR3R_DMA 传输完成字节数 | FA5CH |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_UR3R_RXAH | UR3R_DMA 接收高地址   | FA5DH |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_UR3R_RXAL | UR3R_DMA 接收低地址   | FA5EH |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_UR4T_CFG  | UR4T_DMA 配置寄存器   | FA60H | UR4TIE | -      | -      | -      | UR4TIP[1:0] |       | UR4TPTY[1:0] |        | 0xxx,0000 |
| DMA_UR4T_CR   | UR4T_DMA 控制寄存器   | FA61H | ENUR4T | TRIG   | -      | -      | -           | -     | -            | -      | 00xx,xxxx |
| DMA_UR4T_STA  | UR4T_DMA 状态寄存器   | FA62H | -      | -      | -      | -      | -           | TXOVW | -            | UR4TIF | xxxx,x0x0 |
| DMA_UR4T_AMT  | UR4T_DMA 传输总字节数  | FA63H |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_UR4T_DONE | UR4T_DMA 传输完成字节数 | FA64H |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_UR4T_TXAH | UR4T_DMA 发送高地址   | FA65H |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_UR4T_TXAL | UR4T_DMA 发送低地址   | FA66H |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_UR4R_CFG  | UR4R_DMA 配置寄存器   | FA68H | UR4RIE | -      | -      | -      | UR4RIP[1:0] |       | UR4RPTY[1:0] |        | 0xxx,0000 |
| DMA_UR4R_CR   | UR4R_DMA 控制寄存器   | FA69H | ENUR4R | -      | TRIG   | -      | -           | -     | -            | CLRIFO | 0x0x,xxx0 |
| DMA_UR4R_STA  | UR4R_DMA 状态寄存器   | FA6AH | -      | -      | -      | -      | -           | -     | RXLOSS       | UR4RIF | xxxx,xx00 |
| DMA_UR4R_AMT  | UR4R_DMA 传输总字节数  | FA6BH |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_UR4R_DONE | UR4R_DMA 传输完成字节数 | FA6CH |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_UR4R_RXAH | UR4R_DMA 接收高地址   | FA6DH |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_UR4R_RXAL | UR4R_DMA 接收低地址   | FA6EH |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_LCM_CFG   | LCM_DMA 配置寄存器    | FA70H | LCMIE  | -      | -      | -      | LCMIP[1:0]  |       | LCMPTY[1:0]  |        | 0xxx,0000 |
| DMA_LCM_CR    | LCM_DMA 控制寄存器    | FA71H | ENLCM  | TRIGWC | TRIGWD | TRIGRC | TRIGRD      | -     | -            | -      | 0000,0xxx |
| DMA_LCM_STA   | LCM_DMA 状态寄存器    | FA72H | -      | -      | -      | -      | -           | -     | TXOVW        | LCMIF  | xxxx,xx00 |
| DMA_LCM_AMT   | LCM_DMA 传输总字节数   | FA73H |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_LCM_DONE  | LCM_DMA 传输完成字节数  | FA74H |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_LCM_TXAH  | LCM_DMA 发送高地址    | FA75H |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_LCM_TXAL  | LCM_DMA 发送低地址    | FA76H |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_LCM_RXAH  | LCM_DMA 接收高地址    | FA77H |        |        |        |        |             |       |              |        | 0000,0000 |
| DMA_LCM_RXAL  | LCM_DMA 接收低地址    | FA78H |        |        |        |        |             |       |              |        | 0000,0000 |

注：特殊功能寄存器初始值意义

0： 初始值为 0；

1： 初始值为 1；

n： 初始值与 ISP 下载时的硬件选项有关；

x： 不存在这个位，初始值不确定

# 10 I/O 口

| 产品线              | 最多 I/O 口数量 |
|------------------|------------|
| STC8H1K08 系列     | 17         |
| STC8H1K28 系列     | 29         |
| STC8H3K64S4 系列   | 45         |
| STC8H3K64S2 系列   | 45         |
| STC8H8K64U 系列    | 60         |
| STC8H4K64TL 系列   | 44         |
| STC8H4K64TLCD 系列 | 60         |
| STC8H1K08T 系列    | 16         |

STC8H 系列单片机所有的 I/O 口均有 4 种工作模式：准双向口/弱上拉（标准 8051 输出口模式）、推挽输出/强上拉、高阻输入（电流既不能流入也不能流出）、开漏输出。可使用软件对 I/O 口的工作模式进行配置。

## 关于 I/O 的注意事项：

- 1、 P3.0 和 P3.1 口上电后的状态为弱上拉/准双向口模式
- 2、 除 P3.0 和 P3.1 外，其余所有 IO 口上电后的状态均为高阻输入状态，用户在使用 IO 口前必须先设置 IO 口模式
- 3、 芯片上电时如果不需要使用 USB 进行 ISP 下载，P3.0/P3.1/P3.2 这 3 个 I/O 口不能同时为低电平，否则会进入 USB 下载模式而无法运行用户代码
- 4、 芯片上电时，若 P3.0 和 P3.1 同时为低电平，P3.2 口会短时间由高阻输入状态切换到双向口模式，用以读取 P3.2 口外部状态来判断是否需要进入 USB 下载模式
- 5、 当使用 P5.4 当作复位脚时，这个端口内部的 4K 上拉电阻会一直打开；但 P5.4 做普通 I/O 口时，基于这个 I/O 口与复位脚共享管脚的特殊考量，端口内部的 4K 上拉电阻依然会打开大约 6.5 毫秒时间，再自动关闭（当用户的电路设计需要使用 P5.4 口驱动外部电路时，请务必考虑上电瞬间会有 6.5 毫秒时间的高电平的问题）

## 10.1 I/O 口相关寄存器

| 符号 | 描述    | 地址  | 位地址与符号 |     |     |     |     |     |     |     | 复位值       |
|----|-------|-----|--------|-----|-----|-----|-----|-----|-----|-----|-----------|
|    |       |     | B7     | B6  | B5  | B4  | B3  | B2  | B1  | B0  |           |
| P0 | P0 端口 | 80H | P07    | P06 | P05 | P04 | P03 | P02 | P01 | P00 | 1111,1111 |
| P1 | P1 端口 | 90H | P17    | P16 | P15 | P14 | P13 | P12 | P11 | P10 | 1111,1111 |
| P2 | P2 端口 | A0H | P27    | P26 | P25 | P24 | P23 | P22 | P21 | P20 | 1111,1111 |
| P3 | P3 端口 | B0H | P37    | P36 | P35 | P34 | P33 | P32 | P31 | P30 | 1111,1111 |
| P4 | P4 端口 | C0H | P47    | P46 | P45 | P44 | P43 | P42 | P41 | P40 | 1111,1111 |
| P5 | P5 端口 | C8H | -      | -   | P55 | P54 | P53 | P52 | P51 | P50 | xx11,1111 |

|      |             |     |       |       |       |       |       |       |       |       |           |
|------|-------------|-----|-------|-------|-------|-------|-------|-------|-------|-------|-----------|
| P6   | P6 端口       | E8H | P67   | P66   | P65   | P64   | P63   | P62   | P61   | P60   | 1111,1111 |
| P7   | P7 端口       | F8H | P77   | P76   | P75   | P74   | P73   | P72   | P71   | P70   | 1111,1111 |
| P0M1 | P0 口配置寄存器 1 | 93H | P07M1 | P06M1 | P05M1 | P04M1 | P03M1 | P02M1 | P01M1 | P00M1 | 1111,1111 |
| P0M0 | P0 口配置寄存器 0 | 94H | P07M0 | P06M0 | P05M0 | P04M0 | P03M0 | P02M0 | P01M0 | P00M0 | 0000,0000 |
| P1M1 | P1 口配置寄存器 1 | 91H | P17M1 | P16M1 | P15M1 | P14M1 | P13M1 | P12M1 | P11M1 | P10M1 | 1111,1111 |
| P1M0 | P1 口配置寄存器 0 | 92H | P17M0 | P16M0 | P15M0 | P14M0 | P13M0 | P12M0 | P11M0 | P10M0 | 0000,0000 |
| P2M1 | P2 口配置寄存器 1 | 95H | P27M1 | P26M1 | P25M1 | P24M1 | P23M1 | P22M1 | P21M1 | P20M1 | 1111,1111 |
| P2M0 | P2 口配置寄存器 0 | 96H | P27M0 | P26M0 | P25M0 | P24M0 | P23M0 | P22M0 | P21M0 | P20M0 | 0000,0000 |
| P3M1 | P3 口配置寄存器 1 | B1H | P37M1 | P36M1 | P35M1 | P34M1 | P33M1 | P32M1 | P31M1 | P30M1 | 1111,1100 |
| P3M0 | P3 口配置寄存器 0 | B2H | P37M0 | P36M0 | P35M0 | P34M0 | P33M0 | P32M0 | P31M0 | P30M0 | 0000,0000 |
| P4M1 | P4 口配置寄存器 1 | B3H | P47M1 | P46M1 | P45M1 | P44M1 | P43M1 | P42M1 | P41M1 | P40M1 | 1111,1111 |
| P4M0 | P4 口配置寄存器 0 | B4H | P47M0 | P46M0 | P45M0 | P44M0 | P43M0 | P42M0 | P41M0 | P40M0 | 0000,0000 |
| P5M1 | P5 口配置寄存器 1 | C9H | -     | -     | P55M1 | P54M1 | P53M1 | P52M1 | P51M1 | P50M1 | xx11,1111 |
| P5M0 | P5 口配置寄存器 0 | CAH | -     | -     | P55M0 | P54M0 | P53M0 | P52M0 | P51M0 | P50M0 | xx00,0000 |
| P6M1 | P6 口配置寄存器 1 | CBH | P67M1 | P66M1 | P65M1 | P64M1 | P63M1 | P62M1 | P61M1 | P60M1 | 1111,1111 |
| P6M0 | P6 口配置寄存器 0 | CCH | P67M0 | P66M0 | P65M0 | P64M0 | P63M0 | P62M0 | P61M0 | P60M0 | 0000,0000 |
| P7M1 | P7 口配置寄存器 1 | E1H | P77M1 | P76M1 | P75M1 | P74M1 | P73M1 | P72M1 | P71M1 | P70M1 | 1111,1111 |
| P7M0 | P7 口配置寄存器 0 | E2H | P77M0 | P76M0 | P75M0 | P74M0 | P73M0 | P72M0 | P71M0 | P70M0 | 0000,0000 |

| 符号    | 描述             | 地址    | 位地址与符号 |        |        |        |        |        |        |        | 复位值       |
|-------|----------------|-------|--------|--------|--------|--------|--------|--------|--------|--------|-----------|
|       |                |       | B7     | B6     | B5     | B4     | B3     | B2     | B1     | B0     |           |
| P0PU  | P0 口上拉电阻控制寄存器  | FE10H | P07PU  | P06PU  | P05PU  | P04PU  | P03PU  | P02PU  | P01PU  | P00PU  | 0000,0000 |
| P1PU  | P1 口上拉电阻控制寄存器  | FE11H | P17PU  | P16PU  | P15PU  | P14PU  | P13PU  | P12PU  | P11PU  | P10PU  | 0000,0000 |
| P2PU  | P2 口上拉电阻控制寄存器  | FE12H | P27PU  | P26PU  | P25PU  | P24PU  | P23PU  | P22PU  | P21PU  | P20PU  | 0000,0000 |
| P3PU  | P3 口上拉电阻控制寄存器  | FE13H | P37PU  | P36PU  | P35PU  | P34PU  | P33PU  | P32PU  | P31PU  | P30PU  | 0000,0000 |
| P4PU  | P4 口上拉电阻控制寄存器  | FE14H | P47PU  | P46PU  | P45PU  | P44PU  | P43PU  | P42PU  | P41PU  | P40PU  | 0000,0000 |
| P5PU  | P5 口上拉电阻控制寄存器  | FE15H | -      | -      | P55PU  | P54PU  | P53PU  | P52PU  | P51PU  | P50PU  | xx00,0000 |
| P6PU  | P6 口上拉电阻控制寄存器  | FE16H | P67PU  | P66PU  | P65PU  | P64PU  | P63PU  | P62PU  | P61PU  | P60PU  | 0000,0000 |
| P7PU  | P7 口上拉电阻控制寄存器  | FE17H | P77PU  | P76PU  | P75PU  | P74PU  | P73PU  | P72PU  | P71PU  | P70PU  | 0000,0000 |
| P0NCS | P0 口施密特触发控制寄存器 | FE18H | P07NCS | P06NCS | P05NCS | P04NCS | P03NCS | P02NCS | P01NCS | P00NCS | 0000,0000 |
| P1NCS | P1 口施密特触发控制寄存器 | FE19H | P17NCS | P16NCS | P15NCS | P14NCS | P13NCS | P12NCS | P11NCS | P10NCS | 0000,0000 |
| P2NCS | P2 口施密特触发控制寄存器 | FE1AH | P27NCS | P26NCS | P25NCS | P24NCS | P23NCS | P22NCS | P21NCS | P20NCS | 0000,0000 |
| P3NCS | P3 口施密特触发控制寄存器 | FE1BH | P37NCS | P36NCS | P35NCS | P34NCS | P33NCS | P32NCS | P31NCS | P30NCS | 0000,0000 |
| P4NCS | P4 口施密特触发控制寄存器 | FE1CH | P47NCS | P46NCS | P45NCS | P44NCS | P43NCS | P42NCS | P41NCS | P40NCS | 0000,0000 |
| P5NCS | P5 口施密特触发控制寄存器 | FE1DH | -      | -      | P55NCS | P54NCS | P53NCS | P52NCS | P51NCS | P50NCS | xx00,0000 |
| P6NCS | P6 口施密特触发控制寄存器 | FE1EH | P67NCS | P66NCS | P65NCS | P64NCS | P63NCS | P62NCS | P61NCS | P60NCS | 0000,0000 |
| P7NCS | P7 口施密特触发控制寄存器 | FE1FH | P77NCS | P76NCS | P75NCS | P74NCS | P73NCS | P72NCS | P71NCS | P70NCS | 0000,0000 |
| P0SR  | P0 口电平转换速率寄存器  | FE20H | P07SR  | P06SR  | P05SR  | P04SR  | P03SR  | P02SR  | P01SR  | P00SR  | 1111,1111 |
| P1SR  | P1 口电平转换速率寄存器  | FE21H | P17SR  | P16SR  | P15SR  | P14SR  | P13SR  | P12SR  | P11SR  | P10SR  | 1111,1111 |
| P2SR  | P2 口电平转换速率寄存器  | FE22H | P27SR  | P26SR  | P25SR  | P24SR  | P23SR  | P22SR  | P21SR  | P20SR  | 1111,1111 |
| P3SR  | P3 口电平转换速率寄存器  | FE23H | P37SR  | P36SR  | P35SR  | P34SR  | P33SR  | P32SR  | P31SR  | P30SR  | 1111,1111 |
| P4SR  | P4 口电平转换速率寄存器  | FE24H | P47SR  | P46SR  | P45SR  | P44SR  | P43SR  | P42SR  | P41SR  | P40SR  | 1111,1111 |
| P5SR  | P5 口电平转换速率寄存器  | FE25H | -      | -      | P55SR  | P54SR  | P53SR  | P52SR  | P51SR  | P50SR  | xx11,1111 |
| P6SR  | P6 口电平转换速率寄存器  | FE26H | P67SR  | P66SR  | P65SR  | P64SR  | P63SR  | P62SR  | P61SR  | P60SR  | 1111,1111 |

|      |               |       |       |       |       |       |       |       |       |       |           |
|------|---------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-----------|
| P7SR | P7 口电平转换速率寄存器 | FE27H | P77SR | P76SR | P75SR | P74SR | P73SR | P72SR | P71SR | P70SR | 1111,1111 |
| P0DR | P0 口驱动电流控制寄存器 | FE28H | P07DR | P06DR | P05DR | P04DR | P03DR | P02DR | P01DR | P00DR | 1111,1111 |
| P1DR | P1 口驱动电流控制寄存器 | FE29H | P17DR | P16DR | P15DR | P14DR | P13DR | P12DR | P11DR | P10DR | 1111,1111 |
| P2DR | P2 口驱动电流控制寄存器 | FE2AH | P27DR | P26DR | P25DR | P24DR | P23DR | P22DR | P21DR | P20DR | 1111,1111 |
| P3DR | P3 口驱动电流控制寄存器 | FE2BH | P37DR | P36DR | P35DR | P34DR | P33DR | P32DR | P31DR | P30DR | 1111,1111 |
| P4DR | P4 口驱动电流控制寄存器 | FE2CH | P47DR | P46DR | P45DR | P44DR | P43DR | P42DR | P41DR | P40DR | 1111,1111 |
| P5DR | P5 口驱动电流控制寄存器 | FE2DH | -     | -     | P55DR | P54DR | P53DR | P52DR | P51DR | P50DR | xx11,1111 |
| P6DR | P6 口驱动电流控制寄存器 | FE2EH | P67DR | P66DR | P65DR | P64DR | P63DR | P62DR | P61DR | P60DR | 1111,1111 |
| P7DR | P7 口驱动电流控制寄存器 | FE2FH | P77DR | P76DR | P75DR | P74DR | P73DR | P72DR | P71DR | P70DR | 1111,1111 |
| P0IE | P0 口输入使能控制寄存器 | FE30H | P07IE | P06IE | P05IE | P04IE | P03IE | P02IE | P01IE | P00IE | 1111,1111 |
| P1IE | P1 口输入使能控制寄存器 | FE31H | P17IE | P16IE | P15IE | P14IE | P13IE | P12IE | P11IE | P10IE | 1111,1111 |
| P2IE | P2 口输入使能控制寄存器 | FE32H | P27IE | P26IE | P25IE | P24IE | P23IE | P22IE | P21IE | P20IE | 1111,1111 |
| P3IE | P3 口输入使能控制寄存器 | FE33H | P37IE | P36IE | P35IE | P34IE | P33IE | P32IE | P31IE | P30IE | 1111,1111 |
| P4IE | P4 口输入使能控制寄存器 | FE34H | P47IE | P46IE | P45IE | P44IE | P43IE | P42IE | P41IE | P40IE | 1111,1111 |
| P5IE | P5 口输入使能控制寄存器 | FE35H | -     | -     | P55IE | P54IE | P53IE | P52IE | P41IE | P50IE | xx11,1111 |
| P6IE | P6 口输入使能控制寄存器 | FE36H | P67IE | P66IE | P65IE | P64IE | P63IE | P62IE | P41IE | P60IE | 1111,1111 |
| P7IE | P7 口输入使能控制寄存器 | FE37H | P77IE | P76IE | P75IE | P74IE | P73IE | P72IE | P41IE | P70IE | 1111,1111 |

### 10.1.1 端口数据寄存器 (Px)

| 符号 | 地址  | B7   | B6   | B5   | B4   | B3   | B2   | B1   | B0   |
|----|-----|------|------|------|------|------|------|------|------|
| P0 | 80H | P0.7 | P0.6 | P0.5 | P0.4 | P0.3 | P0.2 | P0.1 | P0.0 |
| P1 | 90H | P1.7 | P1.6 | P1.5 | P1.4 | P1.3 | P1.2 | P1.1 | P1.0 |
| P2 | A0H | P2.7 | P2.6 | P2.5 | P2.4 | P2.3 | P2.2 | P2.1 | P2.0 |
| P3 | B0H | P3.7 | P3.6 | P3.5 | P3.4 | P3.3 | P3.2 | P3.1 | P3.0 |
| P4 | C0H | P4.7 | P4.6 | P4.5 | P4.4 | P4.3 | P4.2 | P4.1 | P4.0 |
| P5 | C8H | -    | -    | P5.5 | P5.4 | P5.3 | P5.2 | P5.1 | P5.0 |
| P6 | E8H | P6.7 | P6.6 | P6.5 | P6.4 | P6.3 | P6.2 | P6.1 | P6.0 |
| P7 | F8H | P7.7 | P7.6 | P7.5 | P7.4 | P7.3 | P7.2 | P7.1 | P7.0 |

读写端口状态

写 0: 输出低电平到端口缓冲区

写 1: 输出高电平到端口缓冲区

读: 直接读端口管脚上的电平

### 10.1.2 端口模式配置寄存器 (PxM0, PxM1)

| 符号   | 地址  | B7    | B6    | B5     | B4     | B3     | B2     | B1     | B0     |
|------|-----|-------|-------|--------|--------|--------|--------|--------|--------|
| P0M0 | 94H | P07M0 | P06M0 | P05M0  | P04M0  | P03M0  | P02M0  | P01M0  | P00M0  |
| P0M1 | 93H | P07M1 | P06M1 | P05M1  | P04M1  | P03M1  | P02M1  | P01M1  | P00M1  |
| P1M0 | 92H | P17M0 | P16M0 | P15M0  | P14M0  | P13M0  | P12M0  | P11M0  | P10M0  |
| P1M1 | 91H | P17M1 | P16M1 | P15M1  | P14M1  | P13M1  | P12M1  | P11M1  | P10M1  |
| P2M0 | 96H | P27M0 | P26M0 | P25M0  | P24M0  | P23M0  | P22M0  | P21M0  | P20M0  |
| P2M1 | 95H | P27M1 | P26M1 | P25M1  | P24M1  | P23M1  | P22M1  | P21M1  | P20M1  |
| P3M0 | B2H | P37M0 | P36M0 | P35M0  | P34M0  | P33M0  | P32M0  | P31M0  | P30M0  |
| P3M1 | B1H | P37M1 | P36M1 | P35M1  | P34M1  | P33M1  | P32M1  | P31M1  | P30M1  |
| P4M0 | B4H | P47M0 | P46M0 | P45M0  | P44M0  | P43M0  | P42M0  | P41M0  | P40M0  |
| P4M1 | B3H | P47M1 | P46M1 | P45M1  | P44M1  | P43M1  | P42M1  | P41M1  | P40M1  |
| P5M0 | CAH | -     | -     | P5.5M0 | P5.4M0 | P5.3M0 | P5.2M0 | P5.1M0 | P5.0M0 |
| P5M1 | C9H | -     | -     | P5.5M1 | P5.4M1 | P5.3M1 | P5.2M1 | P5.1M1 | P5.0M1 |
| P6M0 | CCH | P67M0 | P66M0 | P65M0  | P64M0  | P63M0  | P62M0  | P61M0  | P60M0  |
| P6M1 | CBH | P67M1 | P66M1 | P65M1  | P64M1  | P63M1  | P62M1  | P61M1  | P60M1  |
| P7M0 | E2H | P77M0 | P76M0 | P75M0  | P74M0  | P73M0  | P72M0  | P71M0  | P70M0  |
| P7M1 | E1H | P77M1 | P76M1 | P75M1  | P74M1  | P73M1  | P72M1  | P71M1  | P70M1  |

配置端口的模式

| PnM1.x | PnM0.x | Pn.x 口工作模式 |
|--------|--------|------------|
| 0      | 0      | 准双向口       |
| 0      | 1      | 推挽输出       |
| 1      | 0      | 高阻输入       |
| 1      | 1      | 开漏输出       |

注意: 当有I/O口被选择为ADC输入通道时, 必须设置PxM0/PxM1寄存器将I/O口模式设置为输入模式。另外如果MCU进入掉电模式/时钟停振模式后, 仍需要使能ADC通道, 则需要设置PxIE寄存器关闭数字输入,

才能保证不会有额外的耗电

### 10.1.3 端口上拉电阻控制寄存器 (PxPU)

| 符号   | 地址    | B7    | B6    | B5    | B4    | B3    | B2    | B1    | B0    |
|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| P0PU | FE10H | P07PU | P06PU | P05PU | P04PU | P03PU | P02PU | P01PU | P00PU |
| P1PU | FE11H | P17PU | P16PU | P15PU | P14PU | P13PU | P12PU | P11PU | P10PU |
| P2PU | FE12H | P27PU | P26PU | P25PU | P24PU | P23PU | P22PU | P21PU | P20PU |
| P3PU | FE13H | P37PU | P36PU | P35PU | P34PU | P33PU | P32PU | P31PU | P30PU |
| P4PU | FE14H | P47PU | P46PU | P45PU | P44PU | P43PU | P42PU | P41PU | P40PU |
| P5PU | FE15H | -     | -     | P55PU | P54PU | P53PU | P52PU | P51PU | P50PU |
| P6PU | FE16H | P67PU | P66PU | P65PU | P64PU | P63PU | P62PU | P61PU | P60PU |
| P7PU | FE17H | P77PU | P76PU | P75PU | P74PU | P73PU | P72PU | P71PU | P70PU |

端口内部4.1K上拉电阻控制位（注：P3.0和P3.1口上的上拉电阻可能会略小一些）

0：禁止端口内部的 4.1K 上拉电阻

1：使能端口内部的 4.1K 上拉电阻

### 10.1.4 端口施密特触发控制寄存器 (PxNCS)

| 符号    | 地址    | B7     | B6     | B5     | B4     | B3     | B2     | B1     | B0     |
|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|
| P0NCS | FE18H | P07NCS | P06NCS | P05NCS | P04NCS | P03NCS | P02NCS | P01NCS | P00NCS |
| P1NCS | FE19H | P17NCS | P16NCS | P15NCS | P14NCS | P13NCS | P12NCS | P11NCS | P10NCS |
| P2NCS | FE1AH | P27NCS | P26NCS | P25NCS | P24NCS | P23NCS | P22NCS | P21NCS | P20NCS |
| P3NCS | FE1BH | P37NCS | P36NCS | P35NCS | P34NCS | P33NCS | P32NCS | P31NCS | P30NCS |
| P4NCS | FE1CH | P47NCS | P46NCS | P45NCS | P44NCS | P43NCS | P42NCS | P41NCS | P40NCS |
| P5NCS | FE1DH | -      | -      | P55NCS | P54NCS | P53NCS | P52NCS | P51NCS | P50NCS |
| P6NCS | FE1EH | P67NCS | P66NCS | P65NCS | P64NCS | P63NCS | P62NCS | P61NCS | P60NCS |
| P7NCS | FE1FH | P77NCS | P76NCS | P75NCS | P74NCS | P73NCS | P72NCS | P71NCS | P70NCS |

端口施密特触发控制位

0：使能端口的施密特触发功能。（上电复位后默认使能施密特触发）

1：禁止端口的施密特触发功能。

## 10.1.5 端口电平转换速度控制寄存器 (PxSR)

| 符号   | 地址    | B7    | B6    | B5    | B4    | B3    | B2    | B1    | B0    | 复位值       |
|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-----------|
| P0SR | FE20H | P07SR | P06SR | P05SR | P04SR | P03SR | P02SR | P01SR | P00SR | 1111,1111 |
| P1SR | FE21H | P17SR | P16SR | P15SR | P14SR | P13SR | P12SR | P11SR | P10SR | 1111,1111 |
| P2SR | FE22H | P27SR | P26SR | P25SR | P24SR | P23SR | P22SR | P21SR | P20SR | 1111,1111 |
| P3SR | FE23H | P37SR | P36SR | P35SR | P34SR | P33SR | P32SR | P31SR | P30SR | 1111,1111 |
| P4SR | FE24H | P47SR | P46SR | P45SR | P44SR | P43SR | P42SR | P41SR | P40SR | 1111,1111 |
| P5SR | FE25H | -     | -     | P55SR | P54SR | P53SR | P52SR | P51SR | P50SR | xx11,1111 |
| P6SR | FE26H | P57SR | P66SR | P65SR | P64SR | P63SR | P62SR | P61SR | P60SR | 1111,1111 |
| P7SR | FE27H | P77SR | P76SR | P75SR | P74SR | P73SR | P72SR | P71SR | P70SR | 1111,1111 |

控制端口电平转换的速度

0: 电平转换速度快, 相应的上下冲会比较大

1: 电平转换速度慢, 相应的上下冲比较小

## 10.1.6 端口驱动电流控制寄存器 (PxDR)

| 符号   | 地址    | B7    | B6    | B5    | B4    | B3    | B2    | B1    | B0    | 复位值       |
|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-----------|
| P0DR | FE28H | P07DR | P06DR | P05DR | P04DR | P03DR | P02DR | P01DR | P00DR | 1111,1111 |
| P1DR | FE29H | P17DR | P16DR | P15DR | P14DR | P13DR | P12DR | P11DR | P10DR | 1111,1111 |
| P2DR | FE2AH | P27DR | P26DR | P25DR | P24DR | P23DR | P22DR | P21DR | P20DR | 1111,1111 |
| P3DR | FE2BH | P37DR | P36DR | P35DR | P34DR | P33DR | P32DR | P31DR | P30DR | 1111,1111 |
| P4DR | FE2CH | P47DR | P46DR | P45DR | P44DR | P43DR | P42DR | P41DR | P40DR | 1111,1111 |
| P5DR | FE2DH | -     | -     | P55DR | P54DR | P53DR | P52DR | P51DR | P50DR | xx11,1111 |
| P6DR | FE2EH | P67DR | P66DR | P65DR | P64DR | P63DR | P62DR | P61DR | P60DR | 1111,1111 |
| P7DR | FE2FH | P77DR | P76DR | P75DR | P74DR | P73DR | P72DR | P71DR | P70DR | 1111,1111 |

控制端口的驱动能力

0: 增强驱动能力

1: 一般驱动能力

## 10.1.7 端口数字信号输入使能控制寄存器 (PxIE)

| 符号   | 地址    | B7    | B6    | B5    | B4    | B3    | B2    | B1    | B0    |
|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| P0IE | FE30H | P07IE | P06IE | P05IE | P04IE | P03IE | P02IE | P01IE | P00IE |
| P1IE | FE31H | P17IE | P16IE | P15IE | P14IE | P13IE | P12IE | P11IE | P10IE |
| P2IE | FE32H | P27IE | P26IE | P25IE | P24IE | P23IE | P22IE | P21IE | P20IE |
| P3IE | FE33H | P37IE | P36IE | P35IE | P34IE | P33IE | P32IE | P31IE | P30IE |
| P4IE | FE34H | P47IE | P46IE | P45IE | P44IE | P43IE | P42IE | P41IE | P40IE |
| P5IE | FE35H | -     | -     | P55IE | P54IE | P53IE | P52IE | P51IE | P50IE |
| P6IE | FE36H | P67IE | P66IE | P65IE | P64IE | P63IE | P62IE | P61IE | P60IE |
| P7IE | FE37H | P77IE | P76IE | P75IE | P74IE | P73IE | P72IE | P71IE | P70IE |

数字信号输入使能控制

- 0: 禁止数字信号输入。若 I/O 被当作比较器输入口、ADC 输入口、触摸按键输入口**或者为外部晶振接入脚**等模拟口时, 进入时钟停振模式前, 必须设置为 0, 否则会有额外的耗电。
- 1: 使能数字信号输入。若 I/O 被当作数字口时, 必须设置为 1, 否 MCU 无法读取外部端口的电平。

**特别注意:** 对于具有 RTC 功能的 MCU, 当 RTC 的时钟源选择外部 32.768K 的晶振时, 需要将晶振接入脚 P1.6 和 P1.7 的数字通道关闭, 否则进入 STOP 模式后会有额外的漏电。(将寄存器 P1IE 的 bit6 和 bit7 都设置为 0 即可关闭 P1.6 和 P1.7 的数字通道)

STC MCU



## 10.2 配置 I/O 口

每个 I/O 的配置都需要使用两个寄存器进行设置。

以 P0 口为例，配置 P0 口需要使用 P0M0 和 P0M1 两个寄存器进行配置，如下图所示：

即 P0M0 的第 0 位和 P0M1 的第 0 位组合起来配置 P0.0 口的模式  
即 P0M0 的第 1 位和 P0M1 的第 1 位组合起来配置 P0.1 口的模式  
其他所有 I/O 的配置都与此类似。

PnM0 与 PnM1 的组合方式如下表所示

| PnM1 | PnM0 | I/O 口工作模式  |
|------|------|--|
| 0    | 0    | 准双向口（传统8051端口模式，弱上拉）<br>灌电流可达20mA，拉电流为270~150μA（存在制造误差）  |
| 0    | 1    | 推挽输出（强上拉输出，可达20mA，要加限流电阻）  |
| 1    | 0    | 高阻输入（电流既不能流入也不能流出）   |
| 1    | 1    | 开漏输出（Open-Drain），内部上拉电阻断开<br>开漏模式既可读外部状态也可对外输出（高电平或低电平）。如要正确读外部状态或需要对外输出高电平，需外加上拉电阻，否则读不到外部状态，也对外输不出高电平。 |

注：n = 0, 1, 2, 3, 4, 5, 6, 7

### 注意：

虽然每个 I/O 口在弱上拉（准双向口）/强推挽输出/开漏模式时都能承受 20mA 的灌电流（还是要加限流电阻，如 1K、560Ω、472Ω 等），在强推挽输出时能输出 20mA 的拉电流（也要加限流电阻），但整个芯片的工作电流推荐不要超过 70mA，即从 Vcc 流入的电流建议不要超过 70mA，从 Gnd 流出电流建议不要超过 70mA，整体流入/流出电流建议都不要超过 70mA。

## 10.3 I/O 的结构图

### 10.3.1 准双向口（弱上拉）

准双向口（弱上拉）输出类型可用作输出和输入功能而不需重新配置端口输出状态。这是因为当端口输出为 1 时驱动能力很弱，允许外部装置将其拉低。当引脚输出为低时，它的驱动能力很强，可吸收相当大的电流。准双向口有 3 个上拉晶体管适应不同的需要。

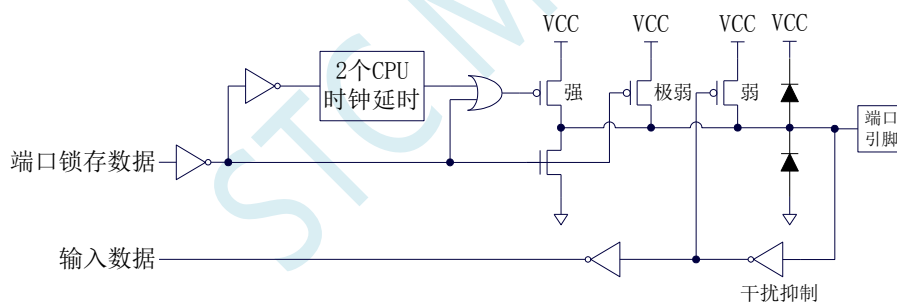
在 3 个上拉晶体管中，有 1 个上拉晶体管称为“弱上拉”，当端口寄存器为 1 且引脚本身为 1 时打开。此上拉提供基本驱动电流使准双向口输出为 1。如果一个引脚输出为 1 而由外部装置下拉到低时，弱上拉关闭而“极弱上拉”维持开状态，为了把这个引脚强拉为低，外部装置必须有足够的灌电流能力使引脚上的电压降到门槛电压以下。对于 5V 单片机，“弱上拉”晶体管的电流约 250uA；对于 3.3V 单片机，“弱上拉”晶体管的电流约 150uA。

第 2 个上拉晶体管，称为“极弱上拉”，当端口锁存为 1 时打开。当引脚悬空时，这个极弱的上拉源产生很弱的上拉电流将引脚上拉为高电平。对于 5V 单片机，“极弱上拉”晶体管的电流约 18uA；对于 3.3V 单片机，“极弱上拉”晶体管的电流约 5uA。

第 3 个上拉晶体管称为“强上拉”。当端口锁存器由 0 到 1 跳变时，这个上拉用来加快准双向口由逻辑 0 到逻辑 1 转换。当发生这种情况时，强上拉打开约 2 个时钟以使引脚能够迅速地上拉到高电平。

准双向口（弱上拉）带有一个施密特触发输入以及一个干扰抑制电路。准双向口（弱上拉）读外部状态前,要先锁存为 ‘1’,才可读到外部正确的状态。

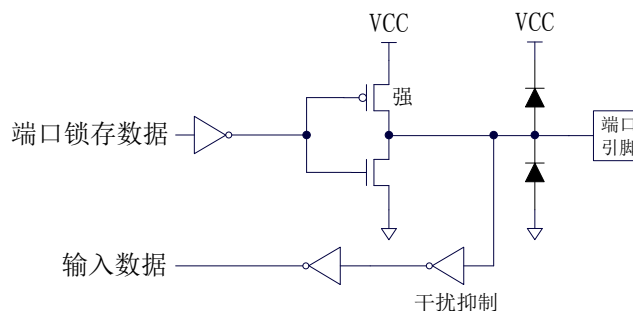
准双向口（弱上拉）输出如下图所示：



### 10.3.2 推挽输出

强推挽输出配置的下拉结构与开漏输出以及准双向口的下拉结构相同，但当锁存器为 1 时提供持续的强上拉。推挽模式一般用于需要更大驱动电流的情况。

强推挽引脚配置如下图所示：

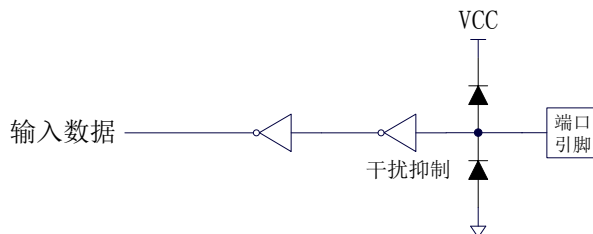


### 10.3.3 高阻输入

电流既不能流入也不能流出

输入口带有一个施密特触发输入以及一个干扰抑制电路

高阻输入引脚配置如下图所示:



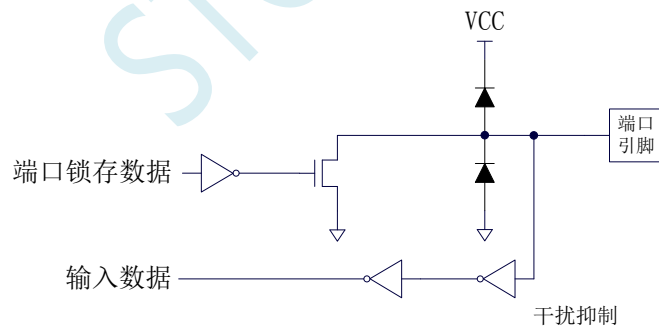
### 10.3.4 开漏输出

开漏模式既可读外部状态也可对外输出（高电平或低电平）。如要正确读外部状态或需要对外输出高电平，需外加上拉电阻。

当端口锁存器为 0 时，开漏输出关闭所有上拉晶体管。当作为一个逻辑输出高电平时，这种配置方式必须有外部上拉，一般通过电阻外接到 Vcc。如果外部有上拉电阻，开漏的 I/O 口还可读外部状态，即此时被配置为开漏模式的 I/O 口还可作为输入 I/O 口。这种方式的下拉与准双向口相同。

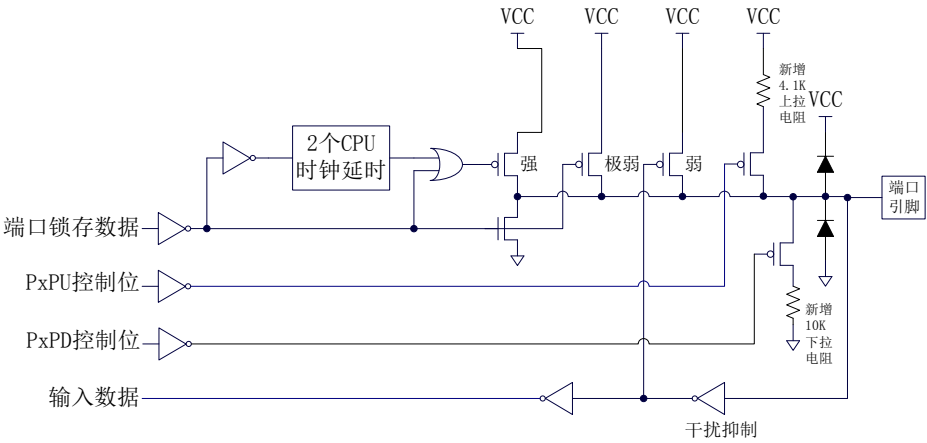
开漏端口带有一个施密特触发输入以及一个干扰抑制电路。

输出端口配置如下图所示:



### 10.3.5 新增 4.1K 上拉电阻

STC8 系列所有的 I/O 口内部均可使能一个大约 4.1K 的上拉电阻（由于制造误差，上拉电阻的范围可能为 3K~5K）



端口上拉电阻控制寄存器

| 符号   | 地址    | B7    | B6    | B5    | B4    | B3    | B2    | B1    | B0    |
|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| P0PU | FE10H | P07PU | P06PU | P05PU | P04PU | P03PU | P02PU | P01PU | P00PU |
| P1PU | FE11H | P17PU | P16PU | P15PU | P14PU | P13PU | P12PU | P11PU | P10PU |
| P2PU | FE12H | P27PU | P26PU | P25PU | P24PU | P23PU | P22PU | P21PU | P20PU |
| P3PU | FE13H | P37PU | P36PU | P35PU | P34PU | P33PU | P32PU | P31PU | P30PU |
| P4PU | FE14H | P47PU | P46PU | P45PU | P44PU | P43PU | P42PU | P41PU | P40PU |
| P5PU | FE15H | -     | -     | -     | P54PU | P53PU | P52PU | P51PU | P50PU |
| P6PU | FE16H | P67PU | P66PU | P65PU | P64PU | P63PU | P62PU | P61PU | P60PU |
| P7PU | FE17H | P77PU | P76PU | P75PU | P74PU | P73PU | P72PU | P71PU | P70PU |

端口内部4.1K上拉电阻控制位（注：P3.0和P3.1口上的上拉电阻可能会略小一些）

- 0：禁止端口内部的 4.1K 上拉电阻
- 1：使能端口内部的 4.1K 上拉电阻

### 10.3.6 如何设置 I/O 口对外输出速度

当用户需要 I/O 口对外输出较快的频率时，可通过加大 I/O 口驱动电流以及增加 I/O 口电平转换速度以达到提高 I/O 口对外输出速度

设置 PxSR 寄存器，可用于控制 I/O 口电平转换速度，设置为 0 时相应的 I/O 口为快速翻转，设置为 1 时为慢速翻转。

设置 PxDR 寄存器，可用于控制 I/O 口驱动电流大小，设置为 1 时 I/O 输出为一般驱动电流，设置为 0 时为强驱动电流

### 10.3.7 如何设置 I/O 口电流驱动能力

若需要改变 I/O 口的电流驱动能力，可通过设置 PxDR 寄存器来实现

设置 PxDR 寄存器，可用于控制 I/O 口驱动电流大小，设置为 1 时 I/O 输出为一般驱动电流，设置为 0 时为强驱动电流

### 10.3.8 如何降低 I/O 口对外辐射

由于设置 PxSR 寄存器，可用于控制 I/O 口电平转换速度，设置 PxDR 寄存器，可用于控制 I/O 口驱动电流大小

当需要降低 I/O 口对外的辐射时，需要将 PxSR 寄存器设置为 1 以降低 I/O 口电平转换速度，同时需要将 PxDR 寄存器设为 1 以降低 I/O 驱动电流，最终达到降低 I/O 口对外辐射

STC MCU

# 10.4 范例程序

## 10.4.1 端口模式设置

### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void main()
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;                  //设置 P0.0~P0.7 为双向口模式
    P0M1 = 0x00;
    P1M0 = 0xff;                  //设置 P1.0~P1.7 为推挽输出模式
    P1M1 = 0x00;
    P2M0 = 0x00;                  //设置 P2.0~P2.7 为高阻输入模式
    P2M1 = 0xff;
    P3M0 = 0xff;                  //设置 P3.0~P3.7 为开漏模式
    P3M1 = 0xff;

    while (1);
}
```

### 汇编代码

```
;测试工作频率为 11.0592MHz

P_SW2      DATA      0BAH

P0M0        DATA      094H
P0M1        DATA      093H
P1M0        DATA      092H
P1M1        DATA      091H
P2M0        DATA      096H
P2M1        DATA      095H
P3M0        DATA      0B2H
P3M1        DATA      0B1H
P4M0        DATA      0B4H
P4M1        DATA      0B3H
P5M0        DATA      0CAH
P5M1        DATA      0C9H
P6M0        DATA      0CCH
P6M1        DATA      0CBH
P7M0        DATA      0E2H
P7M1        DATA      0E1H

            ORG          0000H
            LJMP         MAIN

            ORG          0100H
MAIN:
            MOV          SP, #5FH
```

```

    ORL        P_SW2,#80H                ;使能访问 XFR

    MOV        P0M0,#00H                ;设置 P0.0~P0.7 为双向口模式
    MOV        P0M1,#00H
    MOV        P1M0,#0FFH                ;设置 P1.0~P1.7 为推挽输出模式
    MOV        P1M1,#00H
    MOV        P2M0,#00H                ;设置 P2.0~P2.7 为高阻输入模式
    MOV        P2M1,#0FFH
    MOV        P3M0,#0FFH                ;设置 P3.0~P3.7 为开漏模式
    MOV        P3M1,#0FFH

    JMP        $

END

```

## 10.4.2 双向口读写操作

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```
void main()
```

```

{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P0M0 = 0x00;                //设置 P0.0~P0.7 为双向口模式
    P0M1 = 0x00;

    P00 = 1;                    //P0.0 口输出高电平
    _nop_();                    //
    _nop_();                    //
    P00 = 0;                    //P0.0 口输出低电平
    _nop_();                    //
    _nop_();                    //

    P00 = 1;                    //读取端口前先使能内部弱上拉电阻
    _nop_();                    //等待两个时钟
    _nop_();                    //
    CY = P00;                  //读取端口状态
}

```

```
    while (1);  
}
```

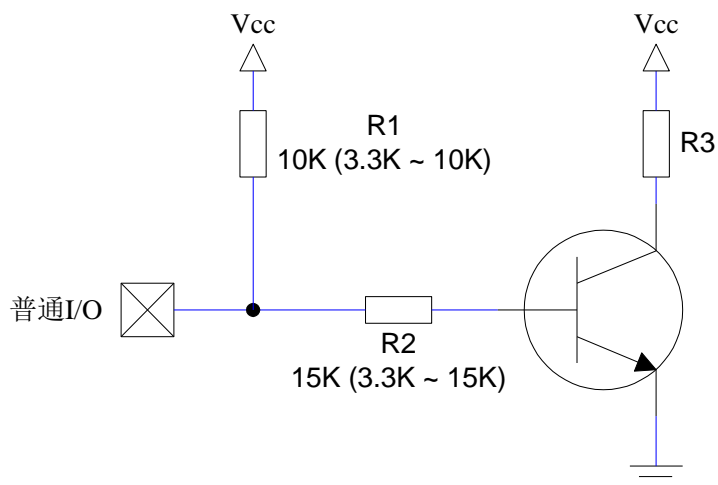
汇编代码

;测试工作频率为 11.0592MHz

```
P0M0    DATA    094H  
P0M1    DATA    093H  
P1M1    DATA    091H  
P1M0    DATA    092H  
P0M1    DATA    093H  
P0M0    DATA    094H  
P2M1    DATA    095H  
P2M0    DATA    096H  
P3M1    DATA    0B1H  
P3M0    DATA    0B2H  
P4M1    DATA    0B3H  
P4M0    DATA    0B4H  
P5M1    DATA    0C9H  
P5M0    DATA    0CAH  
  
        ORG      0000H  
        LJMP     MAIN  
  
MAIN:   ORG      0100H  
  
        MOV      SP, #5FH  
        ORL      P_SW2, #80H    ;使能访问 XFR  
  
        MOV      P0M0, #00H  
        MOV      P0M1, #00H  
        MOV      P1M0, #00H  
        MOV      P1M1, #00H  
        MOV      P2M0, #00H  
        MOV      P2M1, #00H  
        MOV      P3M0, #00H  
        MOV      P3M1, #00H  
        MOV      P4M0, #00H  
        MOV      P4M1, #00H  
        MOV      P5M0, #00H  
        MOV      P5M1, #00H  
  
        MOV      P0M0, #00H    ;设置 P0.0~P0.7 为双向口模式  
        MOV      P0M1, #00H  
  
        SETB     P0.0          ;P0.0 口输出高电平  
        CLR      P0.0          ;P0.0 口输出低电平  
  
        SETB     P0.0          ;读取端口前先使能内部弱上拉电阻  
        NOP  
        NOP                ;等待两个时钟  
        MOV      C, P0.0       ;读取端口状态  
  
        JMP      $  
  
        END
```

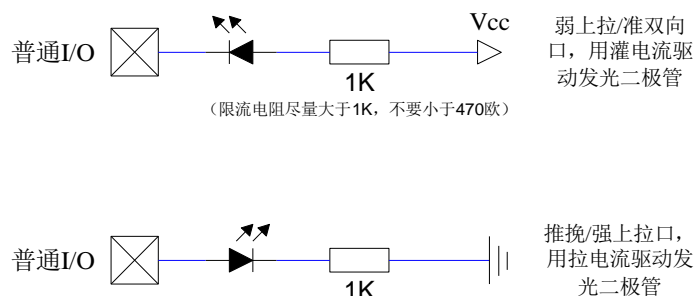


## 10.5 一种典型三极管控制电路



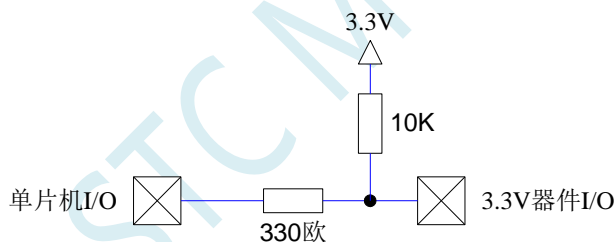
如果使用弱上拉控制, 建议加上拉电阻  $R1$  (3.3K~10K), 如果不加上拉电阻  $R1$  (3.3K~10K), 建议  $R2$  的值在 15K 以上, 或用强推挽输出。

## 10.6 典型发光二极管控制电路

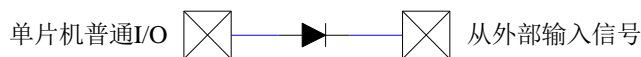


## 10.7 混合电压供电系统 3V/5V 器件 I/O 口互连

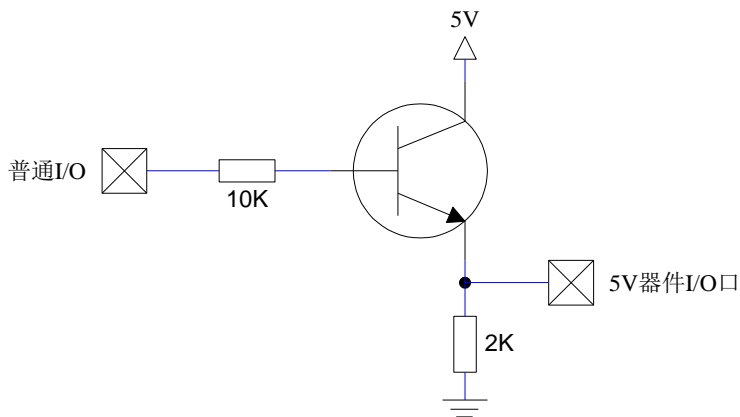
STC 系列宽电压单片机工作在 5V 时, 如需要直接连接 3.3V 器件时, 为防止 3.3V 器件承受不了 5V, 可将相应的单片机 I/O 口先串一个 330Ω 的限流电阻到 3.3V 器件 I/O 口, 程序初始化时将单片机的 I/O 口设置成开漏配置, 断开内部上拉电阻, 相应的 3.3V 器件 I/O 口外部加 10K 上拉电阻到 3.3V 器件的 Vcc, 这样高电平是 3.3V, 低电平是 0V, 输入输出一切正常。



STC 宽电压单片机工作在 3V 时, 如需要直接连接 5V 器件时, 如果相应的 I/O 口是输入, 可在该 I/O 口上串接一个隔离二极管, 隔离高压部分。外部信号电压高于单片机工作电压时截止, I/O 口因内部上拉到高电平, 所以读 I/O 口状态是高电平; 外部信号电压为低时导通, I/O 口被钳位在 0.7V, 小于 0.8V 时单片机读 I/O 口状态是低电平。



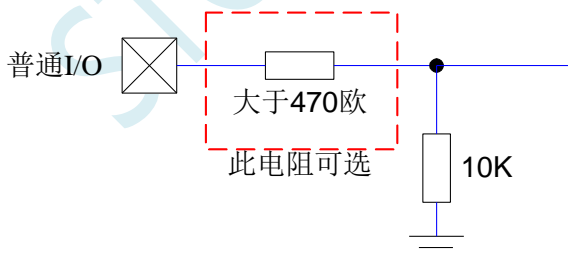
STC 宽电压单片机工作在 3V 时, 如需要直接连接 5V 器件时, 如果相应的 I/O 口是输出, 可用一个 NPN 三极管隔离, 电路如下:



## 10.8 如何让 I/O 口上电复位时为低电平

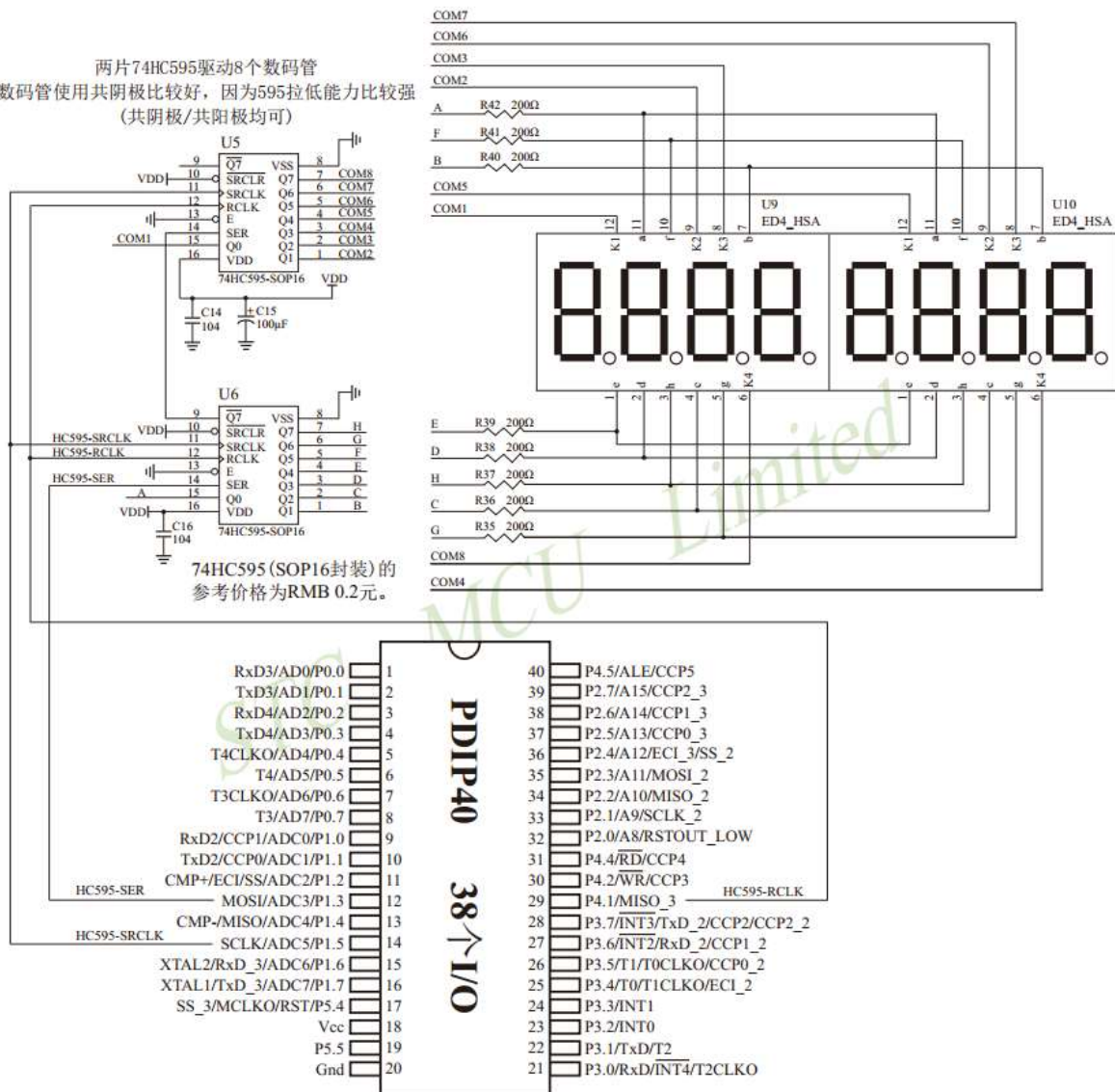
普通 8051 单片机上电复位时普通 I/O 口为弱上拉(准双向口)高电平输出,而很多实际应用要求上电时某些 I/O 口为低电平输出,否则所控制的系统(如马达)就会误动作,现 STC 的单片机由于既有弱上拉输出又有强推挽输出,就可以很轻松的解决此问题。

现可在 STC 的单片机 I/O 口上加一个下拉电阻(10K 左右),这样上电复位时,除了下载口 P3.0 和 P3.1 为弱上拉(准双向口)外,其他 I/O 口均为高阻输入模式,而外部有下拉电阻,所以该 I/O 口上电复位时外部为低电平。如果要将此 I/O 口驱动为高电平,可将此 I/O 口设置为强推挽输出,而强推挽输出时, I/O 口驱动电流可达 20mA,故肯定可以将该口驱动为高电平输出。

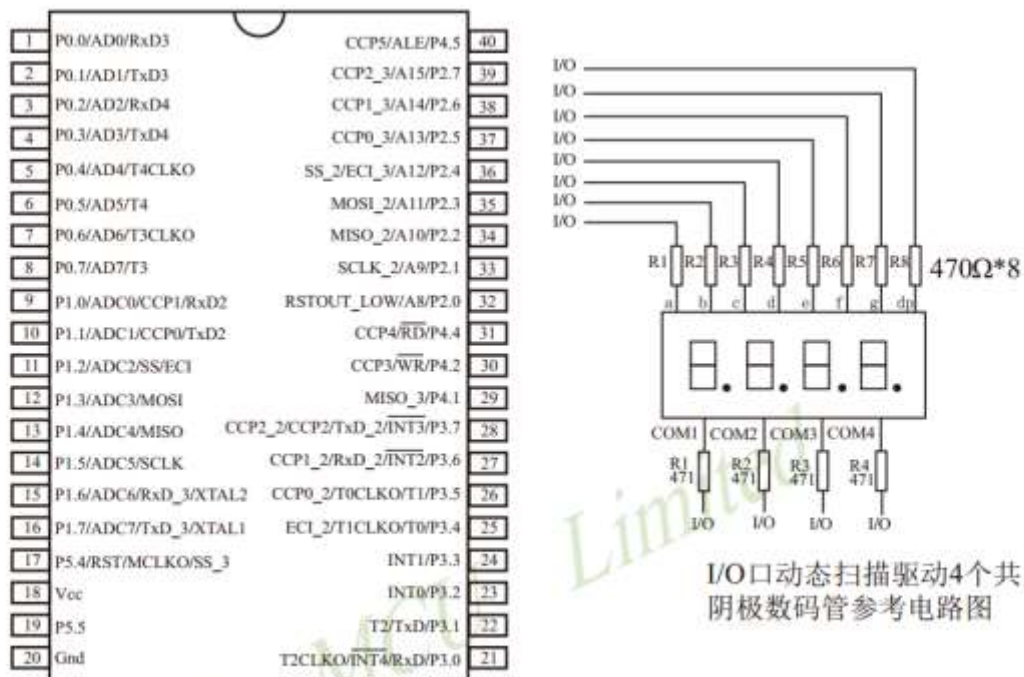


## 10.9 利用 74HC595 驱动 8 个数码管(串行扩展,3 根线)的线路图

两片 74HC595 驱动 8 个数码管  
数码管使用共阴极比较好, 因为 595 拉低能力比较强  
(共阴极/共阳极均可)



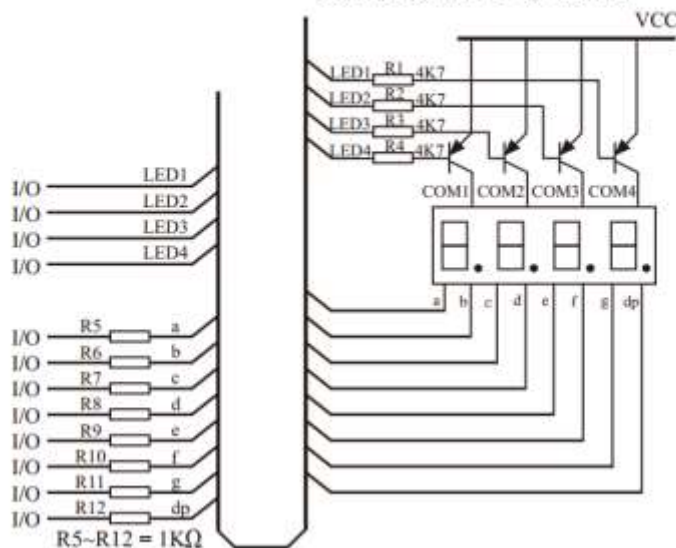
## 10.10 I/O 口直接驱动 LED 数码管应用线路图



I/O口动态扫描驱动4个共阴极数码管参考电路图

I/O 口动态扫描驱动数码时，可以一次点亮一个数码管中的8段，但为降低功耗，建议可以一次只点亮其中的4段或者2段

I/O 口动态扫描驱动4个共阳极数码管参考电路图



## 10.11 用 STC 系列 MCU 的 I/O 口直接驱动段码 LCD

当产品需要段码 LCD 显示时, 如果使用不带 LCD 驱动器的 MCU, 则需要外接 LCD 驱动 IC, 这会增加成本。事实上, 很多小项目, 比如大量的小家电, 需要显示的段码不多, 常见的是 4 个 8 带小数点或时钟的冒号 “:”, 这样如果使用 IO 口直接扫描显示, 则会降低成本, 工作更可靠。

但是, 本方案不合适驱动太多的段 ( 占用 IO 太多), 也不合适非常低功耗的场合 ( 驱动会有几百  $\mu\text{A}$  电流)。

段码 LCD 驱动简单原理: 如图 1 所示。

LCD 是一种特殊的液态晶体, 在电场的作用下晶体的排列方向会发生扭转, 因而改变其透光性, 从而可以看到显示内容。LCD 有一个扭转电压阈值, 当 LCD 两端电压高于此阈值时, 显示内容, 低于此阈值时, 不显示。通常 LCD 有 3 个参数: 工作电压、DUTY (对应 COM 数) 和 BIAS (即偏压, 对应阈值), 比如 3.0V、1/4 DUTY、1/3 BIAS, 表示 LCD 显示电压为 3.0V, 4 个 COM, 阈值大约是 1.5V, 当加在某段 LCD 两端电压为 3.0V 时显示, 而加 1.0V 时不显示。但是 LCD 对于驱动电压的反应不是很敏感的, 比如加 2V 时, 可能会微弱显示, 这就是通常说的 “鬼影”。所以为了保证驱动显示时, 要大于阈值电压比较多, 而不显示时, 要用比阈值小比较多的电压。

注意: LCD 要用交流驱动, 其两端不能加直流电压, 否则时间稍长就会损坏, 所以为了保证加在 LCD 两端的驱动电压的平均电压为 0。LCD 使用时分割扫描法, 任何时候一个 COM 扫描有效, 另外的 COM 处于无效状态。

驱动 1/4Duty 1/2BIAS 3V 的方案电路见图 1, LCD 扫描原理见图 3, MCU 为 3.0V 或 3.3V 工作, 并且每个 COM 都串一个 20K 电阻接到一个电容 C1, RC 滤波后得到一个中点电压 1/2VDD。在轮到某个 COM 扫描时, 连接的 IO 设置成推挽输出, 其余 COM 设置成高阻, 如果与本 COM 连接的 SEG 不显示, 则 SEG 输出与 COM 同相, 如果显示, 则反相。扫描完后, 这个 COM 的 IO 就设置成高阻。每个 COM 通过 20K 电阻连接到电容 C1 上的 1/2VDD 电压, 而 SEG 根据是否显示输出高低电平, 这样加在 LCD 段上的电压, 显示时是 +VDD, 不显示时是 -1/2VDD, 保证了 LCD 两端平均直流电压为 0。

驱动 1/4Duty 1/3BIAS 3V 的方案电路见图 4, LCD 扫描原理见图 5, MCU 为 5V 工作, SEG 接的 IO 通过电阻分压输出 1.5V、3.5V, COM 接的 IO 通过电阻分压输出 0.5V、2.5V (高阻时)、4.5V, 分压电阻公共点接到一个电容 C1, RC 滤波后得到一个中点电压 1/2VDD。在轮到某个 COM 扫描时, 设置成推挽输出, 如果与本 COM 连接的 SEG 不显示, 则 SEG 输出与 COM 同相, 如果显示, 则反相。扫描完后, 这个 COM 的 IO 就设置成高阻, 这样这个 COM 就通过 47K 电阻连接到 2.5V 电压, 而 SEG 根据是否显示输出高低电平, 这样加在 LCD 上的电压, 显示时是 +3.0V, 不显示时是 -1.0V, 完全满足 LCD 的扫描要求。

当需要睡眠省电时, 把所有 COM 和 SEG 驱动 IO 全部输出低电平, LCD 驱动部分不会增加额外电流。

图 1: 驱动 1/4Duty 1/2BIAS 3V LCD 的电路

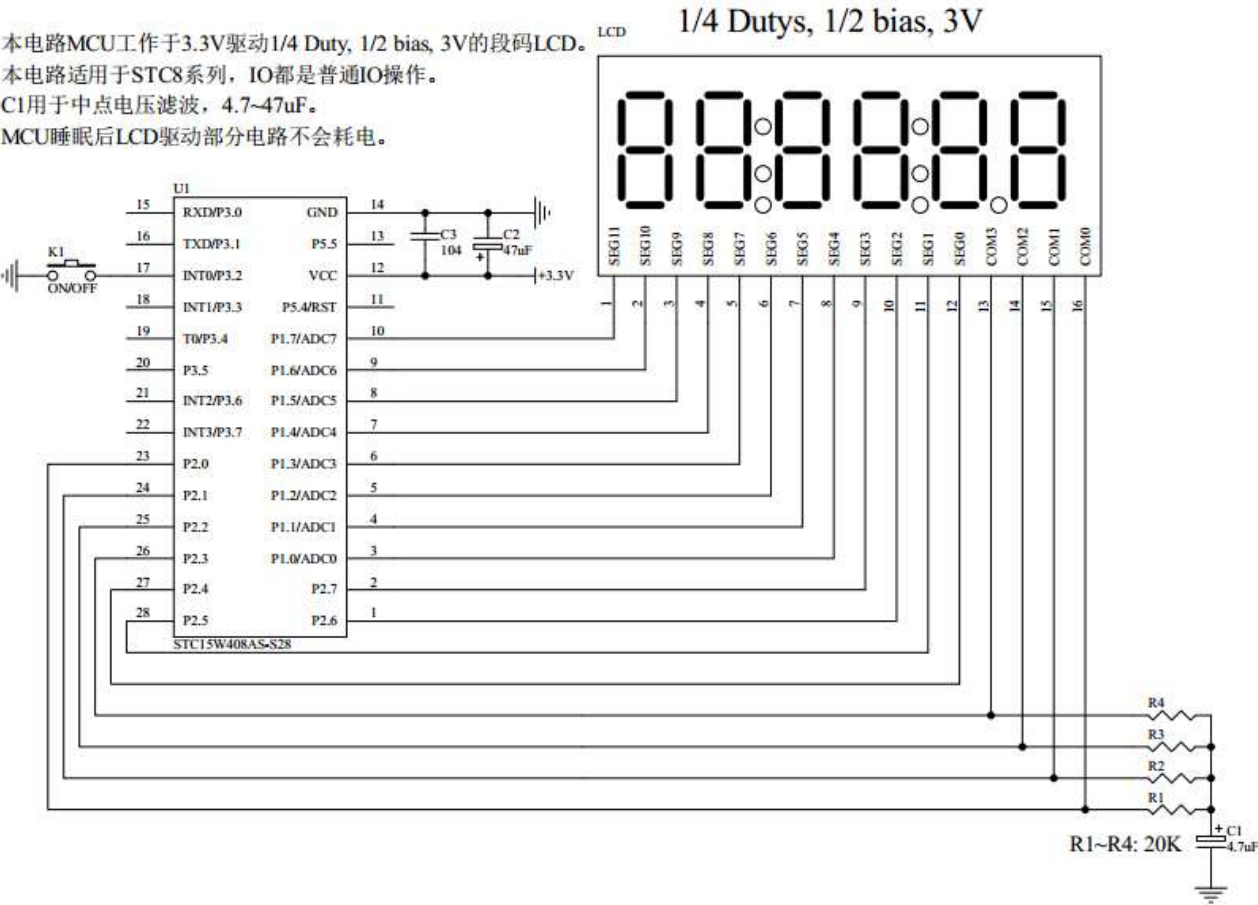


图 2: 段码名称图

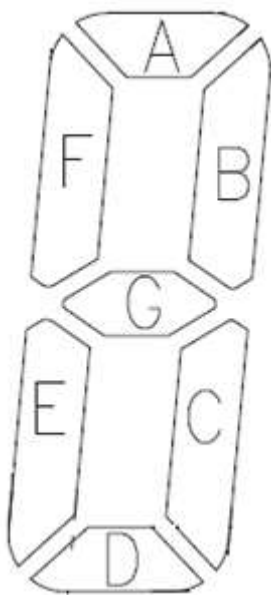




图 3: 1/4Duty 1/2BIAS 扫描原理图

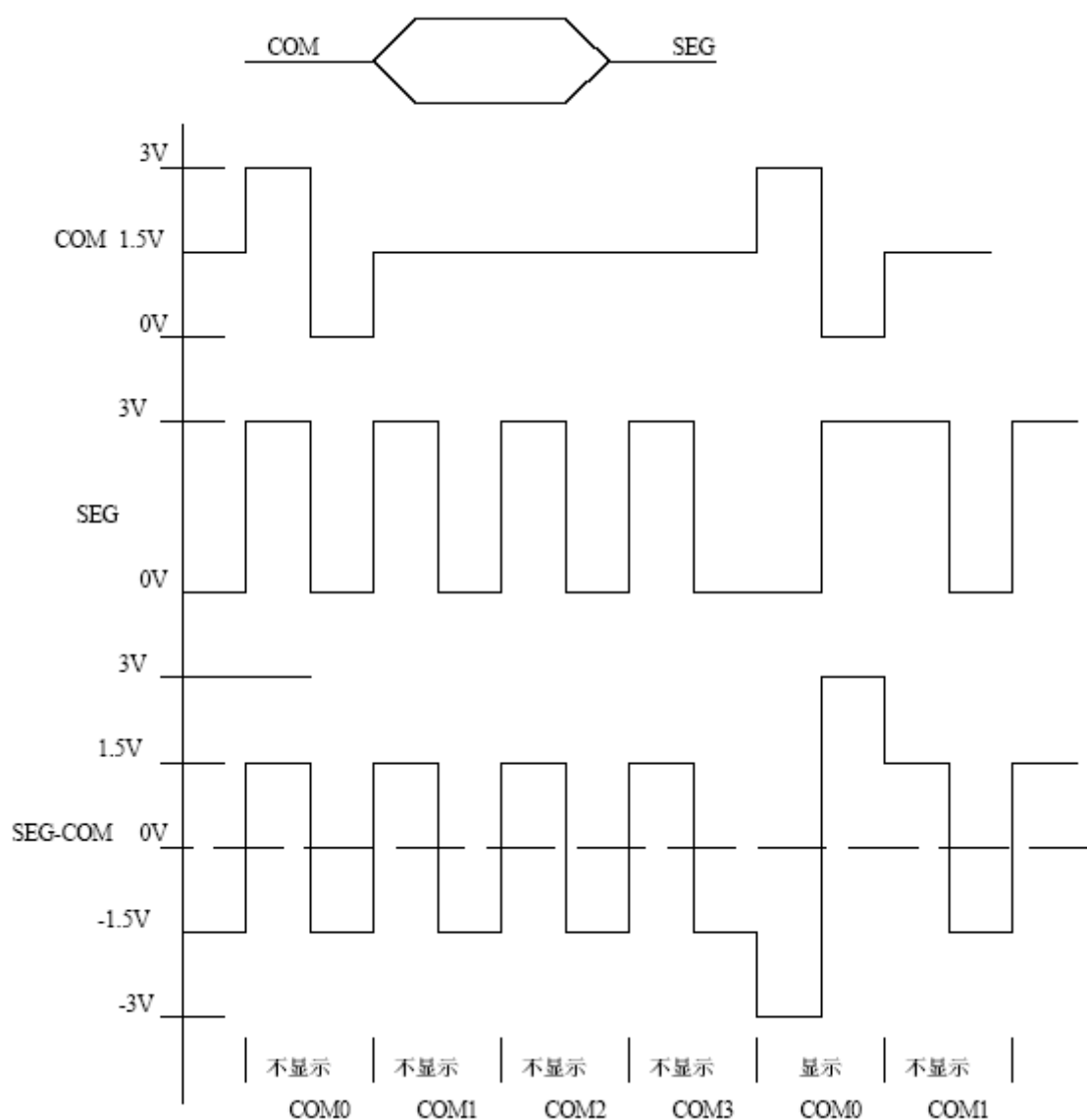




图 4: 驱动 1/4Duty 1/3BIAS 3V LCD 的电路

本电路MCU工作于5V驱动1/4 Duty, 1/3 bias, 3V的段码LCD。

本电路适用于STC8系列, IO都是普通IO操作。

C1用于中点电压滤波, 4.7~47uF。

MCU睡眠后LCD驱动部分电路不会耗电。

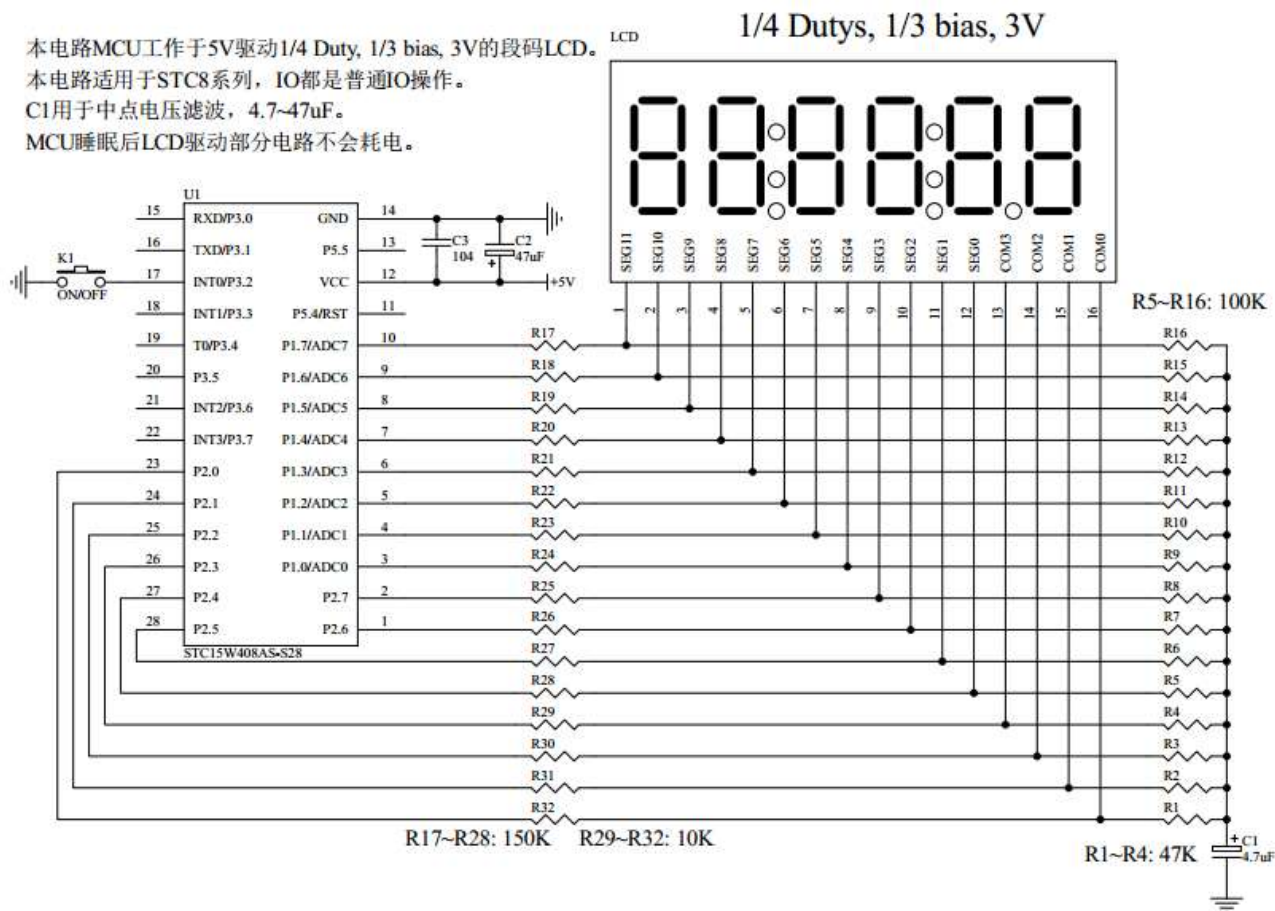
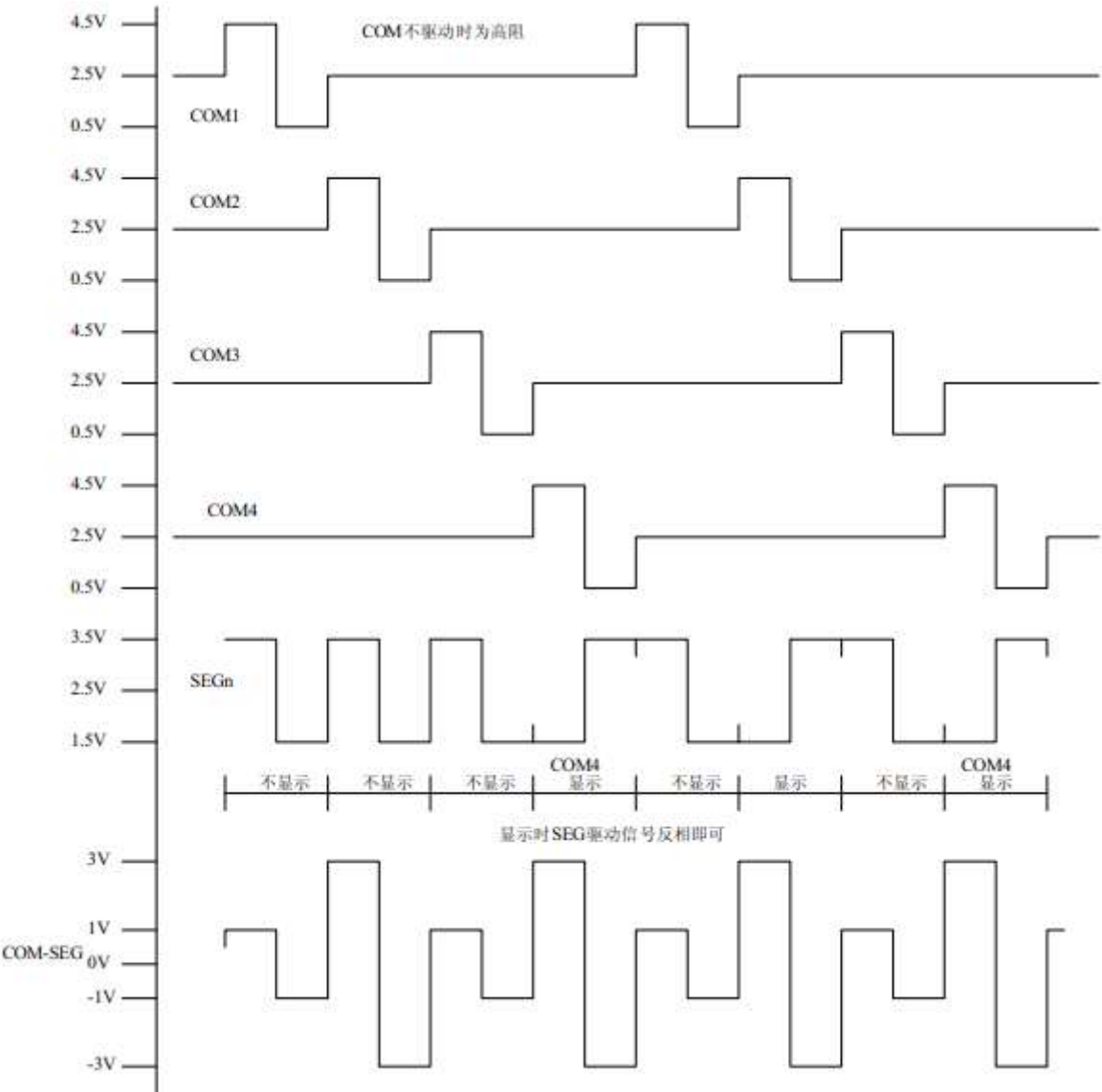


图 5: 1/4Duty 1/3BIAS 扫描原理图



为了方便，显示内容放在一个显存中，其中的各个位与 LCD 的段一一对应，见图 6。

图 6: LCD 真值表和显存影射表

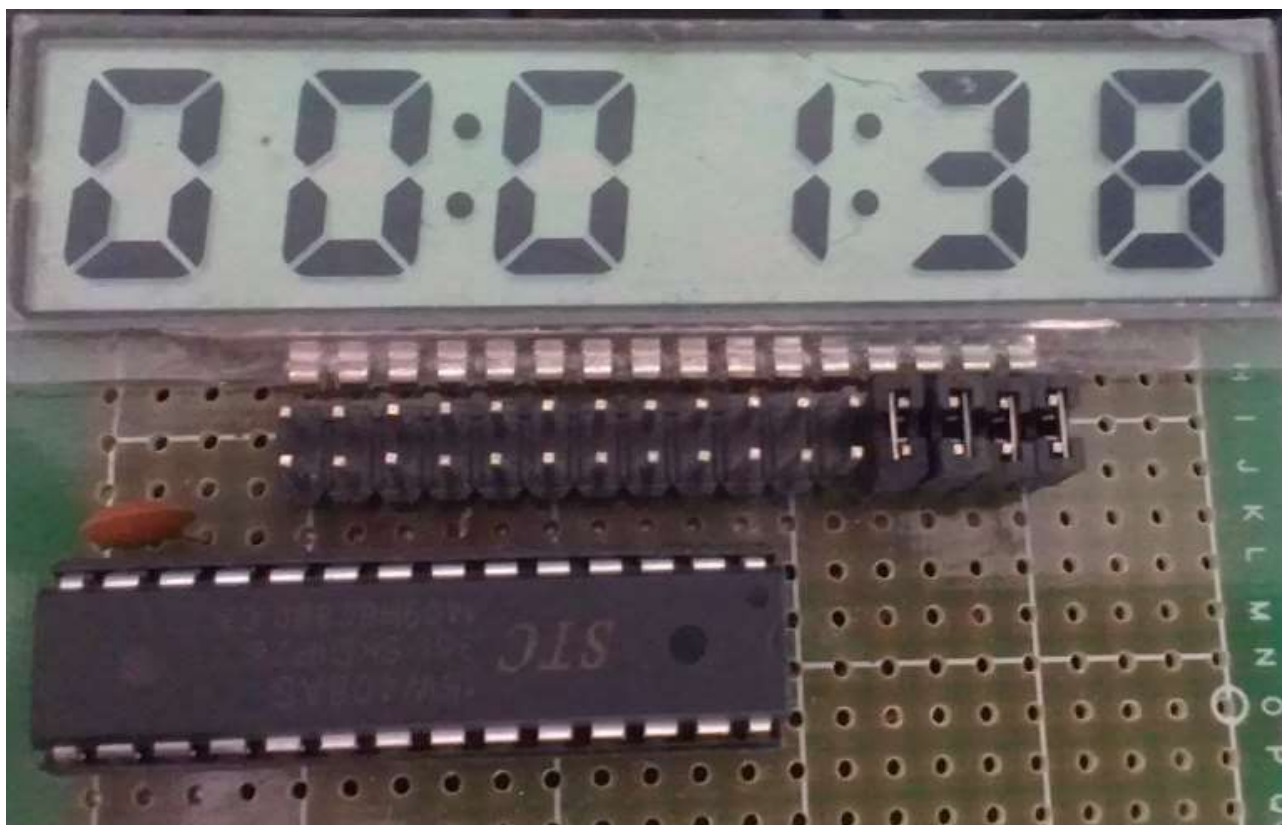
LCD真值表:

| MCU PIN      | P17   | P16   | P15  | P14  | P13  | P12  | P11  | P10  | P27  | P26  | P25  | P24  | P23  | P22  | P21  | P20  |
|--------------|-------|-------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| LCD PIN      | 1     | 2     | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11   | 12   | 13   | 14   | 15   | 16   |
| LCD PIN name | SEG11 | SEG10 | SEG9 | SEG8 | SEG7 | SEG6 | SEG5 | SEG4 | SEG3 | SEG2 | SEG1 | SEG0 | COM3 | COM2 | COM1 | COM0 |
| --           | 1D    | 2:    | 2D   | 2.   | 3D   | 4:   | 4D   | 4.   | 5D   | 5.   | 6D   | COM3 |      |      |      |      |
| 1E           | 1C    | 2E    | 2C   | 3E   | 3C   | 4E   | 4C   | 5E   | 5C   | 6E   | 6C   |      | COM2 |      |      |      |
| 1G           | 1B    | 2G    | 2B   | 3G   | 3B   | 4G   | 4B   | 5G   | 5B   | 6G   | 6B   |      |      | COM1 |      |      |
| 1F           | 1A    | 2F    | 2A   | 3F   | 3A   | 4F   | 4A   | 5F   | 5A   | 6F   | 6A   |      |      |      |      | COM0 |

显存影射表: |

|          |    |    |    |    |    |    |    |    |
|----------|----|----|----|----|----|----|----|----|
|          | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| buff[0]: | -- | 1D | 2: | 2D | 2. | 3D | 4: | 4D |
| buff[1]: | 1E | 1C | 2E | 2C | 3E | 3C | 4E | 4C |
| buff[2]: | 1G | 1B | 2G | 2B | 3G | 3B | 4G | 4B |
| buff[3]: | 1F | 1A | 2F | 2A | 3F | 3A | 4F | 4A |
| buff[4]: | 4. | 5D | 5. | 6D | -- | -- | -- | -- |
| buff[5]: | 5E | 5C | 6E | 6C | -- | -- | -- | -- |
| buff[6]: | 5G | 5B | 6G | 6B | -- | -- | -- | -- |
| buff[7]: | 5F | 5A | 6F | 6A | -- | -- | -- | -- |

图 7: 驱动效果照片



本 LCD 扫描程序仅需要两个函数:

1、LCD 段码扫描函数 `void LCD_scan(void)`

程序隔一定的时间调用这个函数, 就会将 LCD 显示缓冲的内容显示到 LCD 上, 全部扫描一次需要 8 个调用周期, 调用间隔一般是 1~2ms, 假如使用 1ms, 则扫描周期就是 8ms, 刷新率就是 125HZ。

2、LCD 段码显示缓冲装载函数 `void LCD_load(u8 n,u8 dat)`

本函数用来将显示的数字或字符放在 LCD 显示缓冲中, 比如 `LCD_load(1,6)`, 就是要在第一个数字位置显示数字 6, 支持显示 0~9, A~F, 其它字符用户可以自己添加。

另外, 用宏来显示、熄灭或闪烁冒号或小数点。

## C 语言代码

```

/***** 功能说明 *****/
用 STC15 系列测试 I/O 直接驱动段码 LCD(6 个 8 字 LCD, 1/4 Dutys, 1/3 bias)。
上电后显示一个时间(时分秒)。
P3.2 对地接一个开关, 用来进入睡眠或唤醒。
*****/

```

```

#include "stc8h.h"
#include "intrins.h"

```

```

typedef unsigned char    u8;
typedef unsigned int      u16;
typedef unsigned long     u32;

```

```

/***** 本地常量声明 *****/

```

```

#define MAIN_Fosc          11059200L           //定义主时钟

#define DIS_BLACK          0x10
#define DIS_               0x11
#define DIS_A              0x0A
#define DIS_B              0x0B
#define DIS_C              0x0C
#define DIS_D              0x0D
#define DIS_E              0x0E
#define DIS_F              0x0F

#define LCD_SET_DP2        LCD_buff[0] |=  0x08
#define LCD_CLR_DP2        LCD_buff[0] &= ~0x08
#define LCD_FLASH_DP2      LCD_buff[0] ^=  0x08

#define LCD_SET_DP4        LCD_buff[4] |=  0x80
#define LCD_CLR_DP4        LCD_buff[4] &= ~0x80
#define LCD_FLASH_DP4      LCD_buff[4] ^=  0x80

#define LCD_SET_2M         LCD_buff[0] |=  0x20
#define LCD_CLR_2M         LCD_buff[0] &= ~0x20
#define LCD_FLASH_2M       LCD_buff[0] ^=  0x20

#define LCD_SET_4M         LCD_buff[0] |=  0x02
#define LCD_CLR_4M         LCD_buff[0] &= ~0x02
#define LCD_FLASH_4M       LCD_buff[0] ^=  0x02

#define LCD_SET_DP5        LCD_buff[4] |=  0x20
#define LCD_CLR_DP5        LCD_buff[4] &= ~0x20
#define LCD_FLASH_DP5      LCD_buff[4] ^=  0x20

#define P1n_standard(bitn) P1M1 &= ~(bitn), P1M0 &= ~(bitn)
#define P1n_push_pull(bitn) P1M1 &= ~(bitn), P1M0 |= (bitn)
#define P1n_pure_input(bitn) P1M1 |= (bitn), P1M0 &= ~(bitn)
#define P1n_open_drain(bitn) P1M1 |= (bitn), P1M0 |= (bitn)

#define P2n_standard(bitn) P2M1 &= ~(bitn), P2M0 &= ~(bitn)
#define P2n_push_pull(bitn) P2M1 &= ~(bitn), P2M0 |= (bitn)
#define P2n_pure_input(bitn) P2M1 |= (bitn), P2M0 &= ~(bitn)
#define P2n_open_drain(bitn) P2M1 |= (bitn), P2M0 |= (bitn)

/***** 本地变量声明 *****/
u8 cnt_500ms;
u8 second,minute,hour;
bit B_Second;
bit B_2ms;
u8 LCD_buff[8];
u8 scan_index;

/***** 本地函数声明 *****/
void LCD_load(u8 n,u8 dat);
void LCD_scan(void);
void LoadRTC(void);
void delay_ms(u8 ms);

/***** 主函数 *****/
void main(void)
{
    u8 i;

```

```

AUXR = 0x80;
TMOD = 0x00;
TL0 = (65536 - (MAIN_Fosc / 500));
TH0 = (65536 - (MAIN_Fosc / 500)) >> 8;
TR0 = 1;
ET0 = 1;
EA = 1;

//初始化LCD 显存
for(i=0; i<8; i++) LCD_buff[i] = 0;
P2n_push_pull(0xf0);
P1n_push_pull(0xff);

//segment 设置为推挽输出

LCD_SET_2M;
LCD_SET_4M;
LoadRTC();

//显示时分间隔:
//显示分秒间隔:
//显示时间

while (1)
{
    PCON /= 0x01;
    _nop_();
    _nop_();
    _nop_();

    //进入空闲模式, 由Timer0 2ms 唤醒退出

    if(B_2ms)
    {
        B_2ms = 0;

        //2ms 节拍到

        if(++cnt_500ms >= 250)
        {
            cnt_500ms = 0;
            // LCD_FLASH_2M;
            // LCD_FLASH_4M;

            //闪烁时分间隔:
            //闪烁分秒间隔:

            B_Second = ~B_Second;
            if(B_Second)
            {
                if(++second >= 60)
                {
                    //1 分钟到

                    second = 0;
                    if(++minute >= 60)
                    {
                        //1 小时到

                        minute = 0;
                        if(++hour >= 24) hour = 0; //24 小时到

                    }
                }
                LoadRTC();
                //显示时间
            }
        }
    }

    if(!P32)
    {
        //键按下, 准备睡眠

        LCD_CLR_2M;
        LCD_CLR_4M;
        LCD_load(1,DIS_BLACK);
        LCD_load(2,DIS_BLACK);
        LCD_load(3,0);
        LCD_load(4,0x0F);
        //显示时分间隔:
        //显示分秒间隔:
    }
}

```

```

        LCD_load(5,0x0F);
        LCD_load(6,DIS_BLACK);

        while(!P32) delay_ms(10);           //等待释放按键
        delay_ms(50);
        while(!P32) delay_ms(10);           //再次等待释放按键

        TR0 = 0;                             //关闭定时器
        IE0 = 0;                             //外中断0 标志位
        EX0 = 1;                             //INT0 Enable
        IT0 = 1;                             //INT0 下降沿中断

        P1n_push_pull(0xff);                 //com 和seg 全部输出0
        P2n_push_pull(0xff);
        P1 = 0;
        P2 = 0;

        PCON /= 0x02;                         //Sleep
        _nop_();
        _nop_();
        _nop_();

        LCD_SET_2M;                           //显示时分间隔:
        LCD_SET_4M;                           //显示分秒间隔:
        LoadRTC();                           //显示时间
        TR0 = 1;                             //打开定时器
        while(!P32) delay_ms(10);             //等待释放按键
        delay_ms(50);
        while(!P32) delay_ms(10);             //再次等待释放按键
    }
}

}

}

}

/***** 延时函数 *****/
void delay_ms(u8 ms)
{
    unsigned int i;
    do{
        i = MAIN_Fosc / 13000;
        while(--i);           //14T per loop
    }while(--ms);
}

/***** Timer0 中断函数 *****/
void timer0_int (void) interrupt 1
{
    LCD_scan();
    B_2ms = 1;
}

/***** INT0 中断函数 *****/
void INT0_int (void) interrupt 0
{
    EX0 = 0;
    IE0 = 0;
}

/***** LCD 段码扫描函数 *****/

```

```

void LCD_scan(void)                                     //5us @22.1184MHZ
{
    u8 code T_COM[4]={0x08,0x04,0x02,0x01};
    u8 j;

    j = scan_index >> 1;
    P2n_pure_input(0x0f);                               //全部COM 输出高阻, COM 为中点电压
    if(scan_index & 1)                                   //反相扫描
    {
        P1 = ~LCD_buff[j];
        P2 = ~(LCD_buff[j/4] & 0xf0);
    }
    else                                                 //正相扫描
    {
        P1 = LCD_buff[j];
        P2 = LCD_buff[j/4] & 0xf0;
    }
    P2n_push_pull(T_COM[j]);                             //某个COM 设置为推挽输出
    if(++scan_index >= 8) scan_index = 0;
}

/***** 对第1~6 数字装载显示函数 *****/
void LCD_load(u8 n, u8 dat)                             //n 为第几个数字, dat 为要显示的数字
{
    u8 code t_display[]={                                //标准字库
        // 0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
        0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71,
        //black -
        0x00,0x40
    };
    u8 code T_LCD_mask[4] = {~0xc0,~0x30,~0x0c,~0x03};
    u8 code T_LCD_mask4[4] = {~0x40,~0x10,~0x04,~0x01};
    u8 i,k;
    u8 *p;

    if((n == 0) || (n > 6)) return;
    i = t_display[dat];

    if(n <= 4)                                           //1~4
    {
        n--;
        p = LCD_buff;
    }
    else
    {
        n = n - 5;
        p = &LCD_buff[4];
    }

    k = 0;
    if(i & 0x08) k /= 0x40;                               //D
    *p = (*p & T_LCD_mask4[n]) | (k>>2*n);
    p++;

    k = 0;
    if(i & 0x04) k /= 0x40;                               //C
    if(i & 0x10) k /= 0x80;                               //E
    *p = (*p & T_LCD_mask[n]) | (k>>2*n);
    p++;
}

```

```

    k = 0;
    if(i & 0x02) k /= 0x40;          //B
    if(i & 0x40) k /= 0x80;          //G
    *p = (*p & T_LCD_mask[n]) | (k>>2*n);
    p++;

    k = 0;
    if(i & 0x01) k /= 0x40;          //A
    if(i & 0x20) k /= 0x80;          //F
    *p = (*p & T_LCD_mask[n]) | (k>>2*n);
}

```

/\*\*\*\*\*\*显示时间\*\*\*\*\*\*/

void LoadRTC(void)

```

{
    LCD_load(1,hour/10);
    LCD_load(2,hour%10);
    LCD_load(3,minute/10);
    LCD_load(4,minute%10);
    LCD_load(5,second/10);
    LCD_load(6,second%10);
}

```

## 汇编代码

;用STC8 系列测试I/O 直接驱动段码LCD(6 个8 字LCD, 1/4 Dutys, 1/3 bias)。

;上电后显示一个时间(时分秒)。

\*\*\*\*\*

|          |      |      |
|----------|------|------|
| P0M1     | DATA | 0x93 |
| P0M0     | DATA | 0x94 |
| P1M1     | DATA | 0x91 |
| P1M0     | DATA | 0x92 |
| P2M1     | DATA | 0x95 |
| P2M0     | DATA | 0x96 |
| P3M1     | DATA | 0xB1 |
| P3M0     | DATA | 0xB2 |
| P4M1     | DATA | 0xB3 |
| P4M0     | DATA | 0xB4 |
| P5M1     | DATA | 0xC9 |
| P5M0     | DATA | 0xC  |
| P6M1     | DATA | 0xCB |
| P6M0     | DATA | 0xCC |
| P7M1     | DATA | 0xE1 |
| P7M0     | DATA | 0xE2 |
| AUXR     | DATA | 0x8E |
| INT_CLKO | DATA | 0x8F |
| IE2      | DATA | 0xAF |
| P4       | DATA | 0xC0 |
| T2H      | DATA | 0xD6 |
| T2L      | DATA | 0xD7 |
| P_SW2    | DATA | 0BAH |

\*\*\*\*\*

|           |     |      |
|-----------|-----|------|
| DIS_BLACK | EQU | 010H |
| DIS_      | EQU | 011H |



```

DIS_A      EQU      00AH
DIS_B      EQU      00BH
DIS_C      EQU      00CH
DIS_D      EQU      00DH
DIS_E      EQU      00EH
DIS_F      EQU      00FH

```

```

B_2ms      BIT      20H.0      ;2ms 信号
B_Second   BIT      20H.1      ;秒信号
cnt_500ms  DATA    30H
second     DATA    31H
minute     DATA    32H
hour       DATA    33H
scan_index DATA    34H

```

```

LCD_buff   DATA    40H      ;40H~47H

```

```

;*****

```

```

ORG      0000H
LJMP     F_Main

```

```

ORG      000BH
LJMP     F_Timer0_Interrupt

```

```

;*****

```

```

F_Main:   ORG      0100H

```

```

MOV      P_SW2,#80H

```

```

CLR      A
MOV      P3M1,A      ;设置为准双向口
MOV      P3M0,A
MOV      P5M1,A      ;设置为准双向口
MOV      P5M0,A

```

```

MOV      P1M1,#0      ;segment 设置为推挽输出

```

```

MOV      P1M0,#0ffh

```

```

ANL      P2M1,#NOT 0f0h      ;segment 设置为推挽输出

```

```

ORL      P2M0,#0f0h
ORL      P2M1,#00fH      ;全部 COM 输出高阻, COM 为中点电压

```

```

ANL      P2M0,#0f0H

```

```

MOV      SP,#0D0H

```

```

MOV      PSW,#0

```

```

USING    0      ;选择第0 组 R0~R7

```

```

;*****

```

```

MOV      R2,#8

```

```

MOV      R0,#LCD_buff

```

```

L_ClearLcdRam:

```

```

MOV      @R0,#0

```

```

INC      R0

```

```

DJNZ     R2,L_ClearLcdRam

```

```

LCALL    F_Timer0_init

```

```

SETB     EA

```

```

; ORL      LCD_buff,#020H      ;显示时分间隔:

```

```

; ORL      LCD_buff,#002H      ;显示分秒间隔:

```

```

MOV    hour, #12
MOV    minute, #00
MOV    second, #00
LCALL  F_LoadRTC           ;显示时间

```

\*\*\*\*\*

#### L\_Main\_Loop:

```

JNB    B_2ms, L_Main_Loop    ;2ms 节拍到
CLR    B_2ms

INC    cnt_500ms
MOV    A, cnt_500ms
CJNE   A, #250, L_Main_Loop   ;500ms 到
MOV    cnt_500ms, #0;

XRL    LCD_buff, #020H        ;闪烁时分间隔:
XRL    LCD_buff, #002H        ;闪烁分秒间隔:

CPL    B_Second
JNB    B_Second, L_Main_Loop

INC    second
MOV    A, second
CJNE   A, #60, L_Main_Load
MOV    second, #0             ;1 分钟到
INC    minute
MOV    A, minute
CJNE   A, #60, L_Main_Load
MOV    minute, #0;
INC    hour
MOV    A, hour
CJNE   A, #24, L_Main_Load
MOV    hour, #0               ;24 小时到

```

#### L\_Main\_Load:

```

LCALL  F_LoadRTC           ;显示时间
LJMP   L_Main_Loop

```

\*\*\*\*\*

#### F\_Timer0\_init:

```

CLR    TR0                  ; 停止计数
ANL    TMOD, #0f0H
SETB   ET0                  ; 允许中断
ORL    TMOD, #0             ; 工作模式, 0: 16 位自动重装
ANL    INT_CLKO, #NOT 0x01   ; 不输出时钟
ORL    AUXR, #0x80          ; 1T mode
MOV    TH0, #HIGH (-22118)   ; 2ms
MOV    TL0, #LOW  (-22118)   ;
SETB   TR0                  ; 开始运行
RET

```

\*\*\*\*\*

#### F\_Timer0 Interrupt: ;Timer0 1ms 中断函数

```

PUSH    PSW                ;PSW 入栈
PUSH    ACC                 ;ACC 入栈
PUSH    AR0
PUSH    AR7
PUSH    DPH

```

```

PUSH      DPL

LCALL     F_LCD_scan
SETB      B_2ms

POP       DPL
POP       DPH
POP       AR7
POP       AR0
POP       ACC      ;ACC 出栈
POP       PSW      ;PSW 出栈
RET

```

\*\*\*\*\* 显示时间 \*\*\*\*\*

*F\_LoadRTC:*

```

MOV       R6, #1      ;LCD_load(1,hour/10);
MOV       A, hour
MOV       B, #10
DIV       AB
MOV       R7, A
LCALL     F_LCD_load   ;R6 为第几个数字, 为1~6, R7 为要显示的数字

MOV       R6, #2      ;LCD_load(2,hour%10);
MOV       A, hour
MOV       B, #10
DIV       AB
MOV       R7, B
LCALL     F_LCD_load   ;R6 为第几个数字, 为1~6, R7 为要显示的数字

MOV       R6, #3      ;LCD_load(3,minute/10);
MOV       A, minute
MOV       B, #10
DIV       AB
MOV       R7, A
LCALL     F_LCD_load   ;R6 为第几个数字, 为1~6, R7 为要显示的数字

MOV       R6, #4      ;LCD_load(4,minute%10);
MOV       A, minute
MOV       B, #10
DIV       AB
MOV       R7, B
LCALL     F_LCD_load   ;R6 为第几个数字, 为1~6, R7 为要显示的数字

MOV       R6, #5      ;LCD_load(5,second/10);
MOV       A, second
MOV       B, #10
DIV       AB
MOV       R7, A
LCALL     F_LCD_load   ;R6 为第几个数字, 为1~6, R7 为要显示的数字

MOV       R6, #6      ;LCD_load(6,second%10);
MOV       A, second
MOV       B, #10
DIV       AB
MOV       R7, B
LCALL     F_LCD_load   ;R6 为第几个数字, 为1~6, R7 为要显示的数字

RET

```

\*\*\*\*\*

**T\_COM:**

**DB**                    **008H, 004H, 002H, 001H**

**F\_LCD\_scan:**

```

MOV    A, scan_index      ;j = scan_index >> 1;
CLR    C
RRC    A
MOV    R7, A              ;R7 = j
ADD    A, #LCD_buff
MOV    R0, A              ;R0 = LCD_buff[j]
ORL    P2M1, #00fH        ;全部COM 输出高阻, COM 为中点电压
ANL    P2M0, #0f0H

MOV    A, scan_index
JNB    ACC.0, L_LCD_Scan2 ;if(scan_index & 1) //反相扫描
MOV    A, @R0
CPL    A
MOV    P1, A
MOV    A, R0              ;P2 = ~(LCD_buff[j/4] & 0xf0);
ADD    A, #4
MOV    R0, A
MOV    A, @R0
ANL    A, #0f0H
CPL    A
MOV    P2, A
SJMP   L_LCD_Scan3

```

**L\_LCD\_Scan2:**

```

MOV    A, @R0            ;正相扫描
MOV    P1, A
MOV    A, R0
ADD    A, #4
MOV    R0, A
MOV    A, @R0
ANL    A, #0f0H
MOV    P2, A

```

**L\_LCD\_Scan3:**

```

MOV    DPTR, #T_COM      ;某个COM 设置为推挽输出
MOV    A, R7
MOVC   A, @A+DPTR
ORL    P2M0, A
CPL    A
ANL    P2M1, A

INC    scan_index        ;if(++scan_index == 8)   scan_index = 0;
MOV    A, scan_index
CJNE   A, #8, L_QuitLcdScan
MOV    scan_index, #0

```

**L\_QuitLcdScan:**

**RET**

\*\*\*\*\* 标准字库 \*\*\*\*\*

**T\_Display:**

```

;      0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
DB     03FH,006H,05BH,04FH,066H,06DH,07DH,007H,07FH,06FH,077H,07CH,039H,05EH,079H,071H
;      black -

```

*DB*                    *000H,040H*

\*\*\*\*\* 对第1~6 数字装载显示函数 算法简单 \*\*\*\*\*

```

F_LCD_load:
    MOV     DPTR, #T_Display
    MOV     A, R7
    MOVC    A, @A+DPTR
    MOV     B, A
    MOV     A, R6
    CJNE    A, #1, L_NotLoadChar1
    MOV     R0, #LCD_buff
    MOV     A, @R0
    MOV     C, B.3
    MOV     ACC.6, C
    MOV     @R0, A
    INC     R0
    MOV     A, @R0
    MOV     C, B.2
    MOV     ACC.6, C
    MOV     C, B.4
    MOV     ACC.7, C
    MOV     @R0, A
    INC     R0
    MOV     A, @R0
    MOV     C, B.1
    MOV     ACC.6, C
    MOV     C, B.6
    MOV     ACC.7, C
    MOV     @R0, A
    INC     R0
    MOV     A, @R0
    MOV     C, B.0
    MOV     ACC.6, C
    MOV     C, B.5
    MOV     ACC.7, C
    MOV     @R0, A
    RET

```

```

L_NotLoadChar1:
    CJNE    A, #2, L_NotLoadChar2
    MOV     R0, #LCD_buff
    MOV     A, @R0
    MOV     C, B.3
    MOV     ACC.4, C
    MOV     @R0, A

    INC     R0
    MOV     A, @R0
    MOV     C, B.2
    MOV     ACC.4, C
    MOV     C, B.4
    MOV     ACC.5, C
    MOV     @R0, A

    INC     R0

```

```

MOV    A, @R0
MOV    C, B.1                ;B
MOV    ACC.4, C
MOV    C, B.6                ;G
MOV    ACC.5, C
MOV    @R0, A

INC    R0
MOV    A, @R0
MOV    C, B.0                ;A
MOV    ACC.4, C
MOV    C, B.5                ;F
MOV    ACC.5, C
MOV    @R0, A
RET

```

*L\_NotLoadChar2:*

```

CJNE   A, #3, L_NotLoadChar3
MOV    R0, #LCD_buff
MOV    A, @R0
MOV    C, B.3                ;D
MOV    ACC.2, C
MOV    @R0, A

INC    R0
MOV    A, @R0
MOV    C, B.2                ;C
MOV    ACC.2, C
MOV    C, B.4                ;E
MOV    ACC.3, C
MOV    @R0, A

INC    R0
MOV    A, @R0
MOV    C, B.1                ;B
MOV    ACC.2, C
MOV    C, B.6                ;G
MOV    ACC.3, C
MOV    @R0, A

INC    R0
MOV    A, @R0
MOV    C, B.0                ;A
MOV    ACC.2, C
MOV    C, B.5                ;F
MOV    ACC.3, C
MOV    @R0, A
RET

```

*L\_NotLoadChar3:*

```

CJNE   A, #4, L_NotLoadChar4
MOV    R0, #LCD_buff
MOV    A, @R0
MOV    C, B.3                ;D
MOV    ACC.0, C
MOV    @R0, A

INC    R0
MOV    A, @R0

```

```

MOV      C, B.2                      ;C
MOV      ACC.0, C
MOV      C, B.4                      ;E
MOV      ACC.1, C
MOV      @R0, A

```

```

INC      R0
MOV      A, @R0
MOV      C, B.1                      ;B
MOV      ACC.0, C
MOV      C, B.6                      ;G
MOV      ACC.1, C
MOV      @R0, A

```

```

INC      R0
MOV      A, @R0
MOV      C, B.0                      ;A
MOV      ACC.0, C
MOV      C, B.5                      ;F
MOV      ACC.1, C
MOV      @R0, A
RET

```

**L\_NotLoadChar4:**

```

CJNE     A, #5, L_NotLoadChar5
MOV      R0, #LCD_buff+4
MOV      A, @R0
MOV      C, B.3                      ;D
MOV      ACC.6, C
MOV      @R0, A

```

```

INC      R0
MOV      A, @R0
MOV      C, B.2                      ;C
MOV      ACC.6, C
MOV      C, B.4                      ;E
MOV      ACC.7, C
MOV      @R0, A

```

```

INC      R0
MOV      A, @R0
MOV      C, B.1                      ;B
MOV      ACC.6, C
MOV      C, B.6                      ;G
MOV      ACC.7, C
MOV      @R0, A

```

```

INC      R0
MOV      A, @R0
MOV      C, B.0                      ;A
MOV      ACC.6, C
MOV      C, B.5                      ;F
MOV      ACC.7, C
MOV      @R0, A
RET

```

**L\_NotLoadChar5:**

```

CJNE     A, #6, L_NotLoadChar6
MOV      R0, #LCD_buff+4

```

```
MOV    A, @R0
MOV    C, B.3                ;D
MOV    ACC.4, C
MOV    @R0, A

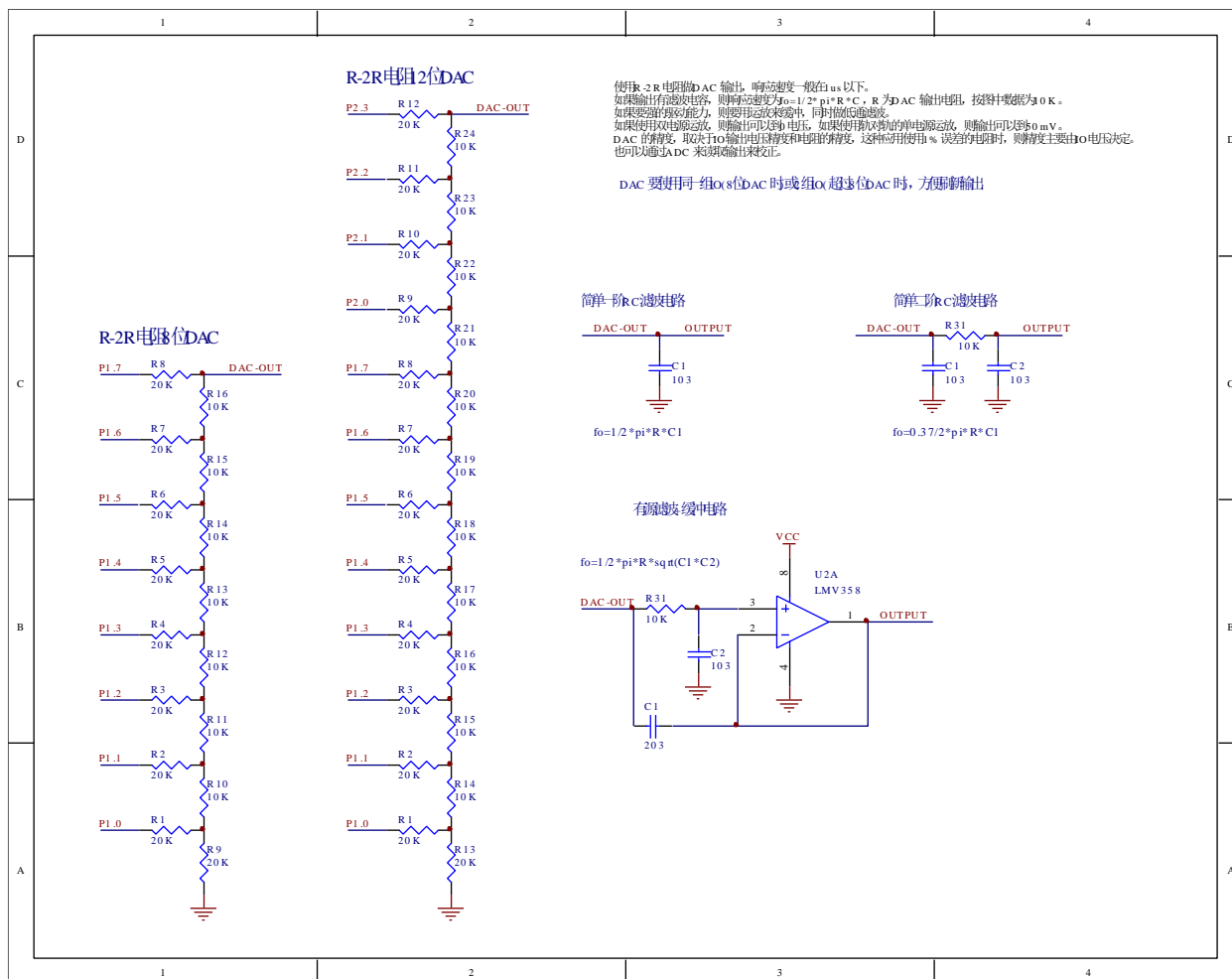
INC     R0
MOV     A, @R0
MOV     C, B.2                ;C
MOV     ACC.4, C
MOV     C, B.4                ;E
MOV     ACC.5, C
MOV     @R0, A

INC     R0
MOV     A, @R0
MOV     C, B.1                ;B
MOV     ACC.4, C
MOV     C, B.6                ;G
MOV     ACC.5, C
MOV     @R0, A

INC     R0
MOV     A, @R0
MOV     C, B.0                ;A
MOV     ACC.4, C
MOV     C, B.5                ;F
MOV     ACC.5, C
MOV     @R0, A
RET
L_NotLoadChar6:
RET
END
```



## 10.12 使用 I/O 和 R-2R 电阻分压实现 DAC 的经典线路图



## 11 指令系统

| 助记符              | 指令说明                 | 字节 | 时钟 |
|------------------|----------------------|----|----|
| ADD A,Rn         | 寄存器内容加到累加器           | 1  | 1  |
| ADD A,direct     | 直接地址单元的数据加到累加器       | 2  | 1  |
| ADD A,@Ri        | 间接地址单元的数据加到累加器       | 1  | 1  |
| ADD A,#data      | 立即数加到累加器             | 2  | 1  |
| ADDC A,Rn        | 寄存器带进位加到累加器          | 1  | 1  |
| ADDC A,direct    | 直接地址单元的数据带进位加到累加器    | 2  | 1  |
| ADDC A,@Ri       | 间接地址单元的数据带进位加到累加器    | 1  | 1  |
| ADDC A,#data     | 立即数带进位加到累加器          | 2  | 1  |
| SUBB A,Rn        | 累加器带借位减寄存器内容         | 1  | 1  |
| SUBB A,direct    | 累加器带借位减直接地址单元的内容     | 2  | 1  |
| SUBB A,@Ri       | 累加器带借位减间接地址单元的内容     | 1  | 1  |
| SUBB A,#data     | 累加器带借位减立即数           | 2  | 1  |
| INC A            | 累加器加1                | 1  | 1  |
| INC Rn           | 寄存器加1                | 1  | 1  |
| INC direct       | 直接地址单元加1             | 2  | 1  |
| INC @Ri          | 间接地址单元加1             | 1  | 1  |
| DEC A            | 累加器减1                | 1  | 1  |
| DEC Rn           | 寄存器减1                | 1  | 1  |
| DEC direct       | 直接地址单元减1             | 2  | 1  |
| DEC @Ri          | 间接地址单元减1             | 1  | 1  |
| INC DPTR         | 地址寄存器DPTR加1          | 1  | 1  |
| MUL AB           | A乘以B, B存放高字节, A存放低字节 | 1  | 2  |
| DIV AB           | A除以B, B存放余数, A存放商    | 1  | 6  |
| DA A             | 累加器十进制调整             | 1  | 3  |
| ANL A,Rn         | 累加器与寄存器相与            | 1  | 1  |
| ANL A,direct     | 累加器与直接地址单元相与         | 2  | 1  |
| ANL A,@Ri        | 累加器与间接地址单元相与         | 1  | 1  |
| ANL A,#data      | 累加器与立即数相与            | 2  | 1  |
| ANL direct,A     | 直接地址单元与累加器相与         | 2  | 1  |
| ANL direct,#data | 直接地址单元与立即数相与         | 3  | 1  |
| ORL A,Rn         | 累加器与寄存器相或            | 1  | 1  |
| ORL A,direct     | 累加器与直接地址单元相或         | 2  | 1  |
| ORL A,@Ri        | 累加器与间接地址单元相或         | 1  | 1  |

|      |              |                 |   |   |
|------|--------------|-----------------|---|---|
| ORL  | A,#data      | 累加器与立即数相或       | 2 | 1 |
| ORL  | direct,A     | 直接地址单元与累加器相或    | 2 | 1 |
| ORL  | direct,#data | 直接地址单元与立即数相或    | 3 | 1 |
| XRL  | A,Rn         | 累加器与寄存器相异或      | 1 | 1 |
| XRL  | A,direct     | 累加器与直接地址单元相异或   | 2 | 1 |
| XRL  | A,@Ri        | 累加器与间接地址单元相异或   | 1 | 1 |
| XRL  | A,#data      | 累加器与立即数相异或      | 2 | 1 |
| XRL  | direct,A     | 直接地址单元与累加器相异或   | 2 | 1 |
| XRL  | direct,#data | 直接地址单元与立即数相异或   | 3 | 1 |
| CLR  | A            | 累加器清0           | 1 | 1 |
| CPL  | A            | 累加器取反           | 1 | 1 |
| RL   | A            | 累加器循环左移         | 1 | 1 |
| RLC  | A            | 累加器带进位循环左移      | 1 | 1 |
| RR   | A            | 累加器循环右移         | 1 | 1 |
| RRC  | A            | 累加器带进位循环右移      | 1 | 1 |
| SWAP | A            | 累加器高低半字节交换      | 1 | 1 |
| CLR  | C            | 清零进位位           | 1 | 1 |
| CLR  | bit          | 清0直接地址位         | 2 | 1 |
| SETB | C            | 置1进位位           | 1 | 1 |
| SETB | bit          | 置1直接地址位         | 2 | 1 |
| CPL  | C            | 进位位求反           | 1 | 1 |
| CPL  | bit          | 直接地址位求反         | 2 | 1 |
| ANL  | C,bit        | 进位位和直接地址位相与     | 2 | 1 |
| ANL  | C,/bit       | 进位位和直接地址位的反码相与  | 2 | 1 |
| ORL  | C,bit        | 进位位和直接地址位相或     | 2 | 1 |
| ORL  | C,/bit       | 进位位和直接地址位的反码相或  | 2 | 1 |
| MOV  | C,bit        | 直接地址位送入进位位      | 2 | 1 |
| MOV  | bit,C        | 进位位送入直接地址位      | 2 | 1 |
| MOV  | A,Rn         | 寄存器内容送入累加器      | 1 | 1 |
| MOV  | A,direct     | 直接地址单元中的数据送入累加器 | 2 | 1 |
| MOV  | A,@Ri        | 间接地址中的数据送入累加器   | 1 | 1 |
| MOV  | A,#data      | 立即数送入累加器        | 2 | 1 |
| MOV  | Rn,A         | 累加器内容送入寄存器      | 1 | 1 |
| MOV  | Rn,direct    | 直接地址单元中的数据送入寄存器 | 2 | 1 |
| MOV  | Rn,#data     | 立即数送入寄存器        | 2 | 1 |
| MOV  | direct,A     | 累加器内容送入直接地址单元   | 2 | 1 |
| MOV  | direct,Rn    | 寄存器内容送入直接地址单元   | 2 | 1 |

|       |               |                          |   |                    |
|-------|---------------|--------------------------|---|--------------------|
| MOV   | direct,direct | 直接地址单元中的数据送入另一个直接地址单元    | 3 | 1                  |
| MOV   | direct,@Ri    | 间接地址中的数据送入直接地址单元         | 2 | 1                  |
| MOV   | direct,#data  | 立即数送入直接地址单元              | 3 | 1                  |
| MOV   | @Ri,A         | 累加器内容送间接地址单元             | 1 | 1                  |
| MOV   | @Ri,direct    | 直接地址单元数据送入间接地址单元         | 2 | 1                  |
| MOV   | @Ri,#data     | 立即数送入间接地址单元              | 2 | 1                  |
| MOV   | DPTR,#data16  | 16位立即数送入数据指针             | 3 | 1                  |
| MOVC  | A,@A+DPTR     | 以DPTR为基地址变址寻址单元中的数据送入累加器 | 1 | 4                  |
| MOVC  | A,@A+PC       | 以PC为基地址变址寻址单元中的数据送入累加器   | 1 | 3                  |
| MOVB  | A,@Ri         | 扩展地址(8位地址)的内容送入累加器A中     | 1 | 3 <sup>[1]</sup>   |
| MOVB  | A,@DPTR       | 扩展RAM(16位地址)的内容送入累加器A中   | 1 | 2 <sup>[1]</sup>   |
| MOVB  | @Ri,A         | 将累加器A的内容送入扩展RAM(8位地址)中   | 1 | 3 <sup>[1]</sup>   |
| MOVB  | @DPTR,A       | 将累加器A的内容送入扩展RAM(16位地址)中  | 1 | 2 <sup>[1]</sup>   |
| PUSH  | direct        | 直接地址单元中的数据压入堆栈           | 2 | 1                  |
| POP   | direct        | 栈底数据弹出送入直接地址单元           | 2 | 1                  |
| XCH   | A,Rn          | 寄存器与累加器交换                | 1 | 1                  |
| XCH   | A,direct      | 直接地址单元与累加器交换             | 2 | 1                  |
| XCH   | A,@Ri         | 间接地址与累加器交换               | 1 | 1                  |
| XCHD  | A,@Ri         | 间接地址的低半字节与累加器交换          | 1 | 1                  |
| ACALL | addr11        | 短调用子程序                   | 2 | 3                  |
| LCALL | addr16        | 长调用子程序                   | 3 | 3                  |
| RET   |               | 子程序返回                    | 1 | 3                  |
| RETI  |               | 中断返回                     | 1 | 3                  |
| AJMP  | addr11        | 短跳转                      | 2 | 3                  |
| LJMP  | addr16        | 长跳转                      | 3 | 3                  |
| SJMP  | rel           | 相对跳转                     | 2 | 3                  |
| JMP   | @A+DPTR       | 相对于DPTR的间接跳转             | 1 | 4                  |
| JZ    | rel           | 累加器为零跳转                  | 2 | 1/3 <sup>[2]</sup> |
| JNZ   | rel           | 累加器非零跳转                  | 2 | 1/3 <sup>[2]</sup> |
| JC    | rel           | 进位位为1跳转                  | 2 | 1/3 <sup>[2]</sup> |
| JNC   | rel           | 进位位为0跳转                  | 2 | 1/3 <sup>[2]</sup> |
| JB    | bit,rel       | 直接地址位为1则跳转               | 3 | 1/3 <sup>[2]</sup> |
| JNB   | bit,rel       | 直接地址位为0则跳转               | 3 | 1/3 <sup>[2]</sup> |
| JBC   | bit,rel       | 直接地址位为1则跳转, 该位清0         | 3 | 1/3 <sup>[2]</sup> |
| CJNE  | A,direct,rel  | 累加器与直接地址单元不相等跳转          | 3 | 2/3 <sup>[3]</sup> |
| CJNE  | A,#data,rel   | 累加器与立即数不相等跳转             | 3 | 1/3 <sup>[2]</sup> |
| CJNE  | Rn,#data,rel  | 寄存器与立即数不相等跳转             | 3 | 2/3 <sup>[3]</sup> |

|      |               |                 |   |                    |
|------|---------------|-----------------|---|--------------------|
| CJNE | @Ri,#data,rel | 间接地址单元与立即数不相等跳转 | 3 | 2/3 <sup>[3]</sup> |
| DJNZ | Rn,rel        | 寄存器减1后非零跳转      | 2 | 2/3 <sup>[3]</sup> |
| DJNZ | direct,rel    | 直接地址单元减1后非零跳转   | 3 | 2/3 <sup>[3]</sup> |
| NOP  |               | 空操作             | 1 | 1                  |

<sup>[1]</sup>: 访问外部扩展 RAM 时, 指令的执行周期与寄存器 BUS\_SPEED 中的 SPEED[2:0]位有关

<sup>[2]</sup>: 对于条件跳转语句的执行时间会依据条件是否满足而不同。当条件不满足时, 不会发生跳转而继续执行下一条指令, 此时条件跳转语句的执行时间为 1 个时钟; 当条件满足时, 则会发生跳转, 此时条件跳转语句的执行时间为 3 个时钟。

<sup>[3]</sup>: 对于条件跳转语句的执行时间会依据条件是否满足而不同。当条件不满足时, 不会发生跳转而继续执行下一条指令, 此时条件跳转语句的执行时间为 2 个时钟; 当条件满足时, 则会发生跳转, 此时条件跳转语句的执行时间为 3 个时钟。

## 12 中断系统

(C 语言程序中使用中断号大于 31 的中断时, 在 Keil 中编译会报错, 解决办法请参考附录)

中断系统是为使 CPU 具有对外界紧急事件的实时处理能力而设置的。

当中央处理机 CPU 正在处理某件事的时候外界发生了紧急事件请求, 要求 CPU 暂停当前的工作, 转而去处理这个紧急事件, 处理完以后, 再回到原来被中断的地方, 继续原来的工作, 这样的过程称为中断。实现这种功能的部件称为中断系统, 请示 CPU 中断的请求源称为中断源。微型机的中断系统一般允许多个中断源, 当几个中断源同时向 CPU 请求中断, 要求为它服务的时候, 这就存在 CPU 优先响应哪一个中断源请求的问题。通常根据中断源的轻重缓急排队, 优先处理最紧急事件的中断请求源, 即规定每一个中断源有一个优先级别。CPU 总是先响应优先级别最高的中断请求。

当 CPU 正在处理一个中断源请求的时候 (执行相应的中断服务程序), 发生了另外一个优先级比它还高的中断源请求。如果 CPU 能够暂停对原来中断源的服务程序, 转而去处理优先级更高的中断请求源, 处理完以后, 再回到原低级中断服务程序, 这样的过程称为中断嵌套。这样的中断系统称为多级中断系统, 没有中断嵌套功能的中断系统称为单级中断系统。

用户可以用关总中断允许位 (EA/IE.7) 或相应中断的允许位屏蔽相应的中断请求, 也可以用打开相应的中断允许位来使 CPU 响应相应的中断申请, 每一个中断源可以用软件独立地控制为开中断或关中断状态, 部分中断的优先级别均可用软件设置。高优先级的中断请求可以打断低优先级的中断, 反之, 低优先级的中断请求不可以打断高优先级的中断。当两个相同优先级的中断同时产生时, 将由查询次序来决定系统先响应哪个中断。

### 12.1 STC8H 系列中断源

下表中 √ 表示对应的系列有相应的中断源

| 中断源                            | STC8H1K16系列 | STC8H1K08系列 | STC8H3K64S1系列 | STC8H3K64S2系列 | STC8H8K64U-A系列 | STC8H8K64U-B/C/D系列 | STC8H1K08T系列 | STC8H4K64TL系列 | STC8H4K64TLD系列 |
|--------------------------------|-------------|-------------|---------------|---------------|----------------|--------------------|--------------|---------------|----------------|
| 外部中断 0 中断 (INT0)<br>支持下降沿和边沿中断 | √           | √           | √             | √             | √              | √                  | √            | √             | √              |
| 定时器 0 中断 (Timer0)              | √           | √           | √             | √             | √              | √                  | √            | √             | √              |
| 外部中断 1 中断 (INT1)<br>支持下降沿和边沿中断 | √           | √           | √             | √             | √              | √                  | √            | √             | √              |
| 定时器 1 中断 (Timer1)              | √           | √           | √             | √             | √              | √                  | √            | √             | √              |
| 串口 1 中断 (UART1)                | √           | √           | √             | √             | √              | √                  | √            | √             | √              |
| 模数转换中断 (ADC)                   | √           | √           | √             | √             | √              | √                  | √            | √             | √              |
| 低压检测中断 (LVD)                   | √           | √           | √             | √             | √              | √                  | √            | √             | √              |
| 串口 2 中断 (UART2)                | √           | √           | √             | √             | √              | √                  | √            | √             | √              |
| 串行外设接口中断 (SPI)                 | √           | √           | √             | √             | √              | √                  | √            | √             | √              |

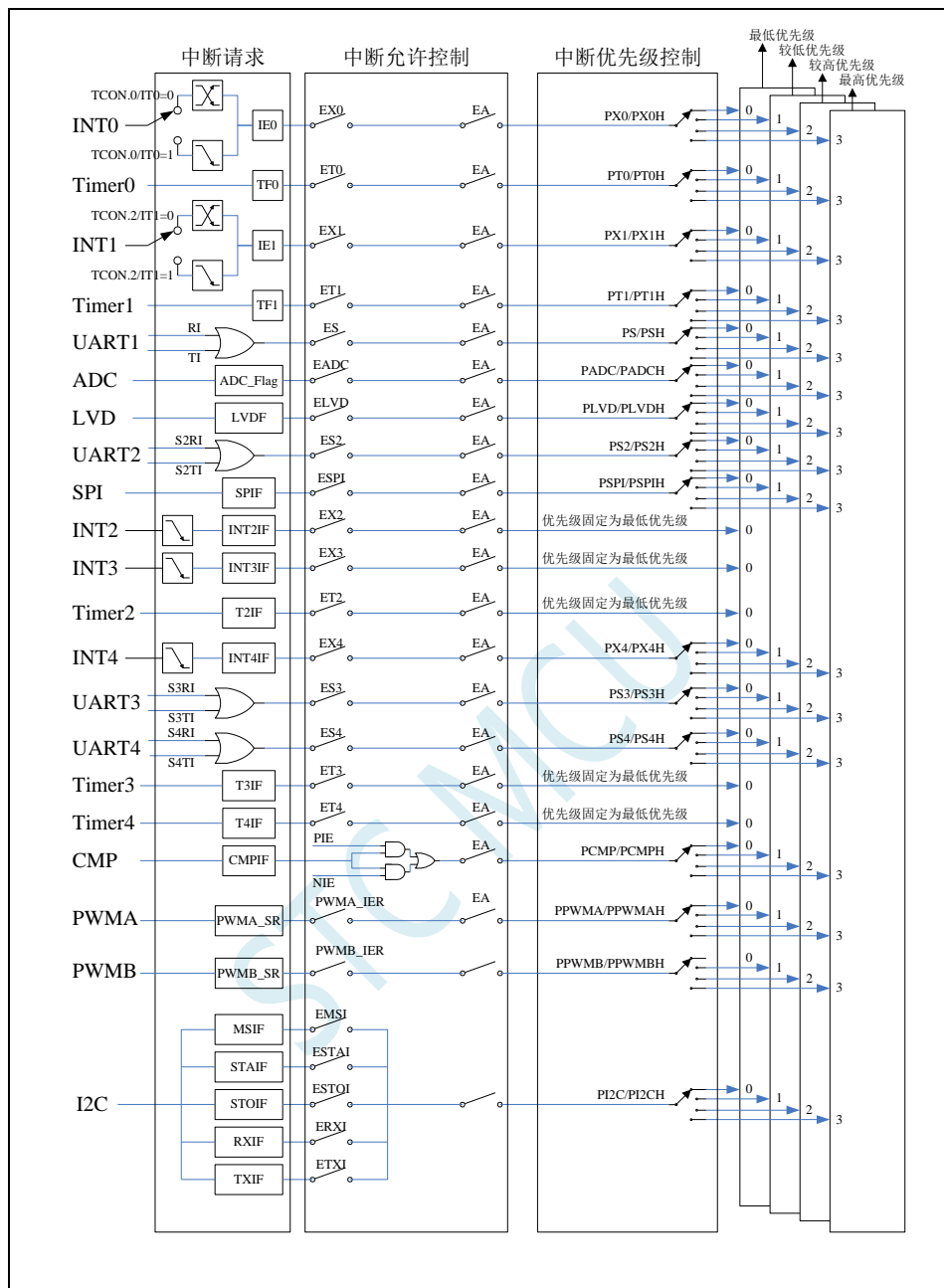
|                               |   |   |   |   |   |   |   |   |   |
|-------------------------------|---|---|---|---|---|---|---|---|---|
| 外部中断 2 中断 (INT2)<br>支持下降沿中断   | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 外部中断 3 中断 (INT3)<br>支持下降沿中断   | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 定时器 2 中断 (Timer2)             | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 外部中断 4 中断 (INT4)              | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 串口 3 中断 (UART3)               |   |   | √ |   | √ | √ |   | √ | √ |
| 串口 4 中断 (UART4)               |   |   | √ |   | √ | √ |   | √ | √ |
| 定时器 3 中断 (Timer3)             | √ |   | √ | √ | √ | √ |   | √ | √ |
| 定时器 4 中断 (Timer4)             | √ |   | √ | √ | √ | √ |   | √ | √ |
| 比较器中断 (CMP)                   | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| I2C 总线中断                      | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| PWMA                          | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| PWMB                          | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| USB 中断                        |   |   |   |   | √ | √ |   |   |   |
| 触摸按键中断                        |   |   |   |   |   |   | √ | √ | √ |
| RTC 中断                        |   |   |   |   |   | √ | √ | √ | √ |
| P0 口中断<br>支持下降沿、上升沿、高电平和低电平中断 |   |   | √ | √ |   | √ |   | √ | √ |
| P1 口中断<br>支持下降沿、上升沿、高电平和低电平中断 |   |   | √ | √ |   | √ | √ | √ | √ |
| P2 口中断<br>支持下降沿、上升沿、高电平和低电平中断 |   |   | √ | √ |   | √ |   | √ | √ |
| P3 口中断<br>支持下降沿、上升沿、高电平和低电平中断 |   |   | √ | √ |   | √ | √ | √ | √ |
| P4 口中断<br>支持下降沿、上升沿、高电平和低电平中断 |   |   | √ | √ |   | √ |   | √ | √ |
| P5 口中断<br>支持下降沿、上升沿、高电平和低电平中断 |   |   | √ | √ |   | √ | √ | √ | √ |
| P6 口中断<br>支持下降沿、上升沿、高电平和低电平中断 |   |   | √ | √ |   | √ |   |   |   |
| P7 口中断<br>支持下降沿、上升沿、高电平和低电平中断 |   |   | √ | √ |   | √ |   |   |   |
| DMA_M2M 中断                    |   |   |   |   |   | √ | √ | √ | √ |
| DMA_ADC 中断                    |   |   |   |   |   | √ | √ | √ | √ |
| DMA_SPI 中断                    |   |   |   |   |   | √ | √ | √ | √ |
| DMA_UR1T 中断                   |   |   |   |   |   | √ | √ | √ | √ |
| DMA_UR1R 中断                   |   |   |   |   |   | √ | √ | √ | √ |
| DMA_UR2T 中断                   |   |   |   |   |   | √ | √ | √ | √ |
| DMA_UR2R 中断                   |   |   |   |   |   | √ | √ | √ | √ |

|             |  |  |  |  |  |   |   |   |   |
|-------------|--|--|--|--|--|---|---|---|---|
| DMA_UR3T 中断 |  |  |  |  |  | √ | √ | √ | √ |
| DMA_UR3R 中断 |  |  |  |  |  | √ | √ | √ | √ |
| DMA_UR4T 中断 |  |  |  |  |  | √ | √ | √ | √ |
| DMA_UR4R 中断 |  |  |  |  |  | √ | √ | √ | √ |
| DMA_LCM 中断  |  |  |  |  |  | √ | √ | √ | √ |
| LCM 中断      |  |  |  |  |  | √ |   | √ | √ |
|             |  |  |  |  |  |   |   |   |   |

STC MCU



## 12.2 STC8H 中断结构图



## 12.3 STC8H 系列中断列表

| 中断源    | 中断向量  | 次序 | 优先级设置      | 优先级     | 中断请求位        | 中断允许位   |
|--------|-------|----|------------|---------|--------------|---------|
| INT0   | 0003H | 0  | PX0PX0H    | 0/1/2/3 | IE0          | EX0     |
| Timer0 | 000BH | 1  | PT0,PT0H   | 0/1/2/3 | TF0          | ET0     |
| INT1   | 0013H | 2  | PX1,PX1H   | 0/1/2/3 | IE1          | EX1     |
| Timer1 | 001BH | 3  | PT1,PT1H   | 0/1/2/3 | TF1          | ET1     |
| UART1  | 0023H | 4  | PS,PSH     | 0/1/2/3 | RI    TI     | ES      |
| ADC    | 002BH | 5  | PADC,PADCH | 0/1/2/3 | ADC_FLAG     | EADC    |
| LVD    | 0033H | 6  | PLVD,PLVDH | 0/1/2/3 | LVDF         | ELVD    |
| PCA    | 003BH | 7  | PPCA,PPCAH | 0/1/2/3 | CF           | ECF     |
|        |       |    |            |         | CCF0         | ECCF0   |
|        |       |    |            |         | CCF1         | ECCF1   |
|        |       |    |            |         | CCF2         | ECCF2   |
|        |       |    |            |         | CCF3         | ECCF3   |
| UART2  | 0043H | 8  | PS2,PS2H   | 0/1/2/3 | S2RI    S2TI | ES2     |
| SPI    | 004BH | 9  | PSPI,PSPIH | 0/1/2/3 | SPIF         | ESPI    |
| INT2   | 0053H | 10 |            | 0       | INT2IF       | EX2     |
| INT3   | 005BH | 11 |            | 0       | INT3IF       | EX3     |
| Timer2 | 0063H | 12 |            | 0       | T2IF         | ET2     |
| INT4   | 0083H | 16 | PX4,PX4H   | 0/1/2/3 | INT4IF       | EX4     |
| UART3  | 008BH | 17 | PS3,PS3H   | 0/1/2/3 | S3RI    S3TI | ES3     |
| UART4  | 0093H | 18 | PS4,PS4H   | 0/1/2/3 | S4RI    S4TI | ES4     |
| Timer3 | 009BH | 19 |            | 0       | T3IF         | ET3     |
| Timer4 | 00A3H | 20 |            | 0       | T4IF         | ET4     |
| CMP    | 00ABH | 21 | PCMP,PCMPH | 0/1/2/3 | CMPIF        | PIE NIE |
| I2C    | 00C3H | 24 | PI2C,PI2CH | 0/1/2/3 | MSIF         | EMSI    |
|        |       |    |            |         | STAIF        | ESTAI   |
|        |       |    |            |         | RXIF         | ERXI    |
|        |       |    |            |         | TXIF         | ETXI    |
|        |       |    |            |         | STOIF        | ESTOI   |

| 中断源         | 中断向量  | 次序 | 优先级设置                | 优先级     | 中断请求位      | 中断允许位    |
|-------------|-------|----|----------------------|---------|------------|----------|
| USB         | 00CBH | 25 | PUSB,PUSBH           | 0/1/2/3 | USB Events | EUSB     |
| PWMA        | 00D3H | 26 | PPWMA,PPWMAH         | 0/1/2/3 | PWMA_SR    | PWMA_IER |
| PWMB        | 00DBH | 27 | PPWMB,PPWMBH         | 0/1/2/3 | PWMB_SR    | PWMB_IER |
| TKSU        | 011BH | 35 | PTKSU,PTKSUH         | 0/1/2/3 | TKIF       | ETKSUI   |
| RTC         | 0123H | 36 | PRTC,PRTCH           | 0/1/2/3 | ALAIF      | EALAI    |
|             |       |    |                      |         | DAYIF      | EDAYI    |
|             |       |    |                      |         | HOURLIF    | EHOURI   |
|             |       |    |                      |         | MINIF      | EMINI    |
|             |       |    |                      |         | SECIF      | ESECI    |
|             |       |    |                      |         | SEC2IF     | ESEC2I   |
|             |       |    |                      |         | SEC8IF     | ESEC8I   |
|             |       |    |                      |         | SEC32IF    | ESEC32I  |
| P0 中断       | 012BH | 37 | PINIPL[0], PINIPH[0] | 0/1/2/3 | P0INTF     | P0INTE   |
| P1 中断       | 0133H | 38 | PINIPL[1], PINIPH[1] | 0/1/2/3 | P1INTF     | P1INTE   |
| P2 中断       | 013BH | 39 | PINIPL[2], PINIPH[2] | 0/1/2/3 | P2INTF     | P2INTE   |
| P3 中断       | 0143H | 40 | PINIPL[3], PINIPH[3] | 0/1/2/3 | P3INTF     | P3INTE   |
| P4 中断       | 014BH | 41 | PINIPL[4], PINIPH[4] | 0/1/2/3 | P4INTF     | P4INTE   |
| P5 中断       | 0153H | 42 | PINIPL[5], PINIPH[5] | 0/1/2/3 | P5INTF     | P5INTE   |
| P6 中断       | 015BH | 43 | PINIPL[6], PINIPH[6] | 0/1/2/3 | P6INTF     | P6INTE   |
| P7 中断       | 0163H | 44 | PINIPL[7], PINIPH[7] | 0/1/2/3 | P7INTF     | P7INTE   |
| DMA_M2M 中断  | 017BH | 47 | M2MIP[1:0]           | 0/1/2/3 | M2MIF      | M2MIE    |
| DMA_ADC 中断  | 0183H | 48 | ADCIP[1:0]           | 0/1/2/3 | ADCIF      | ADCIE    |
| DMA_SPI 中断  | 018BH | 49 | SPIIP[1:0]           | 0/1/2/3 | SPIIF      | SPIIE    |
| DMA_UR1T 中断 | 0193H | 50 | UR1TIP[1:0]          | 0/1/2/3 | UR1TIF     | UR1TIE   |
| DMA_UR1R 中断 | 019BH | 51 | UR1RIP[1:0]          | 0/1/2/3 | UR1RIF     | UR1RIE   |
| DMA_UR2T 中断 | 01A3H | 52 | UR2TIP[1:0]          | 0/1/2/3 | UR2TIF     | UR2TIE   |
| DMA_UR2R 中断 | 01ABH | 53 | UR2RIP[1:0]          | 0/1/2/3 | UR2RIF     | UR2RIE   |
| DMA_UR3T 中断 | 01B3H | 54 | UR3TIP[1:0]          | 0/1/2/3 | UR3TIF     | UR3TIE   |
| DMA_UR3R 中断 | 01BBH | 55 | UR3RIP[1:0]          | 0/1/2/3 | UR3RIF     | UR3RIE   |
| DMA_UR4T 中断 | 01C3H | 56 | UR4TIP[1:0]          | 0/1/2/3 | UR4TIF     | UR4TIE   |
| DMA_UR4R 中断 | 01CBH | 57 | UR4RIP[1:0]          | 0/1/2/3 | UR4RIF     | UR3RIE   |
| DMA_LCM 中断  | 01D3H | 58 | LCMIP[1:0]           | 0/1/2/3 | LCMIF      | LCMIE    |
| LCM 中断      | 01DBH | 59 | LCMIFIP[1:0]         | 0/1/2/3 | LCMIFIF    | LCMIFIE  |

在 C 语言中声明中断服务程序

```
void INT0_Routine(void)    interrupt 0;
void TM0_Routine(void)     interrupt 1;
void INT1_Routine(void)    interrupt 2;
void TM1_Routine(void)     interrupt 3;
void UART1_Routine(void)   interrupt 4;
void ADC_Routine(void)     interrupt 5;
void LVD_Routine(void)     interrupt 6;
void PCA_Routine(void)     interrupt 7;
void UART2_Routine(void)   interrupt 8;
void SPI_Routine(void)     interrupt 9;
void INT2_Routine(void)    interrupt 10;
void INT3_Routine(void)    interrupt 11;
void TM2_Routine(void)     interrupt 12;
void INT4_Routine(void)    interrupt 16;
void UART3_Routine(void)   interrupt 17;
void UART4_Routine(void)   interrupt 18;
void TM3_Routine(void)     interrupt 19;
void TM4_Routine(void)     interrupt 20;
void CMP_Routine(void)     interrupt 21;
void I2C_Routine(void)     interrupt 24;
void USB_Routine(void)     interrupt 25;
void PWMA_Routine(void)    interrupt 26;
void PWMB_Routine(void)    interrupt 27;
```

中断号超过31的C语言中断服务程序不能直接用interrupt声明，请参考附录的处理方法，汇编语言不受影响

## 12.4 中断相关寄存器

| 符号        | 描述            | 地址  | 位地址与符号          |           |          |           |              |        |        |        | 复位值       |
|-----------|---------------|-----|-----------------|-----------|----------|-----------|--------------|--------|--------|--------|-----------|
|           |               |     | B7              | B6        | B5       | B4        | B3           | B2     | B1     | B0     |           |
| IE        | 中断允许寄存器       | A8H | EA              | ELVD      | EADC     | ES        | ET1          | EX1    | ET0    | EX0    | 0000,0000 |
| IE2       | 中断允许寄存器 2     | AFH | EUSB<br>ETKSU   | ET4       | ET3      | ES4       | ES3          | ET2    | ESPI   | ES2    | 0000,0000 |
| INTCLKO   | 中断与时钟输出控制寄存器  | 8FH | -               | EX4       | EX3      | EX2       | -            | T2CLKO | T1CLKO | T0CLKO | x000,x000 |
| IP        | 中断优先级控制寄存器    | B8H | -               | PLVD      | PADC     | PS        | PT1          | PX1    | PT0    | PX0    | x000,0000 |
| IPH       | 高中断优先级控制寄存器   | B7H | -               | PLVDH     | PADCH    | PSH       | PT1H         | PX1H   | PT0H   | PX0H   | x000,0000 |
| IP2       | 中断优先级控制寄存器 2  | B5H | PUSB<br>PTKSU   | PI2C      | PCMP     | PX4       | PPWMB        | PPWMA  | PSPI   | PS2    | 0000,0000 |
| IP2H      | 高中断优先级控制寄存器 2 | B6H | PUSBH<br>PTKSUH | PI2CH     | PCMPH    | PX4H      | PPWMBH       | PPWMAH | PSPIH  | PS2H   | 0000,0000 |
| IP3       | 中断优先级控制寄存器 3  | DFH | -               | -         | -        | -         | -            | PRTC   | PS4    | PS3    | xxxx,x000 |
| IP3H      | 高中断优先级控制寄存器 3 | EEH | -               | -         | -        | -         | -            | PRTCH  | PS4H   | PS3H   | xxxx,x000 |
| TCON      | 定时器控制寄存器      | 88H | TF1             | TR1       | TF0      | TR0       | IE1          | IT1    | IE0    | IT0    | 0000,0000 |
| AUXINTIF  | 扩展外部中断标志寄存器   | EFH | -               | INT4IF    | INT3IF   | INT2IF    | -            | T4IF   | T3IF   | T2IF   | x000,x000 |
| SCON      | 串口 1 控制寄存器    | 98H | SM0/FE          | SM1       | SM2      | REN       | TB8          | RB8    | TI     | RI     | 0000,0000 |
| S2CON     | 串口 2 控制寄存器    | 9AH | S2SM0           | -         | S2SM2    | S2REN     | S2TB8        | S2RB8  | S2TI   | S2RI   | 0100,0000 |
| S3CON     | 串口 3 控制寄存器    | ACH | S3SM0           | S3ST3     | S3SM2    | S3REN     | S3TB8        | S3RB8  | S3TI   | S3RI   | 0000,0000 |
| S4CON     | 串口 4 控制寄存器    | 84H | S4SM0           | S4ST4     | S4SM2    | S4REN     | S4TB8        | S4RB8  | S4TI   | S4RI   | 0000,0000 |
| PCON      | 电源控制寄存器       | 87H | SMOD            | SMOD0     | LVDF     | POF       | GF1          | GF0    | PD     | IDL    | 0011,0000 |
| ADC_CONTR | ADC 控制寄存器     | BCH | ADC_POWER       | ADC_START | ADC_FLAG | ADC_EPWMT | ADC_CHS[3:0] |        |        |        | 000x,0000 |
| SPSTAT    | SPI 状态寄存器     | CDH | SPIF            | WCOL      | -        | -         | -            | -      | -      | -      | 00xx,xxxx |
| CMPCR1    | 比较器控制寄存器 1    | E6H | CMPEN           | CMPIF     | PIE      | NIE       | PIS          | NIS    | CMPOE  | CMPRES | 0000,0000 |

| 符号       | 描述                       | 地址    | 位地址与符号  |         |              |         |               |         |         |         | 复位值       |
|----------|--------------------------|-------|---------|---------|--------------|---------|---------------|---------|---------|---------|-----------|
|          |                          |       | B7      | B6      | B5           | B4      | B3            | B2      | B1      | B0      |           |
| LCMIFCFG | LCM 接口配置寄存器              | FE50H | LCMIFIE | -       | LCMIFIP[1:0] |         | LCMIFDPS[1:0] |         | D16_D8  | M68_I80 | 0x00,0000 |
| LCMIFSTA | LCM 接口状态寄存器              | FE53H | -       | -       | -            | -       | -             | -       | -       | LCMIFIF | xxxx,xxx0 |
| I2CMSCR  | I <sup>2</sup> C 主机控制寄存器 | FE81H | EMSI    | -       | -            | -       | MSCMD[3:0]    |         |         |         | 0xxx,0000 |
| I2CMSST  | I <sup>2</sup> C 主机状态寄存器 | FE82H | MSBUSY  | MSIF    | -            | -       | -             | -       | MSACKI  | MSACKO  | 00xx,xx10 |
| I2CSLCR  | I <sup>2</sup> C 从机控制寄存器 | FE83H | -       | ESTAI   | ERXI         | ETXI    | ESTOI         | -       | -       | SLRST   | x000,0xx0 |
| I2CSLST  | I <sup>2</sup> C 从机状态寄存器 | FE84H | SLBUSY  | STAIF   | RXIF         | TXIF    | STOIF         | TXING   | SLACKI  | SLACKO  | 0000,0000 |
| PWMA_IER | PWMA 中断使能寄存器             | FEC4H | BIE     | TIE     | COMIE        | CC4IE   | CC3IE         | CC2IE   | CC1IE   | UIE     | 0000,0000 |
| PWMA_SR1 | PWMA 状态寄存器 1             | FEC5H | BIF     | TIF     | COMIF        | CC4IF   | CC3IF         | CC2IF   | CC1IF   | UIF     | 0000,0000 |
| PWMA_SR2 | PWMA 状态寄存器 2             | FEC6H | -       | -       | -            | CC4OF   | CC3OF         | CC2OF   | CC1OF   | -       | xxx0,000x |
| PWMB_IER | PWMB 中断使能寄存器             | FEE4H | BIE     | TIE     | COMIE        | CC8IE   | CC7IE         | CC6IE   | CC5IE   | UIE     | 0000,0000 |
| PWMB_SR1 | PWMB 状态寄存器 1             | FEE5H | BIF     | TIF     | COMIF        | CC8IF   | CC7IF         | CC6IF   | CC5IF   | UIF     | 0000,0000 |
| PWMB_SR2 | PWMB 状态寄存器 2             | FEE6H | -       | -       | -            | CC8OF   | CC7OF         | CC6OF   | CC5OF   | -       | xxx0,000x |
| P0INTE   | P0 口中断使能寄存器              | FD00H | P07INTE | P06INTE | P05INTE      | P04INTE | P03INTE       | P02INTE | P01INTE | P00INTE | 0000,0000 |
| P1INTE   | P1 口中断使能寄存器              | FD01H | P17INTE | P16INTE | P15INTE      | P14INTE | P13INTE       | P12INTE | P11INTE | P10INTE | 0000,0000 |

|              |                |       |         |         |         |         |             |         |              |         |           |
|--------------|----------------|-------|---------|---------|---------|---------|-------------|---------|--------------|---------|-----------|
| P2INTE       | P2 口中断使能寄存器    | FD02H | P27INTE | P26INTE | P25INTE | P24INTE | P23INTE     | P22INTE | P21INTE      | P20INTE | 0000,0000 |
| P3INTE       | P3 口中断使能寄存器    | FD03H | P37INTE | P36INTE | P35INTE | P34INTE | P33INTE     | P32INTE | P31INTE      | P30INTE | 0000,0000 |
| P4INTE       | P4 口中断使能寄存器    | FD04H | P47INTE | P46INTE | P45INTE | P44INTE | P43INTE     | P42INTE | P41INTE      | P40INTE | 0000,0000 |
| P5INTE       | P5 口中断使能寄存器    | FD05H | -       | -       | P55INTE | P54INTE | P53INTE     | P52INTE | P51INTE      | P50INTE | xx00,0000 |
| P6INTE       | P6 口中断使能寄存器    | FD06H | P67INTE | P66INTE | P65INTE | P64INTE | P63INTE     | P62INTE | P61INTE      | P60INTE | 0000,0000 |
| P7INTE       | P7 口中断使能寄存器    | FD07H | P77INTE | P76INTE | P75INTE | P74INTE | P73INTE     | P72INTE | P71INTE      | P70INTE | 0000,0000 |
| P0INTF       | P0 口中断标志寄存器    | FD10H | P07INTF | P06INTF | P05INTF | P04INTF | P03INTF     | P02INTF | P01INTF      | P00INTF | 0000,0000 |
| P1INTF       | P1 口中断标志寄存器    | FD11H | P17INTF | P16INTF | P15INTF | P14INTF | P13INTF     | P12INTF | P11INTF      | P10INTF | 0000,0000 |
| P2INTF       | P2 口中断标志寄存器    | FD12H | P27INTF | P26INTF | P25INTF | P24INTF | P23INTF     | P22INTF | P21INTF      | P20INTF | 0000,0000 |
| P3INTF       | P3 口中断标志寄存器    | FD13H | P37INTF | P36INTF | P35INTF | P34INTF | P33INTF     | P32INTF | P31INTF      | P30INTF | 0000,0000 |
| P4INTF       | P4 口中断标志寄存器    | FD14H | P47INTF | P46INTF | P45INTF | P44INTF | P43INTF     | P42INTF | P41INTF      | P40INTF | 0000,0000 |
| P5INTF       | P5 口中断标志寄存器    | FD15H | -       | -       | P55INTF | P54INTF | P53INTF     | P52INTF | P51INTF      | P50INTF | xx00,0000 |
| P6INTF       | P6 口中断标志寄存器    | FD16H | P67INTF | P66INTF | P65INTF | P64INTF | P63INTF     | P62INTF | P61INTF      | P60INTF | 0000,0000 |
| P7INTF       | P7 口中断标志寄存器    | FD17H | P77INTF | P76INTF | P75INTF | P74INTF | P73INTF     | P72INTF | P71INTF      | P70INTF | 0000,0000 |
| PINIPL       | I/O 口中断优先级低寄存器 | FD60H | P7IP    | P6IP    | P5IP    | P4IP    | P3IP        | P2IP    | P1IP         | P0IP    | 0000,0000 |
| PINIPH       | I/O 口中断优先级高寄存器 | FD61H | P7IPH   | P6IPH   | P5IPH   | P4IPH   | P3IPH       | P2IPH   | P1IPH        | P0IPH   | 0000,0000 |
| DMA_M2M_CFG  | M2M_DMA 配置寄存器  | FA00H | M2MIE   | -       | TXACO   | RXACO   | M2MIP[1:0]  |         | M2MPTY[1:0]  |         | 0x00,0000 |
| DMA_ADC_CFG  | ADC_DMA 配置寄存器  | FA10H | ADCIE   | -       | -       | -       | ADCMIP[1:0] |         | ADCPTY[1:0]  |         | 0xxx,0000 |
| DMA_SPL_CFG  | SPI_DMA 配置寄存器  | FA20H | SPIIE   | ACT_TX  | ACT_RX  | -       | SPIIP[1:0]  |         | SPIPTY[1:0]  |         | 000x,0000 |
| DMA_UR1T_CFG | UR1T_DMA 配置寄存器 | FA30H | UR1TIE  | -       | -       | -       | UR1TIP[1:0] |         | UR1TPTY[1:0] |         | 0xxx,0000 |
| DMA_UR1R_CFG | UR1R_DMA 配置寄存器 | FA38H | UR1RIE  | -       | -       | -       | UR1RIP[1:0] |         | UR1RPTY[1:0] |         | 0xxx,0000 |
| DMA_UR2T_CFG | UR2T_DMA 配置寄存器 | FA40H | UR2TIE  | -       | -       | -       | UR2TIP[1:0] |         | UR2TPTY[1:0] |         | 0xxx,0000 |
| DMA_UR2R_CFG | UR2R_DMA 配置寄存器 | FA48H | UR2RIE  | -       | -       | -       | UR2RIP[1:0] |         | UR2RPTY[1:0] |         | 0xxx,0000 |
| DMA_UR3T_CFG | UR3T_DMA 配置寄存器 | FA50H | UR3TIE  | -       | -       | -       | UR3TIP[1:0] |         | UR3TPTY[1:0] |         | 0xxx,0000 |
| DMA_UR3R_CFG | UR3R_DMA 配置寄存器 | FA58H | UR3RIE  | -       | -       | -       | UR3RIP[1:0] |         | UR3RPTY[1:0] |         | 0xxx,0000 |
| DMA_UR4T_CFG | UR4T_DMA 配置寄存器 | FA60H | UR4TIE  | -       | -       | -       | UR4TIP[1:0] |         | UR4TPTY[1:0] |         | 0xxx,0000 |
| DMA_UR4R_CFG | UR4R_DMA 配置寄存器 | FA68H | UR4RIE  | -       | -       | -       | UR4RIP[1:0] |         | UR4RPTY[1:0] |         | 0xxx,0000 |
| DMA_LCM_CFG  | LCM_DMA 配置寄存器  | FA70H | LCMIE   | -       | -       | -       | LCMIP[1:0]  |         | LCMPTY[1:0]  |         | 0xxx,0000 |
| DMA_M2M_STA  | M2M_DMA 状态寄存器  | FA02H | -       | -       | -       | -       | -           | -       | -            | M2MIF   | xxxx,xxx0 |
| DMA_ADC_STA  | ADC_DMA 状态寄存器  | FA12H | -       | -       | -       | -       | -           | -       | -            | ADCIF   | xxxx,xxx0 |
| DMA_SPL_STA  | SPI_DMA 状态寄存器  | FA22H | -       | -       | -       | -       | -           | TXOVW   | RXLOSS       | SPIIF   | xxxx,x000 |
| DMA_UR1T_STA | UR1T_DMA 状态寄存器 | FA32H | -       | -       | -       | -       | -           | TXOVW   | -            | UR1TIF  | xxxx,x0x0 |
| DMA_UR1R_STA | UR1R_DMA 状态寄存器 | FA3AH | -       | -       | -       | -       | -           | -       | RXLOSS       | UR1RIF  | xxxx,xx00 |
| DMA_UR2T_STA | UR2T_DMA 状态寄存器 | FA42H | -       | -       | -       | -       | -           | TXOVW   | -            | UR2TIF  | xxxx,x0x0 |
| DMA_UR2R_STA | UR2R_DMA 状态寄存器 | FA4AH | -       | -       | -       | -       | -           | -       | RXLOSS       | UR2RIF  | xxxx,xx00 |
| DMA_UR3T_STA | UR3T_DMA 状态寄存器 | FA52H | -       | -       | -       | -       | -           | TXOVW   | -            | UR3TIF  | xxxx,x0x0 |
| DMA_UR3R_STA | UR3R_DMA 状态寄存器 | FA5AH | -       | -       | -       | -       | -           | -       | RXLOSS       | UR3RIF  | xxxx,xx00 |
| DMA_UR4T_STA | UR4T_DMA 状态寄存器 | FA62H | -       | -       | -       | -       | -           | TXOVW   | -            | UR4TIF  | xxxx,x0x0 |
| DMA_UR4R_STA | UR4R_DMA 状态寄存器 | FA6AH | -       | -       | -       | -       | -           | -       | RXLOSS       | UR4RIF  | xxxx,xx00 |
| DMA_LCM_STA  | LCM_DMA 状态寄存器  | FA72H | -       | -       | -       | -       | -           | -       | TXOVW        | LCMIF   | xxxx,xx00 |
|              |                |       |         |         |         |         |             |         |              |         |           |

## 12.4.1 中断使能寄存器（中断允许位）

### IE（中断使能寄存器）

| 符号 | 地址  | B7 | B6   | B5   | B4 | B3  | B2  | B1  | B0  |
|----|-----|----|------|------|----|-----|-----|-----|-----|
| IE | A8H | EA | ELVD | EADC | ES | ET1 | EX1 | ET0 | EX0 |

EA：总中断允许控制位。EA 的作用是使中断允许形成多级控制。即各中断源首先受 EA 控制;其次还受各中断源自己的中断允许控制位控制。

0：CPU 屏蔽所有的中断申请

1：CPU 开放中断

ELVD：低压检测中断允许位。

0：禁止低压检测中断

1：允许低压检测中断

EADC：A/D 转换中断允许位。

0：禁止 A/D 转换中断

1：允许 A/D 转换中断

ES：串行口 1 中断允许位。

0：禁止串行口 1 中断

1：允许串行口 1 中断

ET1：定时/计数器 T1 的溢出中断允许位。

0：禁止 T1 中断

1：允许 T1 中断

EX1：外部中断 1 中断允许位。

0：禁止 INT1 中断

1：允许 INT1 中断

ET0：定时/计数器 T0 的溢出中断允许位。

0：禁止 T0 中断

1：允许 T0 中断

EX0：外部中断 0 中断允许位。

0：禁止 INT0 中断

1：允许 INT0 中断

### IE2（中断使能寄存器 2）

| 符号  | 地址  | B7            | B6  | B5  | B4  | B3  | B2  | B1   | B0  |
|-----|-----|---------------|-----|-----|-----|-----|-----|------|-----|
| IE2 | AFH | EUSB<br>ETKSU | ET4 | ET3 | ES4 | ES3 | ET2 | ESPI | ES2 |

EUSB：USB 中断允许位。

0：禁止 USB 中断

1：允许 USB 中断

ETKSU：触摸按键中断允许位。

0：禁止触摸按键中断

1：允许触摸按键中断

ET4：定时/计数器 T4 的溢出中断允许位。

0：禁止 T4 中断

1：允许 T4 中断

ET3: 定时/计数器 T3 的溢出中断允许位。

0: 禁止 T3 中断

1: 允许 T3 中断

ES4: 串行口 4 中断允许位。

0: 禁止串行口 4 中断

1: 允许串行口 4 中断

ES3: 串行口 3 中断允许位。

0: 禁止串行口 3 中断

1: 允许串行口 3 中断

ET2: 定时/计数器 T2 的溢出中断允许位。

0: 禁止 T2 中断

1: 允许 T2 中断

ESPI: SPI 中断允许位。

0: 禁止 SPI 中断

1: 允许 SPI 中断

ES2: 串行口 2 中断允许位。

0: 禁止串行口 2 中断

1: 允许串行口 2 中断

#### INTCLKO (外部中断与时钟输出控制寄存器)

| 符号      | 地址  | B7 | B6  | B5  | B4  | B3 | B2     | B1     | B0     |
|---------|-----|----|-----|-----|-----|----|--------|--------|--------|
| INTCLKO | 8FH | -  | EX4 | EX3 | EX2 | -  | T2CLKO | T1CLKO | T0CLKO |

EX4: 外部中断 4 中断允许位。

0: 禁止 INT4 中断

1: 允许 INT4 中断

EX3: 外部中断 3 中断允许位。

0: 禁止 INT3 中断

1: 允许 INT3 中断

EX2: 外部中断 2 中断允许位。

0: 禁止 INT2 中断

1: 允许 INT2 中断

#### CMPCR1 (比较器控制寄存器 1)

| 符号     | 地址  | B7     | B6    | B5  | B4  | B3  | B2  | B1    | B0      |
|--------|-----|--------|-------|-----|-----|-----|-----|-------|---------|
| CMPCR1 | E6H | CM PEN | CMPIF | PIE | NIE | PIS | NIS | CMPOE | COMPRES |

PIE: 比较器上升沿中断允许位。

0: 禁止比较器上升沿中断

1: 允许比较器上升沿中断

NIE: 比较器下降沿中断允许位。

0: 禁止比较器下降沿中断

1: 允许比较器下降沿中断

#### I2C 控制寄存器



| 符号      | 地址    | B7   | B6    | B5   | B4   | B3         | B2 | B1 | B0    |
|---------|-------|------|-------|------|------|------------|----|----|-------|
| I2CMSCR | FE81H | EMSI | -     | -    | -    | MSCMD[3:0] |    |    |       |
| I2CSLCR | FE83H | -    | ESTAI | ERXI | ETXI | ESTOI      | -  | -  | SLRST |

EMSI: I<sup>2</sup>C主机模式中中断允许位。

- 0: 禁止 I<sup>2</sup>C 主机模式中断
- 1: 允许 I<sup>2</sup>C 主机模式中断

ESTAI: I<sup>2</sup>C从机接收START事件中中断允许位。

- 0: 禁止 I<sup>2</sup>C 从机接收 START 事件中断
- 1: 允许 I<sup>2</sup>C 从机接收 START 事件中断

ERXI: I<sup>2</sup>C从机接收数据完成事件中中断允许位。

- 0: 禁止 I<sup>2</sup>C 从机接收数据完成事件中断
- 1: 允许 I<sup>2</sup>C 从机接收数据完成事件中断

ETXI: I<sup>2</sup>C从机发送数据完成事件中中断允许位。

- 0: 禁止 I<sup>2</sup>C 从机发送数据完成事件中断
- 1: 允许 I<sup>2</sup>C 从机发送数据完成事件中断

ESTOI: I<sup>2</sup>C从机接收STOP事件中中断允许位。

- 0: 禁止 I<sup>2</sup>C 从机接收 STOP 事件中断
- 1: 允许 I<sup>2</sup>C 从机接收 STOP 事件中断

#### PWMA 中断使能寄存器

| 符号       | 地址    | B7  | B6  | B5    | B4    | B3    | B2    | B1    | B0  |
|----------|-------|-----|-----|-------|-------|-------|-------|-------|-----|
| PWMA_IER | FEC4H | BIE | TIE | COMIE | CC4IE | CC3IE | CC2IE | CC1IE | UIE |

BIE: PWMA刹车中断允许位。

- 0: 禁止 PWMA 刹车中断
- 1: 允许 PWMA 刹车中断

TIE: PWMA触发中断允许位。

- 0: 禁止 PWMA 触发中断
- 1: 允许 PWMA 触发中断

COMIE: PWMA比较中断允许位。

- 0: 禁止 PWMA 比较中断
- 1: 允许 PWMA 比较中断

CC4IE: PWMA捕获比较通道4中断允许位。

- 0: 禁止 PWMA 捕获比较通道 4 中断
- 1: 允许 PWMA 捕获比较通道 4 中断

CC3IE: PWMA捕获比较通道3中断允许位。

- 0: 禁止 PWMA 捕获比较通道 3 中断
- 1: 允许 PWMA 捕获比较通道 3 中断

CC2IE: PWMA捕获比较通道2中断允许位。

- 0: 禁止 PWMA 捕获比较通道 2 中断
- 1: 允许 PWMA 捕获比较通道 2 中断

CC1IE: PWMA捕获比较通道1中断允许位。

- 0: 禁止 PWMA 捕获比较通道 1 中断

1: 允许 PWMA 捕获比较通道 1 中断

UIE: PWMA更新中断允许位。

0: 禁止 PWMA 更新中断

1: 允许 PWMA 更新中断

### PWMB 中断使能寄存器

| 符号       | 地址    | B7  | B6  | B5    | B4    | B3    | B2    | B1    | B0  |
|----------|-------|-----|-----|-------|-------|-------|-------|-------|-----|
| PWMB_IER | FEE4H | BIE | TIE | COMIE | CC8IE | CC7IE | CC6IE | CC5IE | UIE |

BIE: PWMB刹车中断允许位。

0: 禁止 PWMB 刹车中断

1: 允许 PWMB 刹车中断

TIE: PWMB触发中断允许位。

0: 禁止 PWMB 触发中断

1: 允许 PWMB 触发中断

COMIE: PWMB比较中断允许位。

0: 禁止 PWMB 比较中断

1: 允许 PWMB 比较中断

CC8IE: PWMB捕获比较通道8中断允许位。

0: 禁止 PWMB 捕获比较通道 8 中断

1: 允许 PWMB 捕获比较通道 8 中断

CC7IE: PWMB捕获比较通道7中断允许位。

0: 禁止 PWMB 捕获比较通道 7 中断

1: 允许 PWMB 捕获比较通道 7 中断

CC6IE: PWMB捕获比较通道6中断允许位。

0: 禁止 PWMB 捕获比较通道 6 中断

1: 允许 PWMB 捕获比较通道 6 中断

CC5IE: PWMB捕获比较通道5中断允许位。

0: 禁止 PWMB 捕获比较通道 5 中断

1: 允许 PWMB 捕获比较通道 5 中断

UIE: PWMB更新中断允许位。

0: 禁止 PWMB 更新中断

1: 允许 PWMB 更新中断

### 端口中断使能寄存器

| 符号     | 地址    | B7      | B6      | B5      | B4      | B3      | B2      | B1      | B0      |
|--------|-------|---------|---------|---------|---------|---------|---------|---------|---------|
| P0INTE | FD00H | P07INTE | P06INTE | P05INTE | P04INTE | P03INTE | P02INTE | P01INTE | P00INTE |
| P1INTE | FD01H | P17INTE | P16INTE | P15INTE | P14INTE | P13INTE | P12INTE | P11INTE | P10INTE |
| P2INTE | FD02H | P27INTE | P26INTE | P25INTE | P24INTE | P23INTE | P22INTE | P21INTE | P20INTE |
| P3INTE | FD03H | P37INTE | P36INTE | P35INTE | P34INTE | P33INTE | P32INTE | P31INTE | P30INTE |
| P4INTE | FD04H | P47INTE | P46INTE | P45INTE | P44INTE | P43INTE | P42INTE | P41INTE | P40INTE |
| P5INTE | FD05H | -       | -       | P55INTE | P54INTE | P53INTE | P52INTE | P51INTE | P50INTE |
| P6INTE | FD06H | P67INTE | P66INTE | P65INTE | P64INTE | P63INTE | P62INTE | P61INTE | P60INTE |

|        |       |         |         |         |         |         |         |         |         |
|--------|-------|---------|---------|---------|---------|---------|---------|---------|---------|
| P7INTE | FD07H | P77INTE | P76INTE | P75INTE | P74INTE | P73INTE | P72INTE | P71INTE | P70INTE |
|--------|-------|---------|---------|---------|---------|---------|---------|---------|---------|

PnINTE.x: 端口中断使能控制位 (n=0~7, x=0~7)

0: 关闭 Pn.x 口中断功能

1: 使能 Pn.x 口中断功能

### LCM 接口配置寄存器

| 符号       | 地址    | B7      | B6 | B5           | B4 | B3            | B2 | B1     | B0      |
|----------|-------|---------|----|--------------|----|---------------|----|--------|---------|
| LCMIFCFG | FE50H | LCMIFIE | -  | LCMIFIP[1:0] |    | LCMIFDPS[1:0] |    | D16_D8 | M68_I80 |

LCMIFIE: LCM接口中断允许位。

0: 禁止 LCM 接口中断

1: 允许 LCM 接口中断

### DMA 中断使能寄存器

| 符号           | 地址    | B7     | B6     | B5     | B4    | B3          | B2 | B1          | B0 |
|--------------|-------|--------|--------|--------|-------|-------------|----|-------------|----|
| DMA_M2M_CFG  | FA00H | M2MIE  | -      | TXACO  | RXACO | M2MIP[1:0]  |    | M2MPTY[1:0] |    |
| DMA_ADC_CFG  | FA10H | ADCIE  | -      | -      | -     | ADCIP[1:0]  |    | ADCP[1:0]   |    |
| DMA_SPI_CFG  | FA20H | SPIIE  | ACT_TX | ACT_RX | -     | SPIIP[1:0]  |    | SPIPT[1:0]  |    |
| DMA_UR1T_CFG | FA30H | UR1TIE | -      | -      | -     | UR1TIP[1:0] |    | UR1TPT[1:0] |    |
| DMA_UR1R_CFG | FA38H | UR1RIE | -      | -      | -     | UR1RIP[1:0] |    | UR1RPT[1:0] |    |
| DMA_UR2T_CFG | FA40H | UR2TIE | -      | -      | -     | UR2TIP[1:0] |    | UR2TPT[1:0] |    |
| DMA_UR2R_CFG | FA48H | UR2RIE | -      | -      | -     | UR2RIP[1:0] |    | UR2RPT[1:0] |    |
| DMA_UR3T_CFG | FA50H | UR3TIE | -      | -      | -     | UR3TIP[1:0] |    | UR3TPT[1:0] |    |
| DMA_UR3R_CFG | FA58H | UR3RIE | -      | -      | -     | UR3RIP[1:0] |    | UR3RPT[1:0] |    |
| DMA_UR3R_CFG | FA60H | UR4TIE | -      | -      | -     | UR4TIP[1:0] |    | UR4TPT[1:0] |    |
| DMA_UR4R_CFG | FA68H | UR4RIE | -      | -      | -     | UR4RIP[1:0] |    | UR4RPT[1:0] |    |
| DMA_LCM_CFG  | FA70H | LCMIE  | -      | -      | -     | LCMIP[1:0]  |    | LCMPT[1:0]  |    |

M2MIE: DMA\_M2M (存储器到存储器DMA) 中断允许位。

0: 禁止 DMA\_M2M 中断

1: 允许 DMA\_M2M 中断

ADCIE: DMA\_ADC (ADC DMA) 中断允许位。

0: 禁止 DMA\_ADC 中断

1: 允许 DMA\_ADC 中断

SPIIE: DMA\_SPI (SPI DMA) 中断允许位。

0: 禁止 DMA\_SPI 中断

1: 允许 DMA\_SPI 中断

UR1TIE: DMA\_UR1T (串口1发送DMA) 中断允许位。

0: 禁止 DMA\_UR1T 中断

1: 允许 DMA\_UR1T 中断

UR1RIE: DMA\_UR1R (串口1接收DMA) 中断允许位。

0: 禁止 DMA\_UR1R 中断

1: 允许 DMA\_UR1R 中断

UR2TIE: DMA\_UR2T (串口2发送DMA) 中断允许位。

0: 禁止 DMA\_UR2T 中断

1: 允许 DMA\_UR2T 中断

UR2RIE: DMA\_UR2R (串口2接收DMA) 中断允许位。

0: 禁止 DMA\_UR2R 中断

1: 允许 DMA\_UR2R 中断

UR3TIE: DMA\_UR3T (串口3发送DMA) 中断允许位。

0: 禁止 DMA\_UR3T 中断

1: 允许 DMA\_UR3T 中断

UR3RIE: DMA\_UR3R (串口3接收DMA) 中断允许位。

0: 禁止 DMA\_UR3R 中断

1: 允许 DMA\_UR3R 中断

UR4TIE: DMA\_UR4T (串口4发送DMA) 中断允许位。

0: 禁止 DMA\_UR4T 中断

1: 允许 DMA\_UR4T 中断

UR4RIE: DMA\_UR4R (串口4接收DMA) 中断允许位。

0: 禁止 DMA\_UR4R 中断

1: 允许 DMA\_UR4R 中断

LCMIE: DMA\_LCM (LCM接口DMA) 中断允许位。

0: 禁止 DMA\_LCM 中断

1: 允许 DMA\_LCM 中断

## 12.4.2 中断请求寄存器 (中断标志位)

### 定时器控制寄存器

| 符号   | 地址  | B7  | B6  | B5  | B4  | B3  | B2  | B1  | B0  |
|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| TCON | 88H | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

TF1: 定时器1溢出中断标志。中断服务程序中, 硬件自动清零。

TF0: 定时器0溢出中断标志。中断服务程序中, 硬件自动清零。

IE1: 外部中断1中断请求标志。中断服务程序中, 硬件自动清零。

IE0: 外部中断0中断请求标志。中断服务程序中, 硬件自动清零。

### 中断标志辅助寄存器

| 符号       | 地址  | B7 | B6     | B5     | B4     | B3 | B2   | B1   | B0   |
|----------|-----|----|--------|--------|--------|----|------|------|------|
| AUXINTIF | EFH | -  | INT4IF | INT3IF | INT2IF | -  | T4IF | T3IF | T2IF |

INT4IF: 外部中断4中断请求标志。中断服务程序中硬件自动清零。

INT3IF: 外部中断3中断请求标志。中断服务程序中硬件自动清零。

INT2IF: 外部中断2中断请求标志。中断服务程序中硬件自动清零。

T4IF: 定时器4溢出中断标志。中断服务程序中硬件自动清零 (注意: 此位为只写寄存器, 不可读)。

T3IF: 定时器3溢出中断标志。中断服务程序中硬件自动清零 (注意: 此位为只写寄存器, 不可读)。

T2IF: 定时器2溢出中断标志。中断服务程序中硬件自动清零 (注意: 此位为只写寄存器, 不可读)。

**注意:**

早期采用 0.35um 工艺的 1T 8051, STC15 系列增加了 16 位重装载定时器, 全球 8051 首次大手笔, 由于制造成本高, STC 可 16 位重装载的定时器 2/3/4 没有设计用户可以访问的中断请求标志位寄存器, 只有内部隐藏的标志位, 提供给用户软件清内部隐藏标志位的方法是: 用户软件禁止定时器 2/3/4 中断时, 硬件自动清定时器 2/3/4 内部隐藏中断请求标志位。

为了产品的一致性:

采用 0.18um 工艺的 STC8A/ STC8F 及后续 STC8G/STC8H/ STC8C/ STC12H 系列虽然增加了定时器 2/3/4 的用户可以访问的中断请求标志位寄存器, 但禁止定时器 2/3/4 中断时, 硬件自动清定时器 2/3/4 内部隐藏中断请求标志位的功能依然保留了。所以在定时器 2/3/4 没有停止计数时不要随意禁止定时器 2/3/4 中断, 否则实际起作用的隐藏的中断请求标志位会被清除掉, 会有可能, 计数器又溢出后, 又产生了隐藏的中断请求标志位被置 1 后, 去请求中断并在等待时, 却被用户误清除的事。

这与传统的 INTEL8048, 8051 不一样, 但 INTEL 已停产, 所以 STC 的新设计并没有考虑兼容传统 INTEL 的规格。

这是中国 STC 对 8051 的再发展。

**串口控制寄存器**

| 符号    | 地址  | B7     | B6    | B5    | B4    | B3    | B2    | B1   | B0   |
|-------|-----|--------|-------|-------|-------|-------|-------|------|------|
| SCON  | 98H | SM0/FE | SM1   | SM2   | REN   | TB8   | RB8   | TI   | RI   |
| S2CON | 9AH | S2SM0  | -     | S2SM2 | S2REN | S2TB8 | S2RB8 | S2TI | S2RI |
| S3CON | ACH | S3SM0  | S3ST3 | S3SM2 | S3REN | S3TB8 | S3RB8 | S3TI | S3RI |
| S4CON | 84H | S4SM0  | S4ST4 | S4SM2 | S4REN | S4TB8 | S4RB8 | S4TI | S4RI |

TI: 串口1发送完成中断请求标志。需要软件清零。

RI: 串口1接收完成中断请求标志。需要软件清零。

S2TI: 串口2发送完成中断请求标志。需要软件清零。

S2RI: 串口2接收完成中断请求标志。需要软件清零。

S3TI: 串口3发送完成中断请求标志。需要软件清零。

S3RI: 串口3接收完成中断请求标志。需要软件清零。

S4TI: 串口4发送完成中断请求标志。需要软件清零。

S4RI: 串口4接收完成中断请求标志。需要软件清零。

**电源管理寄存器**

| 符号   | 地址  | B7   | B6    | B5   | B4  | B3  | B2  | B1 | B0  |
|------|-----|------|-------|------|-----|-----|-----|----|-----|
| PCON | 87H | SMOD | SMOD0 | LVDF | POF | GF1 | GF0 | PD | IDL |

LVDF: 低压检测中断请求标志。需要软件清零。

**ADC 控制寄存器**

| 符号        | 地址  | B7        | B6        | B5       | B4        | B3           | B2 | B1 | B0 |
|-----------|-----|-----------|-----------|----------|-----------|--------------|----|----|----|
| ADC_CONTR | BCH | ADC_POWER | ADC_START | ADC_FLAG | ADC_EPWMT | ADC_CHS[3:0] |    |    |    |

ADC\_FLAG: ADC转换完成中断请求标志。需要软件清零。

**SPI 状态寄存器**

| 符号     | 地址  | B7   | B6   | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-----|------|------|----|----|----|----|----|----|
| SPSTAT | CDH | SPIF | WCOL | -  | -  | -  | -  | -  | -  |

SPIF: SPI数据传输完成中断请求标志。需要软件清零。

**比较器控制寄存器 1**

| 符号     | 地址  | B7    | B6    | B5  | B4  | B3  | B2  | B1    | B0     |
|--------|-----|-------|-------|-----|-----|-----|-----|-------|--------|
| CMPCR1 | E6H | CMPEN | CMPIF | PIE | NIE | PIS | NIS | CMPOE | CMPRES |

CMPIF: 比较器中断请求标志。需要软件清零。

**I2C 状态寄存器**

| 符号      | 地址    | B7     | B6    | B5   | B4   | B3    | B2    | B1     | B0     |
|---------|-------|--------|-------|------|------|-------|-------|--------|--------|
| I2CMSST | FE82H | MSBUSY | MSIF  | -    | -    | -     | -     | MSACKI | MSACKO |
| I2CSLST | FE84H | SLBUSY | STAIF | RXIF | TXIF | STOIF | TXING | SLACKI | SLACKO |

MSIF: I<sup>2</sup>C主机模式中断请求标志。需要软件清零。

ESTAI: I<sup>2</sup>C从机接收START事件中断请求标志。需要软件清零。

ERXI: I<sup>2</sup>C从机接收数据完成事件中断请求标志。需要软件清零。

ETXI: I<sup>2</sup>C从机发送数据完成事件中断请求标志。需要软件清零。

ESTOI: I<sup>2</sup>C从机接收STOP事件中断请求标志。需要软件清零。

**PWMA 状态寄存器**

| 符号       | 地址    | B7  | B6  | B5    | B4    | B3    | B2    | B1    | B0  |
|----------|-------|-----|-----|-------|-------|-------|-------|-------|-----|
| PWMA_SR1 | FEC5H | BIF | TIF | COMIF | CC4IF | CC3IF | CC2IF | CC1IF | UIF |
| PWMA_SR2 | FEC6H | -   | -   | -     | CC4OF | CC3OF | CC2OF | CC1OF | -   |

BIF: PWMA刹车中断请求标志。需要软件清零。

TIF: PWMA触发中断请求标志。需要软件清零。

COMIF: PWMA比较中断请求标志。需要软件清零。

CC4IF: PWMA通道4发生捕获比较中断请求标志。需要软件清零。

CC3IF: PWMA通道3发生捕获比较中断请求标志。需要软件清零。

CC2IF: PWMA通道2发生捕获比较中断请求标志。需要软件清零。

CC1IF: PWMA通道1发生捕获比较中断请求标志。需要软件清零。

UIF: PWMA更新中断请求标志。需要软件清零。

CC4OF: PWMA通道4发生重复捕获中断请求标志。需要软件清零。

CC3OF: PWMA通道3发生重复捕获中断请求标志。需要软件清零。

CC2OF: PWMA通道2发生重复捕获中断请求标志。需要软件清零。

CC1OF: PWMA通道1发生重复捕获中断请求标志。需要软件清零。

**PWMB 状态寄存器**

| 符号       | 地址    | B7  | B6  | B5    | B4    | B3    | B2    | B1    | B0  |
|----------|-------|-----|-----|-------|-------|-------|-------|-------|-----|
| PWMB_SR1 | FEE5H | BIF | TIF | COMIF | CC8IF | CC7IF | CC6IF | CC5IF | UIF |
| PWMB_SR2 | FEE6H | -   | -   | -     | CC8OF | CC7OF | CC6OF | CC5OF | -   |

BIF: PWMB刹车中断请求标志。需要软件清零。

TIF: PWMB触发中断请求标志。需要软件清零。

COMIF: PWMB比较中断请求标志。需要软件清零。

CC8IF: PWMB通道8发生捕获比较中断请求标志。需要软件清零。

CC7IF: PWMB通道7发生捕获比较中断请求标志。需要软件清零。

CC6IF: PWMB通道6发生捕获比较中断请求标志。需要软件清零。

CC5IF: PWMB通道5发生捕获比较中断请求标志。需要软件清零。

UIF: PWMB更新中断请求标志。需要软件清零。

CC8OF: PWMB通道8发生重复捕获中断请求标志。需要软件清零。

CC7OF: PWMB通道7发生重复捕获中断请求标志。需要软件清零。

CC6OF: PWMB通道6发生重复捕获中断请求标志。需要软件清零。

CC5OF: PWMB通道5发生重复捕获中断请求标志。需要软件清零。

### 端口中断标志寄存器

| 符号     | 地址    | B7      | B6      | B5      | B4      | B3      | B2      | B1      | B0      |
|--------|-------|---------|---------|---------|---------|---------|---------|---------|---------|
| P0INTF | FD10H | P07INTF | P06INTF | P05INTF | P04INTF | P03INTF | P02INTF | P01INTF | P00INTF |
| P1INTF | FD11H | P17INTF | P16INTF | P15INTF | P14INTF | P13INTF | P12INTF | P11INTF | P10INTF |
| P2INTF | FD12H | P27INTF | P26INTF | P25INTF | P24INTF | P23INTF | P22INTF | P21INTF | P20INTF |
| P3INTF | FD13H | P37INTF | P36INTF | P35INTF | P34INTF | P33INTF | P32INTF | P31INTF | P30INTF |
| P4INTF | FD14H | P47INTF | P46INTF | P45INTF | P44INTF | P43INTF | P42INTF | P41INTF | P40INTF |
| P5INTF | FD15H | -       | -       | P55INTF | P54INTF | P53INTF | P52INTF | P51INTF | P50INTF |
| P6INTF | FD16H | P67INTF | P66INTF | P65INTF | P64INTF | P63INTF | P62INTF | P61INTF | P60INTF |
| P7INTF | FD17H | P77INTF | P76INTF | P75INTF | P74INTF | P73INTF | P72INTF | P71INTF | P70INTF |

PnINTF.x: 端口中断请求标志位 (n=0~7, x=0~7)

0: Pn.x 口没有中断请求

1: Pn.x 口有中断请求, 若使能中断, 则会进入中断服务程序。标志位需软件清 0。

### LCM 接口状态寄存器

| 符号       | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0      |
|----------|-------|----|----|----|----|----|----|----|---------|
| LCMIFSTA | FE53H | -  | -  | -  | -  | -  | -  | -  | LCMIFIF |

LCMIFIF: LCM接口中断请求标志。需要软件清零。

### DMA 中断标志寄存器

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2    | B1     | B0     |
|--------------|-------|----|----|----|----|----|-------|--------|--------|
| DMA_M2M_STA  | FA02H | -  | -  | -  | -  | -  | -     | -      | M2MIF  |
| DMA_ADC_STA  | FA12H | -  | -  | -  | -  | -  | -     | -      | ADCIF  |
| DMA_SPI_STA  | FA22H | -  | -  | -  | -  | -  | TXOVW | RXLOSS | SPIIF  |
| DMA_UR1T_STA | FA32H | -  | -  | -  | -  | -  | TXOVW | -      | UR1TIF |
| DMA_UR1R_STA | FA3AH | -  | -  | -  | -  | -  | -     | RXLOSS | UR1RIF |
| DMA_UR2T_STA | FA42H | -  | -  | -  | -  | -  | TXOVW | -      | UR2TIF |



|              |       |   |   |   |   |   |       |        |        |
|--------------|-------|---|---|---|---|---|-------|--------|--------|
| DMA_UR2R_STA | FA4AH | - | - | - | - | - | -     | RXLOSS | UR2RIF |
| DMA_UR3T_STA | FA52H | - | - | - | - | - | TXOVW | -      | UR3TIF |
| DMA_UR3R_STA | FA5AH | - | - | - | - | - | -     | RXLOSS | UR3RIF |
| DMA_UR4T_STA | FA62H | - | - | - | - | - | TXOVW | -      | UR4TIF |
| DMA_UR4R_STA | FA6AH | - | - | - | - | - | -     | RXLOSS | UR4RIF |
| DMA_LCM_STA  | FA72H | - | - | - | - | - | -     | TXOVW  | LCMIF  |

M2MIF: DMA\_M2M (存储器到存储器DMA) 中断请求标志。需要软件清零。

ADCIF: DMA\_ADC (ADC DMA) 中断请求标志。需要软件清零。

SPIIF: DMA\_SPI (SPI DMA) 中断请求标志。需要软件清零。。

UR1TIF: DMA\_UR1T (串口1发送DMA) 中断请求标志。需要软件清零。

UR1RIF: DMA\_UR1R (串口1接收DMA) 中断请求标志。需要软件清零。

UR2TIF: DMA\_UR2T (串口2发送DMA) 中断请求标志。需要软件清零。

UR2RIF: DMA\_UR2R (串口2接收DMA) 中断请求标志。需要软件清零。

UR3TIF: DMA\_UR3T (串口3发送DMA) 中断请求标志。需要软件清零。

UR3RIF: DMA\_UR3R (串口3接收DMA) 中断请求标志。需要软件清零。

UR4TIF: DMA\_UR4T (串口4发送DMA) 中断请求标志。需要软件清零。

UR4RIF: DMA\_UR4R (串口4接收DMA) 中断请求标志。需要软件清零。

LCMIF: DMA\_LCM (LCM接口DMA) 中断请求标志。需要软件清零。

### 12.4.3 中断优先级寄存器

除 INT2、INT3、定时器 2、定时器 3、定时器 4 以及全部的端口中断外，其他中断均有 4 级中断优先级可设置

#### 中断优先级控制寄存器

| 符号   | 地址  | B7              | B6    | B5    | B4   | B3     | B2     | B1    | B0   |
|------|-----|-----------------|-------|-------|------|--------|--------|-------|------|
| IP   | B8H | -               | PLVD  | PADC  | PS   | PT1    | PX1    | PT0   | PX0  |
| IPH  | B7H | -               | PLVDH | PADCH | PSH  | PT1H   | PX1H   | PT0H  | PX0H |
| IP2  | B5H | PUSB<br>PTKSU   | PI2C  | PCMP  | PX4  | PPWMB  | PPWMA  | PSPI  | PS2  |
| IP2H | B6H | PUSBH<br>PTKSUH | PI2CH | PCMPH | PX4H | PPWMBH | PPWMAH | PSPIH | PS2H |
| IP3  | DFH | -               | -     | -     | -    | -      | PRTC   | PS4   | PS3  |
| IP3H | EEH | -               | -     | -     | -    | -      | PRTCH  | PS4H  | PS3H |

PX0H,PX0: 外部中断0中断优先级控制位

00: INT0 中断优先级为 0 级 (最低级)

01: INT0 中断优先级为 1 级 (较低级)

10: INT0 中断优先级为 2 级 (较高级)

11: INT0 中断优先级为 3 级 (最高级)



**PT0H,PT0: 定时器0中断优先级控制位**

- 00: 定时器 0 中断优先级为 0 级 (最低级)
- 01: 定时器 0 中断优先级为 1 级 (较低级)
- 10: 定时器 0 中断优先级为 2 级 (较高级)
- 11: 定时器 0 中断优先级为 3 级 (最高级)

**PX1H,PX1: 外部中断1中断优先级控制位**

- 00: INT1 中断优先级为 0 级 (最低级)
- 01: INT1 中断优先级为 1 级 (较低级)
- 10: INT1 中断优先级为 2 级 (较高级)
- 11: INT1 中断优先级为 3 级 (最高级)

**PT1H,PT1: 定时器1中断优先级控制位**

- 00: 定时器 1 中断优先级为 0 级 (最低级)
- 01: 定时器 1 中断优先级为 1 级 (较低级)
- 10: 定时器 1 中断优先级为 2 级 (较高级)
- 11: 定时器 1 中断优先级为 3 级 (最高级)

**PSH,PS: 串口1中断优先级控制位**

- 00: 串口 1 中断优先级为 0 级 (最低级)
- 01: 串口 1 中断优先级为 1 级 (较低级)
- 10: 串口 1 中断优先级为 2 级 (较高级)
- 11: 串口 1 中断优先级为 3 级 (最高级)

**PADCH,PADC: ADC中断优先级控制位**

- 00: ADC 中断优先级为 0 级 (最低级)
- 01: ADC 中断优先级为 1 级 (较低级)
- 10: ADC 中断优先级为 2 级 (较高级)
- 11: ADC 中断优先级为 3 级 (最高级)

**PLVDH,PLVD: 低压检测中断优先级控制位**

- 00: LVD 中断优先级为 0 级 (最低级)
- 01: LVD 中断优先级为 1 级 (较低级)
- 10: LVD 中断优先级为 2 级 (较高级)
- 11: LVD 中断优先级为 3 级 (最高级)

**PS2H,PS2: 串口2中断优先级控制位**

- 00: 串口 2 中断优先级为 0 级 (最低级)
- 01: 串口 2 中断优先级为 1 级 (较低级)
- 10: 串口 2 中断优先级为 2 级 (较高级)
- 11: 串口 2 中断优先级为 3 级 (最高级)

**PS3H,PS3: 串口3中断优先级控制位**

- 00: 串口 3 中断优先级为 0 级 (最低级)
- 01: 串口 3 中断优先级为 1 级 (较低级)
- 10: 串口 3 中断优先级为 2 级 (较高级)
- 11: 串口 3 中断优先级为 3 级 (最高级)

**PS4H,PS4: 串口4中断优先级控制位**

- 00: 串口 4 中断优先级为 0 级 (最低级)
- 01: 串口 4 中断优先级为 1 级 (较低级)
- 10: 串口 4 中断优先级为 2 级 (较高级)

11: 串口 4 中断优先级为 3 级 (最高级)

PSPIH,PSPI: SPI中断优先级控制位

00: SPI 中断优先级为 0 级 (最低级)

01: SPI 中断优先级为 1 级 (较低级)

10: SPI 中断优先级为 2 级 (较高级)

11: SPI 中断优先级为 3 级 (最高级)

PPWMAH,PPWMA: 高级PWMA中断优先级控制位

00: 高级 PWMA 中断优先级为 0 级 (最低级)

01: 高级 PWMA 中断优先级为 1 级 (较低级)

10: 高级 PWMA 中断优先级为 2 级 (较高级)

11: 高级 PWMA 中断优先级为 3 级 (最高级)

PPWMBH,PPWMB: 高级PWMB中断优先级控制位

00: 高级 PWMB 中断优先级为 0 级 (最低级)

01: 高级 PWMB 中断优先级为 1 级 (较低级)

10: 高级 PWMB 中断优先级为 2 级 (较高级)

11: 高级 PWMB 中断优先级为 3 级 (最高级)

PX4H,PX4: 外部中断4中断优先级控制位

00: INT4 中断优先级为 0 级 (最低级)

01: INT4 中断优先级为 1 级 (较低级)

10: INT4 中断优先级为 2 级 (较高级)

11: INT4 中断优先级为 3 级 (最高级)

PCMPH,PCMP: 比较器中断优先级控制位

00: CMP 中断优先级为 0 级 (最低级)

01: CMP 中断优先级为 1 级 (较低级)

10: CMP 中断优先级为 2 级 (较高级)

11: CMP 中断优先级为 3 级 (最高级)

PI2CH,PI2C: I2C中断优先级控制位

00: I2C 中断优先级为 0 级 (最低级)

01: I2C 中断优先级为 1 级 (较低级)

10: I2C 中断优先级为 2 级 (较高级)

11: I2C 中断优先级为 3 级 (最高级)

PUSBH,PUSB: USB中断优先级控制位

00: USB 中断优先级为 0 级 (最低级)

01: USB 中断优先级为 1 级 (较低级)

10: USB 中断优先级为 2 级 (较高级)

11: USB 中断优先级为 3 级 (最高级)

PTKSUH,PTKSU: 触摸按键中断优先级控制位

00: 触摸按键中断优先级为 0 级 (最低级)

01: 触摸按键中断优先级为 1 级 (较低级)

10: 触摸按键中断优先级为 2 级 (较高级)

11: 触摸按键中断优先级为 3 级 (最高级)

PRTCH,PRTC: RTC中断优先级控制位

00: RTC 中断优先级为 0 级 (最低级)

01: RTC 中断优先级为 1 级 (较低级)

10: RTC 中断优先级为 2 级 (较高级)

11: RTC 中断优先级为 3 级 (最高级)

### LCM 接口配置寄存器

| 符号       | 地址    | B7      | B6 | B5           | B4 | B3            | B2 | B1     | B0      |
|----------|-------|---------|----|--------------|----|---------------|----|--------|---------|
| LCMIFCFG | FE50H | LCMIFIE | -  | LCMIFIP[1:0] |    | LCMIFDPS[1:0] |    | D16_D8 | M68_I80 |

LCMIFIP[1:0]: LCM接口中断优先级控制位

00: LCM 接口中断优先级为 0 级 (最低级)

01: LCM 接口中断优先级为 1 级 (较低级)

10: LCM 接口中断优先级为 2 级 (较高级)

11: LCM 接口中断优先级为 3 级 (最高级)

### 端口中断优先级控制寄存器

| 符号     | 地址    | B7    | B6    | B5    | B4    | B3    | B2    | B1    | B0    |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| PINIPL | FD60H | P7IP  | P6IP  | P5IP  | P4IP  | P3IP  | P2IP  | P1IP  | P0IP  |
| PINIPH | FD61H | P7IPH | P6IPH | P5IPH | P4IPH | P3IPH | P2IPH | P1IPH | P0IPH |

P0IPH,P0IP: P0口中断优先级控制位

00: P0 口中断优先级为 0 级 (最低级)

01: P0 口中断优先级为 1 级 (较低级)

10: P0 口中断优先级为 2 级 (较高级)

11: P0 口中断优先级为 3 级 (最高级)

P1IPH,P1IP: P1口中断优先级控制位

00: P1 口中断优先级为 0 级 (最低级)

01: P1 口中断优先级为 1 级 (较低级)

10: P1 口中断优先级为 2 级 (较高级)

11: P1 口中断优先级为 3 级 (最高级)

P2IPH,P2IP: P2口中断优先级控制位

00: P2 口中断优先级为 0 级 (最低级)

01: P2 口中断优先级为 1 级 (较低级)

10: P2 口中断优先级为 2 级 (较高级)

11: P2 口中断优先级为 3 级 (最高级)

P3IPH,P3IP: P3口中断优先级控制位

00: P3 口中断优先级为 0 级 (最低级)

01: P3 口中断优先级为 1 级 (较低级)

10: P3 口中断优先级为 2 级 (较高级)

11: P3 口中断优先级为 3 级 (最高级)

P4IPH,P4IP: P4口中断优先级控制位

00: P4 口中断优先级为 0 级 (最低级)

01: P4 口中断优先级为 1 级 (较低级)

10: P4 口中断优先级为 2 级 (较高级)

11: P4 口中断优先级为 3 级 (最高级)

P5IPH,P5IP: P5口中断优先级控制位

00: P5 口中断优先级为 0 级 (最低级)

01: P5 口中断优先级为 1 级 (较低级)

10: P5 口中断优先级为 2 级 (较高级)

11: P5 口中断优先级为 3 级 (最高级)

P6IPH,P6IP: P6口中断优先级控制位

00: P6 口中断优先级为 0 级 (最低级)

01: P6 口中断优先级为 1 级 (较低级)

10: P6 口中断优先级为 2 级 (较高级)

11: P6 口中断优先级为 3 级 (最高级)

P7IPH,P7IP: P7口中断优先级控制位

00: P7 口中断优先级为 0 级 (最低级)

01: P7 口中断优先级为 1 级 (较低级)

10: P7 口中断优先级为 2 级 (较高级)

11: P7 口中断优先级为 3 级 (最高级)

### DMA 中断优先级控制寄存器

| 符号           | 地址    | B7     | B6     | B5     | B4    | B3           | B2 | B1           | B0 |
|--------------|-------|--------|--------|--------|-------|--------------|----|--------------|----|
| DMA_M2M_CFG  | FA00H | M2MIE  | -      | TXACO  | RXACO | M2MIP[1:0]   |    | M2MPTY[1:0]  |    |
| DMA_ADC_CFG  | FA10H | ADCIE  | -      | -      | -     | ADCMIIP[1:0] |    | ADCPTY[1:0]  |    |
| DMA_SPI_CFG  | FA20H | SPIIE  | ACT_TX | ACT_RX | -     | SPIIIP[1:0]  |    | SPIPTY[1:0]  |    |
| DMA_UR1T_CFG | FA30H | UR1TIE | -      | -      | -     | UR1TIP[1:0]  |    | UR1TPTY[1:0] |    |
| DMA_UR1R_CFG | FA38H | UR1RIE | -      | -      | -     | UR1RIP[1:0]  |    | UR1RPTY[1:0] |    |
| DMA_UR2T_CFG | FA40H | UR2TIE | -      | -      | -     | UR2TIP[1:0]  |    | UR2TPTY[1:0] |    |
| DMA_UR2R_CFG | FA48H | UR2RIE | -      | -      | -     | UR2RIP[1:0]  |    | UR2RPTY[1:0] |    |
| DMA_UR3T_CFG | FA50H | UR3TIE | -      | -      | -     | UR3TIP[1:0]  |    | UR3TPTY[1:0] |    |
| DMA_UR3R_CFG | FA58H | UR3RIE | -      | -      | -     | UR3RIP[1:0]  |    | UR3RPTY[1:0] |    |
| DMA_UR4T_CFG | FA60H | UR4TIE | -      | -      | -     | UR4TIP[1:0]  |    | UR4TPTY[1:0] |    |
| DMA_UR4R_CFG | FA68H | UR4RIE | -      | -      | -     | UR4RIP[1:0]  |    | UR4RPTY[1:0] |    |
| DMA_LCM_CFG  | FA70H | LCMIE  | -      | -      | -     | LCMIIP[1:0]  |    | LCMPPTY[1:0] |    |

M2MIP: DMA\_M2M (存储器到存储器DMA) 中断优先级控制位

00: DMA\_M2M 中断优先级为 0 级 (最低级)

01: DMA\_M2M 中断优先级为 1 级 (较低级)

10: DMA\_M2M 中断优先级为 2 级 (较高级)

11: DMA\_M2M 中断优先级为 3 级 (最高级)

ADCIP: DMA\_ADC (ADC DMA) 中断优先级控制位

00: DMA\_ADC 中断优先级为 0 级 (最低级)

01: DMA\_ADC 中断优先级为 1 级 (较低级)

10: DMA\_ADC 中断优先级为 2 级 (较高级)

11: DMA\_ADC 中断优先级为 3 级 (最高级)

SPIIP: DMA\_SPI (SPI DMA) 中断优先级控制位

00: DMA\_SPI 中断优先级为 0 级 (最低级)

01: DMA\_SPI 中断优先级为 1 级 (较低级)

10: DMA\_SPI 中断优先级为 2 级 (较高级)

11: DMA\_SPI 中断优先级为 3 级 (最高级)

UR1TIP: DMA\_UR1T (串口1发送DMA) 中断优先级控制位

- 00: DMA\_UR1T 中断优先级为 0 级 (最低级)
- 01: DMA\_UR1T 中断优先级为 1 级 (较低级)
- 10: DMA\_UR1T 中断优先级为 2 级 (较高级)
- 11: DMA\_UR1T 中断优先级为 3 级 (最高级)

UR1RIP: DMA\_UR1R (串口1接收DMA) 中断优先级控制位

- 00: DMA\_UR1R 中断优先级为 0 级 (最低级)
- 01: DMA\_UR1R 中断优先级为 1 级 (较低级)
- 10: DMA\_UR1R 中断优先级为 2 级 (较高级)
- 11: DMA\_UR1R 中断优先级为 3 级 (最高级)

UR2TIP: DMA\_UR2T (串口2发送DMA) 中断优先级控制位

- 00: DMA\_UR2T 中断优先级为 0 级 (最低级)
- 01: DMA\_UR2T 中断优先级为 1 级 (较低级)
- 10: DMA\_UR2T 中断优先级为 2 级 (较高级)
- 11: DMA\_UR2T 中断优先级为 3 级 (最高级)

UR2RIP: DMA\_UR2R (串口2接收DMA) 中断优先级控制位

- 00: DMA\_UR2R 中断优先级为 0 级 (最低级)
- 01: DMA\_UR2R 中断优先级为 1 级 (较低级)
- 10: DMA\_UR2R 中断优先级为 2 级 (较高级)
- 11: DMA\_UR2R 中断优先级为 3 级 (最高级)

UR3TIP: DMA\_UR3T (串口3发送DMA) 中断优先级控制位

- 00: DMA\_UR3T 中断优先级为 0 级 (最低级)
- 01: DMA\_UR3T 中断优先级为 1 级 (较低级)
- 10: DMA\_UR3T 中断优先级为 2 级 (较高级)
- 11: DMA\_UR3T 中断优先级为 3 级 (最高级)

UR3RIP: DMA\_UR3R (串口3接收DMA) 中断优先级控制位

- 00: DMA\_UR3R 中断优先级为 0 级 (最低级)
- 01: DMA\_UR3R 中断优先级为 1 级 (较低级)
- 10: DMA\_UR3R 中断优先级为 2 级 (较高级)
- 11: DMA\_UR3R 中断优先级为 3 级 (最高级)

UR4TIP: DMA\_UR4T (串口4发送DMA) 中断优先级控制位

- 00: DMA\_UR4T 中断优先级为 0 级 (最低级)
- 01: DMA\_UR4T 中断优先级为 1 级 (较低级)
- 10: DMA\_UR4T 中断优先级为 2 级 (较高级)
- 11: DMA\_UR4T 中断优先级为 3 级 (最高级)

UR4RIP: DMA\_UR4R (串口4接收DMA) 中断优先级控制位

- 00: DMA\_UR4R 中断优先级为 0 级 (最低级)
- 01: DMA\_UR4R 中断优先级为 1 级 (较低级)
- 10: DMA\_UR4R 中断优先级为 2 级 (较高级)
- 11: DMA\_UR4R 中断优先级为 3 级 (最高级)

LCMIP: DMA\_LCM (LCM接口DMA) 中断优先级控制位

- 00: DMA\_LCM 中断优先级为 0 级 (最低级)
- 01: DMA\_LCM 中断优先级为 1 级 (较低级)
- 10: DMA\_LCM 中断优先级为 2 级 (较高级)
- 11: DMA\_LCM 中断优先级为 3 级 (最高级)

STC MCU

## 12.5 范例程序

### 12.5.1 INT0 中断（上升沿和下降沿），可同时支持上升沿和下降沿

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void INT0_Isr() interrupt 0
```

```
{
    if (P32)                //判断上升沿和下降沿
    {
        P10 = !P10;        //测试端口
    }
    else
    {
        P11 = !P11;        //测试端口
    }
}
```

```
void main()
```

```
{
    P_SW2 /= 0x80;          //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    IT0 = 0;                //使能INT0 上升沿和下降沿中断
    EX0 = 1;                //使能INT0 中断
    EA = 1;

    while (1);
}
```

#### 汇编代码

;测试工作频率为 11.0592MHz

|       |      |      |
|-------|------|------|
| P_SW2 | DATA | 0BAH |
| P1M1  | DATA | 091H |
| P1M0  | DATA | 092H |
| P0M1  | DATA | 093H |

```
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0003H
          LJMP         INT0ISR

INT0ISR:   ORG          0100H

          JB           P3.2,RISING      ;判断上升沿和下降沿
          CPL          P1.0             ;测试端口
          RETI

RISING:    CPL          P1.1             ;测试端口
          RETI

MAIN:      MOV         SP,#5FH
          ORL          P_SW2,#80H      ;使能访问 XFR

          MOV         P0M0,#00H
          MOV         P0M1,#00H
          MOV         P1M0,#00H
          MOV         P1M1,#00H
          MOV         P2M0,#00H
          MOV         P2M1,#00H
          MOV         P3M0,#00H
          MOV         P3M1,#00H
          MOV         P4M0,#00H
          MOV         P4M1,#00H
          MOV         P5M0,#00H
          MOV         P5M1,#00H

          CLR          IT0             ;使能INT0 上升沿和下降沿中断
          SETB         EX0             ;使能INT0 中断
          SETB         EA
          JMP          $

          END
```

12.5.2 INT0 中断（下降沿）

C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```



```
void INT0_Isr() interrupt 0
{
    P10 = !P10;                                //测试端口
}

void main()
{
    P_SW2 /= 0x80;                               //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    IT0 = 1;                                     //使能INT0 下降沿中断
    EX0 = 1;                                     //使能INT0 中断
    EA = 1;

    while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

|          |      |         |       |
|----------|------|---------|-------|
| P_SW2    | DATA | 0BAH    |       |
| P1M1     | DATA | 091H    |       |
| P1M0     | DATA | 092H    |       |
| P0M1     | DATA | 093H    |       |
| P0M0     | DATA | 094H    |       |
| P2M1     | DATA | 095H    |       |
| P2M0     | DATA | 096H    |       |
| P3M1     | DATA | 0B1H    |       |
| P3M0     | DATA | 0B2H    |       |
| P4M1     | DATA | 0B3H    |       |
| P4M0     | DATA | 0B4H    |       |
| P5M1     | DATA | 0C9H    |       |
| P5M0     | DATA | 0CAH    |       |
|          | ORG  | 0000H   |       |
|          | LJMP | MAIN    |       |
|          | ORG  | 0003H   |       |
|          | LJMP | INT0ISR |       |
|          | ORG  | 0100H   |       |
| INT0ISR: |      |         |       |
|          | CPL  | P1.0    | ;测试端口 |
|          | RETI |         |       |

**MAIN:**

```

MOV     SP, #5FH
ORL     P_SW2, #80H           ;使能访问 XFR

MOV     P0M0, #00H
MOV     P0M1, #00H
MOV     P1M0, #00H
MOV     P1M1, #00H
MOV     P2M0, #00H
MOV     P2M1, #00H
MOV     P3M0, #00H
MOV     P3M1, #00H
MOV     P4M0, #00H
MOV     P4M1, #00H
MOV     P5M0, #00H
MOV     P5M1, #00H

SETB    IT0                   ;使能 INT0 下降沿中断
SETB    EX0                   ;使能 INT0 中断
SETB    EA
JMP     $

END

```

### 12.5.3 INT1 中断（上升沿和下降沿），可同时支持上升沿和下降沿

#### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```
void INT1_Isr() interrupt 2
```

```

{
    if (INT1)                    //判断上升沿和下降沿
    {
        P10 = !P10;             //测试端口
    }
    else
    {
        P11 = !P11;             //测试端口
    }
}

```

```
void main()
```

```

{
    P_SW2 /= 0x80;               //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;

```

```
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

IT1 = 0; //使能INT1 上升沿和下降沿中断
EX1 = 1; //使能INT1 中断
EA = 1;

while (1);
}
```

汇编代码

;测试工作频率为11.0592MHz

```
P_SW2      DATA      0BAH

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

ORG        0000H
LJMP       MAIN
ORG        0013H
LJMP       INT1ISR

INT1ISR:   ORG         0100H

           JB          INT1,RISING      ;判断上升沿和下降沿
           CPL         P1.0            ;测试端口
           RETI

RISING:    CPL         P1.1            ;测试端口
           RETI

MAIN:      MOV         SP, #5FH
           ORL         P_SW2,#80H      ;使能访问XFR

           MOV         P0M0, #00H
           MOV         P0M1, #00H
           MOV         P1M0, #00H
           MOV         P1M1, #00H
           MOV         P2M0, #00H
           MOV         P2M1, #00H
           MOV         P3M0, #00H
           MOV         P3M1, #00H
           MOV         P4M0, #00H
```

|             |                   |                   |
|-------------|-------------------|-------------------|
| <i>MOV</i>  | <i>P4M1, #00H</i> |                   |
| <i>MOV</i>  | <i>P5M0, #00H</i> |                   |
| <i>MOV</i>  | <i>P5M1, #00H</i> |                   |
| <i>CLR</i>  | <i>IT1</i>        | ;使能INT1 上升沿和下降沿中断 |
| <i>SETB</i> | <i>EX1</i>        | ;使能INT1 中断        |
| <i>SETB</i> | <i>EA</i>         |                   |
| <i>JMP</i>  | <i>\$</i>         |                   |
| <i>END</i>  |                   |                   |

## 12.5.4 INT1 中断（下降沿）

### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void INT1_Isr() interrupt 2
{
    P10 = !P10; //测试端口
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    IT1 = 1; //使能INT1 下降沿中断
    EX1 = 1; //使能INT1 中断
    EA = 1;

    while (1);
}
```

### 汇编代码

```
;测试工作频率为 11.0592MHz

P_SW2      DATA      0BAH

P1M1        DATA      091H
```

```
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0013H
          LJMP         INT1ISR

INT1ISR:   ORG          0100H

          CPL          P1.0          ;测试端口
          RETI

MAIN:

          MOV          SP, #5FH
          ORL          P_SW2, #80H   ;使能访问 XFR

          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          SETB         IT1           ;使能 INT1 下降沿中断
          SETB         EX1           ;使能 INT1 中断
          SETB         EA
          JMP          $

          END
```

### 12.5.5 INT2 中断（下降沿），只支持下降沿中断

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void INT2_Isr() interrupt 10
```

```
{
    P10 = !P10;                                //测试端口
}

void main()
{
    P_SW2 /= 0x80;                            //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    INTCLKO = EX2;                            //使能INT2 中断
    EA = 1;

    while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

|          |      |         |
|----------|------|---------|
| INTCLKO  | DATA | 8FH     |
| EX2      | EQU  | 10H     |
| EX3      | EQU  | 20H     |
| EX4      | EQU  | 40H     |
| P_SW2    | DATA | 0BAH    |
| P1M1     | DATA | 091H    |
| P1M0     | DATA | 092H    |
| P0M1     | DATA | 093H    |
| P0M0     | DATA | 094H    |
| P2M1     | DATA | 095H    |
| P2M0     | DATA | 096H    |
| P3M1     | DATA | 0B1H    |
| P3M0     | DATA | 0B2H    |
| P4M1     | DATA | 0B3H    |
| P4M0     | DATA | 0B4H    |
| P5M1     | DATA | 0C9H    |
| P5M0     | DATA | 0CAH    |
|          | ORG  | 0000H   |
|          | LJMP | MAIN    |
|          | ORG  | 0053H   |
|          | LJMP | INT2ISR |
|          | ORG  | 0100H   |
| INT2ISR: |      |         |
|          | CPL  | P1.0    |
|          |      | ;测试端口   |

*RETI*

*MAIN:*

```
MOV    SP, #5FH
ORL     P_SW2, #80H           ;使能访问 XFR

MOV     P0M0, #00H
MOV     P0M1, #00H
MOV     P1M0, #00H
MOV     P1M1, #00H
MOV     P2M0, #00H
MOV     P2M1, #00H
MOV     P3M0, #00H
MOV     P3M1, #00H
MOV     P4M0, #00H
MOV     P4M1, #00H
MOV     P5M0, #00H
MOV     P5M1, #00H

MOV     INTCLKO, #EX2         ;使能 INT2 中断
SETB    EA
JMP     $
```

*END*

## 12.5.6 INT3 中断（下降沿），只支持下降沿中断

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void INT3_Isr() interrupt 11
```

```
{
    P10 = !P10;           //测试端口
}
```

```
void main()
```

```
{
    P_SW2 /= 0x80;         //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;
```

```

    INTCLKO = EX3;
    EA = 1;

    while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

```

P_SW2      DATA      0BAH

INTCLKO     DATA      8FH
EX2         EQU        10H
EX3         EQU        20H
EX4         EQU        40H

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         005BH
            LJMP        INT3ISR

INT3ISR:    ORG         0100H

            CPL          P1.0          ;测试端口
            RETI

MAIN:

            MOV         SP, #5FH
            ORL          P_SW2, #80H   ;使能访问 XFR

            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         INTCLKO, #EX3  ;使能 INT3 中断
            SETB        EA

```



JMP\$

END

### 12.5.7 INT4 中断（下降沿），只支持下降沿中断

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void INT4_Isr() interrupt 16
{
    P10 = !P10;                //测试端口
}

void main()
{
    P_SW2 /= 0x80;             //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    INTCLKO = EX4;             //使能 INT4 中断
    EA = 1;

    while (1);
}
```

#### 汇编代码

```
;测试工作频率为 11.0592MHz

P_SW2      DATA      0BAH

INTCLKO     DATA      8FH
EX2         EQU        10H
EX3         EQU        20H
EX4         EQU        40H

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
```

```
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0083H
          LJMP         INT4ISR

INT4ISR:   ORG          0100H

          CPL          P1.0          ;测试端口
          RETI

MAIN:

          MOV          SP, #5FH
          ORL          P_SW2, #80H    ;使能访问 XFR

          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          MOV          INTCLKO, #EX4    ;使能 INT4 中断
          SETB         EA
          JMP          $

          END
```

## 12.5.8 定时器 0 中断

### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void TM0_Isr() interrupt 1
{
    P10 = !P10;          //测试端口
}
```

```
void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x00;
    TL0 = 0x66; //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1; //启动定时器
    ET0 = 1; //使能定时器中断
    EA = 1;

    while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

|         |      |             |           |
|---------|------|-------------|-----------|
| P_SW2   | DATA | 0BAH        |           |
| P1M1    | DATA | 091H        |           |
| P1M0    | DATA | 092H        |           |
| P0M1    | DATA | 093H        |           |
| P0M0    | DATA | 094H        |           |
| P2M1    | DATA | 095H        |           |
| P2M0    | DATA | 096H        |           |
| P3M1    | DATA | 0B1H        |           |
| P3M0    | DATA | 0B2H        |           |
| P4M1    | DATA | 0B3H        |           |
| P4M0    | DATA | 0B4H        |           |
| P5M1    | DATA | 0C9H        |           |
| P5M0    | DATA | 0CAH        |           |
|         | ORG  | 0000H       |           |
|         | LJMP | MAIN        |           |
|         | ORG  | 000BH       |           |
|         | LJMP | TM0ISR      |           |
|         | ORG  | 0100H       |           |
| TM0ISR: |      |             |           |
|         | CPL  | P1.0        | ;测试端口     |
|         | RETI |             |           |
| MAIN:   |      |             |           |
|         | MOV  | SP, #5FH    |           |
|         | ORL  | P_SW2, #80H | ;使能访问 XFR |

```
MOV    P0M0, #00H
MOV    P0M1, #00H
MOV    P1M0, #00H
MOV    P1M1, #00H
MOV    P2M0, #00H
MOV    P2M1, #00H
MOV    P3M0, #00H
MOV    P3M1, #00H
MOV    P4M0, #00H
MOV    P4M1, #00H
MOV    P5M0, #00H
MOV    P5M1, #00H

MOV    TMOD, #00H
MOV    TL0, #66H                ;65536-11.0592M/12/1000
MOV    TH0, #0FCH
SETB   TR0                    ;启动定时器
SETB   ET0                    ;使能定时器中断
SETB   EA

JMP    $

END
```

## 12.5.9 定时器 1 中断

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void TM1_Isr() interrupt 3
{
    P10 = !P10;
}
```

//测试端口

```
void main()
{
```

```
    P_SW2 /= 0x80;
```

//使能访问 XFR

```
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;
```

```
TMOD = 0x00;
TL1 = 0x66; //65536-11.0592M/12/1000
TH1 = 0xfc;
TR1 = 1; //启动定时器
ET1 = 1; //使能定时器中断
EA = 1;

while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

```
P_SW2      DATA      0BAH

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

ORG        0000H
LJMP       MAIN
ORG        001BH
LJMP       TM1ISR

TM1ISR:    ORG        0100H

CPL        P1.0 ;测试端口
RETI

MAIN:

MOV        SP, #5FH
ORL        P_SW2, #80H ;使能访问 XFR

MOV        P0M0, #00H
MOV        P0M1, #00H
MOV        P1M0, #00H
MOV        P1M1, #00H
MOV        P2M0, #00H
MOV        P2M1, #00H
MOV        P3M0, #00H
MOV        P3M1, #00H
MOV        P4M0, #00H
MOV        P4M1, #00H
MOV        P5M0, #00H
MOV        P5M1, #00H

MOV        TMOD, #00H
MOV        TL1, #66H ;65536-11.0592M/12/1000
MOV        TH1, #0FCH
```

|      |     |          |
|------|-----|----------|
| SETB | TRI | ;启动定时器   |
| SETB | ETI | ;使能定时器中断 |
| SETB | EA  |          |
| JMP  | \$  |          |
| END  |     |          |

## 12.5.10 定时器 2 中断

### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void TM2_Isr() interrupt 12
{
    P10 = !P10; //测试端口
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T2L = 0x66; //65536-11.0592M/12/1000
    T2H = 0xfc;
    AUXR = 0x10; //启动定时器
    IE2 = ET2; //使能定时器中断
    EA = 1;

    while (1);
}
```

### 汇编代码

```
;测试工作频率为 11.0592MHz

P_SW2    DATA    0BAH

T2L       DATA    0D7H
T2H       DATA    0D6H
```

```

AUXR      DATA      8EH
IE2       DATA      0AFH
ET2       EQU        04H
AUXINTIF  DATA      0EFH
T2IF      EQU        01H

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG         0000H
          LJMP        MAIN
          ORG         0063H
          LJMP        TM2ISR

TM2ISR:   ORG         0100H

          CPL         P1.0      ;测试端口
          RETI

MAIN:

          MOV         SP, #5FH
          ORL         P_SW2, #80H      ;使能访问 XFR

          MOV         P0M0, #00H
          MOV         P0M1, #00H
          MOV         P1M0, #00H
          MOV         P1M1, #00H
          MOV         P2M0, #00H
          MOV         P2M1, #00H
          MOV         P3M0, #00H
          MOV         P3M1, #00H
          MOV         P4M0, #00H
          MOV         P4M1, #00H
          MOV         P5M0, #00H
          MOV         P5M1, #00H

          MOV         T2L, #66H      ;65536-11.0592M/12/1000
          MOV         T2H, #0FCH
          MOV         AUXR, #10H     ;启动定时器
          MOV         IE2, #ET2     ;使能定时器中断
          SETB        EA

          JMP         $

          END

```

# 12.5.11 定时器 3 中断

## C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void TM3_Isr() interrupt 19
{
    P10 = !P10;           //测试端口
}

void main()
{
    P_SW2 /= 0x80;        //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T3L = 0x66;           //65536-11.0592M/12/1000
    T3H = 0xfc;
    T4T3M = 0x08;        //启动定时器
    IE2 = ET3;           //使能定时器中断
    EA = 1;

    while (1);
}
```

## 汇编代码

;测试工作频率为 11.0592MHz

|                 |             |             |
|-----------------|-------------|-------------|
| <i>P_SW2</i>    | <i>DATA</i> | <i>0BAH</i> |
| <i>T3L</i>      | <i>DATA</i> | <i>0D5H</i> |
| <i>T3H</i>      | <i>DATA</i> | <i>0D4H</i> |
| <i>T4T3M</i>    | <i>DATA</i> | <i>0D1H</i> |
| <i>IE2</i>      | <i>DATA</i> | <i>0AFH</i> |
| <i>ET3</i>      | <i>EQU</i>  | <i>20H</i>  |
| <i>AUXINTIF</i> | <i>DATA</i> | <i>0EFH</i> |
| <i>T3IF</i>     | <i>EQU</i>  | <i>02H</i>  |
| <i>P1M1</i>     | <i>DATA</i> | <i>091H</i> |
| <i>P1M0</i>     | <i>DATA</i> | <i>092H</i> |
| <i>P0M1</i>     | <i>DATA</i> | <i>093H</i> |
| <i>P0M0</i>     | <i>DATA</i> | <i>094H</i> |



```
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          009BH
          LJMP         TM3ISR

TM3ISR:   ORG          0100H

          CPL          P1.0          ;测试端口
          RETI

MAIN:

          MOV          SP, #5FH
          ORL          P_SW2, #80H   ;使能访问 XFR

          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          MOV          T3L, #66H      ;65536-11.0592M/12/1000
          MOV          T3H, #0FCH
          MOV          T4T3M, #08H    ;启动定时器
          MOV          IE2, #ET3      ;使能定时器中断
          SETB         EA

          JMP          $

          END
```

### 12.5.12 定时器 4 中断

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void TM4_Isr() interrupt 20
```

```

{
    P10 = !P10;                //测试端口
}

void main()
{
    P_SW2 /= 0x80;             //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T4L = 0x66;                //65536-11.0592M/12/1000
    T4H = 0xfc;
    T4T3M = 0x80;              //启动定时器
    IE2 = ET4;                 //使能定时器中断
    EA = 1;

    while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|          |      |      |
|----------|------|------|
| P_SW2    | DATA | 0BAH |
| T3L      | DATA | 0D5H |
| T3H      | DATA | 0D4H |
| T4L      | DATA | 0D3H |
| T4H      | DATA | 0D2H |
| T4T3M    | DATA | 0D1H |
| IE2      | DATA | 0AFH |
| ET3      | EQU  | 20H  |
| ET4      | EQU  | 40H  |
| AUXINTIF | DATA | 0EFH |
| T3IF     | EQU  | 02H  |
| T4IF     | EQU  | 04H  |
| P1M1     | DATA | 091H |
| P1M0     | DATA | 092H |
| P0M1     | DATA | 093H |
| P0M0     | DATA | 094H |
| P2M1     | DATA | 095H |
| P2M0     | DATA | 096H |
| P3M1     | DATA | 0B1H |
| P3M0     | DATA | 0B2H |
| P4M1     | DATA | 0B3H |
| P4M0     | DATA | 0B4H |
| P5M1     | DATA | 0C9H |

```
P5M0      DATA      0CAH

           ORG         0000H
           LJMP        MAIN
           ORG         00A3H
           LJMP        TM4ISR

           ORG         0100H
TM4ISR:
           CPL         P1.0           ;测试端口
           RETI

MAIN:
           MOV         SP, #5FH
           ORL         P_SW2, #80H   ;使能访问 XFR

           MOV         P0M0, #00H
           MOV         P0M1, #00H
           MOV         P1M0, #00H
           MOV         P1M1, #00H
           MOV         P2M0, #00H
           MOV         P2M1, #00H
           MOV         P3M0, #00H
           MOV         P3M1, #00H
           MOV         P4M0, #00H
           MOV         P4M1, #00H
           MOV         P5M0, #00H
           MOV         P5M1, #00H

           MOV         T4L, #66H     ;65536-11.0592M/12/1000
           MOV         T4H, #0FCH
           MOV         T4T3M, #80H   ;启动定时器
           MOV         IE2, #ET4     ;使能定时器中断
           SETB        EA

           JMP         $

           END
```

### 12.5.13 UART1 中断

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void UART1_Isr() interrupt 4
{
    if (TI)
    {
        TI = 0;           //清中断标志
        P10 = !P10;       //测试端口
    }
    if (RI)
```

```
{
    RI = 0; //清中断标志
    P1I = !P1I; //测试端口
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    SCON = 0x50;
    T2L = 0xe8; //65536-11059200/115200/4=0FFE8H
    T2H = 0xff;
    AUXR = 0x15; //启动定时器
    ES = 1; //使能串口中断
    EA = 1;
    SBUF = 0x5a; //发送测试数据

    while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

|       |      |       |
|-------|------|-------|
| P_SW2 | DATA | 0BAH  |
| T2L   | DATA | 0D7H  |
| T2H   | DATA | 0D6H  |
| AUXR  | DATA | 8EH   |
| P1M1  | DATA | 091H  |
| P1M0  | DATA | 092H  |
| P0M1  | DATA | 093H  |
| P0M0  | DATA | 094H  |
| P2M1  | DATA | 095H  |
| P2M0  | DATA | 096H  |
| P3M1  | DATA | 0B1H  |
| P3M0  | DATA | 0B2H  |
| P4M1  | DATA | 0B3H  |
| P4M0  | DATA | 0B4H  |
| P5M1  | DATA | 0C9H  |
| P5M0  | DATA | 0CAH  |
|       | ORG  | 0000H |
|       | LJMP | MAIN  |

```

    ORG          0023H
    LJMP         UART1ISR

UART1ISR:
    ORG          0100H

    JNB          TI,CHECKRI
    CLR          TI                ;清中断标志
    CPL          P1.0              ;测试端口

CHECKRI:
    JNB          RI,ISREXIT
    CLR          RI                ;清中断标志
    CPL          P1.1              ;测试端口

ISREXIT:
    RETI

MAIN:
    MOV          SP,#5FH
    ORL          P_SW2,#80H        ;使能访问 XFR

    MOV          P0M0,#00H
    MOV          P0M1,#00H
    MOV          P1M0,#00H
    MOV          P1M1,#00H
    MOV          P2M0,#00H
    MOV          P2M1,#00H
    MOV          P3M0,#00H
    MOV          P3M1,#00H
    MOV          P4M0,#00H
    MOV          P4M1,#00H
    MOV          P5M0,#00H
    MOV          P5M1,#00H

    MOV          SCON,#50H
    MOV          T2L,#0E8H        ;65536-11059200/115200/4=0FFE8H
    MOV          T2H,#0FFH
    MOV          AUXR,#15H        ;启动定时器
    SETB         ES              ;使能串口中断
    SETB         EA
    MOV          SBUF,#5AH        ;发送测试数据

    JMP          $

END
```

12.5.14 UART2 中断

C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void UART2_Isr() interrupt 8
{
    if (S2CON & 0x02)
```

```
{
    S2CON &= ~0x02;           //清中断标志
    P12 = !P12;               //测试端口
}
if (S2CON & 0x01)
{
    S2CON &= ~0x01;           //清中断标志
    P13 = !P13;               //测试端口
}
}

void main()
{
    P_SW2 /= 0x80;             //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    S2CON = 0x10;
    T2L = 0xe8;                //65536-11059200/115200/4=0FFE8H
    T2H = 0xff;
    AUXR = 0x14;               //启动定时器
    IE2 = ES2;                 //使能串口中断
    EA = 1;
    S2BUF = 0x5a;              //发送测试数据

    while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

|       |      |      |
|-------|------|------|
| P_SW2 | DATA | 0BAH |
| T2L   | DATA | 0D7H |
| T2H   | DATA | 0D6H |
| AUXR  | DATA | 8EH  |
| S2CON | DATA | 9AH  |
| S2BUF | DATA | 9BH  |
| IE2   | DATA | 0AFH |
| ES2   | EQU  | 01H  |
| P1M1  | DATA | 091H |
| P1M0  | DATA | 092H |
| P0M1  | DATA | 093H |
| P0M0  | DATA | 094H |
| P2M1  | DATA | 095H |
| P2M0  | DATA | 096H |

```

P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0043H
          LJMP         UART2ISR

          ORG          0100H
UART2ISR:
          PUSH         ACC
          PUSH         PSW
          MOV          A,S2CON
          JNB          ACC.1,CHECKRI
          ANL          S2CON,#NOT 02H      ;清中断标志
          CPL          P1.2                ;测试端口

CHECKRI:
          MOV          A,S2CON
          JNB          ACC.0,ISREXIT
          ANL          S2CON,#NOT 01H      ;清中断标志
          CPL          P1.3                ;测试端口

ISREXIT:
          POP          PSW
          POP          ACC
          RETI

MAIN:
          MOV          SP,#5FH
          ORL          P_SW2,#80H          ;使能访问 XFR

          MOV          P0M0,#00H
          MOV          P0M1,#00H
          MOV          P1M0,#00H
          MOV          P1M1,#00H
          MOV          P2M0,#00H
          MOV          P2M1,#00H
          MOV          P3M0,#00H
          MOV          P3M1,#00H
          MOV          P4M0,#00H
          MOV          P4M1,#00H
          MOV          P5M0,#00H
          MOV          P5M1,#00H

          MOV          S2CON,#10H
          MOV          T2L,#0E8H           ;65536-11059200/115200/4=0FFE8H
          MOV          T2H,#0FFH
          MOV          AUXR,#14H           ;启动定时器
          MOV          IE2,#ES2           ;使能串口中断
          SETB         EA
          MOV          S2BUF,#5AH         ;发送测试数据

          JMP          $

          END

```

## 12.5.15 UART3 中断

### C 语言代码

---

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void UART3_Isr() interrupt 17
```

```
{
    if (S3CON & 0x02)
    {
        S3CON &= ~0x02;           //清中断标志
        P12 = !P12;               //测试端口
    }
    if (S3CON & 0x01)
    {
        S3CON &= ~0x01;           //清中断标志
        P13 = !P13;               //测试端口
    }
}
```

```
void main()
```

```
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    S3CON = 0x10;
    T2L = 0xe8;                   //65536-11059200/115200/4=0FFE8H
    T2H = 0xff;
    AUXR = 0x14;                  //启动定时器
    IE2 = ES3;                    //使能串口中断
    EA = 1;
    S3BUF = 0x5a;                 //发送测试数据

    while (1);
}
```

---

### 汇编代码

---

;测试工作频率为 11.0592MHz

```
P_SW2      DATA      0BAH
```



```

T2L      DATA      0D7H
T2H      DATA      0D6H
AUXR     DATA      8EH
S3CON    DATA      0ACH
S3BUF    DATA      0ADH
IE2      DATA      0AFH
ES3      EQU        08H

P0M1     DATA      093H
P0M0     DATA      094H
P1M1     DATA      091H
P1M0     DATA      092H
P2M1     DATA      095H
P2M0     DATA      096H
P3M1     DATA      0B1H
P3M0     DATA      0B2H
P4M1     DATA      0B3H
P4M0     DATA      0B4H
P5M1     DATA      0C9H
P5M0     DATA      0CAH

ORG      0000H
LJMP     MAIN
ORG      008BH
LJMP     UART3ISR

ORG      0100H
UART3ISR:
    PUSH    ACC
    PUSH    PSW
    MOV     A,S3CON
    JNB     ACC.1,CHECKRI
    ANL     S3CON,#NOT 02H    ;清中断标志
    CPL     P1.2             ;测试端口

CHECKRI:
    MOV     A,S3CON
    JNB     ACC.0,ISREXIT
    ANL     S3CON,#NOT 01H    ;清中断标志
    CPL     P1.3             ;测试端口

ISREXIT:
    POP     PSW
    POP     ACC
    RETI

MAIN:
    MOV     SP,#5FH
    ORL     P_SW2,#80H        ;使能访问 XFR

    MOV     P0M0,#00H
    MOV     P0M1,#00H
    MOV     P1M0,#00H
    MOV     P1M1,#00H
    MOV     P2M0,#00H
    MOV     P2M1,#00H
    MOV     P3M0,#00H
    MOV     P3M1,#00H
    MOV     P4M0,#00H
    MOV     P4M1,#00H
    MOV     P5M0,#00H

```

```
MOV    P5M1, #00H

MOV    S3CON, #10H
MOV    T2L, #0E8H           ;65536-11059200/115200/4=0FFE8H
MOV    T2H, #0FFH
MOV    AUXR, #14H           ;启动定时器
MOV    IE2, #ES3           ;使能串口中断
SETB   EA
MOV    S3BUF, #5AH         ;发送测试数据

JMP    $

END
```

## 12.5.16 UART4 中断

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void UART4_Isr() interrupt 18
{
```

```
    if (S4CON & 0x02)
```

```
    {
```

```
        S4CON &= ~0x02;
```

```
        P12 = !P12;
```

//清中断标志

//测试端口

```
    }
```

```
    if (S4CON & 0x01)
```

```
    {
```

```
        S4CON &= ~0x01;
```

```
        P13 = !P13;
```

//清中断标志

//测试端口

```
    }
```

```
}
```

```
void main()
```

```
{
```

```
    P_SW2 /= 0x80;
```

//使能访问 XFR

```
    P0M0 = 0x00;
```

```
    P0M1 = 0x00;
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P2M0 = 0x00;
```

```
    P2M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P4M0 = 0x00;
```

```
    P4M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
    S4CON = 0x10;
```

```
    T2L = 0xe8;
```

//65536-11059200/115200/4=0FFE8H

```
    T2H = 0xff;
```

```

    AUXR = 0x14;           //启动定时器
    IE2 = ES4;             //使能串口中断
    EA = 1;
    S4BUF = 0x5a;          //发送测试数据

    while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

```

P_SW2      DATA      0BAH

T2L         DATA      0D7H
T2H         DATA      0D6H
AUXR        DATA      8EH
S4CON       DATA      84H
S4BUF       DATA      85H
IE2         DATA      0AFH
ES4         EQU        10H

P0M1        DATA      093H
P0M0        DATA      094H
P1M1        DATA      091H
P1M0        DATA      092H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         0093H
            LJMP        UART4ISR

            ORG         0100H
UART4ISR:
            PUSH        ACC
            PUSH        PSW
            MOV         A,S4CON
            JNB         ACC.1,CHECKRI
            ANL         S4CON,#NOT 02H      ;清中断标志
            CPL         P1.2                ;测试端口

CHECKRI:
            MOV         A,S4CON
            JNB         ACC.0,ISREXIT
            ANL         S4CON,#NOT 01H      ;清中断标志
            CPL         P1.3                ;测试端口

ISREXIT:
            POP         PSW
            POP         ACC
            RETI

```

MAIN:

```

MOV    SP, #5FH
ORL    P_SW2, #80H           ;使能访问 XFR

MOV    P0M0, #00H
MOV    P0M1, #00H
MOV    P1M0, #00H
MOV    P1M1, #00H
MOV    P2M0, #00H
MOV    P2M1, #00H
MOV    P3M0, #00H
MOV    P3M1, #00H
MOV    P4M0, #00H
MOV    P4M1, #00H
MOV    P5M0, #00H
MOV    P5M1, #00H

MOV    S4CON, #10H
MOV    T2L, #0E8H           ;65536-11059200/115200/4=0FFE8H
MOV    T2H, #0FFH
MOV    AUXR, #14H           ;启动定时器
MOV    IE2, #ES4            ;使能串口中断
SETB   EA
MOV    S4BUF, #5AH          ;发送测试数据

JMP    $

END

```

## 12.5.17 ADC 中断

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```
void ADC_Isr() interrupt 5
```

```

{
    ADC_CONTR &= ~0x20;           //清中断标志
    P0 = ADC_RES;                 //测试端口
    P2 = ADC_RES1;                //测试端口
}

```

```
void main()
```

```

{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
}

```

```
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

ADCCFG = 0x00;
ADC_CONTR = 0xc0;           //使能并启动ADC 模块
EADC = 1;                   //使能ADC 中断
EA = 1;

while (1);
}
```

汇编代码

;测试工作频率为11.0592MHz

```
P_SW2      DATA      0BAH

ADC_CONTR  DATA      0BCH
ADC_RES    DATA      0BDH
ADC_RESL   DATA      0BEH
ADCCFG     DATA      0DEH
EADC       BIT        IE.5

P1M1       DATA      091H
P1M0       DATA      092H
P0M1       DATA      093H
P0M0       DATA      094H
P2M1       DATA      095H
P2M0       DATA      096H
P3M1       DATA      0B1H
P3M0       DATA      0B2H
P4M1       DATA      0B3H
P4M0       DATA      0B4H
P5M1       DATA      0C9H
P5M0       DATA      0CAH

ORG        0000H
LJMP       MAIN
ORG        002BH
LJMP       ADCISR

ORG        0100H
ADCISR:
ANL        ADC_CONTR,#NOT 20H    ;清中断标志
MOV        P0,ADC_RES           ;测试端口
MOV        P2,ADC_RESL          ;测试端口
RETI

MAIN:
MOV        SP,#5FH
ORL        P_SW2,#80H           ;使能访问XFR

MOV        P0M0,#00H
MOV        P0M1,#00H
MOV        P1M0,#00H
MOV        P1M1,#00H
MOV        P2M0,#00H
```

```
MOV    P2M1, #00H
MOV    P3M0, #00H
MOV    P3M1, #00H
MOV    P4M0, #00H
MOV    P4M1, #00H
MOV    P5M0, #00H
MOV    P5M1, #00H

MOV    ADCCFG, #00H
MOV    ADC_CONTR, #0C0H    ;使能并启动 ADC 模块
SETB   EADC                ;使能 ADC 中断
SETB   EA

JMP    $

END
```

## 12.5.18 LVD 中断

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void LVD_Isr() interrupt 6
```

```
{
    PCON &= ~LVDF;    //清中断标志
    P10 = !P10;        //测试端口
}
```

```
void main()
```

```
{
    P_SW2 /= 0x80;    //使能访问 XFR
```

```
P0M0 = 0x00;
P0M1 = 0x00;
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;
```

```
PCON &= ~LVDF;    //上电需要清中断标志
RSTCFG = LVD3V0;    //设置 LVD 电压为 3.0V
ELVD = 1; //使能 LVD 中断
EA = 1;
```

```
while (1);
```

```
}
```

## 汇编代码

;测试工作频率为 11.0592MHz

```

P_SW2      DATA      0BAH

RSTCFG      DATA      0FFH
ENLVR       EQU        40H           ;RSTCFG.6
LVD2V2      EQU        00H           ;LVD@2.2V
LVD2V4      EQU        01H           ;LVD@2.4V
LVD2V7      EQU        02H           ;LVD@2.7V
LVD3V0      EQU        03H           ;LVD@3.0V
ELVD        BIT        IE.6
LVDF        EQU        20H           ;PCON.5

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

                ORG      0000H
                LJMP     MAIN
                ORG      0033H
                LJMP     LVDISR

LVDISR:       ORG      0100H

                ANL      PCON,#NOT LVDF      ;清中断标志
                CPL      P1.0                ;测试端口
                RETI

MAIN:

                MOV      SP,#5FH
                ORL      P_SW2,#80H           ;使能访问 XFR

                MOV      P0M0,#00H
                MOV      P0M1,#00H
                MOV      P1M0,#00H
                MOV      P1M1,#00H
                MOV      P2M0,#00H
                MOV      P2M1,#00H
                MOV      P3M0,#00H
                MOV      P3M1,#00H
                MOV      P4M0,#00H
                MOV      P4M1,#00H
                MOV      P5M0,#00H
                MOV      P5M1,#00H

                ANL      PCON,#NOT LVDF      ;上电需要清中断标志
                MOV      RSTCFG,#LVD3V0     ;设置 LVD 电压为 3.0V

```

|      |      |            |
|------|------|------------|
| SETB | ELVD | ;使能 LVD 中断 |
| SETB | EA   |            |
| JMP  | \$   |            |
| END  |      |            |

## 12.5.19 比较器中断

### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void CMP_Isr() interrupt 21
{
    CMPCR1 &= ~0x40;           //清中断标志
    P10 = !P10;                //测试端口
}

void main()
{
    P_SW2 /= 0x80;             //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    CMPCR2 = 0x00;
    CMPCR1 = 0x80;             //使能比较器模块
    CMPCR1 /= 0x30;            //使能比较器边沿中断
    CMPCR1 &= ~0x08;           //P3.6 为 CMP+ 输入脚
    CMPCR1 /= 0x04;            //P3.7 为 CMP- 输入脚
    CMPCR1 /= 0x02;            //使能比较器输出
    EA = 1;

    while (1);
}
```

### 汇编代码

```
;测试工作频率为 11.0592MHz

P_SW2      DATA      0BAH

CMPCR1     DATA      0E6H
```



```

CMPCR2    DATA    0E7H

P1M1      DATA    091H
P1M0      DATA    092H
P0M1      DATA    093H
P0M0      DATA    094H
P2M1      DATA    095H
P2M0      DATA    096H
P3M1      DATA    0B1H
P3M0      DATA    0B2H
P4M1      DATA    0B3H
P4M0      DATA    0B4H
P5M1      DATA    0C9H
P5M0      DATA    0CAH

        ORG        0000H
        LJMP       MAIN
        ORG        00ABH
        LJMP       CMPISR

        ORG        0100H
CMPISR:
        ANL        CMPCR1,#NOT 40H    ;清中断标志
        CPL        P1.0              ;测试端口
        RETI

MAIN:
        MOV        SP,#5FH
        ORL        P_SW2,#80H        ;使能访问 XFR

        MOV        P0M0,#00H
        MOV        P0M1,#00H
        MOV        P1M0,#00H
        MOV        P1M1,#00H
        MOV        P2M0,#00H
        MOV        P2M1,#00H
        MOV        P3M0,#00H
        MOV        P3M1,#00H
        MOV        P4M0,#00H
        MOV        P4M1,#00H
        MOV        P5M0,#00H
        MOV        P5M1,#00H

        MOV        CMPCR2,#00H
        MOV        CMPCR1,#80H        ;使能比较器模块
        ORL        CMPCR1,#30H        ;使能比较器边沿中断
        ANL        CMPCR1,#NOT 08H    ;P3.6 为 CMP+ 输入脚
        ORL        CMPCR1,#04H        ;P3.7 为 CMP- 输入脚
        ORL        CMPCR1,#02H        ;使能比较器输出
        SETB       EA

        JMP        $

        END

```

## 12.5.20 SPI 中断

### C 语言代码

---

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void SPI_Isr() interrupt 9
{
    SPSTAT = 0xc0;           //清中断标志
    P10 = !P10;              //测试端口
}

void main()
{
    P_SW2 /= 0x80;           //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    SPCTL = 0x50;            //使能 SPI 主机模式
    SPSTAT = 0xc0;           //清中断标志
    IE2 = ESPI;              //使能 SPI 中断
    EA = 1;
    SPDAT = 0x5a;            //发送测试数据

    while (1);
}
```

---

### 汇编代码

---

;测试工作频率为 11.0592MHz

|               |             |             |
|---------------|-------------|-------------|
| <i>P_SW2</i>  | <i>DATA</i> | <i>0BAH</i> |
| <i>SPSTAT</i> | <i>DATA</i> | <i>0CDH</i> |
| <i>SPCTL</i>  | <i>DATA</i> | <i>0CEH</i> |
| <i>SPDAT</i>  | <i>DATA</i> | <i>0CFH</i> |
| <i>IE2</i>    | <i>DATA</i> | <i>0AFH</i> |
| <i>ESPI</i>   | <i>EQU</i>  | <i>02H</i>  |
| <i>P1M1</i>   | <i>DATA</i> | <i>091H</i> |
| <i>P1M0</i>   | <i>DATA</i> | <i>092H</i> |
| <i>P0M1</i>   | <i>DATA</i> | <i>093H</i> |
| <i>P0M0</i>   | <i>DATA</i> | <i>094H</i> |
| <i>P2M1</i>   | <i>DATA</i> | <i>095H</i> |

```
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          004BH
          LJMP         SPIISR

          ORG          0100H
SPIISR:
          MOV          SPSTAT,#0C0H      ;清中断标志
          CPL          P1.0              ;测试端口
          RETI

MAIN:
          MOV          SP,#5FH
          ORL          P_SW2,#80H        ;使能访问 XFR

          MOV          P0M0,#00H
          MOV          P0M1,#00H
          MOV          P1M0,#00H
          MOV          P1M1,#00H
          MOV          P2M0,#00H
          MOV          P2M1,#00H
          MOV          P3M0,#00H
          MOV          P3M1,#00H
          MOV          P4M0,#00H
          MOV          P4M1,#00H
          MOV          P5M0,#00H
          MOV          P5M1,#00H

          MOV          SPCTL,#50H        ;使能 SPI 主机模式
          MOV          SPSTAT,#0C0H      ;清中断标志
          MOV          IE2,#ESPI        ;使能 SPI 中断
          SETB         EA
          MOV          SPDAT,#5AH        ;发送测试数据

          JMP          $

          END
```

12.5.21 I2C 中断

C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void I2C_Isr() interrupt 24
```

```
{
    if (I2CMSST & 0x40)
    {
        I2CMSST &= ~0x40;           //清中断标志
        P10 = !P10;                 //测试端口
    }
}

void main()
{
    P_SW2 /= 0x80;                  //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    I2CCFG = 0xc0;                  //使能 I2C 主机模式
    I2CMSCR = 0x80;                 //使能 I2C 中断;
    EA = 1;

    I2CMSCR = 0x81;                 //发送起始命令

    while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

|          |       |        |
|----------|-------|--------|
| P_SW2    | DATA  | 0BAH   |
| I2CCFG   | XDATA | 0FE80H |
| I2CMSCR  | XDATA | 0FE81H |
| I2CMSST  | XDATA | 0FE82H |
| I2CSLCR  | XDATA | 0FE83H |
| I2CSLST  | XDATA | 0FE84H |
| I2CSLADR | XDATA | 0FE85H |
| I2CTXD   | XDATA | 0FE86H |
| I2CRXD   | XDATA | 0FE87H |
| P1M1     | DATA  | 091H   |
| P1M0     | DATA  | 092H   |
| P0M1     | DATA  | 093H   |
| P0M0     | DATA  | 094H   |
| P2M1     | DATA  | 095H   |
| P2M0     | DATA  | 096H   |
| P3M1     | DATA  | 0B1H   |
| P3M0     | DATA  | 0B2H   |
| P4M1     | DATA  | 0B3H   |
| P4M0     | DATA  | 0B4H   |

```

P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          00C3H
          LJMP         I2CISR

          ORG          0100H
I2CISR:
          PUSH         ACC
          PUSH         DPL
          PUSH         DPH
          MOV          DPTR,#I2CMSST
          MOVX         A,@DPTR
          ANL          A,#NOT 40H          ;清中断标志
          MOVX         @DPTR,A
          CPL          P1.0              ;测试端口
          POP          DPH
          POP          DPL
          POP          ACC
          RETI

MAIN:
          MOV          SP,#5FH
          ORL          P_SW2,#80H        ;使能访问 XFR

          MOV          P0M0,#00H
          MOV          P0M1,#00H
          MOV          P1M0,#00H
          MOV          P1M1,#00H
          MOV          P2M0,#00H
          MOV          P2M1,#00H
          MOV          P3M0,#00H
          MOV          P3M1,#00H
          MOV          P4M0,#00H
          MOV          P4M1,#00H
          MOV          P5M0,#00H
          MOV          P5M1,#00H

          MOV          A,#0C0H          ;使能 I2C 主机模式
          MOV          DPTR,#I2CCFG
          MOVX         @DPTR,A
          MOV          A,#80H          ;使能 I2C 中断
          MOV          DPTR,#I2CMSCR
          MOVX         @DPTR,A
          SETB         EA

          MOV          A,#081H          ;发送起始命令
          MOV          DPTR,#I2CMSCR
          MOVX         @DPTR,A

          JMP          $

          END

```

# 13 所有的 I/O 口均可中断（4 种模式），不是传统外部中断

| 产品线                    | I/O 中断 | I/O 中断优先级 | I/O 中断唤醒功能 |
|------------------------|--------|-----------|------------|
| STC8H1K08 系列           |        |           |            |
| STC8H1K28 系列           |        |           |            |
| STC8H3K64S2 系列 A 版本    | ●      | 1 级       |            |
| STC8H3K64S4 系列 A 版本    | ●      | 1 级       |            |
| STC8H3K64S2 系列 B 版本    | ●      | 1 级       | ●          |
| STC8H3K64S4 系列 B 版本    | ●      | 1 级       | ●          |
| STC8H8K64U 系列 A 版本     |        |           |            |
| STC8H8K64U 系列 B/C/D 版本 | ●      | 4 级       | ●          |
| STC8H4K64TL 系列         | ●      | 4 级       | ●          |
| STC8H4K64TLCD 系列       | ●      | 4 级       | ●          |
| STC8H1K08T 系列          | ●      | 4 级       | ●          |

STC8H 部分系列支持所有的 I/O 中断，且支持 4 种中断模式：下降沿中断、上升沿中断、低电平中断、高电平中断。每组 I/O 口都有独立的中断入口地址，且每个 I/O 可独立设置中断模式。

注：经测试发现目前 STC8H 的 I/O 口中断在设计上有一些小瑕疵，请参考下表进行使用：（OK：可正常使用；NG：表示可能会问题）

|      |       | 芯片工作模式 |         |         |
|------|-------|--------|---------|---------|
|      |       | 正常模式   | IDLE 模式 | STOP 模式 |
| 中断模式 | 上升沿中断 | NG     | NG      | OK      |
|      | 下降沿中断 | NG     | NG      | OK      |
|      | 高电平中断 | OK     | OK      | OK      |
|      | 低电平中断 | OK     | OK      | OK      |

如下 STC8H 系列针对上面的 I/O 口中断瑕疵已修改正确：

STC8H4K64TL 系列、STC8H8K64U-D 系列、STC8H4K64TLCD-B 系列、STC8H1K08T 系列

## 13.1 I/O 口中断相关寄存器

| 符号     | 描述          | 地址    | 位地址与符号  |         |         |         |         |         |         |         | 复位值       |
|--------|-------------|-------|---------|---------|---------|---------|---------|---------|---------|---------|-----------|
|        |             |       | B7      | B6      | B5      | B4      | B3      | B2      | B1      | B0      |           |
| P0INTE | P0 口中断使能寄存器 | FD00H | P07INTE | P06INTE | P05INTE | P04INTE | P03INTE | P02INTE | P01INTE | P00INTE | 0000,0000 |
| P1INTE | P1 口中断使能寄存器 | FD01H | P17INTE | P16INTE | P15INTE | P14INTE | P13INTE | P12INTE | P11INTE | P10INTE | 0000,0000 |
| P2INTE | P2 口中断使能寄存器 | FD02H | P27INTE | P26INTE | P25INTE | P24INTE | P23INTE | P22INTE | P21INTE | P20INTE | 0000,0000 |
| P3INTE | P3 口中断使能寄存器 | FD03H | P37INTE | P36INTE | P35INTE | P34INTE | P33INTE | P32INTE | P31INTE | P30INTE | 0000,0000 |

|        |                |       |         |         |         |         |         |         |         |         |           |
|--------|----------------|-------|---------|---------|---------|---------|---------|---------|---------|---------|-----------|
| P4INTE | P4 口中断使能寄存器    | FD04H | P47INTE | P46INTE | P45INTE | P44INTE | P43INTE | P42INTE | P41INTE | P40INTE | 0000,0000 |
| P5INTE | P5 口中断使能寄存器    | FD05H | -       | -       | P55INTE | P54INTE | P53INTE | P52INTE | P51INTE | P50INTE | xx00,0000 |
| P6INTE | P6 口中断使能寄存器    | FD06H | P67INTE | P66INTE | P65INTE | P64INTE | P63INTE | P62INTE | P61INTE | P60INTE | 0000,0000 |
| P7INTE | P7 口中断使能寄存器    | FD07H | P77INTE | P76INTE | P75INTE | P74INTE | P73INTE | P72INTE | P71INTE | P70INTE | 0000,0000 |
| P0INTF | P0 口中断标志寄存器    | FD10H | P07INTF | P06INTF | P05INTF | P04INTF | P03INTF | P02INTF | P01INTF | P00INTF | 0000,0000 |
| P1INTF | P1 口中断标志寄存器    | FD11H | P17INTF | P16INTF | P15INTF | P14INTF | P13INTF | P12INTF | P11INTF | P10INTF | 0000,0000 |
| P2INTF | P2 口中断标志寄存器    | FD12H | P27INTF | P26INTF | P25INTF | P24INTF | P23INTF | P22INTF | P21INTF | P20INTF | 0000,0000 |
| P3INTF | P3 口中断标志寄存器    | FD13H | P37INTF | P36INTF | P35INTF | P34INTF | P33INTF | P32INTF | P31INTF | P30INTF | 0000,0000 |
| P4INTF | P4 口中断标志寄存器    | FD14H | P47INTF | P46INTF | P45INTF | P44INTF | P43INTF | P42INTF | P41INTF | P40INTF | 0000,0000 |
| P5INTF | P5 口中断标志寄存器    | FD15H | -       | -       | P55INTF | P54INTF | P53INTF | P52INTF | P51INTF | P50INTF | xx00,0000 |
| P6INTF | P6 口中断标志寄存器    | FD16H | P67INTF | P66INTF | P65INTF | P64INTF | P63INTF | P62INTF | P61INTF | P60INTF | 0000,0000 |
| P7INTF | P7 口中断标志寄存器    | FD17H | P77INTF | P76INTF | P75INTF | P74INTF | P73INTF | P72INTF | P71INTF | P70INTF | 0000,0000 |
| P0IM0  | P0 口中断模式寄存器 0  | FD20H | P07IM0  | P06IM0  | P05IM0  | P04IM0  | P03IM0  | P02IM0  | P01IM0  | P00IM0  | 0000,0000 |
| P1IM0  | P1 口中断模式寄存器 0  | FD21H | P17IM0  | P16IM0  | P15IM0  | P14IM0  | P13IM0  | P12IM0  | P11IM0  | P10IM0  | 0000,0000 |
| P2IM0  | P2 口中断模式寄存器 0  | FD22H | P27IM0  | P26IM0  | P25IM0  | P24IM0  | P23IM0  | P22IM0  | P21IM0  | P20IM0  | 0000,0000 |
| P3IM0  | P3 口中断模式寄存器 0  | FD23H | P37IM0  | P36IM0  | P35IM0  | P34IM0  | P33IM0  | P32IM0  | P31IM0  | P30IM0  | 0000,0000 |
| P4IM0  | P4 口中断模式寄存器 0  | FD24H | P47IM0  | P46IM0  | P45IM0  | P44IM0  | P43IM0  | P42IM0  | P41IM0  | P40IM0  | 0000,0000 |
| P5IM0  | P5 口中断模式寄存器 0  | FD25H | -       | -       | P55IM0  | P54IM0  | P53IM0  | P52IM0  | P51IM0  | P50IM0  | xx00,0000 |
| P6IM0  | P6 口中断模式寄存器 0  | FD26H | P67IM0  | P66IM0  | P65IM0  | P64IM0  | P63IM0  | P62IM0  | P61IM0  | P60IM0  | 0000,0000 |
| P7IM0  | P7 口中断模式寄存器 0  | FD27H | P77IM0  | P76IM0  | P75IM0  | P74IM0  | P73IM0  | P72IM0  | P71IM0  | P70IM0  | 0000,0000 |
| P0IM1  | P0 口中断模式寄存器 1  | FD30H | P07IM1  | P06IM1  | P05IM1  | P04IM1  | P03IM1  | P02IM1  | P01IM1  | P00IM1  | 0000,0000 |
| P1IM1  | P1 口中断模式寄存器 1  | FD31H | P17IM1  | P16IM1  | P15IM1  | P14IM1  | P13IM1  | P12IM1  | P11IM1  | P10IM1  | 0000,0000 |
| P2IM1  | P2 口中断模式寄存器 1  | FD32H | P27IM1  | P26IM1  | P25IM1  | P24IM1  | P23IM1  | P22IM1  | P21IM1  | P20IM1  | 0000,0000 |
| P3IM1  | P3 口中断模式寄存器 1  | FD33H | P37IM1  | P36IM1  | P35IM1  | P34IM1  | P33IM1  | P32IM1  | P31IM1  | P30IM1  | 0000,0000 |
| P4IM1  | P4 口中断模式寄存器 1  | FD34H | P47IM1  | P46IM1  | P45IM1  | P44IM1  | P43IM1  | P42IM1  | P41IM1  | P40IM1  | 0000,0000 |
| P5IM1  | P5 口中断模式寄存器 1  | FD35H | -       | -       | P55IM1  | P54IM1  | P53IM1  | P52IM1  | P51IM1  | P50IM1  | xx00,0000 |
| P6IM1  | P6 口中断模式寄存器 1  | FD36H | P67IM1  | P66IM1  | P65IM1  | P64IM1  | P63IM1  | P62IM1  | P61IM1  | P60IM1  | 0000,0000 |
| P7IM1  | P7 口中断模式寄存器 1  | FD37H | P77IM1  | P76IM1  | P75IM1  | P74IM1  | P73IM1  | P72IM1  | P71IM1  | P70IM1  | 0000,0000 |
| PINIPL | I/O 口中断优先级低寄存器 | FD60H | P7IP    | P6IP    | P5IP    | P4IP    | P3IP    | P2IP    | P1IP    | P0IP    | 0000,0000 |
| PINIPH | I/O 口中断优先级高寄存器 | FD61H | P7IPH   | P6IPH   | P5IPH   | P4IPH   | P3IPH   | P2IPH   | P1IPH   | P0IPH   | 0000,0000 |
| P0WKUE | P0 口中断唤醒使能寄存器  | FD40H | P07WKUE | P06WKUE | P05WKUE | P04WKUE | P03WKUE | P02WKUE | P01WKUE | P00WKUE | 0000,0000 |
| P1WKUE | P1 口中断唤醒使能寄存器  | FD41H | P17WKUE | P16WKUE | P15WKUE | P14WKUE | P13WKUE | P12WKUE | P11WKUE | P10WKUE | 0000,0000 |
| P2WKUE | P2 口中断唤醒使能寄存器  | FD42H | P27WKUE | P26WKUE | P25WKUE | P24WKUE | P23WKUE | P22WKUE | P21WKUE | P20WKUE | 0000,0000 |
| P3WKUE | P3 口中断唤醒使能寄存器  | FD43H | P37WKUE | P36WKUE | P35WKUE | P34WKUE | P33WKUE | P32WKUE | P31WKUE | P30WKUE | 0000,0000 |
| P4WKUE | P4 口中断唤醒使能寄存器  | FD44H | P47WKUE | P46WKUE | P45WKUE | P44WKUE | P43WKUE | P42WKUE | P41WKUE | P40WKUE | 0000,0000 |
| P5WKUE | P5 口中断唤醒使能寄存器  | FD45H | -       | -       | P55WKUE | P54WKUE | P53WKUE | P52WKUE | P51WKUE | P50WKUE | xx00,0000 |
| P6WKUE | P6 口中断唤醒使能寄存器  | FD46H | P67WKUE | P66WKUE | P65WKUE | P64WKUE | P63WKUE | P62WKUE | P61WKUE | P60WKUE | 0000,0000 |
| P7WKUE | P7 口中断唤醒使能寄存器  | FD47H | P77WKUE | P76WKUE | P75WKUE | P74WKUE | P73WKUE | P72WKUE | P71WKUE | P70WKUE | 0000,0000 |

### 13.1.1 端口中断使能寄存器 (PxINTE)

| 符号     | 地址    | B7      | B6      | B5      | B4      | B3      | B2      | B1      | B0      |
|--------|-------|---------|---------|---------|---------|---------|---------|---------|---------|
| P0INTE | FD00H | P07INTE | P06INTE | P05INTE | P04INTE | P03INTE | P02INTE | P01INTE | P00INTE |

|        |       |         |         |         |         |         |         |         |         |
|--------|-------|---------|---------|---------|---------|---------|---------|---------|---------|
| P1INTE | FD01H | P17INTE | P16INTE | P15INTE | P14INTE | P13INTE | P12INTE | P11INTE | P10INTE |
| P2INTE | FD02H | P27INTE | P26INTE | P25INTE | P24INTE | P23INTE | P22INTE | P21INTE | P20INTE |
| P3INTE | FD03H | P37INTE | P36INTE | P35INTE | P34INTE | P33INTE | P32INTE | P31INTE | P30INTE |
| P4INTE | FD04H | P47INTE | P46INTE | P45INTE | P44INTE | P43INTE | P42INTE | P41INTE | P40INTE |
| P5INTE | FD05H | -       | -       | P55INTE | P54INTE | P53INTE | P52INTE | P51INTE | P50INTE |
| P6INTE | FD06H | P67INTE | P66INTE | P65INTE | P64INTE | P63INTE | P62INTE | P61INTE | P60INTE |
| P7INTE | FD07H | P77INTE | P76INTE | P75INTE | P74INTE | P73INTE | P72INTE | P71INTE | P70INTE |

PnINTE.x: 端口中断使能控制位 (n=0~7, x=0~7)

0: 关闭 Pn.x 口中断功能

1: 使能 Pn.x 口中断功能

### 13.1.2 端口中断标志寄存器 (PxINTF)

| 符号     | 地址    | B7      | B6      | B5      | B4      | B3      | B2      | B1      | B0      |
|--------|-------|---------|---------|---------|---------|---------|---------|---------|---------|
| P0INTF | FD10H | P07INTF | P06INTF | P05INTF | P04INTF | P03INTF | P02INTF | P01INTF | P00INTF |
| P1INTF | FD11H | P17INTF | P16INTF | P15INTF | P14INTF | P13INTF | P12INTF | P11INTF | P10INTF |
| P2INTF | FD12H | P27INTF | P26INTF | P25INTF | P24INTF | P23INTF | P22INTF | P21INTF | P20INTF |
| P3INTF | FD13H | P37INTF | P36INTF | P35INTF | P34INTF | P33INTF | P32INTF | P31INTF | P30INTF |
| P4INTF | FD14H | P47INTF | P46INTF | P45INTF | P44INTF | P43INTF | P42INTF | P41INTF | P40INTF |
| P5INTF | FD15H | -       | -       | P55INTF | P54INTF | P53INTF | P52INTF | P51INTF | P50INTF |
| P6INTF | FD16H | P67INTF | P66INTF | P65INTF | P64INTF | P63INTF | P62INTF | P61INTF | P60INTF |
| P7INTF | FD17H | P77INTF | P76INTF | P75INTF | P74INTF | P73INTF | P72INTF | P71INTF | P70INTF |

PnINTF.x: 端口中断请求标志位 (n=0~7, x=0~7)

0: Pn.x 口没有中断请求

1: Pn.x 口有中断请求, 若使能中断, 则会进入中断服务程序。标志位需软件清 0。

### 13.1.3 端口中断模式配置寄存器 (PxIM0, PxIM1)

| 符号    | 地址    | B7     | B6     | B5     | B4     | B3     | B2     | B1     | B0     |
|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|
| P0IM0 | FD20H | P07IM0 | P06IM0 | P05IM0 | P04IM0 | P03IM0 | P02IM0 | P01IM0 | P00IM0 |
| P0IM1 | FD30H | P07IM1 | P06IM1 | P05IM1 | P04IM1 | P03IM1 | P02IM1 | P01IM1 | P00IM1 |
| P1IM0 | FD21H | P17IM0 | P16IM0 | P15IM0 | P14IM0 | P13IM0 | P12IM0 | P11IM0 | P10IM0 |
| P1IM1 | FD31H | P17IM1 | P16IM1 | P15IM1 | P14IM1 | P13IM1 | P12IM1 | P11IM1 | P10IM1 |
| P2IM0 | FD22H | P27IM0 | P26IM0 | P25IM0 | P24IM0 | P23IM0 | P22IM0 | P21IM0 | P20IM0 |
| P2IM1 | FD32H | P27IM1 | P26IM1 | P25IM1 | P24IM1 | P23IM1 | P22IM1 | P21IM1 | P20IM1 |
| P3IM0 | FD23H | P37IM0 | P36IM0 | P35IM0 | P34IM0 | P33IM0 | P32IM0 | P31IM0 | P30IM0 |
| P3IM1 | FD33H | P37IM1 | P36IM1 | P35IM1 | P34IM1 | P33IM1 | P32IM1 | P31IM1 | P30IM1 |
| P4IM0 | FD24H | P47IM0 | P46IM0 | P45IM0 | P44IM0 | P43IM0 | P42IM0 | P41IM0 | P40IM0 |
| P4IM1 | FD34H | P47IM1 | P46IM1 | P45IM1 | P44IM1 | P43IM1 | P42IM1 | P41IM1 | P40IM1 |
| P5IM0 | FD25H | -      | -      | P55IM0 | P54IM0 | P53IM0 | P52IM0 | P51IM0 | P50IM0 |
| P5IM1 | FD35H | -      | -      | P55IM1 | P54IM1 | P53IM1 | P52IM1 | P51IM1 | P50IM1 |
| P6IM0 | FD26H | P67IM0 | P66IM0 | P65IM0 | P64IM0 | P63IM0 | P62IM0 | P61IM0 | P60IM0 |
| P6IM1 | FD36H | P67IM1 | P66IM1 | P65IM1 | P64IM1 | P63IM1 | P62IM1 | P61IM1 | P60IM1 |
| P7IM0 | FD27H | P77IM0 | P76IM0 | P75IM0 | P74IM0 | P73IM0 | P72IM0 | P71IM0 | P70IM0 |



|       |       |        |        |        |        |        |        |        |        |
|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|
| P7IM1 | FD37H | P77IM1 | P76IM1 | P75IM1 | P74IM1 | P73IM1 | P72IM1 | P71IM1 | P70IM1 |
|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|

配置端口的模式

| PnIM1.x | PnIM0.x | Pn.x 口中断模式 |
|---------|---------|------------|
| 0       | 0       | 下降沿中断      |
| 0       | 1       | 上升沿中断      |
| 1       | 0       | 低电平中断      |
| 1       | 1       | 高电平中断      |

### 13.1.4 端口中断优先级控制寄存器 (PINIPL, PINIPH)

| 符号     | 地址    | B7    | B6    | B5    | B4    | B3    | B2    | B1    | B0    |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| PINIPL | FD60H | P7IP  | P6IP  | P5IP  | P4IP  | P3IP  | P2IP  | P1IP  | P0IP  |
| PINIPH | FD61H | P7IPH | P6IPH | P5IPH | P4IPH | P3IPH | P2IPH | P1IPH | P0IPH |

PxIPH, PxIP: Px口中断优先级控制位

00: Px 口中断优先级为 0 级 (最低级)

01: Px 口中断优先级为 1 级 (较低级)

10: Px 口中断优先级为 2 级 (较高级)

11: Px 口中断优先级为 3 级 (最高级)

### 13.1.5 端口中断掉电唤醒使能寄存器 (PxWKUE)

| 符号    | 地址   | B7     | B6     | B5     | B4     | B3     | B2     | B1     | B0     |
|-------|------|--------|--------|--------|--------|--------|--------|--------|--------|
| P0WKU | FD40 | P07WKU | P06WKU | P05WKU | P04WKU | P03WKU | P02WKU | P01WKU | P00WKU |
| E     | H    | E      | E      | E      | E      | E      | E      | E      | E      |
| P1WKU | FD41 | P17WKU | P16WKU | P15WKU | P14WKU | P13WKU | P12WKU | P11WKU | P10WKU |
| E     | H    | E      | E      | E      | E      | E      | E      | E      | E      |
| P2WKU | FD42 | P27WKU | P26WKU | P25WKU | P24WKU | P23WKU | P22WKU | P21WKU | P20WKU |
| E     | H    | E      | E      | E      | E      | E      | E      | E      | E      |
| P3WKU | FD43 | P37WKU | P36WKU | P35WKU | P34WKU | P33WKU | P32WKU | P31WKU | P30WKU |
| E     | H    | E      | E      | E      | E      | E      | E      | E      | E      |
| P4WKU | FD44 | P47WKU | P46WKU | P45WKU | P44WKU | P43WKU | P42WKU | P41WKU | P40WKU |
| E     | H    | E      | E      | E      | E      | E      | E      | E      | E      |
| P5WKU | FD45 | -      | -      | P55WKU | P54WKU | P53WKU | P52WKU | P51WKU | P50WKU |
| E     | H    |        |        | E      | E      | E      | E      | E      | E      |
| P6WKU | FD46 | P67WKU | P66WKU | P65WKU | P64WKU | P63WKU | P62WKU | P61WKU | P60WKU |
| E     | H    | E      | E      | E      | E      | E      | E      | E      | E      |
| P7WKU | FD47 | P77WKU | P76WKU | P75WKU | P74WKU | P73WKU | P72WKU | P71WKU | P70WKU |
| E     | H    | E      | E      | E      | E      | E      | E      | E      | E      |

PnxWKUE: 端口中断掉电唤醒使能控制位 (n=0~7, x=0~7)

0: 关闭 Pn.x 口中断掉电唤醒功能

1: 使能 Pn.x 口中断掉电唤醒功能

## 13.2 范例程序

### 13.2.1 P0 口下降沿中断

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void main()
```

```
{
    P_SW2 /= 0x80;                //使能访问 XFR
```

```
    P0M0 = 0x00;
```

```
    P0M1 = 0x00;
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P2M0 = 0x00;
```

```
    P2M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P4M0 = 0x00;
```

```
    P4M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
    P0IM0 = 0x00;
```

```
    //下降沿中断
```

```
    P0IM1 = 0x00;
```

```
    P0INTE = 0xff;
```

```
    //使能 P0 口中断
```

```
    EA = 1;
```

```
    while (1);
```

```
}
```

//由于中断向量大于 31, 在 KEIL 中无法直接编译

//必须借用第 13 号中断入口地址

```
void common_isr() interrupt 13
```

```
{
```

```
    unsigned char psw2_st;
```

```
    unsigned char intf;
```

```
    intf = P0INTF;
```

```
    if (intf)
```

```
    {
```

```
        P0INTF = 0x00;
```

```
        if (intf & 0x01)
```

```
        {
```

```
            //P0.0 口中断
```

```
        }
```

```
        if (intf & 0x02)
```

```
        {
```

```
            //P0.1 口中断
```

```
        }
```

```
    }
```

```
        if (intf & 0x04)
        {
                                                    //P0.2 口中断
        }
        if (intf & 0x08)
        {
                                                    //P0.3 口中断
        }
        if (intf & 0x10)
        {
                                                    //P0.4 口中断
        }
        if (intf & 0x20)
        {
                                                    //P0.5 口中断
        }
        if (intf & 0x40)
        {
                                                    //P0.6 口中断
        }
        if (intf & 0x80)
        {
                                                    //P0.7 口中断
        }
    }
}
```

```
// ISR.ASM
//将下面的代码保存为ISR.ASM，然后将文件加入到项目中即可

        CSEG                AT 012BH                ;P0 口中断入口地址
        JMP                P0INT_ISR

P0INT_ISR:
        JMP                006BH                ;借用 13 号中断的入口地址
        END
```

汇编代码

;测试工作频率为 11.0592MHz

```
P0M0      DATA      094H
P0M1      DATA      093H
P1M0      DATA      092H
P1M1      DATA      091H
P2M0      DATA      096H
P2M1      DATA      095H
P3M0      DATA      0B2H
P3M1      DATA      0B1H
P4M0      DATA      0B4H
P4M1      DATA      0B3H
P5M0      DATA      0CAH
P5M1      DATA      0C9H
P6M0      DATA      0CCH
P6M1      DATA      0CBH
P7M0      DATA      0E2H
P7M1      DATA      0E1H
```

```

P_SW2      DATA      0BAH

P0INTE     XDATA      0FD00H
P0INTF     XDATA      0FD10H
P0IM0      XDATA      0FD20H
P0IM1      XDATA      0FD30H

                ORG      0000H
                LJMP     MAIN

                ORG      012BH                      ;P0 口中断入口地址
P0INT_ISR:
                PUSH     ACC
                PUSH     B
                PUSH     DPL
                PUSH     DPH

                MOV      DPTR,#P0INTF
                MOVX     A,@DPTR
                MOV      B,A
                CLR      A
                MOVX     @DPTR,A
                MOV      A,B

CHECKP00:
                JNB      ACC.0,CHECKP01
                NOP

CHECKP01:
                JNB      ACC.1,CHECKP02
                NOP                      ;P0.0 口中断

CHECKP02:
                JNB      ACC.2,CHECKP03
                NOP                      ;P0.1 口中断

CHECKP03:
                JNB      ACC.3,CHECKP04
                NOP                      ;P0.2 口中断

CHECKP04:
                JNB      ACC.4,CHECKP05
                NOP                      ;P0.3 口中断

CHECKP05:
                JNB      ACC.5,CHECKP06
                NOP                      ;P0.4 口中断

CHECKP06:
                JNB      ACC.6,CHECKP07
                NOP                      ;P0.5 口中断

CHECKP07:
                JNB      ACC.7,P0ISREXIT
                NOP                      ;P0.6 口中断

P0ISREXIT:
                POP      DPH
                POP      DPL
                POP      B
                POP      ACC
                RETI

                ORG      0200H
MAIN:
                MOV      SP,#5FH
                ORL      P_SW2,#80H          ;使能访问XFR

```

```

MOV      P0M0,#00H
MOV      P0M1,#00H
MOV      P1M0,#00H
MOV      P1M1,#00H
MOV      P2M0,#00H
MOV      P2M1,#00H
MOV      P3M0,#00H
MOV      P3M1,#00H

CLR      A
MOV      DPTR,# P0IM0      ;下降沿中断
MOVX     @DPTR,A
MOV      DPTR,# P0IMI
MOVX     @DPTR,A
MOV      DPTR,# P0INTE
MOV      A,#0FFH
MOVX     @DPTR,A      ;使能 P0 口中断

SETB     EA

JMP      $

END

```

### 13.2.2 P1 口上升沿中断

#### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```
void main()
```

```

{
    P_SW2 /= 0x80;      //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P1IM0 = 0xff;      //上升沿中断
    P1IM1 = 0x00;
    P1INTE = 0xff;      //使能 P1 口中断
}

```

```
EA = 1;

while (1);
}

// 由于中断向量大于 31, 在 KEIL 中无法直接编译
// 必须借用第 13 号中断入口地址
void common_isr() interrupt 13
{
    unsigned char intf;

    intf = PIINTF;
    if (intf)
    {
        PIINTF = 0x00;
        if (intf & 0x01)
        {
            //P1.0 口中断
        }
        if (intf & 0x02)
        {
            //P1.1 口中断
        }
        if (intf & 0x04)
        {
            //P1.2 口中断
        }
        if (intf & 0x08)
        {
            //P1.3 口中断
        }
        if (intf & 0x10)
        {
            //P1.4 口中断
        }
        if (intf & 0x20)
        {
            //P1.5 口中断
        }
        if (intf & 0x40)
        {
            //P1.6 口中断
        }
        if (intf & 0x80)
        {
            //P1.7 口中断
        }
    }
}
```

// ISR.ASM

// 将下面的代码保存为 ISR.ASM, 然后将文件加入到项目中即可

```
                CSEG          AT 0133H          ;P1 口中断入口地址
                JMP           PIINT_ISR

PIINT_ISR:
                JMP           006BH              ;借用 13 号中断的入口地址
                END
```

## 汇编代码

;测试工作频率为 11.0592MHz

```

P0M0      DATA      094H
P0M1      DATA      093H
P1M0      DATA      092H
P1M1      DATA      091H
P2M0      DATA      096H
P2M1      DATA      095H
P3M0      DATA      0B2H
P3M1      DATA      0B1H
P4M0      DATA      0B4H
P4M1      DATA      0B3H
P5M0      DATA      0CAH
P5M1      DATA      0C9H
P6M0      DATA      0CCH
P6M1      DATA      0CBH
P7M0      DATA      0E2H
P7M1      DATA      0E1H

P_SW2     DATA      0BAH

PIINTE     XDATA      0FD01H
PIINTF     XDATA      0FD11H
PIIM0      XDATA      0FD21H
PIIM1      XDATA      0FD31H

ORG        0000H
LJMP       MAIN

PIINT_ISR: ORG        0133H      ;P1 口中断入口地址

            PUSH      ACC
            PUSH      B
            PUSH      DPL
            PUSH      DPH

            MOV        DPTR,#PIINTF
            MOVX       A,@DPTR
            MOV        B,A
            CLR        A
            MOVX       @DPTR,A
            MOV        A,B

CHECKP10:   JNB        ACC.0,CHECKP11
            NOP

CHECKP11:   JNB        ACC.1,CHECKP12
            NOP      ;P1.0 口中断

CHECKP12:   JNB        ACC.2,CHECKP13
            NOP      ;P1.1 口中断

CHECKP13:   JNB        ACC.3,CHECKP14
            NOP      ;P1.2 口中断

CHECKP14:   JNB        ACC.3,CHECKP14
            NOP      ;P1.3 口中断

```

```

        JNB      ACC.4,CHECKP15
        NOP
CHECKP15:
        JNB      ACC.5,CHECKP16
        NOP
CHECKP16:
        JNB      ACC.6,CHECKP17
        NOP
CHECKP17:
        JNB      ACC.7,PIISREXIT
        NOP
PIISREXIT:
        POP      DPH
        POP      DPL
        POP      B
        POP      ACC
        RETI

MAIN:   ORG      0200H

        MOV      SP, #5FH
        ORL      P_SW2, #80H           ;使能访问 XFR

        MOV      P0M0, #00H
        MOV      P0M1, #00H
        MOV      P1M0, #00H
        MOV      P1M1, #00H
        MOV      P2M0, #00H
        MOV      P2M1, #00H
        MOV      P3M0, #00H
        MOV      P3M1, #00H

        CLR      A
        MOV      DPTR, #PIIM0         ;下降沿中断
        MOVX     @DPTR, A
        MOV      DPTR, #PIIM1
        MOVX     @DPTR, A
        MOV      DPTR, #PIINTE
        MOV      A, #0FFH
        MOVX     @DPTR, A             ;使能 P1 口中断

        SETB     EA

        JMP      $

END

```

### 13.2.3 P2 口低电平中断

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
```



```
#include "intrins.h"
```

```
void main()
```

```
{  
    P_SW2 /= 0x80;                                //使能访问 XFR  
  
    P0M0 = 0x00;  
    P0M1 = 0x00;  
    P1M0 = 0x00;  
    P1M1 = 0x00;  
    P2M0 = 0x00;  
    P2M1 = 0x00;  
    P3M0 = 0x00;  
    P3M1 = 0x00;  
    P4M0 = 0x00;  
    P4M1 = 0x00;  
    P5M0 = 0x00;  
    P5M1 = 0x00;  
  
    P2IM0 = 0x00;                                //低电平中断  
    P2IM1 = 0xff;  
    P2INTE = 0xff;                               //使能 P2 口中断  
  
    EA = 1;  
  
    while (1);  
}
```

//由于中断向量大于31, 在 KEIL 中无法直接编译

//必须借用第13号中断入口地址

```
void common_isr() interrupt 13
```

```
{  
    unsigned char intf;  
  
    intf = P2INTF;  
    if (intf)  
    {  
        P2INTF = 0x00;  
        if (intf & 0x01)  
        {  
                                                    //P2.0 口中断  
        }  
        if (intf & 0x02)  
        {  
                                                    //P2.1 口中断  
        }  
        if (intf & 0x04)  
        {  
                                                    //P2.2 口中断  
        }  
        if (intf & 0x08)  
        {  
                                                    //P0.3 口中断  
        }  
        if (intf & 0x10)  
        {  
                                                    //P2.4 口中断  
        }  
        if (intf & 0x20)
```

```

    {
                                                //P2.5 口中断
    }
    if (intf & 0x40)
    {
                                                //P2.6 口中断
    }
    if (intf & 0x80)
    {
                                                //P2.7 口中断
    }
}
}
```

// ISR.ASM  
//将下面的代码保存为ISR.ASM，然后将文件加入到项目中即可

```

    CSEG          AT 013BH                      ;P2 口中断入口地址
    JMP          P2INT_ISR
P2INT_ISR:
    JMP          006BH                          ;借用 13 号中断的入口地址
    END
```

汇编代码

;测试工作频率为 11.0592MHz

```

P0M0      DATA      094H
P0M1      DATA      093H
P1M0      DATA      092H
P1M1      DATA      091H
P2M0      DATA      096H
P2M1      DATA      095H
P3M0      DATA      0B2H
P3M1      DATA      0B1H
P4M0      DATA      0B4H
P4M1      DATA      0B3H
P5M0      DATA      0CAH
P5M1      DATA      0C9H
P6M0      DATA      0CCH
P6M1      DATA      0CBH
P7M0      DATA      0E2H
P7M1      DATA      0E1H

P_SW2     DATA      0BAH

P2INTE     XDATA      0FD02H
P2INTF     XDATA      0FD12H
P2IM0      XDATA      0FD22H
P2IM1      XDATA      0FD32H

    ORG          0000H
    LJMP         MAIN

    ORG          013BH                      ;P2 口中断入口地址
P2INT_ISR:
    PUSH        ACC
```

```

        PUSH    B
        PUSH    DPL
        PUSH    DPH

        MOV     DPTR,#P2INTF
        MOVX    A,@DPTR
        MOV     B,A
        CLR     A
        MOVX    @DPTR,A
        MOV     A,B
CHECKP20:
        JNB     ACC.0,CHECKP21
        NOP
CHECKP21:
        JNB     ACC.1,CHECKP22
        NOP
CHECKP22:
        JNB     ACC.2,CHECKP23
        NOP
CHECKP23:
        JNB     ACC.3,CHECKP24
        NOP
CHECKP24:
        JNB     ACC.4,CHECKP25
        NOP
CHECKP25:
        JNB     ACC.5,CHECKP26
        NOP
CHECKP26:
        JNB     ACC.6,CHECKP27
        NOP
CHECKP27:
        JNB     ACC.7,P2ISREXIT
        NOP

P2ISREXIT:
        POP     DPH
        POP     DPL
        POP     B
        POP     ACC
        RETI

MAIN:
        ORG     0200H

        MOV     SP,#5FH
        ORL     P_SW2,#80H
        ;使能访问 XFR

        MOV     P0M0,#00H
        MOV     P0M1,#00H
        MOV     P1M0,#00H
        MOV     P1M1,#00H
        MOV     P2M0,#00H
        MOV     P2M1,#00H
        MOV     P3M0,#00H
        MOV     P3M1,#00H

        CLR     A
        MOV     DPTR,# P2IM0
        ;低电平中断

```

```
MOVX    @DPTR,A
MOV     DPTR,# P2IM1
MOVX    @DPTR,A
MOV     DPTR,# P2INTE
MOV     A,#0FFH
MOVX    @DPTR,A           ;使能 P2 口中断

SETB    EA

JMP     $

END
```

## 13.2.4 P3 口高电平中断

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void main()
```

```
{
    P_SW2 /= 0x80;           //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P3IM0 = 0xff;           //高电平中断
    P3IM1 = 0xff;
    P3INTE = 0xff;         //使能 P3 口中断

    EA = 1;

    while (1);
}
```

//由于中断向量大于 31, 在 KEIL 中无法直接编译

//必须借用第 13 号中断入口地址

```
void common_isr() interrupt 13
```

```
{
    unsigned char intf;

    intf = P3INTF;
    if (intf)
```

```
{
    P3INTF = 0x00;
    if (intf & 0x01)
    {
        //P3.0 口中断
    }
    if (intf & 0x02)
    {
        //P3.1 口中断
    }
    if (intf & 0x04)
    {
        //P3.2 口中断
    }
    if (intf & 0x08)
    {
        //P3.3 口中断
    }
    if (intf & 0x10)
    {
        //P3.4 口中断
    }
    if (intf & 0x20)
    {
        //P3.5 口中断
    }
    if (intf & 0x40)
    {
        //P3.6 口中断
    }
    if (intf & 0x80)
    {
        //P3.7 口中断
    }
}
}
```

```
// ISR.ASM
//将下面的代码保存为ISR.ASM，然后将文件加入到项目中即可

                CSEG                AT 0143H                ;P3 口中断入口地址
                JMP                P3INT_ISR

P3INT_ISR:
                JMP                006BH                ;借用 13 号中断的入口地址
                END
```

汇编代码

;测试工作频率为 11.0592MHz

```
P0M0    DATA    094H
P0M1    DATA    093H
P1M0    DATA    092H
P1M1    DATA    091H
P2M0    DATA    096H
P2M1    DATA    095H
P3M0    DATA    0B2H
```

|                   |              |                        |                    |
|-------------------|--------------|------------------------|--------------------|
| <i>P3M1</i>       | <i>DATA</i>  | <i>0B1H</i>            |                    |
| <i>P4M0</i>       | <i>DATA</i>  | <i>0B4H</i>            |                    |
| <i>P4M1</i>       | <i>DATA</i>  | <i>0B3H</i>            |                    |
| <i>P5M0</i>       | <i>DATA</i>  | <i>0CAH</i>            |                    |
| <i>P5M1</i>       | <i>DATA</i>  | <i>0C9H</i>            |                    |
| <i>P6M0</i>       | <i>DATA</i>  | <i>0CCH</i>            |                    |
| <i>P6M1</i>       | <i>DATA</i>  | <i>0CBH</i>            |                    |
| <i>P7M0</i>       | <i>DATA</i>  | <i>0E2H</i>            |                    |
| <i>P7M1</i>       | <i>DATA</i>  | <i>0E1H</i>            |                    |
|                   |              |                        |                    |
| <i>P_SW2</i>      | <i>DATA</i>  | <i>0BAH</i>            |                    |
|                   |              |                        |                    |
| <i>P3INTE</i>     | <i>XDATA</i> | <i>0FD03H</i>          |                    |
| <i>P3INTF</i>     | <i>XDATA</i> | <i>0FD13H</i>          |                    |
| <i>P3IM0</i>      | <i>XDATA</i> | <i>0FD23H</i>          |                    |
| <i>P3IM1</i>      | <i>XDATA</i> | <i>0FD33H</i>          |                    |
|                   |              |                        |                    |
|                   | <i>ORG</i>   | <i>0000H</i>           |                    |
|                   | <i>LJMP</i>  | <i>MAIN</i>            |                    |
|                   |              |                        |                    |
|                   | <i>ORG</i>   | <i>0143H</i>           | <i>;P3 口中断入口地址</i> |
| <i>P3INT_ISR:</i> |              |                        |                    |
|                   | <i>PUSH</i>  | <i>ACC</i>             |                    |
|                   | <i>PUSH</i>  | <i>B</i>               |                    |
|                   | <i>PUSH</i>  | <i>DPL</i>             |                    |
|                   | <i>PUSH</i>  | <i>DPH</i>             |                    |
|                   |              |                        |                    |
|                   | <i>MOV</i>   | <i>DPTR,#P3INTF</i>    |                    |
|                   | <i>MOVX</i>  | <i>A,@DPTR</i>         |                    |
|                   | <i>MOV</i>   | <i>B,A</i>             |                    |
|                   | <i>CLR</i>   | <i>A</i>               |                    |
|                   | <i>MOVX</i>  | <i>@DPTR,A</i>         |                    |
|                   | <i>MOV</i>   | <i>A,B</i>             |                    |
| <i>CHECKP30:</i>  |              |                        |                    |
|                   | <i>JNB</i>   | <i>ACC.0,CHECKP31</i>  |                    |
|                   | <i>NOP</i>   |                        | <i>;P3.0 口中断</i>   |
| <i>CHECKP31:</i>  |              |                        |                    |
|                   | <i>JNB</i>   | <i>ACC.1,CHECKP32</i>  |                    |
|                   | <i>NOP</i>   |                        | <i>;P3.1 口中断</i>   |
| <i>CHECKP32:</i>  |              |                        |                    |
|                   | <i>JNB</i>   | <i>ACC.2,CHECKP33</i>  |                    |
|                   | <i>NOP</i>   |                        | <i>;P3.2 口中断</i>   |
| <i>CHECKP33</i>   |              |                        |                    |
|                   | <i>JNB</i>   | <i>ACC.3,CHECKP34</i>  |                    |
|                   | <i>NOP</i>   |                        | <i>;P3.3 口中断</i>   |
| <i>CHECKP34:</i>  |              |                        |                    |
|                   | <i>JNB</i>   | <i>ACC.4,CHECKP35</i>  |                    |
|                   | <i>NOP</i>   |                        | <i>;P3.4 口中断</i>   |
| <i>CHECKP35:</i>  |              |                        |                    |
|                   | <i>JNB</i>   | <i>ACC.5,CHECKP36</i>  |                    |
|                   | <i>NOP</i>   |                        | <i>;P3.5 口中断</i>   |
| <i>CHECKP36:</i>  |              |                        |                    |
|                   | <i>JNB</i>   | <i>ACC.6,CHECKP37</i>  |                    |
|                   | <i>NOP</i>   |                        | <i>;P3.6 口中断</i>   |
| <i>CHECKP37:</i>  |              |                        |                    |
|                   | <i>JNB</i>   | <i>ACC.7,P3ISREXIT</i> |                    |
|                   | <i>NOP</i>   |                        | <i>;P3.7 口中断</i>   |
| <i>P3ISREXIT:</i> |              |                        |                    |

```

    POP        DPH
    POP        DPL
    POP        B
    POP        ACC
    RETI

MAIN:
    ORG        0200H

    MOV        SP, #5FH
    ORL        P_SW2, #80H           ;使能访问 XFR

    MOV        P0M0, #00H
    MOV        P0M1, #00H
    MOV        P1M0, #00H
    MOV        P1M1, #00H
    MOV        P2M0, #00H
    MOV        P2M1, #00H
    MOV        P3M0, #00H
    MOV        P3M1, #00H

    CLR        A
    MOV        DPTR, # P3IM0         ;高电平中断
    MOVX       @DPTR, A
    MOV        DPTR, # P3IM1
    MOVX       @DPTR, A
    MOV        DPTR, # P3INTE
    MOV        A, #0FFH
    MOVX       @DPTR, A             ;使能 P3 口中断

    SETB       EA

    JMP        $

END
```

### 13.2.5 使用拓展 Keil 中断号方案的 I/O 口中断范例

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

/\*\*\*\*\*\* 功能说明 \*\*\*\*\*

本例为 STC8H4K64TLCD 系列的 I/O 口中断测试程序。  
由于 Keil 软件的的中断号只支持 0~31，而 STC8H 系列的 I/O 口中断的中断号均大于 31  
为了能正确响应 I/O 中断，必须对 Keil 的中断进行一些特殊处理  
本方案使用的是运行拓展 Keil 的中断号插件，将 Keil 支持的中断号拓展为 0~254  
编译本项目前请先运行“Keil 中断向量拓展插件\拓展 Keil 的 C 代码中断号.exe”  
测试方法:从 P0.0 口输入信号,根据 P0.0 口的中断模式产生中断,并在 P2.0 口输出反转信号

\*\*\*\*\*/

```
void P0_isr() interrupt P0INT_VECTOR
{
    char intf;

    intf = P0INTF;
    P0INTF = 0x00;

    if (intf & 0x01)                                //判断中断标志
    {
        P20 = ~P20;                                //反转测试端口
    }
}

void P1_isr() interrupt P1INT_VECTOR
{
    P1INTF = 0x00;
}

void P2_isr() interrupt P2INT_VECTOR
{
    P2INTF = 0x00;
}

void P3_isr() interrupt P3INT_VECTOR
{
    P3INTF = 0x00;
}

void P4_isr() interrupt P4INT_VECTOR
{
    P4INTF = 0x00;
}

void P5_isr() interrupt P5INT_VECTOR
{
    P5INTF = 0x00;
}

void P6_isr() interrupt P6INT_VECTOR
{
    P6INTF = 0x00;
}

void P7_isr() interrupt P7INT_VECTOR
{
    P7INTF = 0x00;
}

void delay()
{
    int i;

    for (i=0; i<100; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}
```



```
        _nop_();
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;
    P6M0 = 0x00;
    P6M1 = 0x00;
    P7M0 = 0x00;
    P7M1 = 0x00;

    P_SW2 = 0x80;

    P0IM0 &= ~0x01;           //设置 P0.0 口中断为下降沿模式
    P0IM1 &= ~0x01;           //设置 P0.0 口中断为上升沿模式
    // P0IM0 |= 0x01;         //设置 P0.0 口中断为低电平模式
    // P0IM1 &= ~0x01;         //设置 P0.0 口中断为高电平模式
    // P0IM0 |= 0x01;         //设置 P0.0 口中断为高电平模式
    // P0IM1 |= 0x01;         //使能 P0.0 口中断
    P0INTE |= 0x01;

    EA = 1;

    while (1)
    {
        P27 = ~P27;
        delay();
    }
}
```

# 14 定时器/计数器

| 产品线              | 定时器数量     |
|------------------|-----------|
| STC8H1K08 系列     | 3 (T0~T2) |
| STC8H1K28 系列     | 5 (T0~T4) |
| STC8H3K64S4 系列   | 5 (T0~T4) |
| STC8H3K64S2 系列   | 5 (T0~T4) |
| STC8H8K64U 系列    | 5 (T0~T4) |
| STC8H4K64TL 系列   | 5 (T0~T4) |
| STC8H4K64TLCD 系列 | 5 (T0~T4) |
| STC8H1K8T 系列     | 3 (T0~T2) |

STC8H 系列单片机内部设置了 5 个 16 位定时器/计数器。5 个 16 位定时器 T0、T1、T2、T3 和 T4 都具有计数方式和定时方式两种工作方式。对定时器/计数器 T0 和 T1，用它们在特殊功能寄存器 TMOD 中相对应的控制位 C/T 来选择 T0 或 T1 为定时器还是计数器。对定时器/计数器 T2，用特殊功能寄存器 AUXR 中的控制位 T2\_C/T 来选择 T2 为定时器还是计数器。对定时器/计数器 T3，用特殊功能寄存器 T4T3M 中的控制位 T3\_C/T 来选择 T3 为定时器还是计数器。对定时器/计数器 T4，用特殊功能寄存器 T4T3M 中的控制位 T4\_C/T 来选择 T4 为定时器还是计数器。定时器/计数器的核心部件是一个加法计数器，其本质是对脉冲进行计数。只是计数脉冲来源不同：如果计数脉冲来自系统时钟，则为定时方式，此时定时器/计数器每 12 个时钟或者每 1 个时钟得到一个计数脉冲，计数值加 1；如果计数脉冲来自单片机外部引脚，则为计数方式，每来一个脉冲加 1。

当定时器/计数器 T0、T1 及 T2 工作在定时模式时，特殊功能寄存器 AUXR 中的 T0x12、T1x12 和 T2x12 分别决定是系统时钟/12 还是系统时钟/1（不分频）后让 T0、T1 和 T2 进行计数。当定时器/计数器 T3 和 T4 工作在定时模式时，特殊功能寄存器 T4T3M 中的 T3x12 和 T4x12 分别决定是系统时钟/12 还是系统时钟/1（不分频）后让 T3 和 T4 进行计数。当定时器/计数器工作在计数模式时，对外部脉冲计数不分频。

定时器/计数器 0 有 4 种工作模式：模式 0（16 位自动重载模式），模式 1（16 位不可重载模式），模式 2（8 位自动重载模式），模式 3（不可屏蔽中断的 16 位自动重载模式）。定时器/计数器 1 除模式 3 外，其他工作模式与定时器/计数器 0 相同。T1 在模式 3 时无效，停止计数。**定时器 T2 的工作模式固定为 16 位自动重载模式。**T2 可以当定时器使用，也可以当串口的波特率发生器和可编程时钟输出。**定时器 3、定时器 4 与定时器 T2 一样，它们的工作模式固定为 16 位自动重载模式。**T3/T4 可以当定时器使用，也可以当串口的波特率发生器和可编程时钟输出。

## 14.1 定时器 3/4 功能脚切换

| 符号      | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0      |
|---------|-------|----|----|----|----|----|----|----|---------|
| T3T4PIN | FEACH | -  | -  | -  | -  | -  | -  | -  | T3T4SEL |

T3T4SEL：T3/T3CLKO/T4/T4CLKO 脚选择位

| T3T4SEL | T3   | T3CLKO | T4   | T4CLKO |
|---------|------|--------|------|--------|
| 0       | P0.4 | P0.5   | P0.6 | P0.7   |
| 1       | P0.0 | P0.1   | P0.2 | P0.3   |

## 14.2 定时器的相关寄存器

| 符号      | 描述             | 地址  | 位地址与符号 |        |           |        |        |        |        |        | 复位值       |
|---------|----------------|-----|--------|--------|-----------|--------|--------|--------|--------|--------|-----------|
|         |                |     | B7     | B6     | B5        | B4     | B3     | B2     | B1     | B0     |           |
| TCON    | 定时器控制寄存器       | 88H | TF1    | TR1    | TF0       | TR0    | IE1    | IT1    | IE0    | IT0    | 0000,0000 |
| TMOD    | 定时器模式寄存器       | 89H | GATE   | C/T    | M1        | M0     | GATE   | C/T    | M1     | M0     | 0000,0000 |
| TL0     | 定时器 0 低 8 位寄存器 | 8AH |        |        |           |        |        |        |        |        | 0000,0000 |
| TL1     | 定时器 1 低 8 位寄存器 | 8BH |        |        |           |        |        |        |        |        | 0000,0000 |
| TH0     | 定时器 0 高 8 位寄存器 | 8CH |        |        |           |        |        |        |        |        | 0000,0000 |
| TH1     | 定时器 1 高 8 位寄存器 | 8DH |        |        |           |        |        |        |        |        | 0000,0000 |
| AUXR    | 辅助寄存器 1        | 8EH | T0x12  | T1x12  | UART_M0x6 | T2R    | T2_C/T | T2x12  | EXTRAM | S1ST2  | 0000,0001 |
| INTCLKO | 中断与时钟输出控制寄存器   | 8FH | -      | EX4    | EX3       | EX2    | -      | T2CLKO | T1CLKO | T0CLKO | x000,x000 |
| WKTCL   | 掉电唤醒定时器低字节     | AAH |        |        |           |        |        |        |        |        | 1111,1111 |
| WKTCH   | 掉电唤醒定时器高字节     | ABH | WKTEN  |        |           |        |        |        |        |        | 0111,1111 |
| T4T3M   | 定时器 4/3 控制寄存器  | D1H | T4R    | T4_C/T | T4x12     | T4CLKO | T3R    | T3_C/T | T3x12  | T3CLKO | 0000,0000 |
| T4H     | 定时器 4 高字节      | D2H |        |        |           |        |        |        |        |        | 0000,0000 |
| T4L     | 定时器 4 低字节      | D3H |        |        |           |        |        |        |        |        | 0000,0000 |
| T3H     | 定时器 3 高字节      | D4H |        |        |           |        |        |        |        |        | 0000,0000 |
| T3L     | 定时器 3 低字节      | D5H |        |        |           |        |        |        |        |        | 0000,0000 |
| T2H     | 定时器 2 高字节      | D6H |        |        |           |        |        |        |        |        | 0000,0000 |
| T2L     | 定时器 2 低字节      | D7H |        |        |           |        |        |        |        |        | 0000,0000 |

| 符号    | 描述             | 地址    | 位地址与符号 |    |    |    |    |    |    |    | 复位值       |
|-------|----------------|-------|--------|----|----|----|----|----|----|----|-----------|
|       |                |       | B7     | B6 | B5 | B4 | B3 | B2 | B1 | B0 |           |
| TM2PS | 定时器 2 时钟预分频寄存器 | FEA2H |        |    |    |    |    |    |    |    | 0000,0000 |
| TM3PS | 定时器 3 时钟预分频寄存器 | FEA3H |        |    |    |    |    |    |    |    | 0000,0000 |
| TM4PS | 定时器 4 时钟预分频寄存器 | FEA4H |        |    |    |    |    |    |    |    | 0000,0000 |

## 14.3 定时器 0/1

### 14.3.1 定时器 0/1 控制寄存器 (TCON)

| 符号   | 地址  | B7  | B6  | B5  | B4  | B3  | B2  | B1  | B0  |
|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| TCON | 88H | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

TF1: T1溢出中断标志。T1被允许计数以后, 从初值开始加1计数。当产生溢出时由硬件将TF1位置“1”, 并向CPU请求中断, 一直保持到CPU响应中断时, 才由硬件清“0”(也可由查询软件清“0”)。

TR1: 定时器T1的运行控制位。该位由软件置位和清零。当GATE (TMOD.7)=0, TR1=1时就允许T1开始计数, TR1=0时禁止T1计数。当GATE (TMOD.7)=1, TR1=1且INT1输入高电平时, 才允许T1计数。

TF0: T0溢出中断标志。T0被允许计数以后, 从初值开始加1计数, 当产生溢出时, 由硬件置“1”TF0, 向CPU请求中断, 一直保持CPU响应该中断时, 才由硬件清0(也可由查询软件清0)。

TR0: 定时器T0的运行控制位。该位由软件置位和清零。当GATE (TMOD.3)=0, TR0=1时就允许T0开始计数, TR0=0时禁止T0计数。当GATE (TMOD.3)=1, TR0=1且INT0输入高电平时, 才允许T0计数, TR0=0时禁止T0计数。

IE1: 外部中断1请求源 (INT1/P3.3) 标志。IE1=1, 外部中断向CPU请求中断, 当CPU响应该中断时由硬件清“0”IE1。

IT1: 外部中断源1触发控制位。IT1=0, 上升沿或下降沿均可触发外部中断1。IT1=1, 外部中断1程控为下降沿触发方式。

IE0: 外部中断0请求源 (INT0/P3.2) 标志。IE0=1外部中断0向CPU请求中断, 当CPU响应外部中断时, 由硬件清“0”IE0(边沿触发方式)。

IT0: 外部中断源0触发控制位。IT0=0, 上升沿或下降沿均可触发外部中断0。IT0=1, 外部中断0程控为下降沿触发方式。

### 14.3.2 定时器 0/1 模式寄存器 (TMOD)

| 符号   | 地址  | B7      | B6     | B5    | B4    | B3      | B2     | B1    | B0    |
|------|-----|---------|--------|-------|-------|---------|--------|-------|-------|
| TMOD | 89H | T1_GATE | T1_C/T | T1_M1 | T1_M0 | T0_GATE | T0_C/T | T0_M1 | T0_M0 |

T1\_GATE: 控制定时器1, 置1时只有在INT1脚为高及TR1控制位置1时才可打开定时器/计数器1。

T0\_GATE: 控制定时器0, 置1时只有在INT0脚为高及TR0控制位置1时才可打开定时器/计数器0。

T1\_C/T: 控制定时器1用作定时器或计数器, 清0则用作定时器(对内部系统时钟进行计数), 置1用作计数器(对引脚T1/P3.5外部脉冲进行计数)。

T0\_C/T: 控制定时器0用作定时器或计数器, 清0则用作定时器(对内部系统时钟进行计数), 置1用作计数器(对引脚T0/P3.4外部脉冲进行计数)。

T1\_M1/T1\_M0: 定时器定时器/计数器1模式选择

| T1_M1 | T1_M0 | 定时器/计数器1工作模式                                     |
|-------|-------|--|
| 0     | 0     | 16位自动重载模式<br>当[TH1, TL1]中的16位计数值溢出时, 系统会自动将内部16位 |

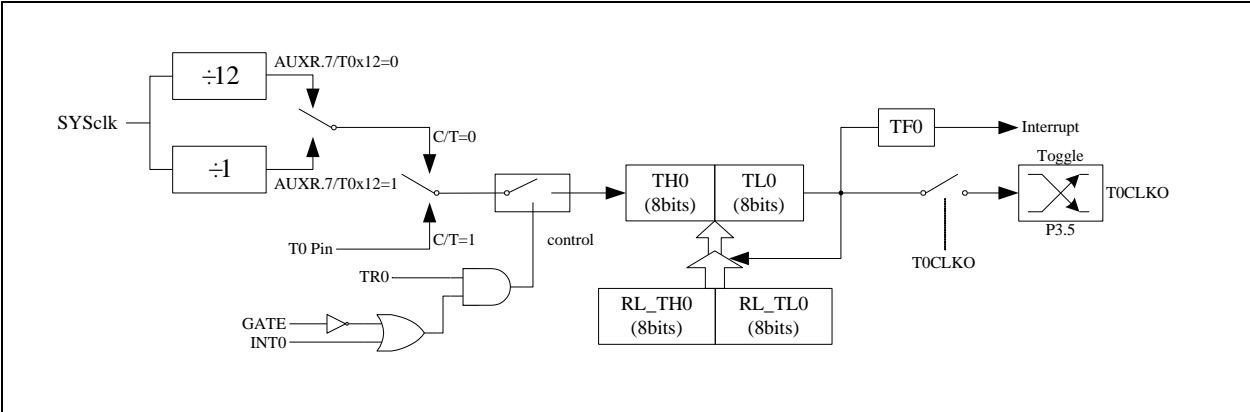
|   |   |  |
|---|---|--|
|   |   | 重载寄存器中的重载值装入[TH1,TL1]中。                          |
| 0 | 1 | 16位不自动重载模式<br>当[TH1,TL1]中的16位计数值溢出时，定时器1将从0开始计数  |
| 1 | 0 | 8位自动重载模式<br>当TL1中的8位计数值溢出时，系统会自动将TH1中的重载值装入TL1中。 |
| 1 | 1 | T1停止工作   |

T0\_M1/T0\_M0: 定时器/计数器0模式选择

| T0_M1 | T0_M0 | 定时器/计数器0工作模式   |
|-------|-------|--|
| 0     | 0     | 16位自动重载模式<br>当[TH0,TL0]中的16位计数值溢出时，系统会自动将内部16位重载寄存器中的重载值装入[TH0,TL0]中。                    |
| 0     | 1     | 16位不自动重载模式<br>当[TH0,TL0]中的16位计数值溢出时，定时器0将从0开始计数  |
| 1     | 0     | 8位自动重载模式<br>当TL0中的8位计数值溢出时，系统会自动将TH0中的重载值装入TL0中。   |
| 1     | 1     | 不可屏蔽中断的16位自动重载模式<br>与模式0相同，不可屏蔽中断，中断优先级最高，高于其他所有中断的优先级，并且不可关闭，可用作操作系统的系统节拍定时器，或者系统监控定时器。 |

14.3.3 定时器 0 模式 0（16 位自动重载模式）

此模式下定时器/计数器 0 作为可自动重载的 16 位计数器，如下图所示：



定时器/计数器 0 的模式 0：16 位自动重载模式

当 GATE=0 (TMOD.3) 时，如 TR0=1，则定时器计数。GATE=1 时，允许由外部输入 INT0 控制定时器 0，这样可实现脉宽测量。TR0 为 TCON 寄存器内的控制位，TCON 寄存器各位的具体功能描述见上节 TCON 寄存器的介绍。

当 C/T=0 时，多路开关连接到系统时钟的分频输出，T0 对内部系统时钟计数，T0 工作在定时方式。当

C/T=1 时, 多路开关连接到外部脉冲输入 P3.4/T0, 即 T0 工作在计数方式。

STC 单片机的定时器 0 有两种计数速率: 一种是 12T 模式, 每 12 个时钟加 1, 与传统 8051 单片机相同; 另外一种 1T 模式, 每个时钟加 1, 速度是传统 8051 单片机的 12 倍。T0 的速率由特殊功能寄存器 AUXR 中的 T0x12 决定, 如果 T0x12=0, T0 则工作在 12T 模式; 如果 T0x12=1, T0 则工作在 1T 模式

定时器 0 有两个隐藏的寄存器 RL\_TH0 和 RL\_TL0。RL\_TH0 与 TH0 共有同一个地址, RL\_TL0 与 TL0 共有同一个地址。当 TR0=0 即定时器/计数器 0 被禁止工作时, 对 TL0 写入的内容会同时写入 RL\_TL0, 对 TH0 写入的内容也会同时写入 RL\_TH0。当 TR0=1 即定时器/计数器 0 被允许工作时, 对 TL0 写入内容, 实际上不是写入当前寄存器 TL0 中, 而是写入隐藏的寄存器 RL\_TL0 中, 对 TH0 写入内容, 实际上也不是写入当前寄存器 TH0 中, 而是写入隐藏的寄存器 RL\_TH0, 这样可以巧妙地实现 16 位重装载定时器。当读 TH0 和 TL0 的内容时, 所读的内容就是 TH0 和 TL0 的内容, 而不是 RL\_TH0 和 RL\_TL0 的内容。

当定时器 0 工作在模式 0 (TMOD[1:0]/[M1,M0]=00B) 时, [TH0,TL0] 的溢出不仅置位 TF0, 而且会自动将 [RL\_TH0,RL\_TL0] 的内容重新装入 [TH0,TL0]。

当 T0CLKO/INT\_CLKO.0=1 时, P3.5/T1 管脚配置为定时器 0 的时钟输出 T0CLKO。输出时钟频率为  $T0 \text{ 溢出率}/2$ 。

如果 C/T=0, 定时器/计数器 T0 对内部系统时钟计数, 则:

T0 工作在 1T 模式 (AUXR.7/T0x12=1) 时的输出时钟频率 =  $(SYSclk)/(65536-[RL\_TH0, RL\_TL0])/2$

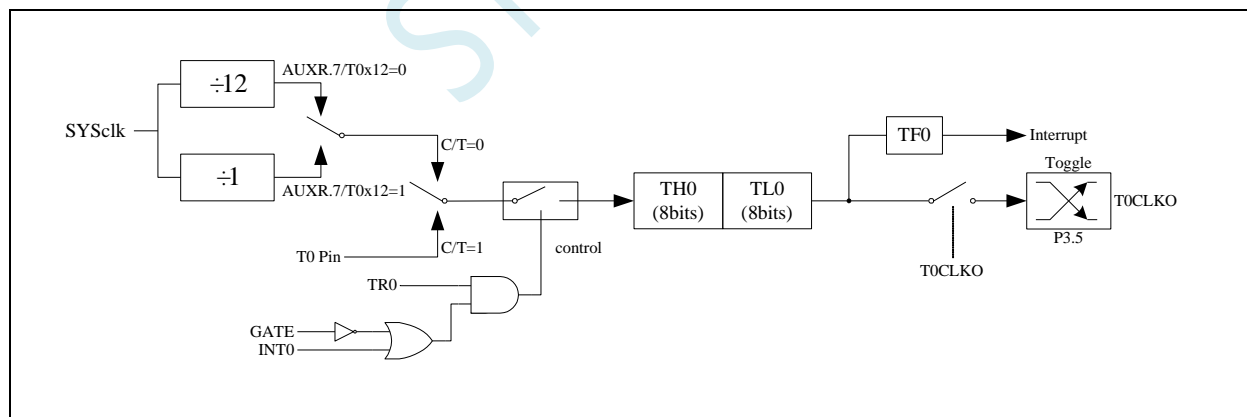
T0 工作在 12T 模式 (AUXR.7/T0x12=0) 时的输出时钟频率 =  $(SYSclk)/12/(65536-[RL\_TH0, RL\_TL0])/2$

如果 C/T=1, 定时器/计数器 T0 是对外部脉冲输入(P3.4/T0)计数, 则:

输出时钟频率 =  $(T0\_Pin\_CLK) / (65536-[RL\_TH0, RL\_TL0])/2$

### 14.3.4 定时器 0 模式 1 (16 位不可重装载模式)

此模式下定时器/计数器 0 工作在 16 位不可重装载模式, 如下图所示



定时器/计数器 0 的模式 1: 16 位不可重装载模式

此模式下, 定时器/计数器 0 配置为 16 位不可重装载模式, 由 TL0 的 8 位和 TH0 的 8 位所构成。TL0 的 8 位溢出向 TH0 进位, TH0 计数溢出置位 TCON 中的溢出标志位 TF0。

当 GATE=0(TM0D.3)时, 如 TR0=1, 则定时器计数。GATE=1 时, 允许由外部输入 INT0 控制定时器 0, 这样可实现脉宽测量。TR0 为 TCON 寄存器内的控制位, TCON 寄存器各位的具体功能描述见上节 TCON 寄存器的介绍。

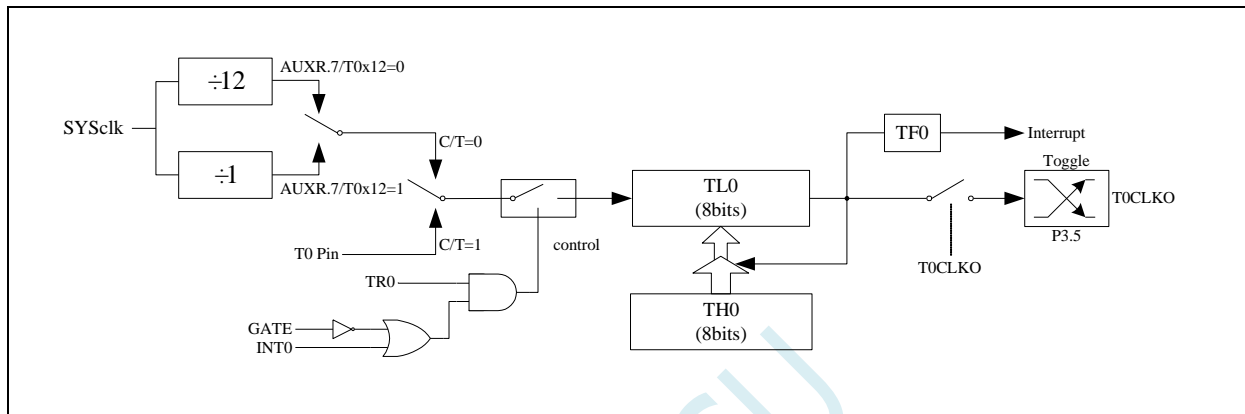
当 C/T=0 时, 多路开关连接到系统时钟的分频输出, T0 对内部系统时钟计数, T0 工作在定时方式。当

C/T=1 时, 多路开关连接到外部脉冲输入 P3.4/T0, 即 T0 工作在计数方式。

STC 单片机的定时器 0 有两种计数速率: 一种是 12T 模式, 每 12 个时钟加 1, 与传统 8051 单片机相同; 另外一种 1T 模式, 每个时钟加 1, 速度是传统 8051 单片机的 12 倍。T0 的速率由特殊功能寄存器 AUXR 中的 T0x12 决定, 如果 T0x12=0, T0 则工作在 12T 模式; 如果 T0x12=1, T0 则工作在 1T 模式

### 14.3.5 定时器 0 模式 2 (8 位自动重载模式)

此模式下定时器/计数器 0 作为可自动重载的 8 位计数器, 如下图所示:



定时器/计数器 0 的模式 2: 8 位自动重载模式

TL0 的溢出不仅置位 TF0, 而且将 TH0 的内容重新装入 TL0, TH0 内容由软件预置, 重装时 TH0 内容不变。

当 T0CLKO/INT\_CLKO.0=1 时, P3.5/T1 管脚配置为定时器 0 的时钟输出 T0CLKO。输出时钟频率为  $T0 \text{ 溢出率}/2$ 。

如果 C/T=0, 定时器/计数器 T0 对内部系统时钟计数, 则:

T0 工作在 1T 模式 (AUXR.7/T0x12=1) 时的输出时钟频率 =  $(SYSclk)/(256-TH0)/2$

T0 工作在 12T 模式 (AUXR.7/T0x12=0) 时的输出时钟频率 =  $(SYSclk)/12/(256-TH0)/2$

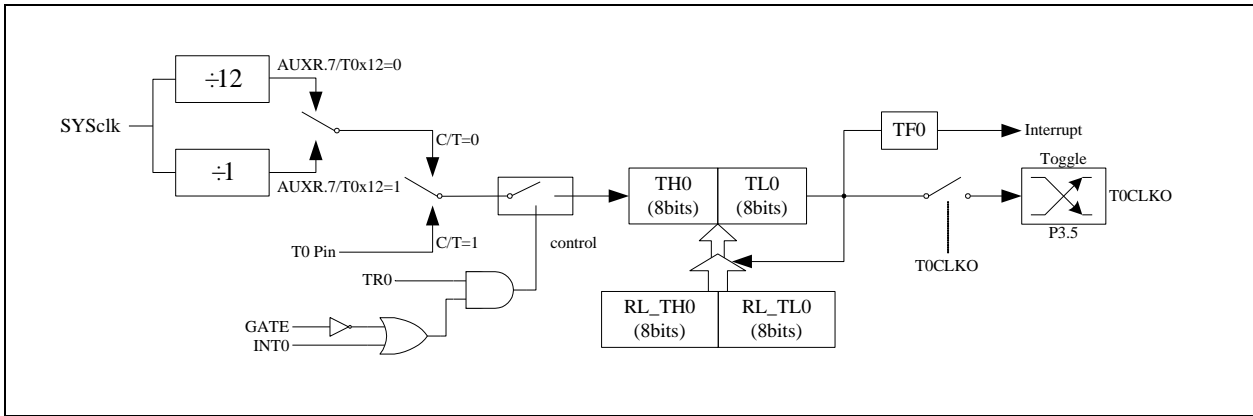
如果 C/T=1, 定时器/计数器 T0 是对外部脉冲输入(P3.4/T0)计数, 则:

输出时钟频率 =  $(T0\_Pin\_CLK) / (256-TH0)/2$

### 14.3.6 定时器 0 模式 3 (不可屏蔽中断 16 位自动重载, 实时操作系统节拍器)

对定时器/计数器 0, 其工作模式模式 3 与工作模式 0 是一样的 (下图定时器模式 3 的原理图, 与工作模式 0 是一样的)。唯一不同的是: 当定时器/计数器 0 工作在模式 3 时, 只需允许 ET0/IE.1(定时器/计数器 0 中断允许位), 不需要允许 EA/IE.7(总中断使能位)就能打开定时器/计数器 0 的中断, 此模式下的定时器/计数器 0 中断与总中断使能位 EA 无关, 一旦工作在模式 3 下的定时器/计数器 0 中断被打开(ET0=1), 那么该中断是不可屏蔽的, 该中断的优先级是最高的, 即该中断不能被任何中断所打断, 而且该中断打开后既不受 EA/IE.7 控制也不再受 ET0 控制, 当 EA=0 或 ET0=0 时都不能屏蔽此中断。故将此模式称为不可屏蔽中断的 16 位自动重载模式。



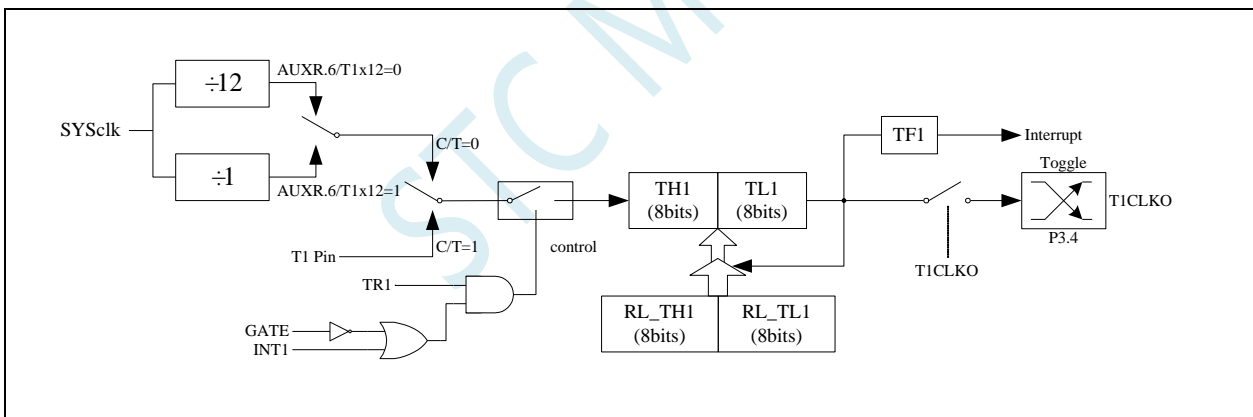


定时器/计数器 0 的模式 3: 不可屏蔽中断的 16 位自动重载模式

注意: 当定时器/计数器 0 工作在模式 3(不可屏蔽中断的 16 位自动重载模式)时, 不需要允许 EA/IE.7(总中断使能位), 只需允许 ET0/IE.1(定时器/计数器 0 中断允许位)就能打开定时器/计数器 0 的中断, 此模式下的定时器/计数器 0 中断与总中断使能位 EA 无关。一旦此模式下的定时器/计数器 0 中断被打开后, 该定时器/计数器 0 中断优先级就是最高的, 它不能被其它任何中断所打断(不管是比定时器/计数器 0 中断优先级低的中断还是比其优先级高的中断, 都不能打断此时的定时器/计数器 0 中断), 而且该中断打开后既不受 EA/IE.7 控制也不再受 ET0 控制了, 清零 EA 或 ET0 都不能关闭此中断。

### 14.3.7 定时器 1 模式 0 (16 位自动重载模式)

此模式下定时器/计数器 1 作为可自动重载的 16 位计数器, 如下图所示:



定时器/计数器 1 的模式 0: 16 位自动重载模式

当 GATE=0 (TMOD.7) 时, 如 TR1=1, 则定时器计数。GATE=1 时, 允许由外部输入 INT1 控制定时器 1, 这样可实现脉宽测量。TR1 为 TCON 寄存器内的控制位, TCON 寄存器各位的具体功能描述见上节 TCON 寄存器的介绍。

当 C/T=0 时, 多路开关连接到系统时钟的分频输出, T1 对内部系统时钟计数, T1 工作在定时方式。当 C/T=1 时, 多路开关连接到外部脉冲输入 P3.5/T1, 即 T1 工作在计数方式。

STC 单片机的定时器 1 有两种计数速率: 一种是 12T 模式, 每 12 个时钟加 1, 与传统 8051 单片机相同; 另外一种 1T 模式, 每个时钟加 1, 速度是传统 8051 单片机的 12 倍。T1 的速率由特殊功能寄存器 AUXR 中的 T1x12 决定, 如果 T1x12=0, T1 则工作在 12T 模式; 如果 T1x12=1, T1 则工作在 1T 模式。

定时器 1 有两个隐藏的寄存器 RL\_TH1 和 RL\_TL1。RL\_TH1 与 TH1 共有同一个地址, RL\_TL1 与 TL1 共有同一个地址。当 TR1=0 即定时器/计数器 1 被禁止工作时, 对 TL1 写入的内容会同时写入 RL\_TL1, 对



TH1 写入的内容也会同时写入 RL\_TH1。当 TR1=1 即定时器/计数器 1 被允许工作时, 对 TL1 写入内容, 实际上不是写入当前寄存器 TL1 中, 而是写入隐藏的寄存器 RL\_TL1 中, 对 TH1 写入内容, 实际上也不是写入当前寄存器 TH1 中, 而是写入隐藏的寄存器 RL\_TH1, 这样可以巧妙地实现 16 位重载定时器。当读 TH1 和 TL1 的内容时, 所读的内容就是 TH1 和 TL1 的内容, 而不是 RL\_TH1 和 RL\_TL1 的内容。

当定时器 1 工作在模式 1 (TMOD[5:4]/[M1,M0]=00B) 时, [TH1,TL1]的溢出不仅置位 TF1, 而且会自动将[RL\_TH1,RL\_TL1]的内容重新装入[TH1,TL1]。

当 T1CLKO/INT\_CLKO.1=1 时, P3.4/T0 管脚配置为定时器 1 的时钟输出 T1CLKO。输出时钟频率为  $T1 \text{ 溢出率}/2$ 。

如果 C/T=0, 定时器/计数器 T1 对内部系统时钟计数, 则:

T1 工作在 1T 模式 (AUXR.6/T1x12=1) 时的输出时钟频率 =  $(\text{SYSclk})/(65536-[\text{RL\_TH1}, \text{RL\_TL1}])/2$

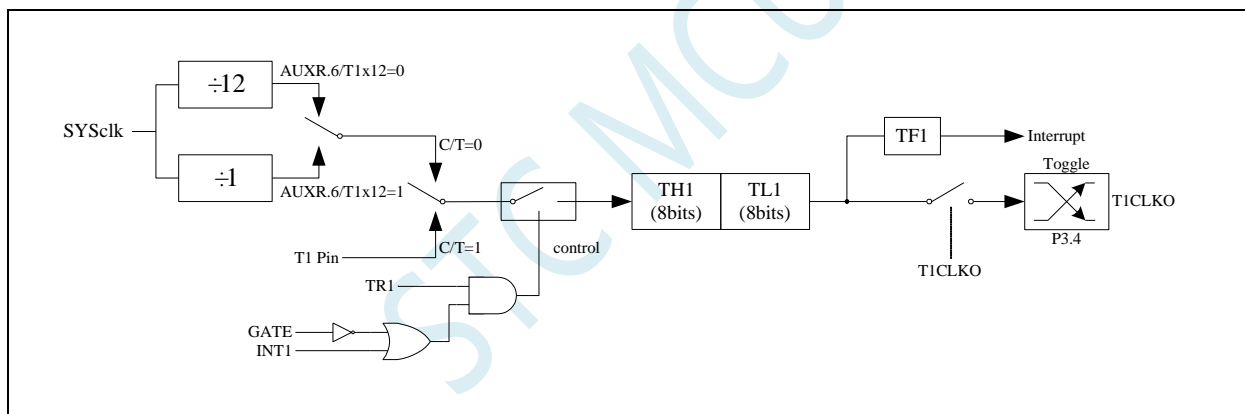
T1 工作在 12T 模式 (AUXR.6/T1x12=0) 时的输出时钟频率 =  $(\text{SYSclk})/12/(65536-[\text{RL\_TH1}, \text{RL\_TL1}])/2$

如果 C/T=1, 定时器/计数器 T1 是对外部脉冲输入(P3.5/T1)计数, 则:

输出时钟频率 =  $(\text{T1\_Pin\_CLK}) / (65536-[\text{RL\_TH1}, \text{RL\_TL1}])/2$

### 14.3.8 定时器 1 模式 1 (16 位不可重载模式)

此模式下定时器/计数器 1 工作在 16 位不可重载模式, 如下图所示



定时器/计数器 1 的模式 1: 16 位不可重载模式

此模式下, 定时器/计数器 1 配置为 16 位不可重载模式, 由 TL1 的 8 位和 TH1 的 8 位所构成。TL1 的 8 位溢出向 TH1 进位, TH1 计数溢出置位 TCON 中的溢出标志位 TF1。

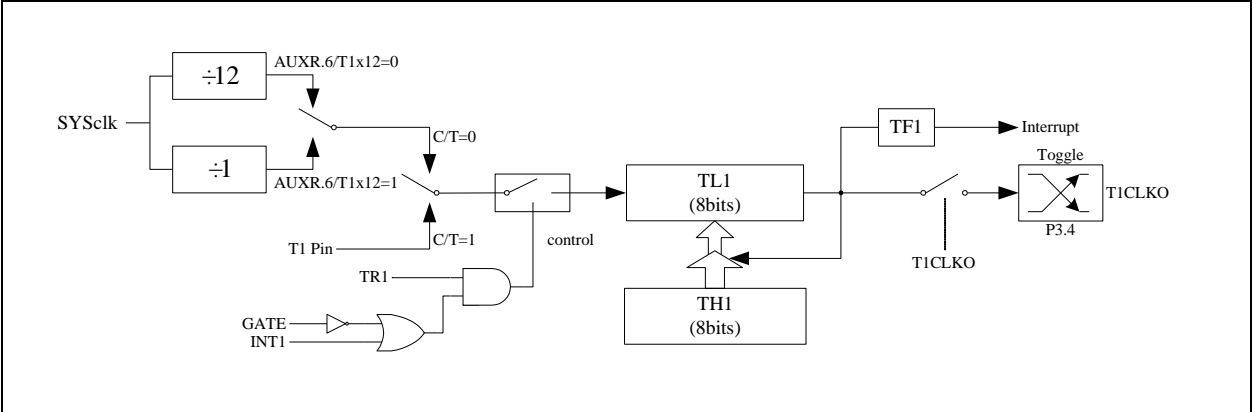
当 GATE=0(TM0D.7)时, 如 TR1=1, 则定时器计数。GATE=1 时, 允许由外部输入 INT1 控制定时器 1, 这样可实现脉宽测量。TR1 为 TCON 寄存器内的控制位, TCON 寄存器各位的具体功能描述见上节 TCON 寄存器的介绍。

当 C/T=0 时, 多路开关连接到系统时钟的分频输出, T1 对内部系统时钟计数, T1 工作在定时方式。当 C/T=1 时, 多路开关连接到外部脉冲输入 P3.5/T1, 即 T1 工作在计数方式。

STC 单片机的定时器 1 有两种计数速率: 一种是 12T 模式, 每 12 个时钟加 1, 与传统 8051 单片机相同; 另外一种 1T 模式, 每个时钟加 1, 速度是传统 8051 单片机的 12 倍。T1 的速率由特殊功能寄存器 AUXR 中的 T1x12 决定, 如果 T1x12=0, T1 则工作在 12T 模式; 如果 T1x12=1, T1 则工作在 1T 模式

### 14.3.9 定时器 1 模式 2（8 位自动重载模式）

此模式下定时器/计数器 1 作为可自动重载的 8 位计数器，如下图所示：



定时器/计数器 1 的模式 2：8 位自动重载模式

TL1 的溢出不仅置位 TF1，而且将 TH1 的内容重新装入 TL1，TH1 内容由软件预置，重装时 TH1 内容不变。

当 T1CLKO/INT\_CLKO.1=1 时，P3.4/T0 管脚配置为定时器 1 的时钟输出 T1CLKO。输出时钟频率为  $\frac{\text{溢出率}}{2}$ 。

如果 C/T=0，定时器/计数器 T1 对内部系统时钟计数，则：

T1 工作在 1T 模式（AUXR.6/T1x12=1）时的输出时钟频率 =  $\frac{(\text{SYSclk})}{(256-\text{TH1})/2}$

T1 工作在 12T 模式（AUXR.6/T1x12=0）时的输出时钟频率 =  $\frac{(\text{SYSclk})}{12 \cdot (256-\text{TH1})/2}$

如果 C/T=1，定时器/计数器 T1 是对外部脉冲输入(P3.5/T1)计数，则：

输出时钟频率 =  $\frac{(\text{T1\_Pin\_CLK})}{(256-\text{TH1})/2}$

### 14.3.10 定时器 0 计数寄存器（TL0，TH0）

| 符号  | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----|-----|----|----|----|----|----|----|----|----|
| TL0 | 8AH |    |    |    |    |    |    |    |    |
| TH0 | 8CH |    |    |    |    |    |    |    |    |

当定时器/计数器0工作在16位模式（模式0、模式1、模式3）时，TL0和TH0组合成为一个16位寄存器，TL0为低字节，TH0为高字节。若为8位模式（模式2）时，TL0和TH0为两个独立的8位寄存器。

### 14.3.11 定时器 1 计数寄存器（TL1，TH1）

| 符号  | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----|-----|----|----|----|----|----|----|----|----|
| TL1 | 8BH |    |    |    |    |    |    |    |    |
| TH1 | 8DH |    |    |    |    |    |    |    |    |

当定时器/计数器1工作在16位模式（模式0、模式1）时，TL1和TH1组合成为一个16位寄存器，TL1为低字节，TH1为高字节。若为8位模式（模式2）时，TL1和TH1为两个独立的8位寄存器。

### 14.3.12 辅助寄存器 1（AUXR）

| 符号 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----|----|----|----|----|----|----|----|----|----|
|----|----|----|----|----|----|----|----|----|----|

|      |     |       |       |           |     |        |       |        |       |
|------|-----|-------|-------|-----------|-----|--------|-------|--------|-------|
| AUXR | 8EH | T0x12 | T1x12 | UART_M0x6 | T2R | T2_C/T | T2x12 | EXTRAM | S1ST2 |
|------|-----|-------|-------|-----------|-----|--------|-------|--------|-------|

T0x12: 定时器0速度控制位

- 0: 12T 模式, 即 CPU 时钟 12 分频 (FOSC/12)
- 1: 1T 模式, 即 CPU 时钟不分频 (FOSC/1)

T1x12: 定时器1速度控制位

- 0: 12T 模式, 即 CPU 时钟 12 分频 (FOSC/12)
- 1: 1T 模式, 即 CPU 时钟不分频 (FOSC/1)

### 14.3.13 中断与时钟输出控制寄存器 (INTCLKO)

| 符号      | 地址  | B7 | B6  | B5  | B4  | B3 | B2     | B1     | B0     |
|---------|-----|----|-----|-----|-----|----|--------|--------|--------|
| INTCLKO | 8FH | -  | EX4 | EX3 | EX2 | -  | T2CLKO | T1CLKO | T0CLKO |

T0CLKO: 定时器0时钟输出控制

- 0: 关闭时钟输出
- 1: 使能 P3.5 口的是定时器 0 时钟输出功能  
当定时器 0 计数发生溢出时, P3.5 口的电平自动发生翻转。

T1CLKO: 定时器1时钟输出控制

- 0: 关闭时钟输出
- 1: 使能 P3.4 口的是定时器 1 时钟输出功能  
当定时器 1 计数发生溢出时, P3.4 口的电平自动发生翻转。

### 14.3.14 定时器 0 计算公式

| 定时器模式              | 定时器速度 | 周期计算公式  |
|--------------------|-------|---|
| 模式0/3<br>(16位自动重载) | 1T    | 定时器周期 = $\frac{65536 - [TH0, TL0]}{SYSclk}$ (自动重载)            |
|                    | 12T   | 定时器周期 = $\frac{65536 - [TH0, TL0]}{SYSclk} \times 12$ (自动重载)  |
| 模式1<br>(16位不自动重载)  | 1T    | 定时器周期 = $\frac{65536 - [TH0, TL0]}{SYSclk}$ (需软件装载)           |
|                    | 12T   | 定时器周期 = $\frac{65536 - [TH0, TL0]}{SYSclk} \times 12$ (需软件装载) |
| 模式2<br>(8位自动重载)    | 1T    | 定时器周期 = $\frac{256 - TH0}{SYSclk}$ (自动重载)                     |
|                    | 12T   | 定时器周期 = $\frac{256 - TH0}{SYSclk} \times 12$ (自动重载)           |

### 14.3.15 定时器 1 计算公式

| 定时器模式             | 定时器速度 | 周期计算公式  |
|-------------------|-------|---|
| 模式0<br>(16位自动重载)  | 1T    | 定时器周期 = $\frac{65536 - [TH1, TL1]}{SYSclk}$ (自动重载)            |
|                   | 12T   | 定时器周期 = $\frac{65536 - [TH1, TL1]}{SYSclk} \times 12$ (自动重载)  |
| 模式1<br>(16位不自动重载) | 1T    | 定时器周期 = $\frac{65536 - [TH1, TL1]}{SYSclk}$ (需软件装载)           |
|                   | 12T   | 定时器周期 = $\frac{65536 - [TH1, TL1]}{SYSclk} \times 12$ (需软件装载) |
| 模式2<br>(8位自动重载)   | 1T    | 定时器周期 = $\frac{256 - TH1}{SYSclk}$ (自动重载)                     |
|                   | 12T   | 定时器周期 = $\frac{256 - TH1}{SYSclk} \times 12$ (自动重载)           |

## 14.4 定时器 2（24 位定时器，8 位预分频+16 位定时）

### 14.4.1 辅助寄存器 1（AUXR）

| 符号   | 地址  | B7    | B6    | B5        | B4  | B3     | B2    | B1     | B0    |
|------|-----|-------|-------|-----------|-----|--------|-------|--------|-------|
| AUXR | 8EH | T0x12 | T1x12 | UART_M0x6 | T2R | T2_C/T | T2x12 | EXTRAM | S1ST2 |

T2R: 定时器2的运行控制位

0: 定时器 2 停止计数

1: 定时器 2 开始计数

T2\_C/T: 控制定时器2用作定时器或计数器，清0则用作定时器（对内部系统时钟进行计数），置1用作计数器（对引脚T2/P1.2外部脉冲进行计数）。

T2x12: 定时器2速度控制位

0: 12T 模式，即 CPU 时钟 12 分频（FOSC/12）

1: 1T 模式，即 CPU 时钟不分频（FOSC/1）

### 14.4.2 中断与时钟输出控制寄存器（INTCLKO）

| 符号      | 地址  | B7 | B6  | B5  | B4  | B3 | B2     | B1     | B0     |
|---------|-----|----|-----|-----|-----|----|--------|--------|--------|
| INTCLKO | 8FH | -  | EX4 | EX3 | EX2 | -  | T2CLKO | T1CLKO | T0CLKO |

T2CLKO: 定时器2时钟输出控制

0: 关闭时钟输出

1: 使能 P1.3 口的是定时器 2 时钟输出功能

当定时器 2 计数发生溢出时，P1.3 口的电平自动发生翻转。

### 14.4.3 定时器 2 计数寄存器（T2L，T2H）

| 符号  | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----|-----|----|----|----|----|----|----|----|----|
| T2L | D7H |    |    |    |    |    |    |    |    |
| T2H | D6H |    |    |    |    |    |    |    |    |

定时器/计数器2的工作模式固定为16位重载模式，T2L和T2H组合成为一个16位寄存器，T2L为低字节，

T2H为高字节。当[T2H,T2L]中的16位计数值溢出时，系统会自动将内部16位重载寄存器中的重载值装入[T2H,T2L]中。

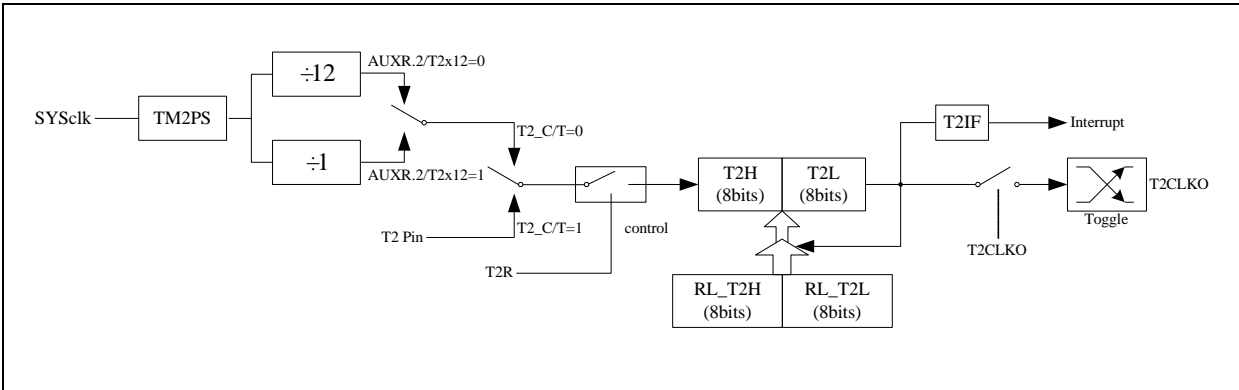
### 14.4.4 定时器 2 的 8 位预分频寄存器（TM2PS）

| 符号    | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------|-------|----|----|----|----|----|----|----|----|
| TM2PS | FEA2H |    |    |    |    |    |    |    |    |

定时器2的时钟 = 系统时钟SYSclk ÷ (TM2PS + 1)

### 14.4.5 定时器 2 工作模式

定时器/计数器 2 的原理框图如下:



定时器/计数器 2 的工作模式：16 位自动重装载模式

T2R/AUXR.4 为 AUXR 寄存器内的控制位, AUXR 寄存器各位的具体功能描述见上节 AUXR 寄存器的介绍。

当 T2\_C/T=0 时,多路开关连接到系统时钟输出,T2 对内部系统时钟计数,T2 工作在定时方式。当 T2\_C/T=1 时,多路开关连接到外部脉冲输 T2,即 T2 工作在计数方式。

STC 单片机的定时器 2 有两种计数速率:一种是 12T 模式,每 12 个时钟加 1,与传统 8051 单片机相同;另外一种 是 1T 模式,每个时钟加 1,速度是传统 8051 单片机的 12 倍。T2 的速率由特殊功能寄存器 AUXR 中的 T2x12 决定,如果 T2x12=0,T2 则工作在 12T 模式;如果 T2x12=1,T2 则工作在 1T 模式

定时器 2 有两个隐藏的寄存器 RL\_T2H 和 RL\_T2L。RL\_T2H 与 T2H 共有同一个地址,RL\_T2L 与 T2L 共有同一个地址。当 T2R=0 即定时器/计数器 2 被禁止工作时,对 T2L 写入的内容会同时写入 RL\_T2L,对 T2H 写入的内容也会同时写入 RL\_T2H。当 T2R=1 即定时器/计数器 2 被允许工作时,对 T2L 写入内容,实际上不是写入当前寄存器 T2L 中,而是写入隐藏的寄存器 RL\_T2L 中,对 T2H 写入内容,实际上也不是写入当前寄存器 T2H 中,而是写入隐藏的寄存器 RL\_T2H,这样可以巧妙地实现 16 位重装载定时器。当读 T2H 和 T2L 的内容时,所读的内容就是 T2H 和 T2L 的内容,而不是 RL\_T2H 和 RL\_T2L 的内容。

[T2H,T2L]的溢出不仅置位中断请求标志位 (T2IF),使 CPU 转去执行定时器 2 的中断程序,而且会自动将[RL\_T2H,RL\_T2L]的内容重新装入[T2H,T2L]。

### 14.4.6 定时器 2 计算公式

| 定时器速度 | 周期计算公式   |
|-------|--|
| 1T    | 定时器周期 = $\frac{65536 - [T2H, T2L]}{SYSclk/(TM2PS+1)}$ (自动重载)           |
| 12T   | 定时器周期 = $\frac{65536 - [T2H, T2L]}{SYSclk/(TM2PS+1)} \times 12$ (自动重载) |

## 14.5 定时器 3/4（24 位定时器，8 位预分频+16 位定时）

### 14.5.1 定时器 4/3 控制寄存器（T4T3M）

| 符号    | 地址  | B7  | B6     | B5    | B4     | B3  | B2     | B1    | B0     |
|-------|-----|-----|--------|-------|--------|-----|--------|-------|--------|
| T4T3M | D1H | T4R | T4_C/T | T4x12 | T4CLKO | T3R | T3_C/T | T3x12 | T3CLKO |

T4R: 定时器4的运行控制位

0: 定时器 4 停止计数

1: 定时器 4 开始计数

T4\_C/T: 控制定时器4用作定时器或计数器，清0则用作定时器（对内部系统时钟进行计数），置1用作计数器（对引脚T4/P0.6外部脉冲进行计数）。

T4x12: 定时器4速度控制位

0: 12T 模式，即 CPU 时钟 12 分频（FOSC/12）

1: 1T 模式，即 CPU 时钟不分频（FOSC/1）

T4CLKO: 定时器4时钟输出控制

0: 关闭时钟输出

1: 使能 P0.7 口的是定时器 4 时钟输出功能

当定时器 4 计数发生溢出时，P0.7 口的电平自动发生翻转。

T3R: 定时器3的运行控制位

0: 定时器 3 停止计数

1: 定时器 3 开始计数

T3\_C/T: 控制定时器3用作定时器或计数器，清0则用作定时器（对内部系统时钟进行计数），置1用作计数器（对引脚T3/P0.4外部脉冲进行计数）。

T3x12: 定时器3速度控制位

0: 12T 模式，即 CPU 时钟 12 分频（FOSC/12）

1: 1T 模式，即 CPU 时钟不分频（FOSC/1）

T3CLKO: 定时器3时钟输出控制

0: 关闭时钟输出

1: 使能 P0.5 口的是定时器 3 时钟输出功能

当定时器 3 计数发生溢出时，P0.5 口的电平自动发生翻转。

### 14.5.2 定时器 3 计数寄存器（T3L，T3H）

| 符号  | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----|-----|----|----|----|----|----|----|----|----|
| T3L | D5H |    |    |    |    |    |    |    |    |
| T3H | D4H |    |    |    |    |    |    |    |    |

定时器/计数器3的工作模式固定为16位重载模式，T3L和T3H组合成为一个16位寄存器，T3L为低字节，

T3H为高字节。当[T3H,T3L]中的16位计数值溢出时，系统会自动将内部16位重载寄存器中的重载值装入[T3H,T3L]中。

### 14.5.3 定时器 4 计数寄存器（T4L，T4H）

| 符号  | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----|-----|----|----|----|----|----|----|----|----|
| T4L | D3H |    |    |    |    |    |    |    |    |

|     |     |  |
|-----|-----|--|
| T4H | D2H |  |
|-----|-----|--|

定时器/计数器 4 的工作模式固定为 16 位重载模式，T4L 和 T4H 组合成为一个 16 位寄存器，T4L 为低字节，T4H 为高字节。当[T4H,T4L]中的 16 位计数值溢出时，系统会自动将内部 16 位重载寄存器中的重载值装入[T4H,T4L]中。

14.5.4 定时器 3 的 8 位预分频寄存器（TM3PS）

| 符号    | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------|-------|----|----|----|----|----|----|----|----|
| TM3PS | FEA3H |    |    |    |    |    |    |    |    |

定时器3的时钟 = 系统时钟SYSclk ÷ ( TM3PS + 1 )

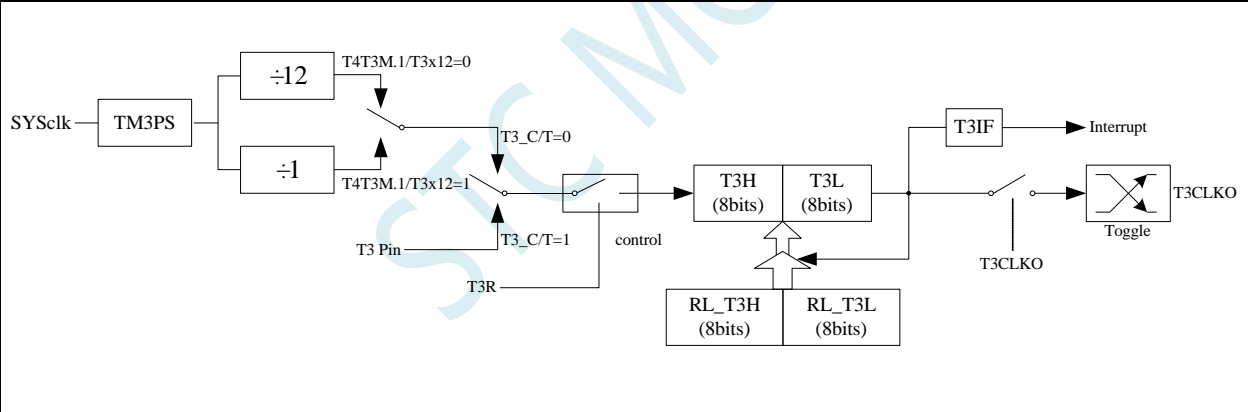
14.5.5 定时器 4 的 8 位预分频寄存器（TM4PS）

| 符号    | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------|-------|----|----|----|----|----|----|----|----|
| TM4PS | FEA4H |    |    |    |    |    |    |    |    |

定时器4的时钟 = 系统时钟SYSclk ÷ ( TM4PS + 1 )

14.5.6 定时器 3 工作模式

定时器/计数器 3 的原理框图如下：



定时器/计数器 3 的工作模式：16 位自动重载模式

T3R/T4T3M.3 为 T4T3M 寄存器内的控制位，T4T3M 寄存器各位的具体功能描述见上节 T4T3M 寄存器的介绍。

当 T3\_C/T=0 时，多路开关连接到系统时钟输出，T3 对内部系统时钟计数，T3 工作在定时方式。当 T3\_C/T=1 时，多路开关连接到外部脉冲输入 T3，即 T3 工作在计数方式。

STC 单片机的定时器 3 有两种计数速率：一种是 12T 模式，每 12 个时钟加 1，与传统 8051 单片机相同；另外一种 1T 模式，每个时钟加 1，速度是传统 8051 单片机的 12 倍。T3 的速率由特殊功能寄存器 T4T3M 中的 T3x12 决定，如果 T3x12=0，T3 则工作在 12T 模式；如果 T3x12=1，T3 则工作在 1T 模式

定时器 3 有两个隐藏的寄存器 RL\_T3H 和 RL\_T3L。RL\_T3H 与 T3H 共有同一个地址，RL\_T3L 与 T3L 共有同一个地址。当 T3R=0 即定时器/计数器 3 被禁止工作时，对 T3L 写入的内容会同时写入 RL\_T3L，对 T3H 写入的内容也会同时写入 RL\_T3H。当 T3R=1 即定时器/计数器 3 被允许工作时，对 T3L 写入内容，实际上不是写入当前寄存器 T3L 中，而是写入隐藏的寄存器 RL\_T3L 中，对 T3H 写入内容，实际上也不是写入



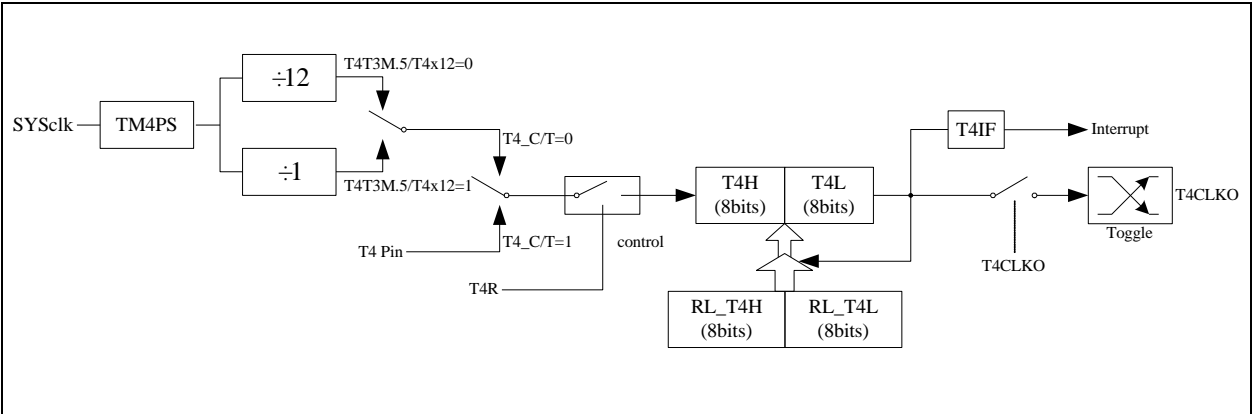
当前寄存器 T3H 中，而是写入隐藏的寄存器 RL\_T3H，这样可以巧妙地实现 16 位重载定时器。当读 T3H 和 T3L 的内容时，所读的内容就是 T3H 和 T3L 的内容，而不是 RL\_T3H 和 RL\_T3L 的内容。

[T3H,T3L]的溢出不仅置位中断请求标志位 (T3IF)，使 CPU 转去执行定时器 3 的中断程序，而且会自动将[RL\_T3H,RL\_T3L]的内容重新装入[T3H,T3L]。

STC MCU

### 14.5.7 定时器 4 工作模式

定时器/计数器 4 的原理框图如下：



定时器/计数器 4 的工作模式：16 位自动重载模式

T4R/T4T3M.7 为 T4T3M 寄存器内的控制位，T4T3M 寄存器各位的具体功能描述见上节 T4T3M 寄存器的介绍。

当 T4\_C/T=0 时，多路开关连接到系统时钟输出，T4 对内部系统时钟计数，T4 工作在定时方式。当 T4\_C/T=1 时，多路开关连接到外部脉冲输 T4，即 T4 工作在计数方式。

STC 单片机的定时器 4 有两种计数速率：一种是 12T 模式，每 12 个时钟加 1，与传统 8051 单片机相同；另外一种 1T 模式，每个时钟加 1，速度是传统 8051 单片机的 12 倍。T4 的速率由特殊功能寄存器 T4T3M 中的 T4x12 决定，如果 T4x12=0，T4 则工作在 12T 模式；如果 T4x12=1，T4 则工作在 1T 模式

定时器 4 有两个隐藏的寄存器 RL\_T4H 和 RL\_T4L。RL\_T4H 与 T4H 共有同一个地址，RL\_T4L 与 T4L 共有同一个地址。当 T4R=0 即定时器/计数器 4 被禁止工作时，对 T4L 写入的内容会同时写入 RL\_T4L，对 T4H 写入的内容也会同时写入 RL\_T4H。当 T4R=1 即定时器/计数器 4 被允许工作时，对 T4L 写入内容，实际上不是写入当前寄存器 T4L 中，而是写入隐藏的寄存器 RL\_T4L 中，对 T4H 写入内容，实际上也不是写入当前寄存器 T4H 中，而是写入隐藏的寄存器 RL\_T4H，这样可以巧妙地实现 16 位重载定时器。当读 T4H 和 T4L 的内容时，所读的内容就是 T4H 和 T4L 的内容，而不是 RL\_T4H 和 RL\_T4L 的内容。

[T4H,T4L]的溢出不仅置位中断请求标志位（T4IF），使 CPU 转去执行定时器 4 的中断程序，而且会自动将[RL\_T4H,RL\_T4L]的内容重新装入[T4H,T4L]。

### 14.5.8 定时器 3 计算公式

| 定时器速度 | 周期计算公式   |
|-------|--|
| 1T    | $\text{定时器周期} = \frac{65536 - [T3H, T3L]}{\text{SYSclk}/(\text{TM3PS}+1)} \quad (\text{自动重载})$           |
| 12T   | $\text{定时器周期} = \frac{65536 - [T3H, T3L]}{\text{SYSclk}/(\text{TM3PS}+1)} \times 12 \quad (\text{自动重载})$ |

## 14.5.9 定时器 4 计算公式

| 定时器速度 | 周期计算公式   |
|-------|--|
| 1T    | 定时器周期 = $\frac{65536 - [T4H, T4L]}{SYSclk/(TM4PS+1)}$ (自动重载)           |
| 12T   | 定时器周期 = $\frac{65536 - [T4H, T4L]}{SYSclk/(TM4PS+1)} \times 12$ (自动重载) |

# 14.6 范例程序

## 14.6.1 定时器 0（模式 0—16 位自动重载），用作定时

### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void TM0_Isr() interrupt 1
{
    P10 = !P10;           //测试端口
}

void main()
{
    P_SW2 /= 0x80;        //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x00;          //模式 0
    TL0 = 0x66;           //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1;              //启动定时器
    ET0 = 1;              //使能定时器中断
    EA = 1;

    while (1);
}
```

### 汇编代码

```
;测试工作频率为 11.0592MHz

P_SW2      DATA      0BAH

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
```

```
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          000BH
          LJMP         TM0ISR

          ORG          0100H
TM0ISR:
          CPL          P1.0          ;测试端口
          RETI

MAIN:
          MOV          SP, #5FH
          ORL          P_SW2, #80H    ;使能访问 XFR

          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          MOV          TMOD, #00H      ;模式 0
          MOV          TL0, #66H       ;65536-11.0592M/12/1000
          MOV          TH0, #0FCH
          SETB         TR0             ;启动定时器
          SETB         ET0             ;使能定时器中断
          SETB         EA

          JMP          $

          END
```

14.6.2 定时器 0（模式 1—16 位不自动重载），用作定时

C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void TM0_Isr() interrupt 1
{
    TL0 = 0x66;          //重设定定时参数
```

```
    TH0 = 0xfc;
    P10 = !P10;                                     //测试端口
}

void main()
{
    P_SW2 /= 0x80;                                   //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x01;                                     //模式1
    TL0 = 0x66;                                       //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1;                                          //启动定时器
    ET0 = 1;                                          //使能定时器中断
    EA = 1;

    while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

|         |      |           |        |
|---------|------|-----------|--------|
| P_SW2   | DATA | 0BAH      |        |
| P1M1    | DATA | 091H      |        |
| P1M0    | DATA | 092H      |        |
| P0M1    | DATA | 093H      |        |
| P0M0    | DATA | 094H      |        |
| P2M1    | DATA | 095H      |        |
| P2M0    | DATA | 096H      |        |
| P3M1    | DATA | 0B1H      |        |
| P3M0    | DATA | 0B2H      |        |
| P4M1    | DATA | 0B3H      |        |
| P4M0    | DATA | 0B4H      |        |
| P5M1    | DATA | 0C9H      |        |
| P5M0    | DATA | 0CAH      |        |
|         | ORG  | 0000H     |        |
|         | LJMP | MAIN      |        |
|         | ORG  | 000BH     |        |
|         | LJMP | TM0ISR    |        |
|         | ORG  | 0100H     |        |
| TM0ISR: |      |           |        |
|         | MOV  | TL0,#66H  | ;重设定参数 |
|         | MOV  | TH0,#0FCH |        |

```

CPL      P1.0      ;测试端口
RETI

MAIN:

MOV      SP, #5FH
ORL      P_SW2, #80H      ;使能访问 XFR

MOV      P0M0, #00H
MOV      P0M1, #00H
MOV      P1M0, #00H
MOV      P1M1, #00H
MOV      P2M0, #00H
MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      TMOD, #01H      ;模式 1
MOV      TL0, #66H      ;65536-11.0592M/12/1000
MOV      TH0, #0FCH
SETB     TR0      ;启动定时器
SETB     ET0      ;使能定时器中断
SETB     EA

JMP      $

END
```

### 14.6.3 定时器 0（模式 2—8 位自动重载），用作定时

#### C 语言代码

```

//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void TM0_Isr() interrupt 1
{
    P10 = !P10;      //测试端口
}

void main()
{
    P_SW2 /= 0x80;      //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
```

```
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

TMOD = 0x02; //模式2
TL0 = 0xf4; //256-11.0592M/12/76K
TH0 = 0xf4;
TR0 = 1; //启动定时器
ET0 = 1; //使能定时器中断
EA = 1;

while (1);
}
```

汇编代码

;测试工作频率为11.0592MHz

|         |      |                      |
|---------|------|----------------------|
| P_SW2   | DATA | 0BAH                 |
| P1M1    | DATA | 091H                 |
| P1M0    | DATA | 092H                 |
| P0M1    | DATA | 093H                 |
| P0M0    | DATA | 094H                 |
| P2M1    | DATA | 095H                 |
| P2M0    | DATA | 096H                 |
| P3M1    | DATA | 0B1H                 |
| P3M0    | DATA | 0B2H                 |
| P4M1    | DATA | 0B3H                 |
| P4M0    | DATA | 0B4H                 |
| P5M1    | DATA | 0C9H                 |
| P5M0    | DATA | 0CAH                 |
|         | ORG  | 0000H                |
|         | LJMP | MAIN                 |
|         | ORG  | 000BH                |
|         | LJMP | TM0ISR               |
|         | ORG  | 0100H                |
| TM0ISR: |      |                      |
|         | CPL  | P1.0 ;测试端口           |
|         | RETI |                      |
| MAIN:   |      |                      |
|         | MOV  | SP, #5FH             |
|         | ORL  | P_SW2, #80H ;使能访问XFR |
|         | MOV  | P0M0, #00H           |
|         | MOV  | P0M1, #00H           |
|         | MOV  | P1M0, #00H           |
|         | MOV  | P1M1, #00H           |
|         | MOV  | P2M0, #00H           |
|         | MOV  | P2M1, #00H           |
|         | MOV  | P3M0, #00H           |
|         | MOV  | P3M1, #00H           |
|         | MOV  | P4M0, #00H           |
|         | MOV  | P4M1, #00H           |



```
MOV    P5M0, #00H
MOV    P5M1, #00H

MOV    TMOD, #02H           ;模式2
MOV    TL0, #0F4H           ;256-11.0592M/12/76K
MOV    TH0, #0F4H
SETB   TR0                  ;启动定时器
SETB   ET0                  ;使能定时器中断
SETB   EA

JMP    $

END
```

## 14.6.4 定时器 0 (模式 3—16 位自动重载不可屏蔽中断), 用作定时

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void TM0_Isr() interrupt 1
{
    P10 = !P10;           //测试端口
}
```

```
void main()
{
    P_SW2 /= 0x80;         //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x03;           //模式3
    TL0 = 0x66;            //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1;               //启动定时器
    ET0 = 1;               //使能定时器中断
    // EA = 1;             //不受 EA 控制

    while (1);
}
```

## 汇编代码

; 测试工作频率为 11.0592MHz

```

P_SW2      DATA      0BAH

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

ORG        0000H
LJMP      MAIN
ORG        000BH
LJMP      TM0ISR

TM0ISR:    ORG        0100H

CPL        P1.0      ; 测试端口
RETI

MAIN:

MOV        SP, #5FH
ORL        P_SW2, #80H      ; 使能访问 XFR

MOV        P0M0, #00H
MOV        P0M1, #00H
MOV        P1M0, #00H
MOV        P1M1, #00H
MOV        P2M0, #00H
MOV        P2M1, #00H
MOV        P3M0, #00H
MOV        P3M1, #00H
MOV        P4M0, #00H
MOV        P4M1, #00H
MOV        P5M0, #00H
MOV        P5M1, #00H

MOV        TMOD, #03H      ; 模式 3
MOV        TL0, #66H      ; 65536-11.0592M/12/1000
MOV        TH0, #0FCH

SETB      TR0      ; 启动定时器
SETB      ET0      ; 使能定时器中断
;
SETB      EA      ; 不受 EA 控制

JMP        $

END

```

## 14.6.5 定时器 0（外部计数—扩展 T0 为外部下降沿中断）

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void TM0_Isr() interrupt 1
```

```
{
    P10 = !P10;                //测试端口
}
```

```
void main()
```

```
{
    P_SW2 /= 0x80;             //使能访问 XFR
```

```
    P0M0 = 0x00;
```

```
    P0M1 = 0x00;
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P2M0 = 0x00;
```

```
    P2M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P4M0 = 0x00;
```

```
    P4M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
    TMOD = 0x04;
```

//外部计数模式

```
    TL0 = 0xff;
```

```
    TH0 = 0xff;
```

```
    TR0 = 1;
```

//启动定时器

```
    ET0 = 1;
```

//使能定时器中断

```
    EA = 1;
```

```
    while (1);
```

```
}
```

### 汇编代码

;测试工作频率为 11.0592MHz

```
P_SW2    DATA    0BAH
```

```
P1M1     DATA    091H
```

```
P1M0     DATA    092H
```

```
P0M1     DATA    093H
```

```
P0M0     DATA    094H
```

```
P2M1     DATA    095H
```

```
P2M0     DATA    096H
```

```
P3M1     DATA    0B1H
```

```
P3M0     DATA    0B2H
```

```
P4M1     DATA    0B3H
```

```
P4M0     DATA    0B4H
```

```
P5M1     DATA    0C9H
```

```
P5M0      DATA      0CAH

           ORG         0000H
           LJMP        MAIN
           ORG         000BH
           LJMP        TM0ISR

           ORG         0100H
TM0ISR:
           CPL         P1.0           ;测试端口
           RETI

MAIN:
           MOV         SP, #5FH
           ORL         P_SW2, #80H   ;使能访问 XFR

           MOV         P0M0, #00H
           MOV         P0M1, #00H
           MOV         P1M0, #00H
           MOV         P1M1, #00H
           MOV         P2M0, #00H
           MOV         P2M1, #00H
           MOV         P3M0, #00H
           MOV         P3M1, #00H
           MOV         P4M0, #00H
           MOV         P4M1, #00H
           MOV         P5M0, #00H
           MOV         P5M1, #00H

           MOV         TMOD, #04H    ;外部计数模式
           MOV         TL0, #0FFH
           MOV         TH0, #0FFH
           SETB        TR0           ;启动定时器
           SETB        ET0           ;使能定时器中断
           SETB        EA

           JMP         $

           END
```

### 14.6.6 定时器 0（测量脉宽—INT0 高电平宽度）

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void INT0_Isr() interrupt 0
{
    P0 = TL0;           //TL0 为测量值低字节
    P1 = TH0;           //TH0 为测量值高字节
    TL0 = 0x00;
    TH0 = 0x00;
}
```

```

void main()
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    AUXR = 0x80;                //IT 模式
    TMOD = 0x08;                //使能 GATE, INT0 为 1 时使能计时
    TL0 = 0x00;
    TH0 = 0x00;
    while (P32);                //等待 INT0 为低
    TR0 = 1;                    //启动定时器
    IT0 = 1;                    //使能 INT0 下降沿中断
    EX0 = 1;
    EA = 1;

    while (1);
}
    
```

# 汇编代码

;测试工作频率为 11.0592MHz

|          |      |         |              |
|----------|------|---------|--------------|
| P_SW2    | DATA | 0BAH    |              |
| AUXR     | DATA | 8EH     |              |
| P1M1     | DATA | 091H    |              |
| P1M0     | DATA | 092H    |              |
| P0M1     | DATA | 093H    |              |
| P0M0     | DATA | 094H    |              |
| P2M1     | DATA | 095H    |              |
| P2M0     | DATA | 096H    |              |
| P3M1     | DATA | 0B1H    |              |
| P3M0     | DATA | 0B2H    |              |
| P4M1     | DATA | 0B3H    |              |
| P4M0     | DATA | 0B4H    |              |
| P5M1     | DATA | 0C9H    |              |
| P5M0     | DATA | 0CAH    |              |
|          | ORG  | 0000H   |              |
|          | LJMP | MAIN    |              |
|          | ORG  | 0003H   |              |
|          | LJMP | INT0ISR |              |
|          | ORG  | 0100H   |              |
| INT0ISR: | MOV  | P0, TL0 | ;TL0 为测量值低字节 |

```

MOV    P1,TH0                ;TH0 为测量值高字节
MOV    TL0,#00H
MOV    TH0,#00H
RETI

```

MAIN:

```

MOV    SP,#5FH
ORL    P_SW2,#80H            ;使能访问 XFR

MOV    P0M0,#00H
MOV    P0M1,#00H
MOV    P1M0,#00H
MOV    P1M1,#00H
MOV    P2M0,#00H
MOV    P2M1,#00H
MOV    P3M0,#00H
MOV    P3M1,#00H
MOV    P4M0,#00H
MOV    P4M1,#00H
MOV    P5M0,#00H
MOV    P5M1,#00H

MOV    AUXR,#80H            ;IT 模式
MOV    TMOD,#08H            ;使能 GATE,INT0 为 1 时使能计时
MOV    TL0,#00H
MOV    TH0,#00H
JB     P3.2,$                ;等待 INT0 为低
SETB   TR0                  ;启动定时器
SETB   IT0                  ;使能 INT0 下降沿中断
SETB   EX0
SETB   EA

JMP     $

END

```

## 14.6.7 定时器 0（模式 0），时钟分频输出

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```
void main()
```

```

{
    P_SW2 /= 0x80;            //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;

```

```
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

TMOD = 0x00; //模式0
TL0 = 0x66; //65536-11.0592M/12/1000
TH0 = 0xfc;
TR0 = 1; //启动定时器
INTCLKO = 0x01; //使能时钟输出

while (1);
}
```

汇编代码

;测试工作频率为11.0592MHz

```
P_SW2      DATA      0BAH

INTCLKO     DATA      8FH
P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

            ORG         0000H
            LJMP        MAIN

MAIN:       ORG         0100H

            MOV         SP, #5FH
            ORL         P_SW2, #80H ;使能访问 XFR

            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         TMOD, #00H ;模式0
            MOV         TL0, #66H ;65536-11.0592M/12/1000
            MOV         TH0, #0FCH
```

|             |                     |                |
|-------------|---------------------|----------------|
| <i>SETB</i> | <i>TR0</i>          | <i>;启动定时器</i>  |
| <i>MOV</i>  | <i>INTCLKO,#01H</i> | <i>;使能时钟输出</i> |
| <i>JMP</i>  | <i>\$</i>           |                |
| <i>END</i>  |                     |                |

### 14.6.8 定时器 1（模式 0—16 位自动重载），用作定时

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void TM1_Isr() interrupt 3
{
    P10 = !P10;                //测试端口
}

void main()
{
    P_SW2 /= 0x80;             //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x00;               //模式 0
    TL1 = 0x66;                 //65536-11.0592M/12/1000
    TH1 = 0xfc;
    TR1 = 1;                    //启动定时器
    ET1 = 1;                    //使能定时器中断
    EA = 1;

    while (1);
}
```

#### 汇编代码

```
;测试工作频率为 11.0592MHz

P_SW2      DATA      0BAH

P1M1      DATA      091H
P1M0      DATA      092H
```



```
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          001BH
          LJMP         TMIISR

TMIISR:    ORG          0100H

          CPL          P1.0          ;测试端口
          RETI

MAIN:

          MOV          SP, #5FH
          ORL          P_SW2, #80H    ;使能访问 XFR

          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          MOV          TMOD, #00H      ;模式 0
          MOV          TL1, #66H      ;65536-11.0592M/12/1000
          MOV          TH1, #0FCH
          SETB         TRI            ;启动定时器
          SETB         ETI            ;使能定时器中断
          SETB         EA

          JMP          $

          END
```

14.6.9 定时器 1（模式 1—16 位不自动重载），用作定时

C 语言代码

//测试工作频率为 11.0592MHz

#include "stc8h.h"

```
#include "intrins.h"

void TMI_Isr() interrupt 3
{
    TL1 = 0x66;           //重设定定时参数
    TH1 = 0xfc;
    P10 = !P10;           //测试端口
}

void main()
{
    P_SW2 /= 0x80;        //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x10;          //模式1
    TL1 = 0x66;           //65536-11.0592M/12/1000
    TH1 = 0xfc;
    TR1 = 1;              //启动定时器
    ET1 = 1;              //使能定时器中断
    EA = 1;

    while (1);
}
```

汇编代码

;测试工作频率为11.0592MHz

|       |      |        |
|-------|------|--------|
| P_SW2 | DATA | 0BAH   |
| P1M1  | DATA | 091H   |
| P1M0  | DATA | 092H   |
| P0M1  | DATA | 093H   |
| P0M0  | DATA | 094H   |
| P2M1  | DATA | 095H   |
| P2M0  | DATA | 096H   |
| P3M1  | DATA | 0B1H   |
| P3M0  | DATA | 0B2H   |
| P4M1  | DATA | 0B3H   |
| P4M0  | DATA | 0B4H   |
| P5M1  | DATA | 0C9H   |
| P5M0  | DATA | 0CAH   |
|       | ORG  | 0000H  |
|       | LJMP | MAIN   |
|       | ORG  | 001BH  |
|       | LJMP | TMIISR |

```

ORG          0100H

TMIISR:
MOV          TL1,#66H          ;重设定定时参数
MOV          TH1,#0FCH
CPL          P1.0              ;测试端口
RETI

MAIN:
MOV          SP,#5FH
ORL          P_SW2,#80H        ;使能访问 XFR

MOV          P0M0,#00H
MOV          P0M1,#00H
MOV          P1M0,#00H
MOV          P1M1,#00H
MOV          P2M0,#00H
MOV          P2M1,#00H
MOV          P3M0,#00H
MOV          P3M1,#00H
MOV          P4M0,#00H
MOV          P4M1,#00H
MOV          P5M0,#00H
MOV          P5M1,#00H

MOV          TMOD,#10H          ;模式 1
MOV          TL1,#66H          ;65536-11.0592M/12/1000
MOV          TH1,#0FCH
SETB         TR1              ;启动定时器
SETB         ETI              ;使能定时器中断
SETB         EA

JMP          $

END

```

# 14.6.10 定时器 1（模式 2—8 位自动重载），用作定时

## C 语言代码

```

//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void TMI_Isr() interrupt 3
{
    P10 = !P10;          //测试端口
}

void main()
{
    P_SW2 /= 0x80;        //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;

```

```
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

TMOD = 0x20; //模式2
TL1 = 0xf4; //256-11.0592M/12/76K
TH1 = 0xf4;
TR1 = 1; //启动定时器
ET1 = 1; //使能定时器中断
EA = 1;

while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

```
P_SW2      DATA      0BAH

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

ORG        0000H
LJMP       MAIN
ORG        001BH
LJMP       TMIISR

ORG        0100H
TMIISR:
CPL        P1.0 ;测试端口
RETI

MAIN:
MOV        SP, #5FH
ORL        P_SW2, #80H ;使能访问 XFR

MOV        P0M0, #00H
MOV        P0M1, #00H
MOV        P1M0, #00H
MOV        P1M1, #00H
MOV        P2M0, #00H
```

```

MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      TMOD, #20H      ; 模式 2
MOV      TL1, #0F4H      ; 256-11.0592M/12/76K
MOV      TH1, #0F4H

SETB     TRI             ; 启动定时器
SETB     ETI             ; 使能定时器中断
SETB     EA

JMP      $

END

```

## 14.6.11 定时器 1（外部计数—扩展 T1 为外部下降沿中断）

### C 语言代码

// 测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```
void TM1_Isr() interrupt 3
```

```

{
    P10 = !P10;           // 测试端口
}

```

```
void main()
```

```

{
    P_SW2 /= 0x80;        // 使能访问 XFR

```

```

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

```

```
    TMOD = 0x40;          // 外部计数模式
```

```
    TL1 = 0xff;
```

```
    TH1 = 0xff;
```

```
    TRI = 1;
```

```
    ETI = 1;
```

```
    EA = 1;
```

```
    // 启动定时器
```

```
    // 使能定时器中断
```

```
while (1);  
}
```

汇编代码

;测试工作频率为 11.0592MHz

```
P_SW2      DATA      0BAH  
  
P1M1       DATA      091H  
P1M0       DATA      092H  
P0M1       DATA      093H  
P0M0       DATA      094H  
P2M1       DATA      095H  
P2M0       DATA      096H  
P3M1       DATA      0B1H  
P3M0       DATA      0B2H  
P4M1       DATA      0B3H  
P4M0       DATA      0B4H  
P5M1       DATA      0C9H  
P5M0       DATA      0CAH  
  
           ORG         0000H  
           LJMP        MAIN  
           ORG         001BH  
           LJMP        TMIISR  
  
TMIISR:    ORG         0100H  
  
           CPL         P1.0      ;测试端口  
           RETI  
  
MAIN:      MOV         SP, #5FH  
           ORL         P_SW2, #80H      ;使能访问 XFR  
  
           MOV         P0M0, #00H  
           MOV         P0M1, #00H  
           MOV         P1M0, #00H  
           MOV         P1M1, #00H  
           MOV         P2M0, #00H  
           MOV         P2M1, #00H  
           MOV         P3M0, #00H  
           MOV         P3M1, #00H  
           MOV         P4M0, #00H  
           MOV         P4M1, #00H  
           MOV         P5M0, #00H  
           MOV         P5M1, #00H  
  
           MOV         TMOD, #40H      ;外部计数模式  
           MOV         TL1, #0FFH  
           MOV         TH1, #0FFH  
           SETB        TRI      ;启动定时器  
           SETB        ETI      ;使能定时器中断  
           SETB        EA  
  
           JMP         $
```

---

---

END

---

---

## 14.6.12 定时器 1（测量脉宽—INT1 高电平宽度）

### C 语言代码

---

---

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void INT1_Isr() interrupt 2
```

```
{
    P0 = TL1;                //TL1 为测量值低字节
    P1 = TH1;                //TH1 为测量值高字节
    TL1 = 0x00;
    TH1 = 0x00;
}
```

```
void main()
```

```
{
    P_SW2 /= 0x80;          //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    AUXR = 0x40;            //IT 模式
    TMOD = 0x80;            //使能 GATE, INT1 为 1 时使能计时
    TL1 = 0x00;
    TH1 = 0x00;
    while (INT1);           //等待 INT1 为低
    TR1 = 1;                //启动定时器
    IT1 = 1;                //使能 INT1 下降沿中断
    EX1 = 1;
    EA = 1;

    while (1);
}
```

---

---

### 汇编代码

---

---

;测试工作频率为 11.0592MHz

```
P_SW2      DATA      0BAH
```

```
AUXR       DATA      8EH
```

```

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0013H
          LJMP         INT1ISR

INT1ISR:  ORG          0100H

          MOV          P0,TL1          ;TL1 为测量值低字节
          MOV          P1,TH1          ;TH1 为测量值高字节
          MOV          TL1,#00H
          MOV          TH1,#00H
          RETI

MAIN:

          MOV          SP,#5FH
          ORL          P_SW2,#80H      ;使能访问 XFR

          MOV          P0M0,#00H
          MOV          P0M1,#00H
          MOV          P1M0,#00H
          MOV          P1M1,#00H
          MOV          P2M0,#00H
          MOV          P2M1,#00H
          MOV          P3M0,#00H
          MOV          P3M1,#00H
          MOV          P4M0,#00H
          MOV          P4M1,#00H
          MOV          P5M0,#00H
          MOV          P5M1,#00H

          MOV          AUXR,#40H      ;IT 模式
          MOV          TMOD,#80H      ;使能 GATE,INT1 为 1 时使能计时
          MOV          TL1,#00H
          MOV          TH1,#00H

          JB           INT1,$          ;等待 INT1 为低
          SETB         TRI            ;启动定时器
          SETB         IT1            ;使能 INT1 下降沿中断
          SETB         EXI
          SETB         EA

          JMP          $

          END

```



## 14.6.13 定时器 1（模式 0），时钟分频输出

### C 语言代码

---

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void main()
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x00;                //模式 0
    TLI = 0x66;                //65536-11.0592M/12/1000
    TH1 = 0xfc;
    TR1 = 1;                    //启动定时器
    INTCLKO = 0x02;            //使能时钟输出

    while (1);
}
```

---

### 汇编代码

---

```
;测试工作频率为 11.0592MHz

P_SW2      DATA      0BAH

INTCLKO     DATA      8FH
P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

            ORG         0000H
            LJMP        MAIN
```

---

```

ORG          0100H

MAIN:

MOV          SP, #5FH
ORL          P_SW2, #80H          ;使能访问 XFR

MOV          P0M0, #00H
MOV          P0M1, #00H
MOV          P1M0, #00H
MOV          P1M1, #00H
MOV          P2M0, #00H
MOV          P2M1, #00H
MOV          P3M0, #00H
MOV          P3M1, #00H
MOV          P4M0, #00H
MOV          P4M1, #00H
MOV          P5M0, #00H
MOV          P5M1, #00H

MOV          TMOD, #00H          ;模式 0
MOV          TL1, #66H          ;65536-11.0592M/12/1000
MOV          TH1, #0FCH
SETB         TR1              ;启动定时器
MOV          INTCLKO, #02H      ;使能时钟输出

JMP          $

END

```

## 14.6.14 定时器 1（模式 0）做串口 1 波特率发生器

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define FOSC      11059200UL
#define BRT       (65536 - (FOSC / 115200+2) / 4)

```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```

bit      busy;
char     wptr;
char     rptr;
char     buffer[16];

```

```

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {

```

```
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void UartSendStr(char *p)
{
    while (*p)
    {
        UartSend(*p++);
    }
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;
    UartSendStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
```

```

    {
        UartSend(buffer[rptr++]);
        rptr &= 0x0f;
    }
}
}
```

汇编代码

;测试工作频率为 11.0592MHz

|                      |             |                        |           |
|----------------------|-------------|------------------------|-----------|
| <i>P_SW2</i>         | <i>DATA</i> | <i>0BAH</i>            |           |
| <i>AUXR</i>          | <i>DATA</i> | <i>8EH</i>             |           |
| <i>BUSY</i>          | <i>BIT</i>  | <i>20H.0</i>           |           |
| <i>WPTR</i>          | <i>DATA</i> | <i>21H</i>             |           |
| <i>RPTR</i>          | <i>DATA</i> | <i>22H</i>             |           |
| <i>BUFFER</i>        | <i>DATA</i> | <i>23H</i>             | ;16 bytes |
| <i>P1M1</i>          | <i>DATA</i> | <i>091H</i>            |           |
| <i>P1M0</i>          | <i>DATA</i> | <i>092H</i>            |           |
| <i>P0M1</i>          | <i>DATA</i> | <i>093H</i>            |           |
| <i>P0M0</i>          | <i>DATA</i> | <i>094H</i>            |           |
| <i>P2M1</i>          | <i>DATA</i> | <i>095H</i>            |           |
| <i>P2M0</i>          | <i>DATA</i> | <i>096H</i>            |           |
| <i>P3M1</i>          | <i>DATA</i> | <i>0B1H</i>            |           |
| <i>P3M0</i>          | <i>DATA</i> | <i>0B2H</i>            |           |
| <i>P4M1</i>          | <i>DATA</i> | <i>0B3H</i>            |           |
| <i>P4M0</i>          | <i>DATA</i> | <i>0B4H</i>            |           |
| <i>P5M1</i>          | <i>DATA</i> | <i>0C9H</i>            |           |
| <i>P5M0</i>          | <i>DATA</i> | <i>0CAH</i>            |           |
|                      | <i>ORG</i>  | <i>0000H</i>           |           |
|                      | <i>LJMP</i> | <i>MAIN</i>            |           |
|                      | <i>ORG</i>  | <i>0023H</i>           |           |
|                      | <i>LJMP</i> | <i>UART_ISR</i>        |           |
|                      | <i>ORG</i>  | <i>0100H</i>           |           |
| <i>UART_ISR:</i>     |             |                        |           |
|                      | <i>PUSH</i> | <i>ACC</i>             |           |
|                      | <i>PUSH</i> | <i>PSW</i>             |           |
|                      | <i>MOV</i>  | <i>PSW,#08H</i>        |           |
|                      | <i>JNB</i>  | <i>TI,CHKRI</i>        |           |
|                      | <i>CLR</i>  | <i>TI</i>              |           |
|                      | <i>CLR</i>  | <i>BUSY</i>            |           |
| <i>CHKRI:</i>        |             |                        |           |
|                      | <i>JNB</i>  | <i>RI,UARTISR_EXIT</i> |           |
|                      | <i>CLR</i>  | <i>RI</i>              |           |
|                      | <i>MOV</i>  | <i>A,WPTR</i>          |           |
|                      | <i>ANL</i>  | <i>A,#0FH</i>          |           |
|                      | <i>ADD</i>  | <i>A,#BUFFER</i>       |           |
|                      | <i>MOV</i>  | <i>R0,A</i>            |           |
|                      | <i>MOV</i>  | <i>@R0,SBUF</i>        |           |
|                      | <i>INC</i>  | <i>WPTR</i>            |           |
| <i>UARTISR_EXIT:</i> |             |                        |           |
|                      | <i>POP</i>  | <i>PSW</i>             |           |

```

        POP        ACC
        RETI

UART_INIT:
        MOV        SCON,#50H
        MOV        TMOD,#00H
        MOV        TL1,#0E8H                ;65536-11059200/115200/4=0FFE8H
        MOV        TH1,#0FFH
        SETB       TR1
        MOV        AUXR,#40H
        CLR        BUSY
        MOV        WPTR,#00H
        MOV        RPTR,#00H
        RET

UART_SEND:
        JB         BUSY,$
        SETB       BUSY
        MOV        SBUF,A
        RET

UART_SENDSTR:
        CLR        A
        MOVC       A,@A+DPTR
        JZ         SENDEND
        LCALL      UART_SEND
        INC        DPTR
        JMP        UART_SENDSTR

SENDEND:
        RET

MAIN:
        MOV        SP,#5FH
        ORL        P_SW2,#80H                ;使能访问 XFR

        MOV        P0M0,#00H
        MOV        P0M1,#00H
        MOV        P1M0,#00H
        MOV        P1M1,#00H
        MOV        P2M0,#00H
        MOV        P2M1,#00H
        MOV        P3M0,#00H
        MOV        P3M1,#00H
        MOV        P4M0,#00H
        MOV        P4M1,#00H
        MOV        P5M0,#00H
        MOV        P5M1,#00H

        LCALL      UART_INIT
        SETB       ES
        SETB       EA

        MOV        DPTR,#STRING
        LCALL      UART_SENDSTR

LOOP:
        MOV        A,RPTR
        XRL        A,WPTR
        ANL        A,#0FH

```

```

JZ      LOOP
MOV     A,RPTR
ANL     A,#0FH
ADD     A,#BUFFER
MOV     R0,A
MOV     A,@R0
LCALL   UART_SEND
INC     RPTR
JMP     LOOP

STRING:  DB      'Uart Test !',0DH,0AH,00H

END

```

## 14.6.15 定时器 1（模式 2）做串口 1 波特率发生器

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define FOSC      11059200UL
#define BRT      (256 - (FOSC / 115200+16) / 32)

```

//加 16 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```

bit      busy;
char      wptr;
char      rptr;
char      buffer[16];

```

```

void UartIsr() interrupt 4
{

```

```

    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}

```

```

void UartInit()
{

```

```

    SCON = 0x50;
    TMOD = 0x20;
    TL1 = BRT;
    TH1 = BRT;
    TR1 = 1;
    AUXR = 0x40;
}

```

```
wptr = 0x00;
rptr = 0x00;
busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void UartSendStr(char *p)
{
    while (*p)
    {
        UartSend(*p++);
    }
}

void main()
{
    P_SW2 /= 0x80; //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;
    UartSendStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UartSend(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

## 汇编代码

;测试工作频率为11.0592MHz

**P\_SW2**        **DATA**        **0BAH**

**AUXR**        **DATA**        **8EH**

|                      |             |                        |                                     |
|----------------------|-------------|------------------------|-------------------------------------|
| <i>BUSY</i>          | <i>BIT</i>  | <i>20H.0</i>           |                                     |
| <i>WPTR</i>          | <i>DATA</i> | <i>21H</i>             |                                     |
| <i>RPTR</i>          | <i>DATA</i> | <i>22H</i>             |                                     |
| <i>BUFFER</i>        | <i>DATA</i> | <i>23H</i>             | <i>;16 bytes</i>                    |
|                      |             |                        |                                     |
| <i>P1M1</i>          | <i>DATA</i> | <i>091H</i>            |                                     |
| <i>P1M0</i>          | <i>DATA</i> | <i>092H</i>            |                                     |
| <i>P0M1</i>          | <i>DATA</i> | <i>093H</i>            |                                     |
| <i>P0M0</i>          | <i>DATA</i> | <i>094H</i>            |                                     |
| <i>P2M1</i>          | <i>DATA</i> | <i>095H</i>            |                                     |
| <i>P2M0</i>          | <i>DATA</i> | <i>096H</i>            |                                     |
| <i>P3M1</i>          | <i>DATA</i> | <i>0B1H</i>            |                                     |
| <i>P3M0</i>          | <i>DATA</i> | <i>0B2H</i>            |                                     |
| <i>P4M1</i>          | <i>DATA</i> | <i>0B3H</i>            |                                     |
| <i>P4M0</i>          | <i>DATA</i> | <i>0B4H</i>            |                                     |
| <i>P5M1</i>          | <i>DATA</i> | <i>0C9H</i>            |                                     |
| <i>P5M0</i>          | <i>DATA</i> | <i>0CAH</i>            |                                     |
|                      |             |                        |                                     |
|                      | <i>ORG</i>  | <i>0000H</i>           |                                     |
|                      | <i>LJMP</i> | <i>MAIN</i>            |                                     |
|                      | <i>ORG</i>  | <i>0023H</i>           |                                     |
|                      | <i>LJMP</i> | <i>UART_ISR</i>        |                                     |
|                      |             |                        |                                     |
|                      | <i>ORG</i>  | <i>0100H</i>           |                                     |
|                      |             |                        |                                     |
| <i>UART_ISR:</i>     |             |                        |                                     |
|                      | <i>PUSH</i> | <i>ACC</i>             |                                     |
|                      | <i>PUSH</i> | <i>PSW</i>             |                                     |
|                      | <i>MOV</i>  | <i>PSW,#08H</i>        |                                     |
|                      |             |                        |                                     |
|                      | <i>JNB</i>  | <i>TI,CHKRI</i>        |                                     |
|                      | <i>CLR</i>  | <i>TI</i>              |                                     |
|                      | <i>CLR</i>  | <i>BUSY</i>            |                                     |
|                      |             |                        |                                     |
| <i>CHKRI:</i>        |             |                        |                                     |
|                      | <i>JNB</i>  | <i>RI,UARTISR_EXIT</i> |                                     |
|                      | <i>CLR</i>  | <i>RI</i>              |                                     |
|                      | <i>MOV</i>  | <i>A,WPTR</i>          |                                     |
|                      | <i>ANL</i>  | <i>A,#0FH</i>          |                                     |
|                      | <i>ADD</i>  | <i>A,#BUFFER</i>       |                                     |
|                      | <i>MOV</i>  | <i>R0,A</i>            |                                     |
|                      | <i>MOV</i>  | <i>@R0,SBUF</i>        |                                     |
|                      | <i>INC</i>  | <i>WPTR</i>            |                                     |
|                      |             |                        |                                     |
| <i>UARTISR_EXIT:</i> |             |                        |                                     |
|                      | <i>POP</i>  | <i>PSW</i>             |                                     |
|                      | <i>POP</i>  | <i>ACC</i>             |                                     |
|                      | <i>RETI</i> |                        |                                     |
|                      |             |                        |                                     |
| <i>UART_INIT:</i>    |             |                        |                                     |
|                      | <i>MOV</i>  | <i>SCON,#50H</i>       |                                     |
|                      | <i>MOV</i>  | <i>TMOD,#20H</i>       |                                     |
|                      | <i>MOV</i>  | <i>TL1,#0FDH</i>       | <i>;256-11059200/115200/32=0FDH</i> |
|                      | <i>MOV</i>  | <i>TH1,#0FDH</i>       |                                     |
|                      | <i>SETB</i> | <i>TR1</i>             |                                     |
|                      | <i>MOV</i>  | <i>AUXR,#40H</i>       |                                     |
|                      | <i>CLR</i>  | <i>BUSY</i>            |                                     |
|                      | <i>MOV</i>  | <i>WPTR,#00H</i>       |                                     |
|                      | <i>MOV</i>  | <i>RPTR,#00H</i>       |                                     |
|                      | <i>RET</i>  |                        |                                     |



**UART\_SEND:**

```

JB     BUSY,$
SETB   BUSY
MOV     SBUF,A
RET

```

**UART\_SENDSTR:**

```

CLR     A
MOVC    A,@A+DPTR
JZ      SENDEND
LCALL   UART_SEND
INC     DPTR
JMP     UART_SENDSTR

```

**SENDEND:**

```

RET

```

**MAIN:**

```

MOV     SP,#5FH
ORL     P_SW2,#80H           ;使能访问XFR

MOV     P0M0,#00H
MOV     P0M1,#00H
MOV     P1M0,#00H
MOV     P1M1,#00H
MOV     P2M0,#00H
MOV     P2M1,#00H
MOV     P3M0,#00H
MOV     P3M1,#00H
MOV     P4M0,#00H
MOV     P4M1,#00H
MOV     P5M0,#00H
MOV     P5M1,#00H

LCALL   UART_INIT
SETB    ES
SETB    EA

MOV     DPTR,#STRING
LCALL   UART_SENDSTR

```

**LOOP:**

```

MOV     A,RPTR
XRL     A,WPTR
ANL     A,#0FH
JZ      LOOP
MOV     A,RPTR
ANL     A,#0FH
ADD     A,#BUFFER
MOV     R0,A
MOV     A,@R0
LCALL   UART_SEND
INC     RPTR
JMP     LOOP

```

**STRING:**

```

DB      'Uart Test !',0DH,0AH,00H

```

```

END

```

## 14.6.16 定时器 2（16 位自动重载），用作定时

### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void TM2_Isr() interrupt 12
{
    P10 = !P10;                //测试端口
}

void main()
{
    P_SW2 /= 0x80;              //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T2L = 0x66;                 //65536-11.0592M/12/1000
    T2H = 0xfc;
    AUXR = 0x10;                //启动定时器
    IE2 = ET2;                  //使能定时器中断
    EA = 1;

    while (1);
}
```

### 汇编代码

;测试工作频率为 11.0592MHz

|          |      |      |
|----------|------|------|
| P_SW2    | DATA | 0BAH |
| T2L      | DATA | 0D7H |
| T2H      | DATA | 0D6H |
| AUXR     | DATA | 8EH  |
| IE2      | DATA | 0AFH |
| ET2      | EQU  | 04H  |
| AUXINTIF | DATA | 0EFH |
| T2IF     | EQU  | 01H  |
| P1M1     | DATA | 091H |
| P1M0     | DATA | 092H |

```
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0063H
          LJMP         TM2ISR

TM2ISR:    ORG          0100H

          CPL          P1.0          ;测试端口
          RETI

MAIN:

          MOV          SP, #5FH
          ORL          P_SW2, #80H    ;使能访问 XFR

          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          MOV          T2L, #66H      ;65536-11.0592M/12/1000
          MOV          T2H, #0FCH
          MOV          AUXR, #10H     ;启动定时器
          MOV          IE2, #ET2     ;使能定时器中断
          SETB         EA

          JMP          $

          END
```

14.6.17 定时器 2（外部计数—扩展 T2 为外部下降沿中断）

C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void TM2_Isr() interrupt 12
{
    P10 = !P10;                                //测试端口
}

void main()
{
    P_SW2 /= 0x80;                               //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T2L = 0xff;
    T2H = 0xff;
    AUXR = 0x18;                                //设置外部计数模式并启动定时器
    IE2 = ET2;                                  //使能定时器中断
    EA = 1;

    while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

|                 |             |             |
|-----------------|-------------|-------------|
| <i>P_SW2</i>    | <i>DATA</i> | <i>0BAH</i> |
| <i>T2L</i>      | <i>DATA</i> | <i>0D7H</i> |
| <i>T2H</i>      | <i>DATA</i> | <i>0D6H</i> |
| <i>AUXR</i>     | <i>DATA</i> | <i>8EH</i>  |
| <i>IE2</i>      | <i>DATA</i> | <i>0AFH</i> |
| <i>ET2</i>      | <i>EQU</i>  | <i>04H</i>  |
| <i>AUXINTIF</i> | <i>DATA</i> | <i>0EFH</i> |
| <i>T2IF</i>     | <i>EQU</i>  | <i>01H</i>  |
| <i>P1M1</i>     | <i>DATA</i> | <i>091H</i> |
| <i>P1M0</i>     | <i>DATA</i> | <i>092H</i> |
| <i>P0M1</i>     | <i>DATA</i> | <i>093H</i> |
| <i>P0M0</i>     | <i>DATA</i> | <i>094H</i> |
| <i>P2M1</i>     | <i>DATA</i> | <i>095H</i> |
| <i>P2M0</i>     | <i>DATA</i> | <i>096H</i> |
| <i>P3M1</i>     | <i>DATA</i> | <i>0B1H</i> |
| <i>P3M0</i>     | <i>DATA</i> | <i>0B2H</i> |
| <i>P4M1</i>     | <i>DATA</i> | <i>0B3H</i> |
| <i>P4M0</i>     | <i>DATA</i> | <i>0B4H</i> |
| <i>P5M1</i>     | <i>DATA</i> | <i>0C9H</i> |
| <i>P5M0</i>     | <i>DATA</i> | <i>0CAH</i> |

```

    ORG      0000H
    LJMP     MAIN
    ORG      0063H
    LJMP     TM2ISR

TM2ISR:
    ORG      0100H

    CPL      P1.0          ;测试端口
    RETI

MAIN:
    MOV      SP, #5FH
    ORL      P_SW2, #80H   ;使能访问 XFR

    MOV      P0M0, #00H
    MOV      P0M1, #00H
    MOV      P1M0, #00H
    MOV      P1M1, #00H
    MOV      P2M0, #00H
    MOV      P2M1, #00H
    MOV      P3M0, #00H
    MOV      P3M1, #00H
    MOV      P4M0, #00H
    MOV      P4M1, #00H
    MOV      P5M0, #00H
    MOV      P5M1, #00H

    MOV      T2L, #0FFH
    MOV      T2H, #0FFH
    MOV      AUXR, #18H    ;设置外部计数模式并启动定时器
    MOV      IE2, #ET2    ;使能定时器中断
    SETB     EA

    JMP      $

END
```

### 14.6.18 定时器 2，时钟分频输出

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void main()
{
    P_SW2 /= 0x80;          //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
```

```
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

T2L = 0x66; //65536-11.0592M/12/1000
T2H = 0xfc;
AUXR = 0x10; //启动定时器
INTCLKO = 0x04; //使能时钟输出

while (1);
}
```

汇编代码

;测试工作频率为11.0592MHz

|         |      |                      |
|---------|------|----------------------|
| P_SW2   | DATA | 0BAH                 |
| T2L     | DATA | 0D7H                 |
| T2H     | DATA | 0D6H                 |
| AUXR    | DATA | 8EH                  |
| INTCLKO | DATA | 8FH                  |
| P1M1    | DATA | 091H                 |
| P1M0    | DATA | 092H                 |
| P0M1    | DATA | 093H                 |
| P0M0    | DATA | 094H                 |
| P2M1    | DATA | 095H                 |
| P2M0    | DATA | 096H                 |
| P3M1    | DATA | 0B1H                 |
| P3M0    | DATA | 0B2H                 |
| P4M1    | DATA | 0B3H                 |
| P4M0    | DATA | 0B4H                 |
| P5M1    | DATA | 0C9H                 |
| P5M0    | DATA | 0CAH                 |
|         | ORG  | 0000H                |
|         | LJMP | MAIN                 |
|         | ORG  | 0100H                |
| MAIN:   |      |                      |
|         | MOV  | SP, #5FH             |
|         | ORL  | P_SW2, #80H ;使能访问XFR |
|         | MOV  | P0M0, #00H           |
|         | MOV  | P0M1, #00H           |
|         | MOV  | P1M0, #00H           |
|         | MOV  | P1M1, #00H           |
|         | MOV  | P2M0, #00H           |
|         | MOV  | P2M1, #00H           |
|         | MOV  | P3M0, #00H           |
|         | MOV  | P3M1, #00H           |
|         | MOV  | P4M0, #00H           |
|         | MOV  | P4M1, #00H           |
|         | MOV  | P5M0, #00H           |
|         | MOV  | P5M1, #00H           |

```
MOV    T2L,#66H           ;65536-11.0592M/12/1000
MOV    T2H,#0FCH
MOV    AUXR,#10H          ;启动定时器
MOV    INTCLKO,#04H        ;使能时钟输出

JMP    $

END
```

## 14.6.19 定时器 2 做串口 1 波特率发生器

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT       (65536 - (FOSC / 115200+2) / 4)
```

//加2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
bit     busy;
char    wptr;
char    rptr;
char    buffer[16];
```

```
void UartIsr() interrupt 4
```

```
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}
```

```
void UartInit()
```

```
{
    SCON = 0x50;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x15;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}
```

```
void UartSend(char dat)
```

```

{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void UartSendStr(char *p)
{
    while (*p)
    {
        UartSend(*p++);
    }
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;
    UartSendStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UartSend(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}

```

## 汇编代码

;测试工作频率为11.0592MHz

|              |             |              |
|--------------|-------------|--------------|
| <i>P_SW2</i> | <i>DATA</i> | <i>0BAH</i>  |
| <i>AUXR</i>  | <i>DATA</i> | <i>8EH</i>   |
| <i>T2H</i>   | <i>DATA</i> | <i>0D6H</i>  |
| <i>T2L</i>   | <i>DATA</i> | <i>0D7H</i>  |
| <i>BUSY</i>  | <i>BIT</i>  | <i>20H.0</i> |
| <i>WPTR</i>  | <i>DATA</i> | <i>21H</i>   |
| <i>RPTR</i>  | <i>DATA</i> | <i>22H</i>   |



```

BUFFER      DATA      23H                               ;16 bytes

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

            ORG          0000H
            LJMP         MAIN
            ORG          0023H
            LJMP         UART_ISR

            ORG          0100H

UART_ISR:
            PUSH         ACC
            PUSH         PSW
            MOV          PSW,#08H

            JNB          TI,CHKRI
            CLR          TI
            CLR          BUSY

CHKRI:
            JNB          RI,UARTISR_EXIT
            CLR          RI
            MOV          A,WPTR
            ANL          A,#0FH
            ADD          A,#BUFFER
            MOV          R0,A
            MOV          @R0,SBUF
            INC          WPTR

UARTISR_EXIT:
            POP          PSW
            POP          ACC
            RETI

UART_INIT:
            MOV          SCON,#50H
            MOV          T2L,#0E8H                               ;65536-11059200/115200/4=0FFE8H
            MOV          T2H,#0FFH
            MOV          AUXR,#15H
            CLR          BUSY
            MOV          WPTR,#00H
            MOV          RPTR,#00H
            RET

UART_SEND:
            JB           BUSY,$
            SETB         BUSY
            MOV          SBUF,A
            RET

```

**UART\_SENDSTR:**

```

CLR      A
MOVC     A,@A+DPTR
JZ       SENDEND
LCALL    UART_SEND
INC      DPTR
JMP      UART_SENDSTR

```

**SENDEND:**

```
RET
```

**MAIN:**

```

MOV      SP,#5FH
ORL      P_SW2,#80H      ;使能访问 XFR

MOV      P0M0,#00H
MOV      P0M1,#00H
MOV      P1M0,#00H
MOV      P1M1,#00H
MOV      P2M0,#00H
MOV      P2M1,#00H
MOV      P3M0,#00H
MOV      P3M1,#00H
MOV      P4M0,#00H
MOV      P4M1,#00H
MOV      P5M0,#00H
MOV      P5M1,#00H

LCALL    UART_INIT
SETB     ES
SETB     EA

MOV      DPTR,#STRING
LCALL    UART_SENDSTR

```

**LOOP:**

```

MOV      A,RPTR
XRL      A,WPTR
ANL      A,#0FH
JZ       LOOP
MOV      A,RPTR
ANL      A,#0FH
ADD      A,#BUFFER
MOV      R0,A
MOV      A,@R0
LCALL    UART_SEND
INC      RPTR
JMP      LOOP

```

```
STRING:  DB      'Uart Test !',0DH,0AH,00H
```

```
END
```

## 14.6.20 定时器 2 做串口 2 波特率发生器

## C 语言代码

//测试工作频率为11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC 11059200UL
#define BRT (65536 - (FOSC / 115200 + 2) / 4)
```

//加2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
bit busy;
char wptr;
char rptr;
char buffer[16];
```

```
void Uart2Isr() interrupt 8
```

```
{
    if (S2CON & 0x02)
    {
        S2CON &= ~0x02;
        busy = 0;
    }
    if (S2CON & 0x01)
    {
        S2CON &= ~0x01;
        buffer[wptr++] = S2BUF;
        wptr &= 0x0f;
    }
}
```

```
void Uart2Init()
```

```
{
    S2CON = 0x10;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x14;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}
```

```
void Uart2Send(char dat)
```

```
{
    while (busy);
    busy = 1;
    S2BUF = dat;
}
```

```
void Uart2SendStr(char *p)
```

```
{
    while (*p)
    {
        Uart2Send(*p++);
    }
}
```

```
void main()
```

```
{
```

*P\_SW2* /= 0x80; //使能访问XFR

*P0M0* = 0x00;

*P0M1* = 0x00;

*P1M0* = 0x00;

*P1M1* = 0x00;

*P2M0* = 0x00;

*P2M1* = 0x00;

*P3M0* = 0x00;

*P3M1* = 0x00;

*P4M0* = 0x00;

*P4M1* = 0x00;

*P5M0* = 0x00;

*P5M1* = 0x00;

*Uart2Init*();

*IE2* = 0x01;

*EA* = 1;

*Uart2SendStr*("Uart Test !\r\n");

*while* (1)

{

*if* (*rp*tr != *w*ptr)

{

*Uart2Send*(*buffer*[*rp*tr++]);

*rp*tr &= 0x0f;

}

}

}

## 汇编代码

;测试工作频率为 11.0592MHz

|               |      |       |           |
|---------------|------|-------|-----------|
| <i>P_SW2</i>  | DATA | 0BAH  |           |
| <i>AUXR</i>   | DATA | 8EH   |           |
| <i>T2H</i>    | DATA | 0D6H  |           |
| <i>T2L</i>    | DATA | 0D7H  |           |
| <i>S2CON</i>  | DATA | 9AH   |           |
| <i>S2BUF</i>  | DATA | 9BH   |           |
| <i>IE2</i>    | DATA | 0AFH  |           |
| <i>BUSY</i>   | BIT  | 20H.0 |           |
| <i>WPTR</i>   | DATA | 21H   |           |
| <i>RPTR</i>   | DATA | 22H   |           |
| <i>BUFFER</i> | DATA | 23H   | ;16 bytes |
| <i>P1M1</i>   | DATA | 091H  |           |
| <i>P1M0</i>   | DATA | 092H  |           |
| <i>P0M1</i>   | DATA | 093H  |           |
| <i>P0M0</i>   | DATA | 094H  |           |
| <i>P2M1</i>   | DATA | 095H  |           |
| <i>P2M0</i>   | DATA | 096H  |           |
| <i>P3M1</i>   | DATA | 0B1H  |           |
| <i>P3M0</i>   | DATA | 0B2H  |           |
| <i>P4M1</i>   | DATA | 0B3H  |           |
| <i>P4M0</i>   | DATA | 0B4H  |           |
| <i>P5M1</i>   | DATA | 0C9H  |           |

```

P5M0      DATA      0CAH

           ORG         0000H
           LJMP        MAIN
           ORG         0043H
           LJMP        UART2_ISR

           ORG         0100H

UART2_ISR:
           PUSH        ACC
           PUSH        PSW
           MOV         PSW,#08H

           MOV         A,S2CON
           JNB         ACC.1,CHKRI
           ANL         S2CON,#NOT 02H
           CLR         BUSY

CHKRI:
           JNB         ACC.0,UART2ISR_EXIT
           ANL         S2CON,#NOT 01H
           MOV         A,WPTR
           ANL         A,#0FH
           ADD         A,#BUFFER
           MOV         R0,A
           MOV         @R0,S2BUF
           INC         WPTR

UART2ISR_EXIT:
           POP         PSW
           POP         ACC
           RETI

UART2_INIT:
           MOV         S2CON,#10H
           MOV         T2L,#0E8H
           MOV         T2H,#0FFH
           MOV         AUXR,#14H
           CLR         BUSY
           MOV         WPTR,#00H
           MOV         RPTR,#00H
           RET

UART2_SEND:
           JB          BUSY,$
           SETB        BUSY
           MOV         S2BUF,A
           RET

UART2_SENDSTR:
           CLR         A
           MOVC        A,@A+DPTR
           JZ          SEND2END
           LCALL       UART2_SEND
           INC         DPTR
           JMP         UART2_SENDSTR

SEND2END:
           RET

MAIN:

```

```

MOV     SP, #5FH
ORL     P_SW2, #80H           ;使能访问 XFR

MOV     P0M0, #00H
MOV     P0M1, #00H
MOV     P1M0, #00H
MOV     P1M1, #00H
MOV     P2M0, #00H
MOV     P2M1, #00H
MOV     P3M0, #00H
MOV     P3M1, #00H
MOV     P4M0, #00H
MOV     P4M1, #00H
MOV     P5M0, #00H
MOV     P5M1, #00H

LCALL   UART2_INIT
MOV     IE2, #01H
SETB    EA

MOV     DPTR, #STRING
LCALL   UART2_SENDSTR

LOOP:
MOV     A, RPTR
XRL     A, WPTR
ANL     A, #0FH
JZ      LOOP
MOV     A, RPTR
ANL     A, #0FH
ADD     A, #BUFFER
MOV     R0, A
MOV     A, @R0
LCALL   UART2_SEND
INC     RPTR
JMP     LOOP

STRING:  DB      'Uart Test !', 0DH, 0AH, 00H

END

```

## 14.6.21 定时器 2 做串口 3 波特率发生器

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT      (65536 - (FOSC / 115200 + 2) / 4)
```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
bit      busy;
char     wptr;
```

```
char    rptr;
char    buffer[16];

void Uart3Isr() interrupt 17
{
    if (S3CON & 0x02)
    {
        S3CON &= ~0x02;
        busy = 0;
    }
    if (S3CON & 0x01)
    {
        S3CON &= ~0x01;
        buffer[wptr++] = S3BUF;
        wptr &= 0x0f;
    }
}

void Uart3Init()
{
    S3CON = 0x10;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x14;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart3Send(char dat)
{
    while (busy);
    busy = 1;
    S3BUF = dat;
}

void Uart3SendStr(char *p)
{
    while (*p)
    {
        Uart3Send(*p++);
    }
}

void main()
{
    P_SW2 |= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
```

```

P5M1 = 0x00;

Uart3Init();
IE2 = 0x08;
EA = 1;
Uart3SendStr("Uart Test !\r\n");

while (1)
{
    if (rptr != wptr)
    {
        Uart3Send(buffer[rptr++]);
        rptr &= 0x0f;
    }
}
}

```

# 汇编代码

;测试工作频率为 11.0592MHz

|            |      |           |           |
|------------|------|-----------|-----------|
| P_SW2      | DATA | 0BAH      |           |
| AUXR       | DATA | 8EH       |           |
| T2H        | DATA | 0D6H      |           |
| T2L        | DATA | 0D7H      |           |
| S3CON      | DATA | 0ACH      |           |
| S3BUF      | DATA | 0ADH      |           |
| IE2        | DATA | 0AFH      |           |
| BUSY       | BIT  | 20H.0     |           |
| WPTR       | DATA | 21H       |           |
| RPTR       | DATA | 22H       |           |
| BUFFER     | DATA | 23H       | ;16 bytes |
| P0M1       | DATA | 093H      |           |
| P0M0       | DATA | 094H      |           |
| P1M1       | DATA | 091H      |           |
| P1M0       | DATA | 092H      |           |
| P2M1       | DATA | 095H      |           |
| P2M0       | DATA | 096H      |           |
| P3M1       | DATA | 0B1H      |           |
| P3M0       | DATA | 0B2H      |           |
| P4M1       | DATA | 0B3H      |           |
| P4M0       | DATA | 0B4H      |           |
| P5M1       | DATA | 0C9H      |           |
| P5M0       | DATA | 0CAH      |           |
|            | ORG  | 0000H     |           |
|            | LJMP | MAIN      |           |
|            | ORG  | 008BH     |           |
|            | LJMP | UART3_ISR |           |
|            | ORG  | 0100H     |           |
| UART3_ISR: |      |           |           |
|            | PUSH | ACC       |           |
|            | PUSH | PSW       |           |
|            | MOV  | PSW,#08H  |           |



```

MOV      A,S3CON
JNB      ACC.1,CHKRI
ANL      S3CON,#NOT 02H
CLR      BUSY

CHKRI:
JNB      ACC.0,UART3ISR_EXIT
ANL      S3CON,#NOT 01H
MOV      A,WPTR
ANL      A,#0FH
ADD      A,#BUFFER
MOV      R0,A
MOV      @R0,S3BUF
INC      WPTR

UART3ISR_EXIT:
POP      PSW
POP      ACC
RETI

UART3_INIT:
MOV      S3CON,#10H
MOV      T2L,#0E8H                      ;65536-11059200/115200/4=0FFE8H
MOV      T2H,#0FFH
MOV      AUXR,#14H
CLR      BUSY
MOV      WPTR,#00H
MOV      RPTR,#00H
RET

UART3_SEND:
JB       BUSY,$
SETB     BUSY
MOV      S3BUF,A
RET

UART3_SENDSTR:
CLR      A
MOVC     A,@A+DPTR
JZ       SEND3END
LCALL    UART3_SEND
INC      DPTR
JMP      UART3_SENDSTR

SEND3END:
RET

MAIN:
MOV      SP,#5FH
ORL      P_SW2,#80H                      ;使能访问 XFR

MOV      P0M0,#00H
MOV      P0M1,#00H
MOV      P1M0,#00H
MOV      P1M1,#00H
MOV      P2M0,#00H
MOV      P2M1,#00H
MOV      P3M0,#00H
MOV      P3M1,#00H
MOV      P4M0,#00H
MOV      P4M1,#00H

```

```

MOV      P5M0, #00H
MOV      P5M1, #00H

LCALL    UART3_INIT
MOV      IE2, #08H
SETB     EA

MOV      DPTR, #STRING
LCALL    UART3_SENDSTR

LOOP:
MOV      A, RPTR
XRL      A, WPTR
ANL      A, #0FH
JZ       LOOP
MOV      A, RPTR
ANL      A, #0FH
ADD      A, #BUFFER
MOV      R0, A
MOV      A, @R0
LCALL    UART3_SEND
INC      RPTR
JMP      LOOP

STRING:   DB      'Uart Test !', 0DH, 0AH, 00H

END

```

## 14.6.22 定时器 2 做串口 4 波特率发生器

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define FOSC      11059200UL
#define BRT      (65536 - (FOSC / 115200 + 2) / 4)

```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```

bit      busy;
char      wptr;
char      rptr;
char      buffer[16];

```

```
void Uart4Isr() interrupt 18
```

```

{
    if (S4CON & 0x02)
    {
        S4CON &= ~0x02;
        busy = 0;
    }
    if (S4CON & 0x01)
    {

```

```
        S4CON &= ~0x01;
        buffer[wptr++] = S4BUF;
        wptr &= 0x0f;
    }
}

void Uart4Init()
{
    S4CON = 0x10;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x14;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart4Send(char dat)
{
    while (busy);
    busy = 1;
    S4BUF = dat;
}

void Uart4SendStr(char *p)
{
    while (*p)
    {
        Uart4Send(*p++);
    }
}

void main()
{
    P_SW2 /= 0x80; //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    Uart4Init();
    IE2 = 0x10;
    EA = 1;
    Uart4SendStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            Uart4Send(buffer[rptr++]);
        }
    }
}
```

```
    rptr &= 0x0f;
}
}
}
```

汇编代码

;测试工作频率为 11.0592MHz

```
P_SW2      DATA      0BAH

AUXR        DATA      8EH
T2H         DATA      0D6H
T2L         DATA      0D7H
S4CON       DATA      84H
S4BUF       DATA      85H
IE2         DATA      0AFH

BUSY        BIT        20H.0
WPTR        DATA      21H
RPTR        DATA      22H
BUFFER      DATA      23H                ;16 bytes

P0M1        DATA      093H
P0M0        DATA      094H
P1M1        DATA      091H
P1M0        DATA      092H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

ORG         0000H
LJMP        MAIN
ORG         0093H
LJMP        UART4_ISR

ORG         0100H

UART4_ISR:
    PUSH    ACC
    PUSH    PSW
    MOV     PSW,#08H

    MOV     A,S4CON
    JNB     ACC.1,CHKRI
    ANL     S4CON,#NOT 02H
    CLR     BUSY

CHKRI:
    JNB     ACC.0,UART4ISR_EXIT
    ANL     S4CON,#NOT 01H
    MOV     A,WPTR
    ANL     A,#0FH
    ADD     A,#BUFFER
    MOV     R0,A
```

```

        MOV        @R0,S4BUF
        INC        WPTR
UART4ISR_EXIT:
        POP        PSW
        POP        ACC
        RETI

UART4_INIT:
        MOV        S4CON,#10H
        MOV        T2L,#0E8H                ;65536-11059200/115200/4=0FFE8H
        MOV        T2H,#0FFH
        MOV        AUXR,#14H
        CLR        BUSY
        MOV        WPTR,#00H
        MOV        RPTR,#00H
        RET

UART4_SEND:
        JB         BUSY,$
        SETB       BUSY
        MOV        S4BUF,A
        RET

UART4_SENDSTR:
        CLR        A
        MOVC       A,@A+DPTR
        JZ         SEND4END
        LCALL      UART4_SEND
        INC        DPTR
        JMP        UART4_SENDSTR
SEND4END:
        RET

MAIN:
        MOV        SP,#5FH
        ORL        P_SW2,#80H                ;使能访问 XFR

        MOV        P0M0,#00H
        MOV        P0M1,#00H
        MOV        P1M0,#00H
        MOV        P1M1,#00H
        MOV        P2M0,#00H
        MOV        P2M1,#00H
        MOV        P3M0,#00H
        MOV        P3M1,#00H
        MOV        P4M0,#00H
        MOV        P4M1,#00H
        MOV        P5M0,#00H
        MOV        P5M1,#00H

        LCALL      UART4_INIT
        MOV        IE2,#10H
        SETB       EA

        MOV        DPTR,#STRING
        LCALL      UART4_SENDSTR

LOOP:
        MOV        A,RPTR

```

```

XRL      A,WPTR
ANL      A,#0FH
JZ       LOOP
MOV      A,RPTR
ANL      A,#0FH
ADD      A,#BUFFER
MOV      R0,A
MOV      A,@R0
LCALL    UART4_SEND
INC      RPTR
JMP      LOOP

STRING:   DB      'Uart Test !',0DH,0AH,00H

END
```

## 14.6.23 定时器 3（16 位自动重载），用作定时

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void TM3_Isr() interrupt 19
```

```
{
    P10 = !P10;           //测试端口
}
```

```
void main()
```

```
{
    P_SW2 /= 0x80;        //使能访问 XFR
```

```
    P0M0 = 0x00;
```

```
    P0M1 = 0x00;
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P2M0 = 0x00;
```

```
    P2M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P4M0 = 0x00;
```

```
    P4M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
    T3L = 0x66;
```

//65536-11.0592M/12/1000

```
    T3H = 0xfc;
```

```
    T4T3M = 0x08;
```

//启动定时器

```
    IE2 = ET3;
```

//使能定时器中断

```
    EA = 1;
```

```
    while (1);
```

```
}
```

## 汇编代码

;测试工作频率为11.0592MHz

```

P_SW2      DATA      0BAH

T4T3M      DATA      0D1H
T4L        DATA      0D3H
T4H        DATA      0D2H
T3L        DATA      0D5H
T3H        DATA      0D4H
T2L        DATA      0D7H
T2H        DATA      0D6H
AUXR       DATA      8EH
IE2        DATA      0AFH
ET2        EQU        04H
ET3        EQU        20H
ET4        EQU        40H
AUXINTIF   DATA      0EFH
T2IF       EQU        01H
T3IF       EQU        02H
T4IF       EQU        04H

P1M1       DATA      091H
P1M0       DATA      092H
P0M1       DATA      093H
P0M0       DATA      094H
P2M1       DATA      095H
P2M0       DATA      096H
P3M1       DATA      0B1H
P3M0       DATA      0B2H
P4M1       DATA      0B3H
P4M0       DATA      0B4H
P5M1       DATA      0C9H
P5M0       DATA      0CAH

            ORG        0000H
            LJMP       MAIN
            ORG        009BH
            LJMP       TM3ISR

            ORG        0100H
TM3ISR:
            CPL        P1.0          ;测试端口
            RETI

MAIN:
            MOV        SP, #5FH
            ORL        P_SW2, #80H   ;使能访问XFR

            MOV        P0M0, #00H
            MOV        P0M1, #00H
            MOV        P1M0, #00H
            MOV        P1M1, #00H
            MOV        P2M0, #00H
            MOV        P2M1, #00H
            MOV        P3M0, #00H
            MOV        P3M1, #00H
            MOV        P4M0, #00H
            MOV        P4M1, #00H

```

```
MOV    P5M0, #00H
MOV    P5M1, #00H

MOV    T3L, #66H                ;65536-11.0592M/12/1000
MOV    T3H, #0FCH
MOV    T4T3M, #08H             ;启动定时器
MOV    IE2, #ET3               ;使能定时器中断
SETB   EA

JMP    $

END
```

## 14.6.24 定时器 3（外部计数—扩展 T3 为外部下降沿中断）

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void TM3_Isr() interrupt 19
```

```
{
    P10 = !P10;                //测试端口
}
```

```
void main()
```

```
{
    P_SW2 |= 0x80;             //使能访问 XFR
```

```
    P0M0 = 0x00;
```

```
    P0M1 = 0x00;
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P2M0 = 0x00;
```

```
    P2M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P4M0 = 0x00;
```

```
    P4M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
    T3L = 0xff;
```

```
    T3H = 0xff;
```

```
    T4T3M = 0x0c;
```

```
    IE2 = ET3;
```

```
    EA = 1;
```

```
    while (1);
```

```
}
```

### 汇编代码

;测试工作频率为 11.0592MHz



|                 |             |                    |                  |
|-----------------|-------------|--------------------|------------------|
| <i>P_SW2</i>    | <i>DATA</i> | <i>0BAH</i>        |                  |
|                 |             |                    |                  |
| <i>T4T3M</i>    | <i>DATA</i> | <i>0D1H</i>        |                  |
| <i>T4L</i>      | <i>DATA</i> | <i>0D3H</i>        |                  |
| <i>T4H</i>      | <i>DATA</i> | <i>0D2H</i>        |                  |
| <i>T3L</i>      | <i>DATA</i> | <i>0D5H</i>        |                  |
| <i>T3H</i>      | <i>DATA</i> | <i>0D4H</i>        |                  |
| <i>T2L</i>      | <i>DATA</i> | <i>0D7H</i>        |                  |
| <i>T2H</i>      | <i>DATA</i> | <i>0D6H</i>        |                  |
| <i>AUXR</i>     | <i>DATA</i> | <i>8EH</i>         |                  |
| <i>IE2</i>      | <i>DATA</i> | <i>0AFH</i>        |                  |
| <i>ET2</i>      | <i>EQU</i>  | <i>04H</i>         |                  |
| <i>ET3</i>      | <i>EQU</i>  | <i>20H</i>         |                  |
| <i>ET4</i>      | <i>EQU</i>  | <i>40H</i>         |                  |
| <i>AUXINTIF</i> | <i>DATA</i> | <i>0EFH</i>        |                  |
| <i>T2IF</i>     | <i>EQU</i>  | <i>01H</i>         |                  |
| <i>T3IF</i>     | <i>EQU</i>  | <i>02H</i>         |                  |
| <i>T4IF</i>     | <i>EQU</i>  | <i>04H</i>         |                  |
|                 |             |                    |                  |
| <i>P1M1</i>     | <i>DATA</i> | <i>091H</i>        |                  |
| <i>P1M0</i>     | <i>DATA</i> | <i>092H</i>        |                  |
| <i>P0M1</i>     | <i>DATA</i> | <i>093H</i>        |                  |
| <i>P0M0</i>     | <i>DATA</i> | <i>094H</i>        |                  |
| <i>P2M1</i>     | <i>DATA</i> | <i>095H</i>        |                  |
| <i>P2M0</i>     | <i>DATA</i> | <i>096H</i>        |                  |
| <i>P3M1</i>     | <i>DATA</i> | <i>0B1H</i>        |                  |
| <i>P3M0</i>     | <i>DATA</i> | <i>0B2H</i>        |                  |
| <i>P4M1</i>     | <i>DATA</i> | <i>0B3H</i>        |                  |
| <i>P4M0</i>     | <i>DATA</i> | <i>0B4H</i>        |                  |
| <i>P5M1</i>     | <i>DATA</i> | <i>0C9H</i>        |                  |
| <i>P5M0</i>     | <i>DATA</i> | <i>0CAH</i>        |                  |
|                 |             |                    |                  |
|                 | <i>ORG</i>  | <i>0000H</i>       |                  |
|                 | <i>LJMP</i> | <i>MAIN</i>        |                  |
|                 | <i>ORG</i>  | <i>009BH</i>       |                  |
|                 | <i>LJMP</i> | <i>TM3ISR</i>      |                  |
|                 |             |                    |                  |
|                 | <i>ORG</i>  | <i>0100H</i>       |                  |
| <i>TM3ISR:</i>  |             |                    |                  |
|                 | <i>CPL</i>  | <i>P1.0</i>        | <i>;测试端口</i>     |
|                 | <i>RETI</i> |                    |                  |
|                 |             |                    |                  |
| <i>MAIN:</i>    |             |                    |                  |
|                 | <i>MOV</i>  | <i>SP, #5FH</i>    |                  |
|                 | <i>ORL</i>  | <i>P_SW2, #80H</i> | <i>;使能访问 XFR</i> |
|                 |             |                    |                  |
|                 | <i>MOV</i>  | <i>P0M0, #00H</i>  |                  |
|                 | <i>MOV</i>  | <i>P0M1, #00H</i>  |                  |
|                 | <i>MOV</i>  | <i>P1M0, #00H</i>  |                  |
|                 | <i>MOV</i>  | <i>P1M1, #00H</i>  |                  |
|                 | <i>MOV</i>  | <i>P2M0, #00H</i>  |                  |
|                 | <i>MOV</i>  | <i>P2M1, #00H</i>  |                  |
|                 | <i>MOV</i>  | <i>P3M0, #00H</i>  |                  |
|                 | <i>MOV</i>  | <i>P3M1, #00H</i>  |                  |
|                 | <i>MOV</i>  | <i>P4M0, #00H</i>  |                  |
|                 | <i>MOV</i>  | <i>P4M1, #00H</i>  |                  |
|                 | <i>MOV</i>  | <i>P5M0, #00H</i>  |                  |
|                 | <i>MOV</i>  | <i>P5M1, #00H</i>  |                  |

```
MOV    T3L,#0FFH
MOV    T3H,#0FFH
MOV    T4T3M,#0CH           ;设置外部计数模式并启动定时器
MOV    IE2,#ET3             ;使能定时器中断
SETB   EA

JMP    $

END
```

### 14.6.25 定时器 3，时钟分频输出

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void main()
{
    P_SW2 /= 0x80;           //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T3L = 0x66;              //65536-11.0592M/12/1000
    T3H = 0xfc;
    T4T3M = 0x09;           //使能时钟输出并启动定时器

    while (1);
}
```

#### 汇编代码

```
;测试工作频率为 11.0592MHz

P_SW2      DATA      0BAH

T4T3M      DATA      0D1H
T4L         DATA      0D3H
T4H         DATA      0D2H
T3L         DATA      0D5H
T3H         DATA      0D4H
T2L         DATA      0D7H
```

```
T2H      DATA      0D6H

P1M1     DATA      091H
P1M0     DATA      092H
P0M1     DATA      093H
P0M0     DATA      094H
P2M1     DATA      095H
P2M0     DATA      096H
P3M1     DATA      0B1H
P3M0     DATA      0B2H
P4M1     DATA      0B3H
P4M0     DATA      0B4H
P5M1     DATA      0C9H
P5M0     DATA      0CAH

        ORG          0000H
        LJMP         MAIN

MAIN:    ORG          0100H

        MOV          SP, #5FH
        ORL          P_SW2, #80H          ;使能访问 XFR

        MOV          P0M0, #00H
        MOV          P0M1, #00H
        MOV          P1M0, #00H
        MOV          P1M1, #00H
        MOV          P2M0, #00H
        MOV          P2M1, #00H
        MOV          P3M0, #00H
        MOV          P3M1, #00H
        MOV          P4M0, #00H
        MOV          P4M1, #00H
        MOV          P5M0, #00H
        MOV          P5M1, #00H

        MOV          T3L, #66H          ;65536-11.0592M/12/1000
        MOV          T3H, #0FCH
        MOV          T4T3M, #09H        ;使能时钟输出并启动定时器

        JMP          $

        END
```

### 14.6.26 定时器 3 做串口 3 波特率发生器

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

#define FOSC          11059200UL
#define BRT           (65536 - (FOSC / 115200+2) / 4)

//加 2 操作是为了让 Keil 编译器
```

// 自动实现四舍五入运算

```
bit      busy;
char     wptr;
char     rptr;
char     buffer[16];

void Uart3Isr() interrupt 17
{
    if (S3CON & 0x02)
    {
        S3CON &= ~0x02;
        busy = 0;
    }
    if (S3CON & 0x01)
    {
        S3CON &= ~0x01;
        buffer[wptr++] = S3BUF;
        wptr &= 0x0f;
    }
}

void Uart3Init()
{
    S3CON = 0x50;
    T3L = BRT;
    T3H = BRT >> 8;
    T4T3M = 0x0a;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart3Send(char dat)
{
    while (busy);
    busy = 1;
    S3BUF = dat;
}

void Uart3SendStr(char *p)
{
    while (*p)
    {
        Uart3Send(*p++);
    }
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
```

```
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

Uart3Init();
IE2 = 0x08;
EA = 1;
Uart3SendStr("Uart Test !\r\n");

while (1)
{
    if (rptr != wptr)
    {
        Uart3Send(buffer[rptr++]);
        rptr &= 0x0f;
    }
}
```

汇编代码

;测试工作频率为 11.0592MHz

|        |      |       |           |
|--------|------|-------|-----------|
| P_SW2  | DATA | 0BAH  |           |
| T4T3M  | DATA | 0D1H  |           |
| T4L    | DATA | 0D3H  |           |
| T4H    | DATA | 0D2H  |           |
| T3L    | DATA | 0D5H  |           |
| T3H    | DATA | 0D4H  |           |
| T2L    | DATA | 0D7H  |           |
| T2H    | DATA | 0D6H  |           |
| S3CON  | DATA | 0ACH  |           |
| S3BUF  | DATA | 0ADH  |           |
| IE2    | DATA | 0AFH  |           |
| BUSY   | BIT  | 20H.0 |           |
| WPTR   | DATA | 21H   |           |
| RPTR   | DATA | 22H   |           |
| BUFFER | DATA | 23H   | ;16 bytes |
| P0M1   | DATA | 093H  |           |
| P0M0   | DATA | 094H  |           |
| P1M1   | DATA | 091H  |           |
| P1M0   | DATA | 092H  |           |
| P2M1   | DATA | 095H  |           |
| P2M0   | DATA | 096H  |           |
| P3M1   | DATA | 0B1H  |           |
| P3M0   | DATA | 0B2H  |           |
| P4M1   | DATA | 0B3H  |           |
| P4M0   | DATA | 0B4H  |           |
| P5M1   | DATA | 0C9H  |           |
| P5M0   | DATA | 0CAH  |           |
|        | ORG  | 0000H |           |
|        | LJMP | MAIN  |           |
|        | ORG  | 008BH |           |

```

        LJMP        UART3_ISR

        ORG         0100H

UART3_ISR:
        PUSH        ACC
        PUSH        PSW
        MOV         PSW,#08H

        MOV         A,S3CON
        JNB         ACC.1,CHKRI
        ANL         S3CON,#NOT 02H
        CLR         BUSY

CHKRI:
        JNB         ACC.0,UART3ISR_EXIT
        ANL         S3CON,#NOT 01H
        MOV         A,WPTR
        ANL         A,#0FH
        ADD         A,#BUFFER
        MOV         R0,A
        MOV         @R0,S3BUF
        INC         WPTR

UART3ISR_EXIT:
        POP         PSW
        POP         ACC
        RETI

UART3_INIT:
        MOV         S3CON,#50H
        MOV         T3L,#0E8H
        MOV         T3H,#0FFH
        MOV         T4T3M,#0AH
        CLR         BUSY
        MOV         WPTR,#00H
        MOV         RPTR,#00H
        RET

UART3_SEND:
        JB          BUSY,$
        SETB        BUSY
        MOV         S3BUF,A
        RET

UART3_SENDSTR:
        CLR         A
        MOVC        A,@A+DPTR
        JZ          SEND3END
        LCALL       UART3_SEND
        INC         DPTR
        JMP         UART3_SENDSTR

SEND3END:
        RET

MAIN:
        MOV         SP,#5FH
        ORL         P_SW2,#80H
        ;使能访问 XFR

        MOV         P0M0,#00H
        MOV         P0M1,#00H

```

```

MOV      P1M0, #00H
MOV      P1M1, #00H
MOV      P2M0, #00H
MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

LCALL    UART3_INIT
MOV      IE2, #08H
SETB     EA

MOV      DPTR, #STRING
LCALL    UART3_SENDSTR

LOOP:
MOV      A, RPTR
XRL      A, WPTR
ANL      A, #0FH
JZ       LOOP
MOV      A, RPTR
ANL      A, #0FH
ADD      A, #BUFFER
MOV      R0, A
MOV      A, @R0
LCALL    UART3_SEND
INC      RPTR
JMP      LOOP

STRING:   DB      'Uart Test !', 0DH, 0AH, 00H

END

```

## 14.6.27 定时器 4（16 位自动重载），用作定时

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```
void TM4_Isr() interrupt 20
```

```

{
    P10 = !P10;           //测试端口
}

```

```
void main()
```

```

{
    P_SW2 /= 0x80;        //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
}

```

```

P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

T4L = 0x66;           //65536-11.0592M/12/1000
T4H = 0xfc;
T4T3M = 0x80;         //启动定时器
IE2 = ET4;            //使能定时器中断
EA = 1;

while (1);
}

```

# 汇编代码

;测试工作频率为 11.0592MHz

|          |      |       |
|----------|------|-------|
| P_SW2    | DATA | 0BAH  |
| T4T3M    | DATA | 0D1H  |
| T4L      | DATA | 0D3H  |
| T4H      | DATA | 0D2H  |
| T3L      | DATA | 0D5H  |
| T3H      | DATA | 0D4H  |
| T2L      | DATA | 0D7H  |
| T2H      | DATA | 0D6H  |
| AUXR     | DATA | 8EH   |
| IE2      | DATA | 0AFH  |
| ET2      | EQU  | 04H   |
| ET3      | EQU  | 20H   |
| ET4      | EQU  | 40H   |
| AUXINTIF | DATA | 0EFH  |
| T2IF     | EQU  | 01H   |
| T3IF     | EQU  | 02H   |
| T4IF     | EQU  | 04H   |
| P1M1     | DATA | 091H  |
| P1M0     | DATA | 092H  |
| P0M1     | DATA | 093H  |
| P0M0     | DATA | 094H  |
| P2M1     | DATA | 095H  |
| P2M0     | DATA | 096H  |
| P3M1     | DATA | 0B1H  |
| P3M0     | DATA | 0B2H  |
| P4M1     | DATA | 0B3H  |
| P4M0     | DATA | 0B4H  |
| P5M1     | DATA | 0C9H  |
| P5M0     | DATA | 0CAH  |
|          | ORG  | 0000H |
|          | LJMP | MAIN  |
|          | ORG  | 00A3H |



```

    LJMP      TM4ISR

TM4ISR:
    ORG      0100H

    CPL      P1.0          ;测试端口
    RETI

MAIN:
    MOV      SP, #5FH
    ORL      P_SW2, #80H   ;使能访问 XFR

    MOV      P0M0, #00H
    MOV      P0M1, #00H
    MOV      P1M0, #00H
    MOV      P1M1, #00H
    MOV      P2M0, #00H
    MOV      P2M1, #00H
    MOV      P3M0, #00H
    MOV      P3M1, #00H
    MOV      P4M0, #00H
    MOV      P4M1, #00H
    MOV      P5M0, #00H
    MOV      P5M1, #00H

    MOV      T4L, #66H     ;65536-11.0592M/12/1000
    MOV      T4H, #0FCH
    MOV      T4T3M, #80H   ;启动定时器
    MOV      IE2, #ET4     ;使能定时器中断
    SETB     EA

    JMP      $

END
```

### 14.6.28 定时器 4（外部计数—扩展 T4 为外部下降沿中断）

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void TM4_Isr() interrupt 20
{
    P10 = !P10;          //测试端口
}

void main()
{
    P_SW2 /= 0x80;       //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
```

```
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

T4L = 0xff;
T4H = 0xff;
T4T3M = 0xc0;           //设置外部计数模式并启动定时器
IE2 = ET4;              //使能定时器中断
EA = 1;

while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

|          |      |        |
|----------|------|--------|
| P_SW2    | DATA | 0BAH   |
| T4T3M    | DATA | 0D1H   |
| T4L      | DATA | 0D3H   |
| T4H      | DATA | 0D2H   |
| T3L      | DATA | 0D5H   |
| T3H      | DATA | 0D4H   |
| T2L      | DATA | 0D7H   |
| T2H      | DATA | 0D6H   |
| AUXR     | DATA | 8EH    |
| IE2      | DATA | 0AFH   |
| ET2      | EQU  | 04H    |
| ET3      | EQU  | 20H    |
| ET4      | EQU  | 40H    |
| AUXINTIF | DATA | 0EFH   |
| T2IF     | EQU  | 01H    |
| T3IF     | EQU  | 02H    |
| T4IF     | EQU  | 04H    |
| P1M1     | DATA | 091H   |
| P1M0     | DATA | 092H   |
| P0M1     | DATA | 093H   |
| P0M0     | DATA | 094H   |
| P2M1     | DATA | 095H   |
| P2M0     | DATA | 096H   |
| P3M1     | DATA | 0B1H   |
| P3M0     | DATA | 0B2H   |
| P4M1     | DATA | 0B3H   |
| P4M0     | DATA | 0B4H   |
| P5M1     | DATA | 0C9H   |
| P5M0     | DATA | 0CAH   |
|          | ORG  | 0000H  |
|          | LJMP | MAIN   |
|          | ORG  | 00A3H  |
|          | LJMP | TM4ISR |

```

    ORG      0100H

TM4ISR:
    CPL      P1.0          ;测试端口
    RETI

MAIN:
    MOV      SP, #5FH
    ORL      P_SW2, #80H   ;使能访问 XFR

    MOV      P0M0, #00H
    MOV      P0M1, #00H
    MOV      P1M0, #00H
    MOV      P1M1, #00H
    MOV      P2M0, #00H
    MOV      P2M1, #00H
    MOV      P3M0, #00H
    MOV      P3M1, #00H
    MOV      P4M0, #00H
    MOV      P4M1, #00H
    MOV      P5M0, #00H
    MOV      P5M1, #00H

    MOV      T4L, #0FFH
    MOV      T4H, #0FFH
    MOV      T4T3M, #0C0H   ;设置外部计数模式并启动定时器
    MOV      IE2, #ET4      ;使能定时器中断
    SETB     EA

    JMP      $

END
```

## 14.6.29 定时器 4，时钟分频输出

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void main()
{
    P_SW2 /= 0x80;          //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
```

```

P3M1 = 0x00;

T4L = 0x66; //65536-11.0592M/12/1000
T4H = 0xfc;
T4T3M = 0x90; //使能时钟输出并启动定时器

while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

```

P_SW2      DATA      0BAH

T4T3M      DATA      0D1H
T4L        DATA      0D3H
T4H        DATA      0D2H
T3L        DATA      0D5H
T3H        DATA      0D4H
T2L        DATA      0D7H
T2H        DATA      0D6H

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

ORG        0000H
LJMP       MAIN

ORG        0100H
MAIN:
MOV        SP, #5FH
ORL        P_SW2, #80H ;使能访问 XFR

MOV        P0M0, #00H
MOV        P0M1, #00H
MOV        P1M0, #00H
MOV        P1M1, #00H
MOV        P2M0, #00H
MOV        P2M1, #00H
MOV        P3M0, #00H
MOV        P3M1, #00H
MOV        P4M0, #00H
MOV        P4M1, #00H
MOV        P5M0, #00H
MOV        P5M1, #00H

MOV        T4L, #66H ;65536-11.0592M/12/1000
MOV        T4H, #0FCH
```

```
MOV      T4T3M,#90H      ;使能时钟输出并启动定时器
JMP      $
END
```

## 14.6.30 定时器 4 做串口 4 波特率发生器

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT      (65536 - (FOSC / 115200+2) / 4)
```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
bit      busy;
char      wptr;
char      rptr;
char      buffer[16];
```

```
void Uart4Isr() interrupt 18
```

```
{
    if (S4CON & 0x02)
    {
        S4CON &= ~0x02;
        busy = 0;
    }
    if (S4CON & 0x01)
    {
        S4CON &= ~0x01;
        buffer[wptr++] = S4BUF;
        wptr &= 0x0f;
    }
}
```

```
void Uart4Init()
```

```
{
    S4CON = 0x50;
    T4L = BRT;
    T4H = BRT >> 8;
    T4T3M = 0xa0;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}
```

```
void Uart4Send(char dat)
```

```
{
    while (busy);
    busy = 1;
    S4BUF = dat;
}
```

```
}

void Uart4SendStr(char *p)
{
    while (*p)
    {
        Uart4Send(*p++);
    }
}

void main()
{
    P_SW2 |= 0x80; //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    Uart4Init();
    IE2 = 0x10;
    EA = 1;
    Uart4SendStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            Uart4Send(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

汇编代码

;测试工作频率为11.0592MHz

|              |             |             |
|--------------|-------------|-------------|
| <i>P_SW2</i> | <i>DATA</i> | <i>0BAH</i> |
| <i>T4T3M</i> | <i>DATA</i> | <i>0D1H</i> |
| <i>T4L</i>   | <i>DATA</i> | <i>0D3H</i> |
| <i>T4H</i>   | <i>DATA</i> | <i>0D2H</i> |
| <i>T3L</i>   | <i>DATA</i> | <i>0D5H</i> |
| <i>T3H</i>   | <i>DATA</i> | <i>0D4H</i> |
| <i>T2L</i>   | <i>DATA</i> | <i>0D7H</i> |
| <i>T2H</i>   | <i>DATA</i> | <i>0D6H</i> |
| <i>S4CON</i> | <i>DATA</i> | <i>84H</i>  |
| <i>S4BUF</i> | <i>DATA</i> | <i>85H</i>  |
| <i>IE2</i>   | <i>DATA</i> | <i>0AFH</i> |

```

BUSY      BIT      20H.0
WPTR     DATA     21H
RPTR     DATA     22H
BUFFER   DATA     23H                                ;16 bytes

P0M1     DATA     093H
P0M0     DATA     094H
P1M1     DATA     091H
P1M0     DATA     092H
P2M1     DATA     095H
P2M0     DATA     096H
P3M1     DATA     0B1H
P3M0     DATA     0B2H
P4M1     DATA     0B3H
P4M0     DATA     0B4H
P5M1     DATA     0C9H
P5M0     DATA     0CAH

      ORG      0000H
      LJMP     MAIN
      ORG      0093H
      LJMP     UART4_ISR

      ORG      0100H

UART4_ISR:
      PUSH     ACC
      PUSH     PSW
      MOV      PSW,#08H

      MOV      A,S4CON
      JNB      ACC.1,CHKRI
      ANL      S4CON,#NOT 02H
      CLR      BUSY

CHKRI:
      JNB      ACC.0,UART4ISR_EXIT
      ANL      S4CON,#NOT 01H
      MOV      A,WPTR
      ANL      A,#0FH
      ADD      A,#BUFFER
      MOV      R0,A
      MOV      @R0,S4BUF
      INC      WPTR

UART4ISR_EXIT:
      POP      PSW
      POP      ACC
      RETI

UART4_INIT:
      MOV      S4CON,#50H
      MOV      T4L,#0E8H                                ;65536-11059200/115200/4=0FFE8H
      MOV      T4H,#0FFH
      MOV      T4T3M,#0A0H
      CLR      BUSY
      MOV      WPTR,#00H
      MOV      RPTR,#00H
      RET

UART4_SEND:

```

```

        JB      BUSY,$
        SETB    BUSY
        MOV     S4BUF,A
        RET

UART4_SENDSTR:
        CLR     A
        MOVC    A,@A+DPTR
        JZ      SEND4END
        LCALL   UART4_SEND
        INC     DPTR
        JMP     UART4_SENDSTR
SEND4END:
        RET

MAIN:
        MOV     SP,#5FH
        ORL     P_SW2,#80H          ;使能访问 XFR

        MOV     P0M0,#00H
        MOV     P0M1,#00H
        MOV     P1M0,#00H
        MOV     P1M1,#00H
        MOV     P2M0,#00H
        MOV     P2M1,#00H
        MOV     P3M0,#00H
        MOV     P3M1,#00H
        MOV     P4M0,#00H
        MOV     P4M1,#00H
        MOV     P5M0,#00H
        MOV     P5M1,#00H

        LCALL   UART4_INIT
        MOV     IE2,#10H
        SETB    EA

        MOV     DPTR,#STRING
        LCALL   UART4_SENDSTR

LOOP:
        MOV     A,RPTR
        XRL     A,WPTR
        ANL     A,#0FH
        JZ      LOOP
        MOV     A,RPTR
        ANL     A,#0FH
        ADD     A,#BUFFER
        MOV     R0,A
        MOV     A,@R0
        LCALL   UART4_SEND
        INC     RPTR
        JMP     LOOP

STRING:  DB      'Uart Test !',0DH,0AH,00H

        END

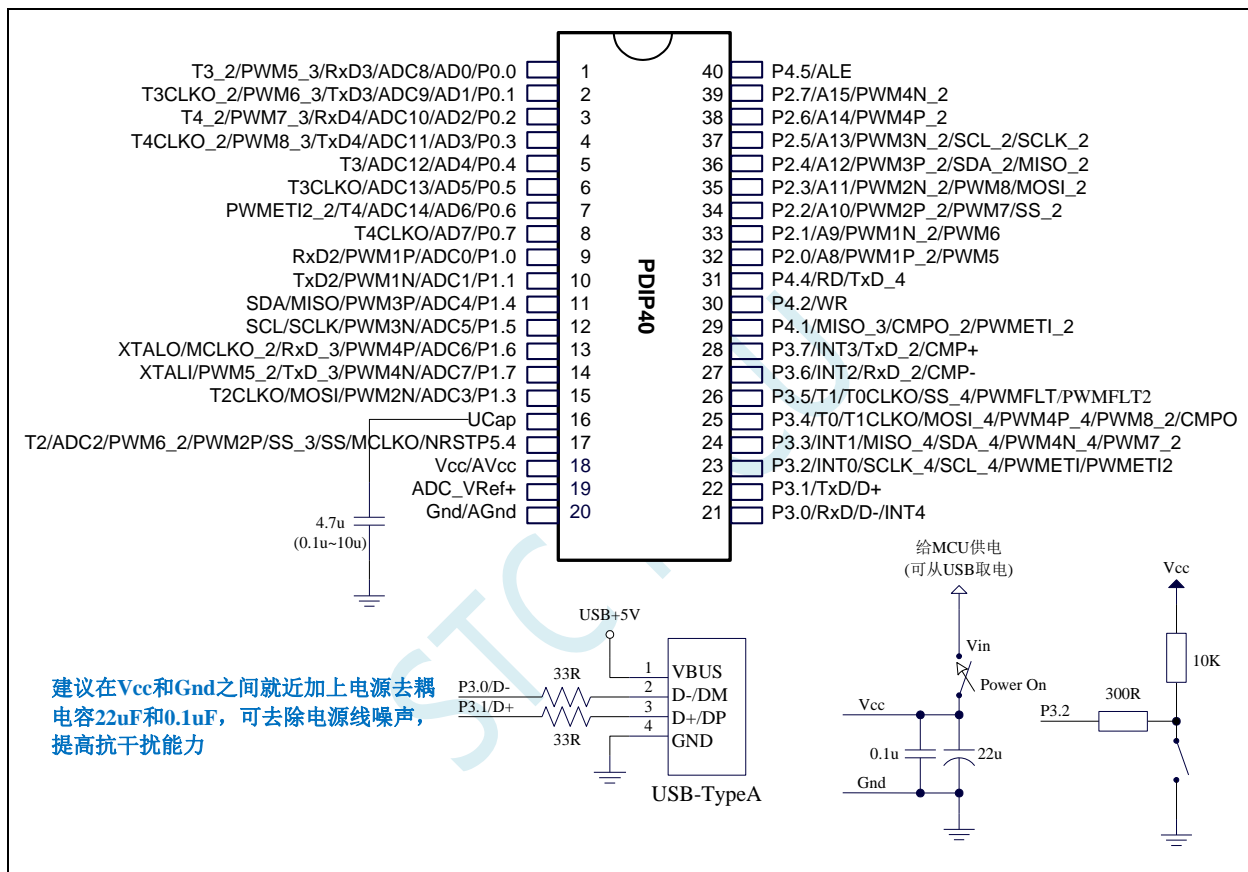
```



## 15 超级简单的 STC USB-CDC 虚拟串口应用

## 15.1 USB-CDC 虚拟串口概述

当单片机需要和电脑之间进行数据交换时，首选一般都是串口通讯。STC32G 系列单片机内置 USB 控制器和收发器，当用户代码中嵌入 USB-CDC 代码后，使用 USB 线将单片机与电脑直接相连接，在电脑端就可识别出【USB-CDC 虚拟串口】，【USB-CDC 虚拟串口】就是【串口】。



### USB-CDC 虚拟串口和传统串口相比有如下优点:

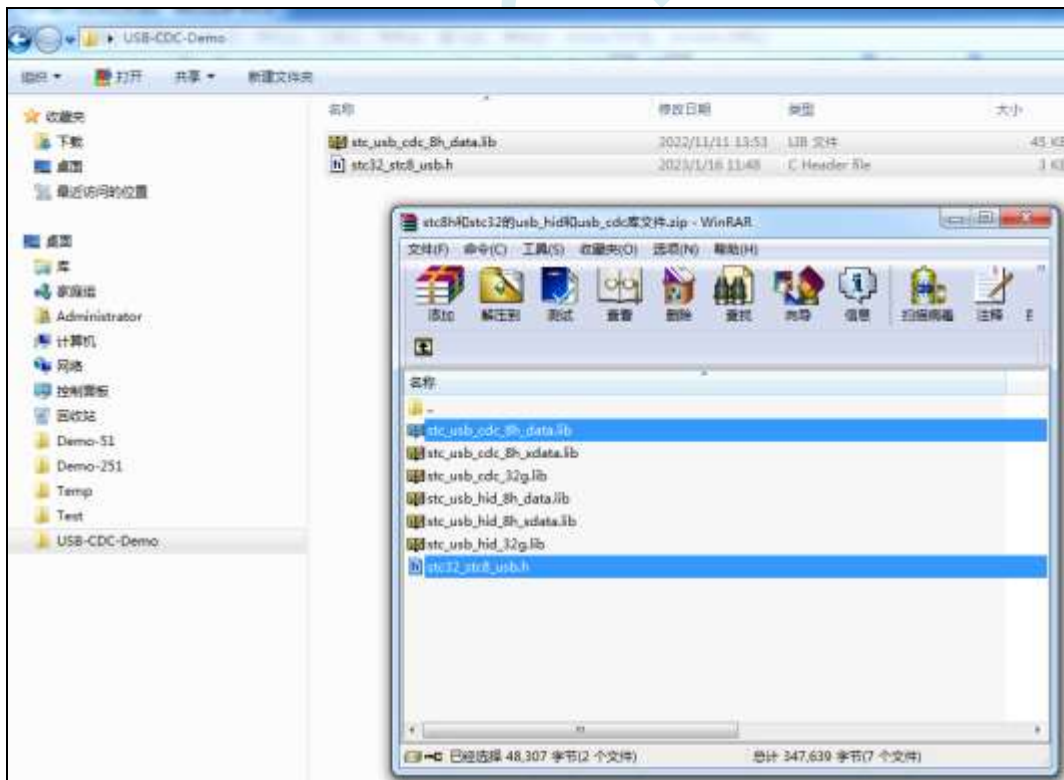
- **数据传输更快：**USB-CDC 虚拟串口忽略传统串口的波特率，传输速度的比特率即为全速 USB 的通讯速度 12MBPS（即每秒 12M 位）
- **使用更简单便捷：**USB-CDC 虚拟串口忽略传统串口的起始位、停止位等冗余信息
- **数据传输更可靠：**USB-CDC 虚拟串口丢弃传统串口简单的软件奇偶校验机制，USB-CDC 虚拟串口数据传输时有 USB 硬件 CRC 校验，以及校验出错重传机制，保证数据 100% 正确
- **自动缓存数据：**USB-CDC 虚拟串口会自动缓存数据。单片机在没有处理完成上位机下传的上一笔数据时，如果此时上位机又有新的数据下传，虚拟串口会自动将新的数据缓存，从而保证数据 100% 不会丢失或被覆盖。

## 15.2 新建 Keil 项目并加入 CDC 模块

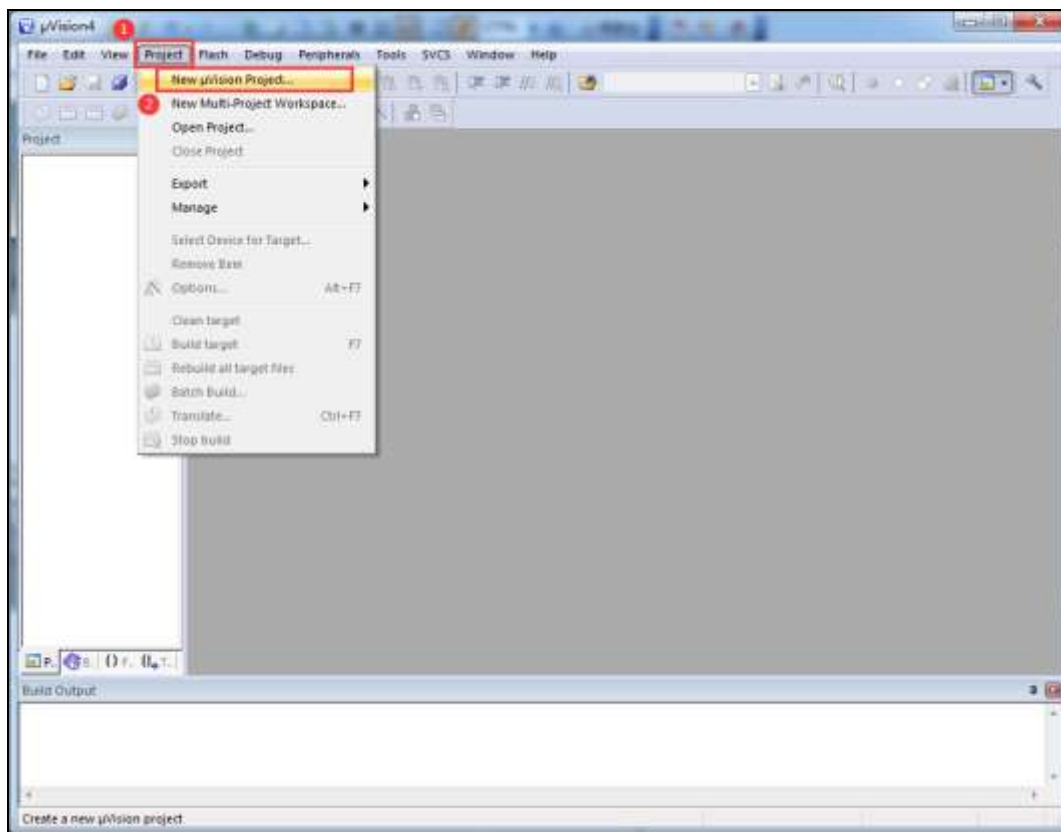
1、首选从官网下载 CDC 代码库 (<https://www.stcai.com/filedownload/656048>)



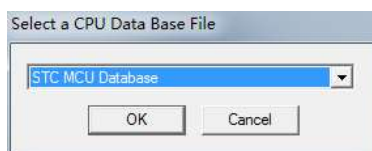
2、下载完成后，从压缩包中将“stc\_usb\_cdc\_8h\_data.lib”和“stc32\_stc8\_usb.h”解压到项目目录中



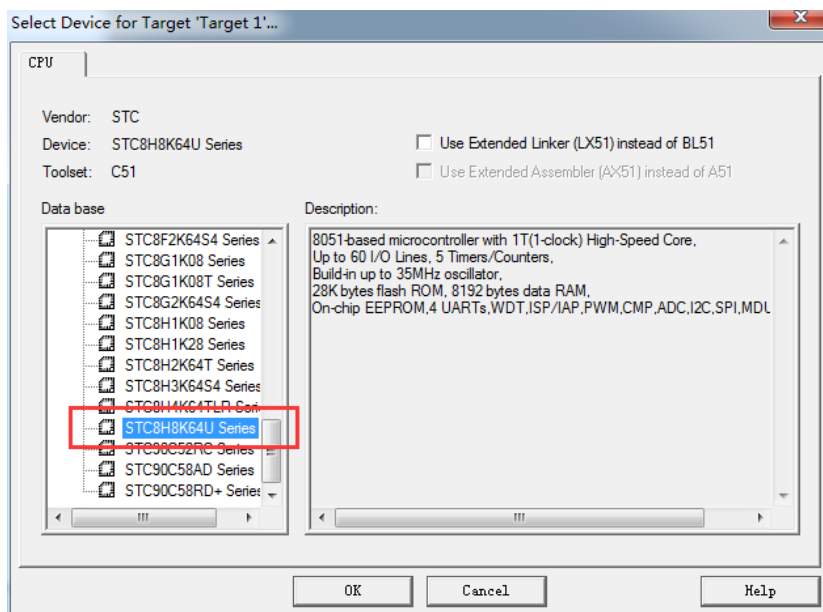
### 3、打开 Keil 软件，并新建项目



Database 请选择 “STC MCU Database”



单片机型号选择 “STC8H8K64U Series”



4、项目建立完成后，将下面的代码命名为 `main.c` 文件并保存到项目目录中

//测试工作频率为24MHz

```
#include "stc8h.h"
```

```
#include "stc32_stc8_usb.h"
```

```
#define FOSC 24000000UL
```

//ISP 下载时需将工作频率设置为24MHz

```
char *USER_DEVICEDESC = NULL;
```

```
char *USER_PRODUCTDESC = NULL;
```

```
char *USER_STCISPCMD = "@STCISP#";
```

//不停电自动ISP 下载命令

```
void main()
```

```
{
```

```
    P_SW2 /= 0x80;
```

```
    P0M1 = 0x00;    P0M0 = 0x00;
```

```
    P1M1 = 0x00;    P1M0 = 0x00;
```

```
    P2M1 = 0x00;    P2M0 = 0x00;
```

```
    P3M1 = 0x00;    P3M0 = 0x00;
```

```
    P4M1 = 0x00;    P4M0 = 0x00;
```

```
    P5M1 = 0x00;    P5M0 = 0x00;
```

```
    P6M1 = 0x00;    P6M0 = 0x00;
```

```
    P7M1 = 0x00;    P7M0 = 0x00;
```

```
    P3M0 &= ~0x03;
```

```
    P3M1 /= 0x03;
```

//P3.0/P3.1 和USB 的D-/D+共用PIN 脚,

//需要将P3.0/P3.1 设置为高阻输入模式

```
    IRC48MCR = 0x80;
```

```
    while (!(IRC48MCR & 0x01));
```

```
    USBCLK = 0x00;
```

```
    USBCON = 0x90;
```

//使能内部48M 的USB 专用IRC

//设置USB 时钟源为内部48M 的USB 专用IRC

//使能USB 功能

```
    usb_init();
```

//调用USB CDC 初始化库函数

```
    IE2 /= 0x80;
```

```
    EA = 1;
```

//使能USB 中断

---

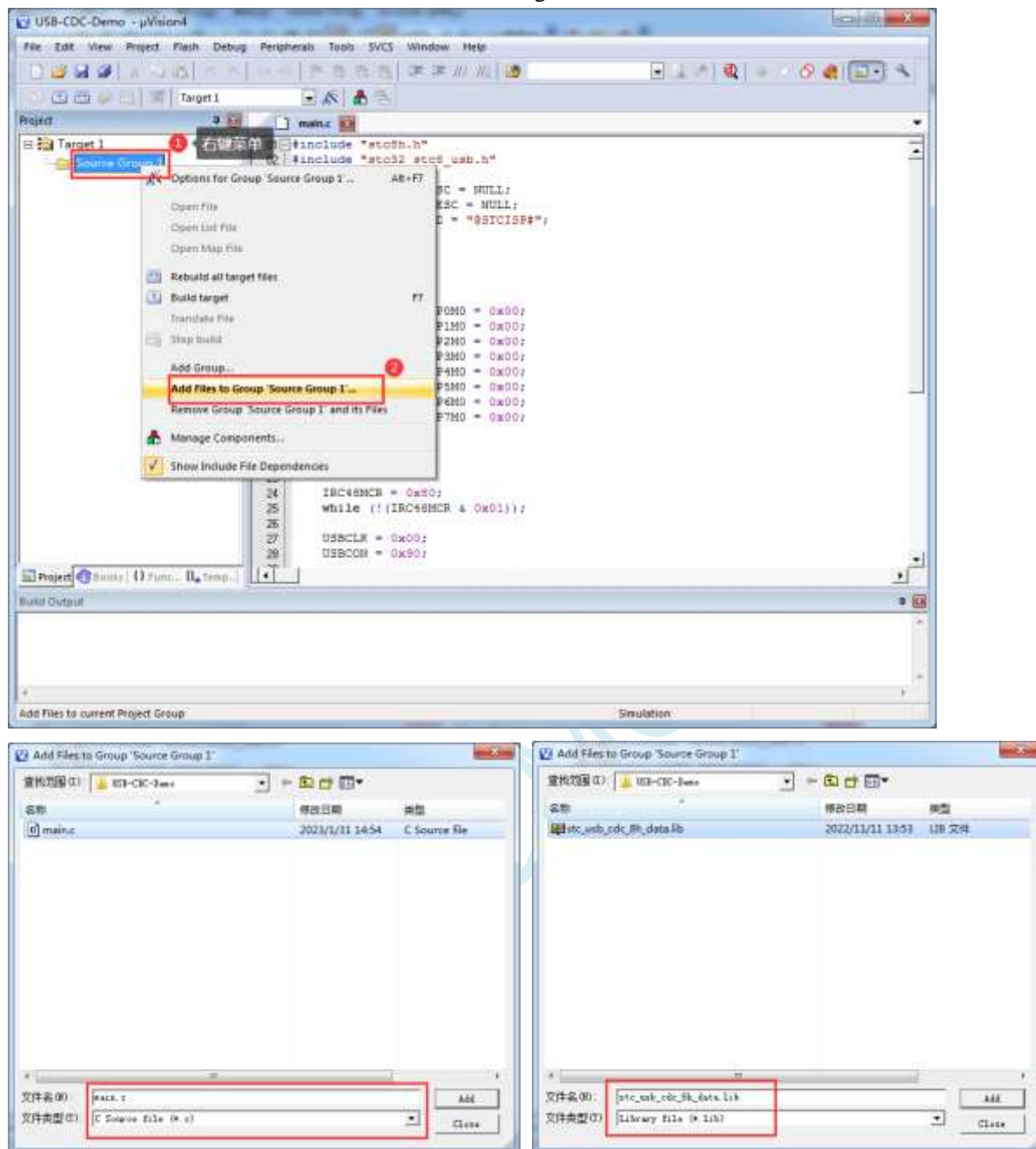
```
while (DeviceState != DEVSTATE_CONFIGURED);           //等待USB 完成配置

while (1)
{
    if (bUsbOutReady)                                  //当硬件接收完成上位机通过串口助手发送数据后
                                                         //会自动将bUsbOutReady 置1
                                                         //接收的数据字节数保存在 OutNumber 变量中
                                                         //接收的数据保存在 UsbOutBuffer 缓冲区

    {
        USB_SendData(UsbOutBuffer,OutNumber);          //使用USB_SendData 库函数可向上位机发送数据
                                                         //这里的测试代码为将接收数据原样返回
        usb_OUT_done();                                //处理完成接收的数据后
                                                         //调用usb_OUT_done 准备接收下一笔数据
    }
}
}
```

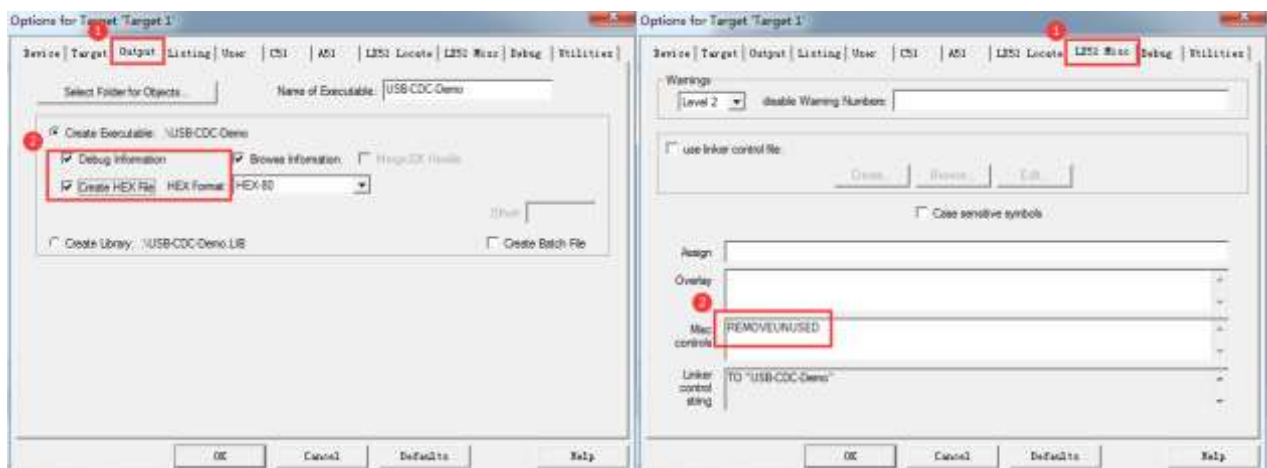
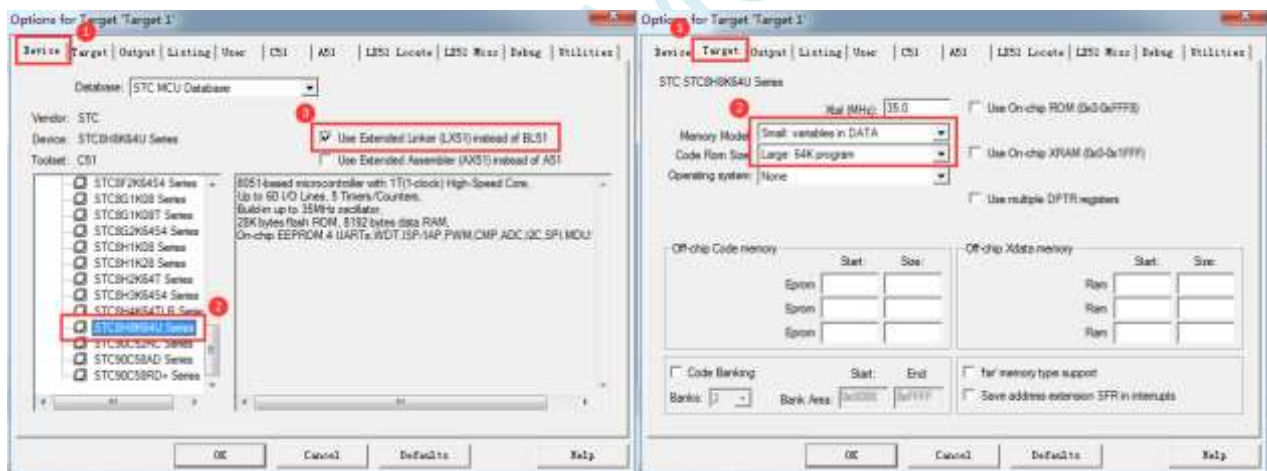
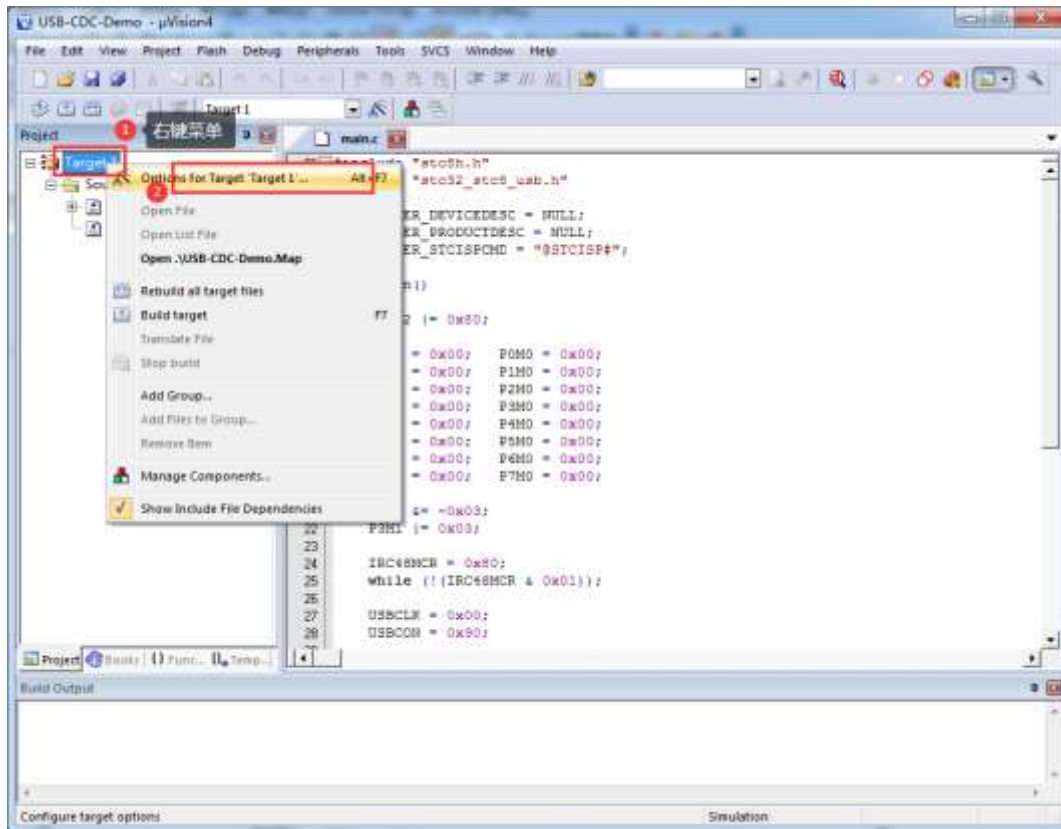
---

## 5、将项目目录下的“main.c”和“stc\_usb\_cdc\_32g.lib”加入到项目中



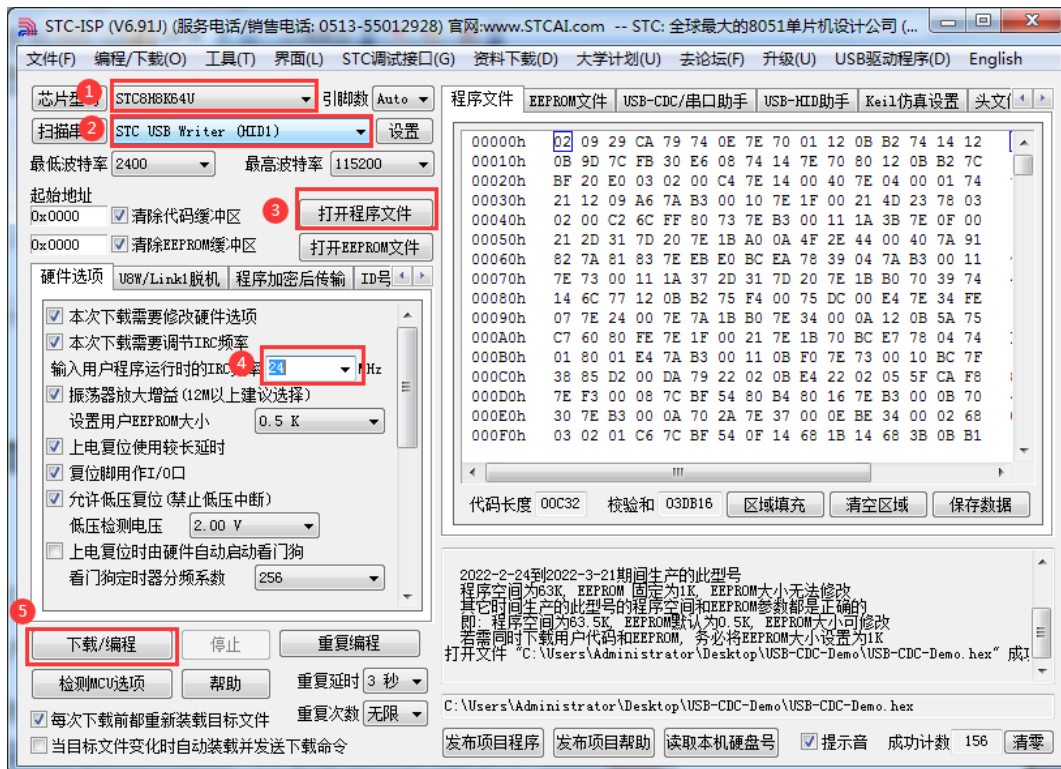


## 6、进行项目设置

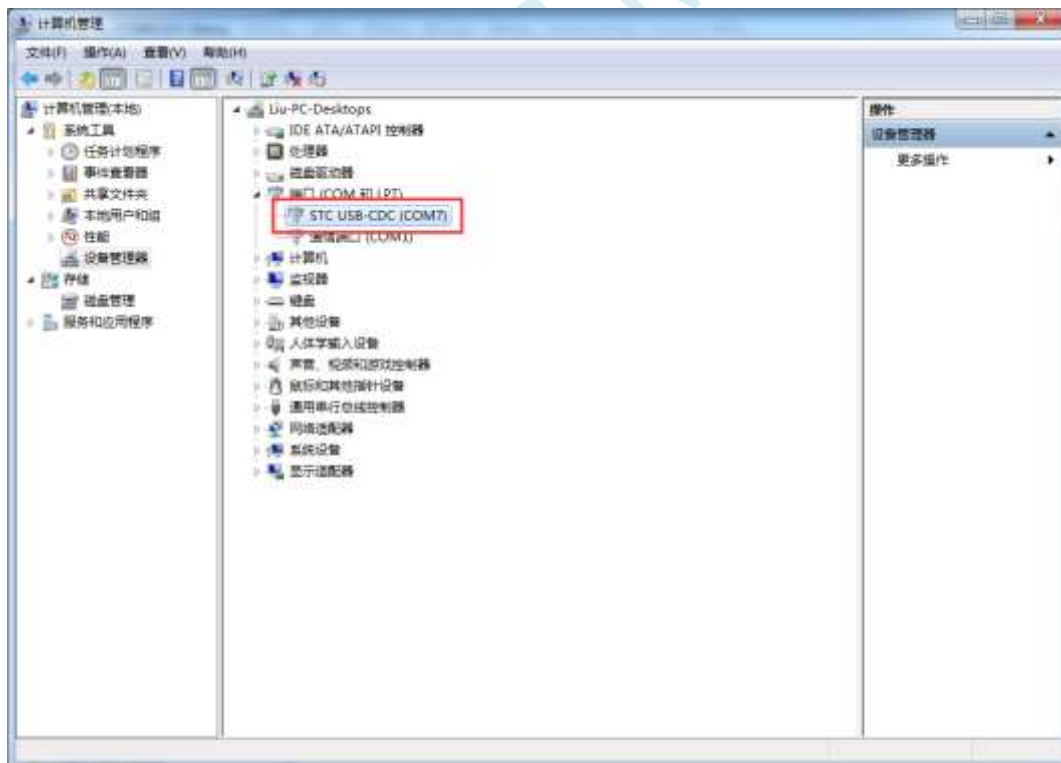


设置完成后, 编译通过即可生产目标 HEX 文件

## 7、使用最新的 STC-ISP 下载软件将 HEX 下载到目标芯片



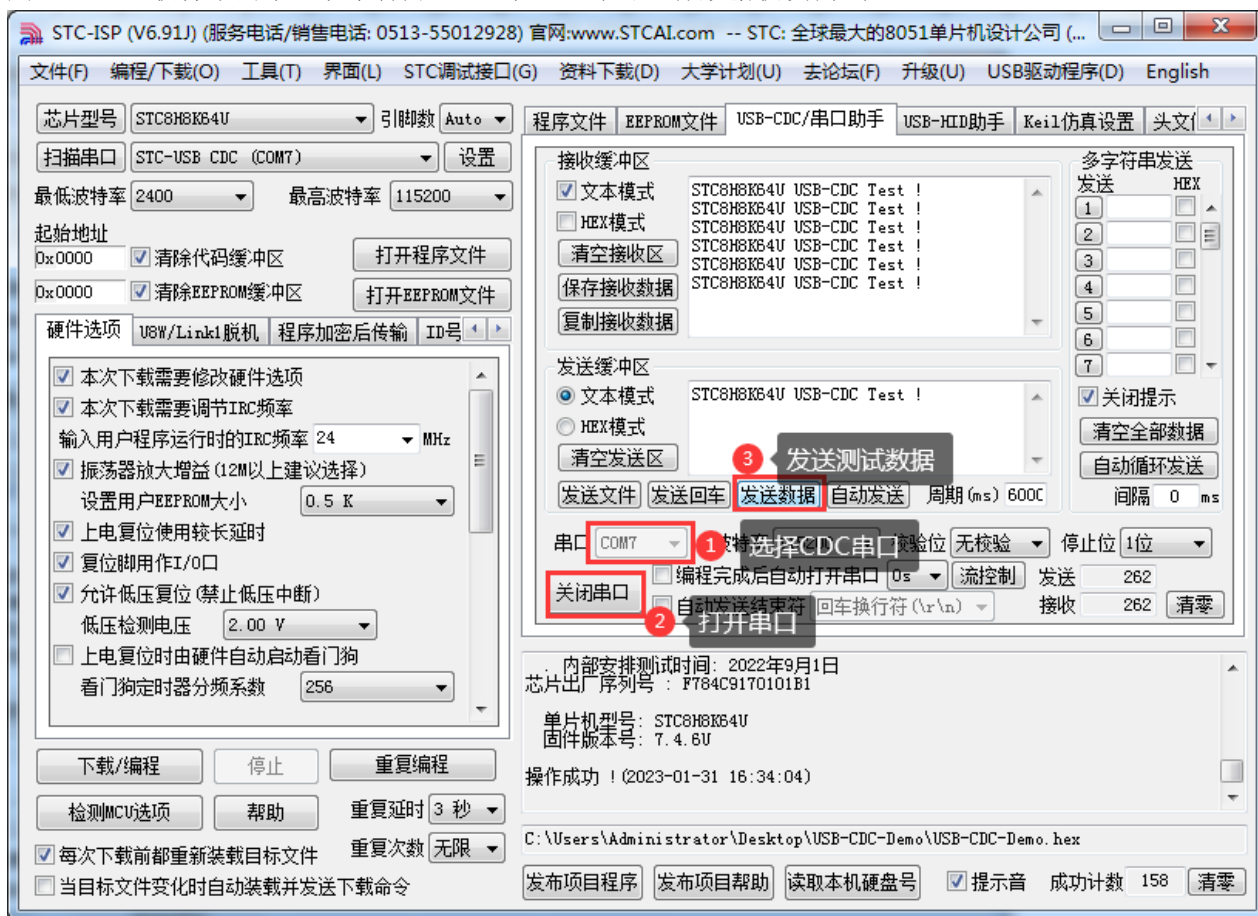
## 8、下载完成后, 系统中就会出现一个 STC USB-CDC 串口





### 15.3 USB-CDC 虚拟串口与电脑进行数据传输

用 STC-ISP 软件中的串口助手打开 CDC 串口，即可进行数据收发测试



## 15.4 STC USB-CDC 虚拟串口实现不停电自动 ISP 下载

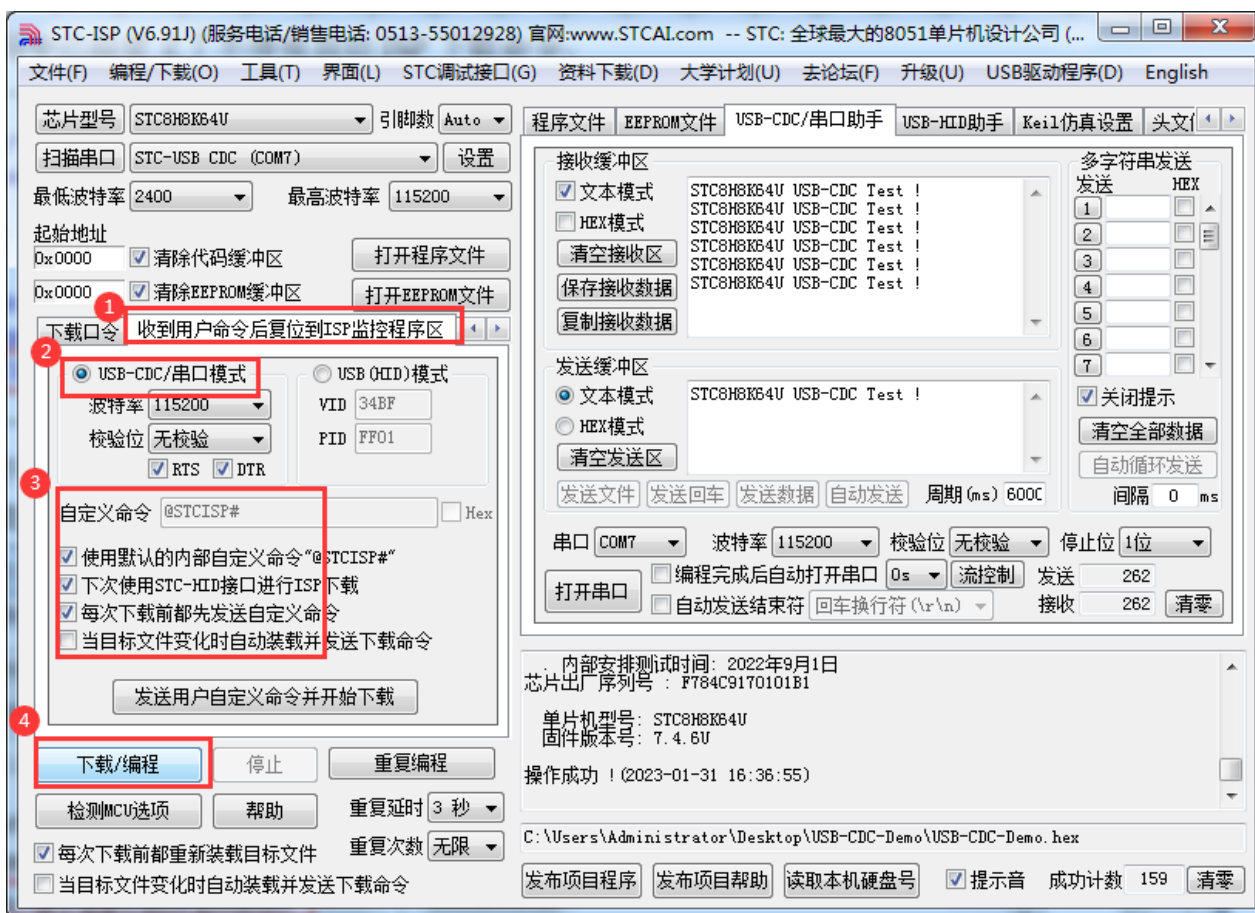
由于我们在代码中已经定义了不停电自动 ISP 下载命令

```
#include "stc8h.h"
#include "stc32_stc8_usb.h"

#define FOSC 2400000UL //ISP 下载时需将工作频率设置为24MHz

char *USER_DEVICEDESC=NULL;
char *USER_PRODUCTDESC=NULL;
char *USER_STCISPCMD="@STCISP#"; //不停电自动ISP 下载命令
```

我们只需要在下载软件的“收到用户命令后复位到 ISP 监控程序区”中的进行如下设置，即可实现不停电自动 ISP 下载功能了。



## 16 串口通信

| 产品线              | 串口数量      |
|------------------|-----------|
| STC8H1K08 系列     | 2 (U1~U2) |
| STC8H1K28 系列     | 2 (U1~U2) |
| STC8H3K64S4 系列   | 4 (U1~U4) |
| STC8H3K64S2 系列   | 2 (U1~U2) |
| STC8H8K64U 系列    | 4 (U1~U4) |
| STC8H4K64TL 系列   | 4 (U1~U4) |
| STC8H4K64TLCD 系列 | 4 (U1~U4) |
| STC8H1K08T 系列    | 2 (U1~U2) |

STC8H 系列单片机具有 4 个全双工异步串行通信接口。每个串行口由 2 个数据缓冲器、一个移位寄存器、一个串行控制寄存器和一个波特率发生器等组成。每个串行口的数据缓冲器由 2 个互相独立的接收、发送缓冲器构成，可以同时发送和接收数据。

STC8 系列单片机的串口 1 有 4 种工作方式，其中两种方式的波特率是可变的，另两种是固定的，以供不同应用场合选用。串口 2/串口 3/串口 4 都只有两种工作方式，这两种方式的波特率都是可变的。用户可用软件设置不同的波特率和选择不同的工作方式。主机可通过查询或中断方式对接收/发送进行程序处理，使用十分灵活。

串口 1、串口 2、串口 3、串口 4 的通讯口均可以通过功能管脚的切换功能切换到多组端口，从而可以将一个通讯口分时复用为多个通讯口。

### 16.1 串口功能脚切换

| 符号    | 地址  | B7        | B6 | B5         | B4 | B3         | B2   | B1   | B0   |
|-------|-----|-----------|----|------------|----|------------|------|------|------|
| P_SW1 | A2H | S1_S[1:0] |    | -          | -  | SPI_S[1:0] |      | 0    | -    |
| P_SW2 | BAH | EAXFR     | -  | I2C_S[1:0] |    | CMPO_S     | S4_S | S3_S | S2_S |

S1\_S[1:0]: 串口 1 功能脚选择位

| S1_S[1:0] | RxD  | TxD  |
|-----------|------|------|
| 00        | P3.0 | P3.1 |
| 01        | P3.6 | P3.7 |
| 10        | P1.6 | P1.7 |
| 11        | P4.3 | P4.4 |

S4\_S: 串口 4 功能脚选择位

| S4_S | RxD4 | TxD4 |
|------|------|------|
| 0    | P0.2 | P0.3 |
| 1    | P5.2 | P5.3 |

S3\_S: 串口 3 功能脚选择位

| S3_S | RxD3 | TxD3 |
|------|------|------|
| 0    | P0.0 | P0.1 |

|   |      |      |
|---|------|------|
| 1 | P5.0 | P5.1 |
|---|------|------|

S2\_S: 串口 2 功能脚选择位

| S2_S | RxD2 | TxD2 |
|------|------|------|
| 0    | P1.0 | P1.1 |
| 1    | P4.6 | P4.7 |

## 16.2 串口相关寄存器

| 符号    | 描述             | 地址  | 位地址与符号 |       |           |       |        |       |        |       | 复位值       |
|-------|----------------|-----|--------|-------|-----------|-------|--------|-------|--------|-------|-----------|
|       |                |     | B7     | B6    | B5        | B4    | B3     | B2    | B1     | B0    |           |
| SCON  | 串口 1 控制寄存器     | 98H | SM0/FE | SM1   | SM2       | REN   | TB8    | RB8   | TI     | RI    | 0000,0000 |
| SBUF  | 串口 1 数据寄存器     | 99H |        |       |           |       |        |       |        |       | 0000,0000 |
| S2CON | 串口 2 控制寄存器     | 9AH | S2SM0  | -     | S2SM2     | S2REN | S2TB8  | S2RB8 | S2TI   | S2RI  | 0100,0000 |
| S2BUF | 串口 2 数据寄存器     | 9BH |        |       |           |       |        |       |        |       | 0000,0000 |
| S3CON | 串口 3 控制寄存器     | ACH | S3SM0  | S3ST3 | S3SM2     | S3REN | S3TB8  | S3RB8 | S3TI   | S3RI  | 0000,0000 |
| S3BUF | 串口 3 数据寄存器     | ADH |        |       |           |       |        |       |        |       | 0000,0000 |
| S4CON | 串口 4 控制寄存器     | 84H | S4SM0  | S4ST4 | S4SM2     | S4REN | S4TB8  | S4RB8 | S4TI   | S4RI  | 0000,0000 |
| S4BUF | 串口 4 数据寄存器     | 85H |        |       |           |       |        |       |        |       | 0000,0000 |
| PCON  | 电源控制寄存器        | 87H | SMOD   | SMOD0 | LVDF      | POF   | GF1    | GF0   | PD     | IDL   | 0011,0000 |
| AUXR  | 辅助寄存器 1        | 8EH | T0x12  | T1x12 | UART_M0x6 | T2R   | T2_C/T | T2x12 | EXTRAM | S1ST2 | 0000,0001 |
| SADDR | 串口 1 从机地址寄存器   | A9H |        |       |           |       |        |       |        |       | 0000,0000 |
| SADEN | 串口 1 从机地址屏蔽寄存器 | B9H |        |       |           |       |        |       |        |       | 0000,0000 |

## 16.3 串口 1

### 16.3.1 串口 1 控制寄存器 (SCON)

| 符号   | 地址  | B7     | B6  | B5  | B4  | B3  | B2  | B1 | B0 |
|------|-----|--------|-----|-----|-----|-----|-----|----|----|
| SCON | 98H | SM0/FE | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |

**SM0/FE:** 当PCON寄存器中的SMOD0位为1时, 该位为帧错误检测标志位。当UART在接收过程中检测到一个无效停止位时, 通过UART接收器将该位置1, 必须由软件清零。当PCON寄存器中的SMOD0位为0时, 该位和SM1一起指定串口1的通信工作模式, 如下表所示:

| SM0 | SM1 | 串口1工作模式 | 功能说明        |
|-----|-----|---------|-------------|
| 0   | 0   | 模式0     | 同步移位串行方式    |
| 0   | 1   | 模式1     | 可变波特率8位数据方式 |
| 1   | 0   | 模式2     | 固定波特率9位数据方式 |
| 1   | 1   | 模式3     | 可变波特率9位数据方式 |

**SM2:** 允许模式 2 或模式 3 多机通信控制位。当串口 1 使用模式 2 或模式 3 时, 如果 SM2 位为 1 且 REN 位为 1, 则接收机处于地址帧筛选状态。此时可以利用接收到的第 9 位 (即 RB8) 来筛选地址帧, 若 RB8=1, 说明该帧是地址帧, 地址信息可以进入 SBUF, 并使 RI 为 1, 进而在中断服务程序中再进行地址号比较; 若 RB8=0, 说明该帧不是地址帧, 应丢掉且保持 RI=0。在模式 2 或模式 3 中, 如果 SM2 位为 0 且 REN 位为 1, 接收机处于地址帧筛选被禁止状态, 不论收到的 RB8 为 0 或 1, 均可使接收到的信息进入 SBUF, 并使 RI=1, 此时 RB8 通常为校验位。模式 1 和模式 0 为非多机通信方式, 在这两种方式时, SM2 应设置为 0。

**REN:** 允许/禁止串口接收控制位

0: 禁止串口接收数据

1: 允许串口接收数据

**TB8:** 当串口 1 使用模式 2 或模式 3 时, TB8 为要发送的第 9 位数据, 按需要由软件置位或清 0。在模式 0 和模式 1 中, 该位不用。

**RB8:** 当串口 1 使用模式 2 或模式 3 时, RB8 为接收到的第 9 位数据, 一般用作校验位或者地址帧/数据帧标志位。在模式 0 和模式 1 中, 该位不用。

**TI:** 串口 1 发送中断请求标志位。在模式 0 中, 当串口发送数据第 8 位结束时, 由硬件自动将 TI 置 1, 向主机请求中断, 响应中断后 TI 必须用软件清零。在其他模式中, 则在停止位开始发送时由硬件自动将 TI 置 1, 向 CPU 发请求中断, 响应中断后 TI 必须用软件清零。

**RI:** 串口 1 接收中断请求标志位。在模式 0 中, 当串口接收第 8 位数据结束时, 由硬件自动将 RI 置 1, 向主机请求中断, 响应中断后 RI 必须用软件清零。在其他模式中, 串行接收到停止位的中间时刻由硬件自动将 RI 置 1, 向 CPU 发中断申请, 响应中断后 RI 必须由软件清零。

### 16.3.2 串口 1 数据寄存器 (SBUF)

| 符号   | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------|-----|----|----|----|----|----|----|----|----|
| SBUF | 99H |    |    |    |    |    |    |    |    |

**SBUF:** 串口 1 数据接收/发送缓冲区。SBUF 实际是 2 个缓冲器, 读缓冲器和写缓冲器, 两个操作分别对应两个不同的寄存器, 1 个是只写寄存器 (写缓冲器), 1 个是只读寄存器 (读缓冲器)。对 SBUF 进行读操作, 实际是读取串口接收缓冲区, 对 SBUF 进行写操作则是触发串口开始发送数据。

### 16.3.3 电源管理寄存器 (PCON)

| 符号   | 地址  | B7   | B6    | B5   | B4  | B3  | B2  | B1 | B0  |
|------|-----|------|-------|------|-----|-----|-----|----|-----|
| PCON | 87H | SMOD | SMOD0 | LVDF | POF | GF1 | GF0 | PD | IDL |

SMOD: 串口 1 波特率控制位

0: 串口 1 的各个模式的波特率都不加倍

1: 串口 1 模式 1 (使用模式 2 的定时器 1 作为波特率发生器时有效)、模式 2、模式 3 (使用模式 2 的定时器 1 作为波特率发生器时有效) 的波特率加倍

SMOD0: 帧错误检测控制位

0: 无帧错检测功能

1: 使能帧错误检测功能。此时 SCON 的 SM0/FE 为 FE 功能, 即为帧错误检测标志位。

### 16.3.4 辅助寄存器 1 (AUXR)

| 符号   | 地址  | B7    | B6    | B5        | B4  | B3     | B2    | B1     | B0    |
|------|-----|-------|-------|-----------|-----|--------|-------|--------|-------|
| AUXR | 8EH | T0x12 | T1x12 | UART_M0x6 | T2R | T2_C/T | T2x12 | EXTRAM | S1ST2 |

UART\_M0x6: 串口 1 模式 0 的通讯速度控制

0: 串口 1 模式 0 的波特率不加倍, 固定为  $F_{osc}/12$

1: 串口 1 模式 0 的波特率 6 倍速, 即固定为  $F_{osc}/12 \times 6 = F_{osc}/2$

S1ST2: 串口 1 波特率发生器选择位

0: 选择定时器 1 作为波特率发生器

1: 选择定时器 2 作为波特率发生器

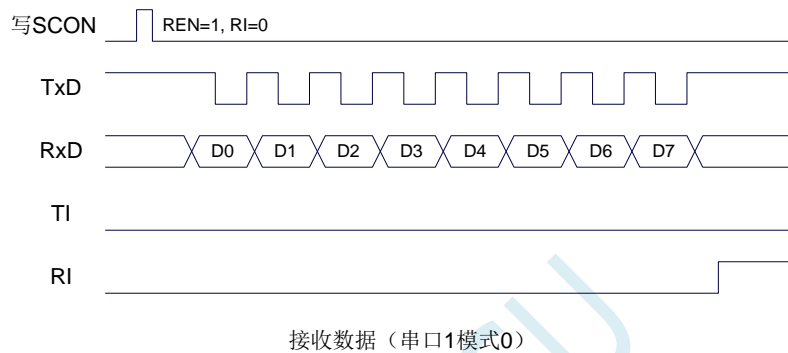
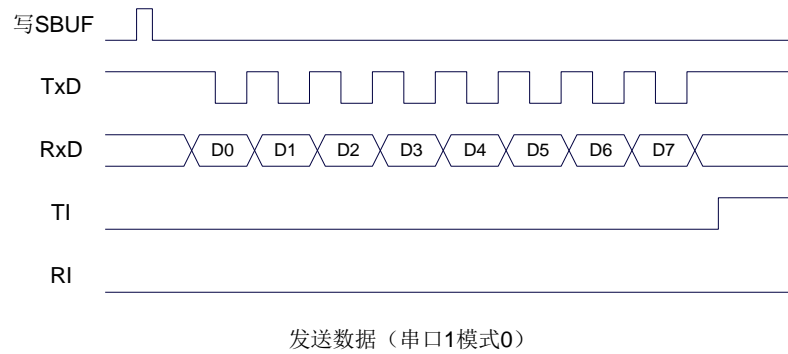
### 16.3.5 串口 1 模式 0, 模式 0 波特率计算公式

当串口 1 选择工作模式为模式 0 时, 串行通信接口工作在同步移位寄存器模式, 当串行口模式 0 的通信速度设置位 UART\_M0x6 为 0 时, 其波特率固定为系统时钟频率的 12 分频 ( $SYSClk/12$ ); 当设置 UART\_M0x6 为 1 时, 其波特率固定为系统时钟频率的 2 分频 ( $SYSClk/2$ )。RxD 为串行通讯的数据口, TxD 为同步移位脉冲输出脚, 发送、接收的是 8 位数据, 低位在先。

模式 0 的发送过程: 当主机执行将数据写入发送缓冲器 SBUF 指令时启动发送, 串行口即将 8 位数据以  $SYSClk/12$  或  $SYSClk/2$  (由 UART\_M0x6 确定是 12 分频还是 2 分频) 的波特率从 RxD 管脚输出(从低位到高位), 发送完中断标志 TI 置 1, TxD 管脚输出同步移位脉冲信号。当写信号有效后, 相隔一个时钟, 发送控制端 SEND 有效(高电平), 允许 RxD 发送数据, 同时允许 TxD 输出同步移位脉冲。一帧(8 位)数据发送完毕时, 各控制端均恢复原状态, 只有 TI 保持高电平, 呈中断申请状态。在再次发送数据前, 必须用软件将 TI 清 0。

模式 0 的接收过程: 首先将接收中断请求标志 RI 清零并置位允许接收控制位 REN 时启动模式 0 接收过程。启动接收过程后, RxD 为串行数据输入端, TxD 为同步脉冲输出端。串行接收的波特率为  $SYSClk/12$  或  $SYSClk/2$  (由 UART\_M0x6 确定是 12 分频还是 2 分频)。当接收完成一帧数据(8 位)后, 控制信号复位, 中断标志 RI 被置 1, 呈中断申请状态。当再次接收时, 必须通过软件将 RI 清 0





工作于模式 0 时，必须清 0 多机通信控制位 SM2，使之不影响 TB8 位和 RB8 位。由于波特率固定为 SYSClk/12 或 SYSClk/2，无需定时器提供，直接由单片机的时钟作为同步移位脉冲。

串口 1 模式 0 的波特率计算公式如下表所示（SYSClk 为系统工作频率）：

| UART_M0x6 | 波特率计算公式                                 |
|-----------|---|
| 0         | $\text{波特率} = \frac{\text{SYSClk}}{12}$ |
| 1         | $\text{波特率} = \frac{\text{SYSClk}}{2}$  |

## 16.3.6 串口 1 模式 1，模式 1 波特率计算公式

当软件设置 SCON 的 SM0、SM1 为“01”时，串行口 1 则以模式 1 进行工作。此模式为 8 位 UART 格式，一帧信息为 10 位：1 位起始位，8 位数据位（低位在先）和 1 位停止位。波特率可变，即可根据需要进行设置波特率。TxD 为数据发送口，RxD 为数据接收口，串行口全双工接受/发送。

模式 1 的发送过程：串行通信模式发送时，数据由串行发送端 TxD 输出。当主机执行一条写 SBUF 的指令就启动串行通信的发送，写“SBUF”信号还把“1”装入发送移位寄存器的第 9 位，并通知 TX 控制单元开始发送。移位寄存器将数据不断右移送 TxD 端口发送，在数据的左边不断移入“0”作补充。当数据的最高位移到移位寄存器的输出位置，紧跟其后的是第 9 位“1”，在它的左边各位全为“0”，这个状态条件，使 TX 控制单元作最后一次移位输出，然后使允许发送信号“SEND”失效，完成一帧信息的发送，并置位中断请求位 TI，即 TI=1，向主机请求中断处理。

模式 1 的接收过程：当软件置位接收允许标志位 REN，即 REN=1 时，接收器便对 RxD 端口的信号进行检测，当检测到 RxD 端口发送从“1”→“0”的下降沿跳变时就启动接收器准备接收数据，并立即复位波特率发生器的接收计数器，将 1FFH 装入移位寄存器。接收的数据从接收移位寄存器的右边移入，已装入的 1FFH 向左边移出，当起始位"0"移到移位寄存器的最左边时，使 RX 控制器作最后一次移位，完成一帧的接收。若同时满足以下两个条件：

- RI=0;
- SM2=0 或接收到的停止位为 1。

则接收到的数据有效，实现装载入 SBUF，停止位进入 RB8，RI 标志位被置 1，向主机请求中断，若上述两条件不能同时满足，则接收到的数据作废并丢失，无论条件满足与否，接收器重又检测 RxD 端口上的"1"→"0"的跳变，继续下一帧的接收。接收有效，在响应中断后，RI 标志位必须由软件清 0。通常情况下，串行通信工作于模式 1 时，SM2 设置为"0"。



串口 1 的波特率是可变的，其波特率可由定时器 1 或者定时器 2 产生。当定时器采用 1T 模式时（12 倍速），相应的波特率的速度也会相应提高 12 倍。



串口 1 模式 1 的波特率计算公式如下表所示: (SYSclk 为系统工作频率)

| 选择定时器       | 定时器速度 | 重装值计算公式   | 波特率  |
|-------------|-------|---|--|
| 定时器2        | 1T    | 定时器2重载值 = $65536 - \frac{\text{SYSclk}}{4 \times \text{波特率}}$                                 | 波特率 = $\frac{\text{SYSclk}}{4 \times (65536 - \text{定时器重装数})}$                                 |
|             | 12T   | 定时器2重载值 = $65536 - \frac{\text{SYSclk}}{12 \times 4 \times \text{波特率}}$                       | 波特率 = $\frac{\text{SYSclk}}{12 \times 4 \times (65536 - \text{定时器重装数})}$                       |
| 定时器1<br>模式0 | 1T    | 定时器1重载值 = $65536 - \frac{\text{SYSclk}}{4 \times \text{波特率}}$                                 | 波特率 = $\frac{\text{SYSclk}}{4 \times (65536 - \text{定时器重装数})}$                                 |
|             | 12T   | 定时器1重载值 = $65536 - \frac{\text{SYSclk}}{12 \times 4 \times \text{波特率}}$                       | 波特率 = $\frac{\text{SYSclk}}{12 \times 4 \times (65536 - \text{定时器重装数})}$                       |
| 定时器1<br>模式2 | 1T    | 定时器1重载值 = $256 - \frac{2^{\text{SMOD}} \times \text{SYSclk}}{32 \times \text{波特率}}$           | 波特率 = $\frac{2^{\text{SMOD}} \times \text{SYSclk}}{32 \times (256 - \text{定时器重装数})}$           |
|             | 12T   | 定时器1重载值 = $256 - \frac{2^{\text{SMOD}} \times \text{SYSclk}}{12 \times 32 \times \text{波特率}}$ | 波特率 = $\frac{2^{\text{SMOD}} \times \text{SYSclk}}{12 \times 32 \times (256 - \text{定时器重装数})}$ |

下面为常用频率与常用波特率所对应定时器的重载值

| 频率<br>(MHz) | 波特率    | 定时器 2 |        | 定时器 1 模式 0 |        | 定时器 1 模式 2 |        |        |        |
|-------------|--------|-------|--------|------------|--------|------------|--------|--------|--------|
|             |        | 1T 模式 | 12T 模式 | 1T 模式      | 12T 模式 | SMOD=1     |        | SMOD=0 |        |
|             |        |       |        |            |        | 1T 模式      | 12T 模式 | 1T 模式  | 12T 模式 |
| 11.0592     | 115200 | FFE8H | FFFEH  | FFE8H      | FFFEH  | FAH        | -      | FDH    | -      |
|             | 57600  | FFD0H | FFFCH  | FFD0H      | FFFCH  | F4H        | FFH    | FAH    | -      |
|             | 38400  | FFB8H | FFFAH  | FFB8H      | FFFAH  | EEH        | -      | F7H    | -      |
|             | 19200  | FF70H | FFF4H  | FF70H      | FFF4H  | DCH        | FDH    | EEH    | -      |
|             | 9600   | FEE0H | FFE8H  | FEE0H      | FFE8H  | B8H        | FAH    | DCH    | FDH    |
| 18.432      | 115200 | FFD8H | -      | FFD8H      | -      | F6H        | -      | FBH    | -      |
|             | 57600  | FFB0H | -      | FFB0H      | -      | ECH        | -      | F6H    | -      |
|             | 38400  | FF88H | FFF6H  | FF88H      | FFF6H  | E2H        | -      | F1H    | -      |
|             | 19200  | FF10H | FFECH  | FF10H      | FFECH  | C4H        | FBH    | E2H    | -      |
|             | 9600   | FE20H | FFD8H  | FE20H      | FFD8H  | 88H        | F6H    | C4H    | FBH    |
| 22.1184     | 115200 | FFD0H | FFFCH  | FFD0H      | FFFCH  | F4H        | FFH    | FAH    | -      |
|             | 57600  | FFA0H | FFF8H  | FFA0H      | FFF8H  | E8H        | FEH    | F4H    | FFH    |
|             | 38400  | FF70H | FFF4H  | FF70H      | FFF4H  | DCH        | FDH    | EEH    | -      |
|             | 19200  | FEE0H | FFE8H  | FEE0H      | FFE8H  | B8H        | FAH    | DCH    | FDH    |
|             | 9600   | FDC0H | FFD0H  | FDC0H      | FFD0H  | 70H        | F4H    | B8H    | FAH    |

### 16.3.7 串口 1 模式 2，模式 2 波特率计算公式

当 SM0、SM1 两位为 10 时，串行口 1 工作在模式 2。串行口 1 工作模式 2 为 9 位数据异步通信 UART 模式，其一帧的信息由 11 位组成：1 位起始位，8 位数据位（低位在先），1 位可编程位（第 9 位数据）和 1 位停止位。发送时可编程位（第 9 位数据）由 SCON 中的 TB8 提供，可软件设置为 1 或 0，或者可将 PSW 中的奇/偶校验位 P 值装入 TB8（TB8 既可作为多机通信中的地址数据标志位，又可作为数据的奇偶校验位）。接收时第 9 位数据装入 SCON 的 RB8。TxD 为发送端口，RxD 为接收端口，以全双工模式进行接收/发送。

模式 2 的波特率固定为系统时钟的 64 分频或 32 分频（取决于 PCON 中 SMOD 的值）

串口 1 模式 2 的波特率计算公式如下表所示（SYSclk 为系统工作频率）：

| SMOD | 波特率计算公式                                 |
|------|---|
| 0    | $\text{波特率} = \frac{\text{SYSclk}}{64}$ |
| 1    | $\text{波特率} = \frac{\text{SYSclk}}{32}$ |

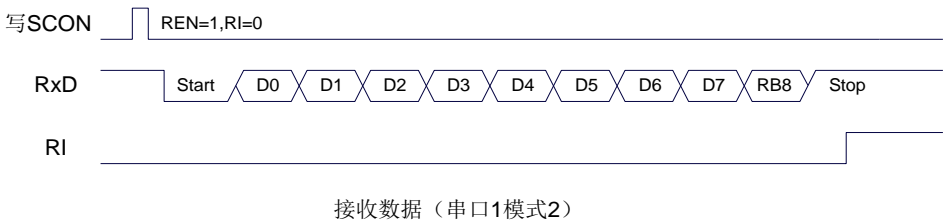
模式 2 和模式 1 相比，除波特率发生源略有不同，发送时由 TB8 提供给移位寄存器第 9 数据位不同外，其余功能结构均基本相同，其接收/发送操作过程及时序也基本相同。

当接收器接收完一帧信息后必须同时满足下列条件：

- RI=0
- SM2=0 或者 SM2=1 且接收到的第 9 数据位 RB8=1。

当上述两条件同时满足时，才将接收到的移位寄存器的数据装入 SBUF 和 RB8 中，RI 标志位被置 1，并向主机请求中断处理。如果上述条件有一个不满足，则刚接收到移位寄存器中的数据无效而丢失，也不置位 RI。无论上述条件满足与否，接收器又重新开始检测 RxD 输入端口的跳变信息，接收下一帧的输入信息。在模式 2 中，接收到的停止位与 SBUF、RB8 和 RI 无关。

通过软件对 SCON 中的 SM2、TB8 的设置以及通信协议的约定，为多机通信提供了方便。



### 16.3.8 串口 1 模式 3，模式 3 波特率计算公式

当 SM0、SM1 两位为 11 时，串行口 1 工作在模式 3。串行通信模式 3 为 9 位数据异步通信 UART 模式，其一帧的信息由 11 位组成：1 位起始位，8 位数据位（低位在先），1 位可编程位（第 9 位数据）和 1 位停止位。发送时可编程位（第 9 位数据）由 SCON 中的 TB8 提供，可软件设置为 1 或 0，或者可将 PSW 中的奇/偶校验位 P 值装入 TB8（TB8 既可作为多机通信中的地址数据标志位，又可作为数据的奇偶校验位）。接收时第 9 位数据装入 SCON 的 RB8。TxD 为发送端口，RxD 为接收端口，以全双工模式进行接收/发送。

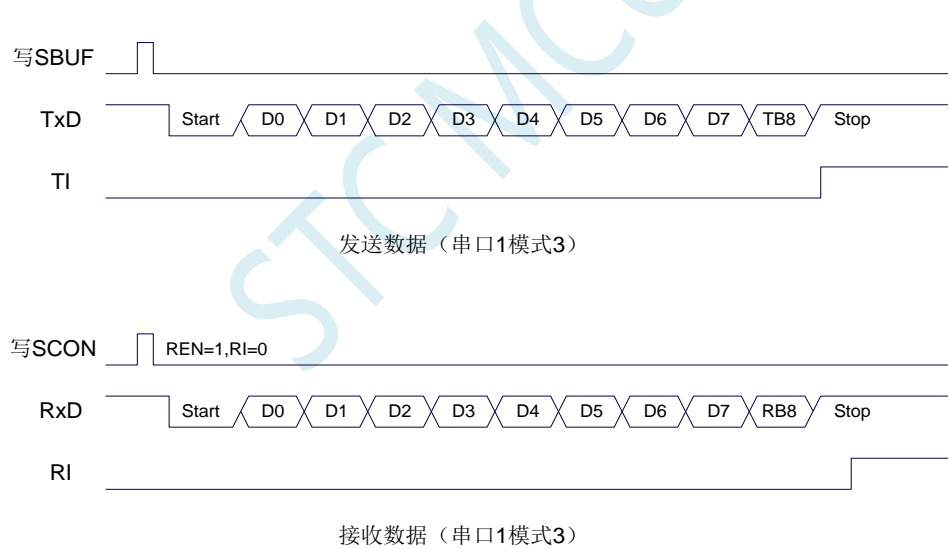
模式 3 和模式 1 相比，除发送时由 TB8 提供给移位寄存器第 9 数据位不同外，其余功能结构均基本相同，其接收‘发送操作过程及时序也基本相同。

当接收器接收完一帧信息后必须同时满足下列条件：

- RI=0
- SM2=0 或者 SM2=1 且接收到的第 9 数据位 RB8=1。

当上述两条件同时满足时，才将接收到的移位寄存器的数据装入 SBUF 和 RB8 中，RI 标志位被置 1，并向主机请求中断处理。如果上述条件有一个不满足，则刚接收到移位寄存器中的数据无效而丢失，也不置位 RI。无论上述条件满足与否，接收器又重新开始检测 RxD 输入端口的跳变信息，接收下一帧的输入信息。在模式 3 中，接收到的停止位与 SBUF、RB8 和 RI 无关。

通过软件对 SCON 中的 SM2、TB8 的设置以及通信协议的约定，为多机通信提供了方便。



串口 1 模式 3 的波特率计算公式与模式 1 是完全相同的。请参考模式 1 的波特率计算公式。

### 16.3.9 自动地址识别

#### 16.3.10 串口 1 从机地址控制寄存器（SADDR，SADEN）

| 符号    | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------|-----|----|----|----|----|----|----|----|----|
| SADDR | A9H |    |    |    |    |    |    |    |    |
| SADEN | B9H |    |    |    |    |    |    |    |    |

SADDR：从机地址寄存器  
SADEN：从机地址屏蔽位寄存器

自动地址识别功能典型应用在多机通讯领域，其主要原理是从机系统通过硬件比较功能来识别来自于主机串口数据流中的地址信息，通过寄存器 SADDR 和 SADEN 设置的本机的从机地址，硬件自动对从机地址进行过滤，当来自于主机的从机地址信息与本机所设置的从机地址相匹配时，硬件产生串口中断；否则硬件自动丢弃串口数据，而不产生中断。当众多处于空闲模式的从机链接在一起时，只有从机地址相匹配的从机才会从空闲模式唤醒，从而可以大大降低从机 MCU 的功耗，即使从机处于正常工作状态也可避免不停地进入串口中断而降低系统执行效率。

要使用串口的自动地址识别功能，首先需要将参与通讯的 MCU 的串口通讯模式设置为模式 2 或者模式 3（通常都选择波特率可变的模式 3，因为模式 2 的波特率是固定的，不便于调节），并开启从机的 SCON 的 SM2 位。对于串口模式 2 或者模式 3 的 9 位数据位中，第 9 位数据（存放在 RB8 中）为地址/数据的标志位，当第 9 位数据为 1 时，表示前面的 8 位数据（存放在 SBUF 中）为地址信息。当 SM2 被设置为 1 时，从机 MCU 会自动过滤掉非地址数据（第 9 位为 0 的数据），而对 SBUF 中的地址数据（第 9 位为 1 的数据）自动与 SADDR 和 SADEN 所设置的本机地址进行比较，若地址相匹配，则会将 RI 置“1”，并产生中断，否则不予处理本次接收的串口数据。

从机地址的设置是通过 SADDR 和 SADEN 两个寄存器进行设置的。SADDR 为从机地址寄存器，里面存放本机的从机地址。SADEN 为从机地址屏蔽位寄存器，用于设置地址信息中的忽略位，设置方法如下：

例如

SADDR = 11001010

SADEN = 10000001

则匹配地址为 1xxxxxx0

即，只要主机送出的地址数据中的 bit0 为 0 且 bit7 为 1 就可以和本机地址相匹配

再例如

SADDR = 11001010

SADEN = 00001111

则匹配地址为 xxxx1010

即，只要主机送出的地址数据中的低 4 位为 1010 就可以和本机地址相匹配，而高 4 为被忽略，可以为任意值。

主机可以使用广播地址（FFH）同时选中所有的从机来进行通讯。

## 16.4 串口 2

### 16.4.1 串口 2 控制寄存器 (S2CON)

| 符号    | 地址  | B7    | B6 | B5    | B4    | B3    | B2    | B1   | B0   |
|-------|-----|-------|----|-------|-------|-------|-------|------|------|
| S2CON | 9AH | S2SM0 | -  | S2SM2 | S2REN | S2TB8 | S2RB8 | S2TI | S2RI |

S2SM0: 指定串口2的通信工作模式, 如下表所示:

| S2SM0 | 串口2工作模式 | 功能说明        |
|-------|---------|-------------|
| 0     | 模式0     | 可变波特率8位数据方式 |
| 1     | 模式1     | 可变波特率9位数据方式 |

S2SM2: 允许串口 2 在模式 1 时允许多机通信控制位。在模式 1 时, 如果 S2SM2 位为 1 且 S2REN 位为 1, 则接收机处于地址帧筛选状态。此时可以利用接收到的第 9 位 (即 S2RB8) 来筛选地址帧: 若 S2RB8=1, 说明该帧是地址帧, 地址信息可以进入 S2BUF, 并使 S2RI 为 1, 进而在中断服务程序中再进行地址号比较; 若 S2RB8=0, 说明该帧不是地址帧, 应丢掉且保持 S2RI=0。在模式 1 中, 如果 S2SM2 位为 0 且 S2REN 位为 1, 接收机处于地址帧筛选被禁止状态。不论收到的 S2RB8 为 0 或 1, 均可使接收到的信息进入 S2BUF, 并使 S2RI=1, 此时 S2RB8 通常为校验位。模式 0 为非多机通信方式, 在这种方式时, 要设置 S2SM2 应为 0。

S2REN: 允许/禁止串口接收控制位

0: 禁止串口接收数据

1: 允许串口接收数据

S2TB8: 当串口 2 使用模式 1 时, S2TB8 为要发送的第 9 位数据, 一般用作校验位或者地址帧/数据帧标志位, 按需要由软件置位或清 0。在模式 0 中, 该位不用。

S2RB8: 当串口 2 使用模式 1 时, S2RB8 为接收到的第 9 位数据, 一般用作校验位或者地址帧/数据帧标志位。在模式 0 中, 该位不用。

S2TI: 串口 2 发送中断请求标志位。在停止位开始发送时由硬件自动将 S2TI 置 1, 向 CPU 发请求中断, 响应中断后 S2TI 必须用软件清零。

S2RI: 串口 2 接收中断请求标志位。串行接收到停止位的中间时刻由硬件自动将 S2RI 置 1, 向 CPU 发中断申请, 响应中断后 S2RI 必须由软件清零。

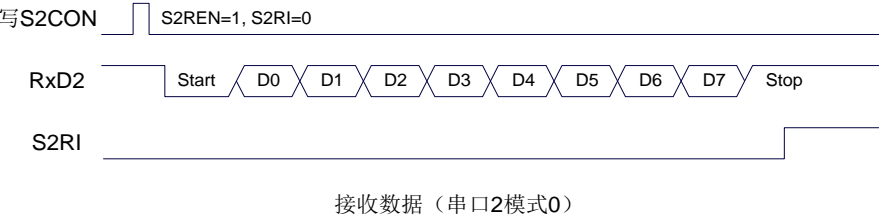
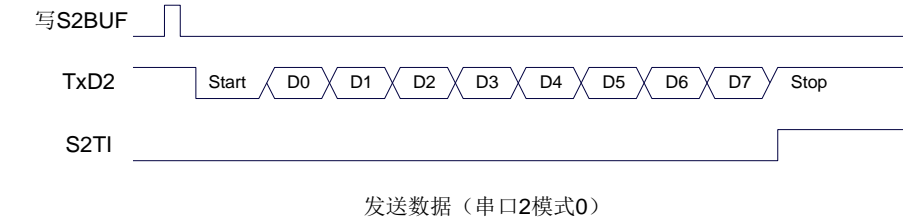
### 16.4.2 串口 2 数据寄存器 (S2BUF)

| 符号    | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------|-----|----|----|----|----|----|----|----|----|
| S2BUF | 9BH |    |    |    |    |    |    |    |    |

S2BUF: 串口 1 数据接收/发送缓冲区。S2BUF 实际是 2 个缓冲器, 读缓冲器和写缓冲器, 两个操作分别对应两个不同的寄存器, 1 个是只写寄存器 (写缓冲器), 1 个是只读寄存器 (读缓冲器)。对 S2BUF 进行读操作, 实际是读取串口接收缓冲区, 对 S2BUF 进行写操作则是触发串口开始发送数据。

### 16.4.3 串口 2 模式 0, 模式 0 波特率计算公式

串行口 2 的模式 0 为 8 位数据位可变波特率 UART 工作模式。此模式一帧信息为 10 位: 1 位起始位, 8 位数据位 (低位在先) 和 1 位停止位。波特率可变, 可根据需要进行设置波特率。TxD2 为数据发送口, RxD2 为数据接收口, 串行口全双工接受/发送。



串口 2 的波特率是可变的，其波特率由定时器 2 产生。当定时器采用 1T 模式时（12 倍速），相应的波特率的速度也会相应提高 12 倍。

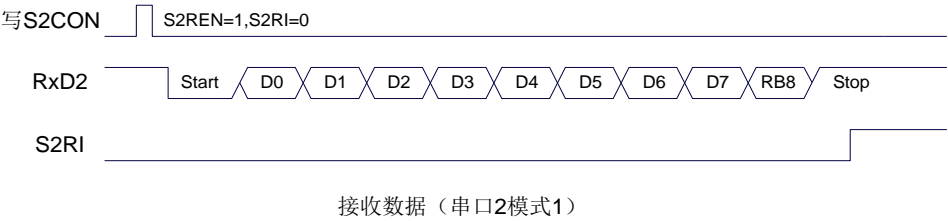
串口 2 模式 0 的波特率计算公式如下表所示：（SYSclk 为系统工作频率）

| 选择定时器 | 定时器速度 | 重装值计算公式   | 波特率  |
|-------|-------|---|--|
| 定时器2  | 1T    | 定时器2重载值 = $65536 - \frac{\text{SYSclk}}{4 \times \text{波特率}}$           | 波特率 = $\frac{\text{SYSclk}}{4 \times (65536 - \text{定时器重装数})}$           |
|       | 12T   | 定时器2重载值 = $65536 - \frac{\text{SYSclk}}{12 \times 4 \times \text{波特率}}$ | 波特率 = $\frac{\text{SYSclk}}{12 \times 4 \times (65536 - \text{定时器重装数})}$ |

16.4.4 串口 2 模式 1，模式 1 波特率计算公式

串行口 2 的模式 1 为 9 位数据位可变波特率 UART 工作模式。此模式一帧信息为 11 位：1 位起始位，9 位数据位（低位在先）和 1 位停止位。波特率可变，可根据需要进行设置波特率。TxD2 为数据发送口，RxD2 为数据接收口，串行口全双工接受/发送。





串口 2 模式 1 的波特率计算公式与模式 0 是完全相同的。请参考模式 0 的波特率计算公式。



## 16.5 串口 3

### 16.5.1 串口 3 控制寄存器 (S3CON)

| 符号    | 地址  | B7    | B6    | B5    | B4    | B3    | B2    | B1   | B0   |
|-------|-----|-------|-------|-------|-------|-------|-------|------|------|
| S3CON | ACH | S3SM0 | S3ST3 | S3SM2 | S3REN | S3TB8 | S3RB8 | S3TI | S3RI |

S3SM0: 指定串口3的通信工作模式, 如下表所示:

| S3SM0 | 串口3工作模式 | 功能说明        |
|-------|---------|-------------|
| 0     | 模式0     | 可变波特率8位数据方式 |
| 1     | 模式1     | 可变波特率9位数据方式 |

S3ST3: 选择串口 3 的波特率发生器

0: 选择定时器 2 为串口 3 的波特率发生器

1: 选择定时器 3 为串口 3 的波特率发生器

S3SM2: 允许串口 3 在模式 1 时允许多机通信控制位。在模式 1 时, 如果 S3SM2 位为 1 且 S3REN 位为 1, 则接收机处于地址帧筛选状态。此时可以利用接收到的第 9 位 (即 S3RB8) 来筛选地址帧: 若 S3RB8=1, 说明该帧是地址帧, 地址信息可以进入 S3BUF, 并使 S3RI 为 1, 进而在中断服务程序中再进行地址号比较; 若 S3RB8=0, 说明该帧不是地址帧, 应丢掉且保持 S3RI=0。在模式 1 中, 如果 S3SM2 位为 0 且 S3REN 位为 1, 接收机处于地址帧筛选被禁止状态。不论收到的 S3RB8 为 0 或 1, 均可使接收到的信息进入 S3BUF, 并使 S3RI=1, 此时 S3RB8 通常为校验位。模式 0 为非多机通信方式, 在这种方式时, 要设置 S3SM2 应为 0。

S3REN: 允许/禁止串口接收控制位

0: 禁止串口接收数据

1: 允许串口接收数据

S3TB8: 当串口 3 使用模式 1 时, S3TB8 为要发送的第 9 位数据, 一般用作校验位或者地址帧/数据帧标志位, 按需要由软件置位或清 0。在模式 0 中, 该位不用。

S3RB8: 当串口 3 使用模式 1 时, S3RB8 为接收到的第 9 位数据, 一般用作校验位或者地址帧/数据帧标志位。在模式 0 中, 该位不用。

S3TI: 串口 3 发送中断请求标志位。在停止位开始发送时由硬件自动将 S3TI 置 1, 向 CPU 发请求中断, 响应中断后 S3TI 必须用软件清零。

S3RI: 串口 3 接收中断请求标志位。串行接收到停止位的中间时刻由硬件自动将 S3RI 置 1, 向 CPU 发中断申请, 响应中断后 S3RI 必须由软件清零。

### 16.5.2 串口 3 数据寄存器 (S3BUF)

| 符号    | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------|-----|----|----|----|----|----|----|----|----|
| S3BUF | ADH |    |    |    |    |    |    |    |    |

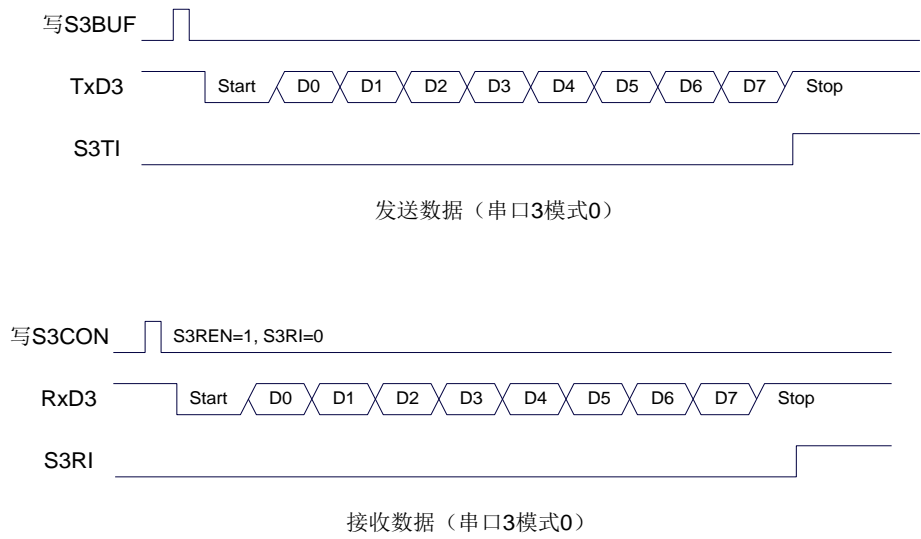
S3BUF: 串口 1 数据接收/发送缓冲区。S3BUF 实际是 2 个缓冲器, 读缓冲器和写缓冲器, 两个操作分别对应两个不同的寄存器, 1 个是只写寄存器 (写缓冲器), 1 个是只读寄存器 (读缓冲器)。对 S3BUF 进行读操作, 实际是读取串口接收缓冲区, 对 S3BUF 进行写操作则是触发串口开始发送数据。

### 16.5.3 串口 3 模式 0, 模式 0 波特率计算公式

串行口 3 的模式 0 为 8 位数据位可变波特率 UART 工作模式。此模式一帧信息为 10 位: 1 位起始位,



8 位数据位（低位在先）和 1 位停止位。波特率可变，可根据需要进行设置波特率。TxD3 为数据发送口，RxD3 为数据接收口，串行口全双工接受/发送。



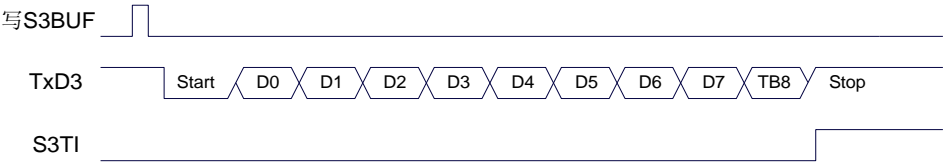
串口 3 的波特率是可变的，其波特率可由定时器 2 或定时器 3 产生。当定时器采用 1T 模式时（12 倍速），相应的波特率的速度也会相应提高 12 倍。

串口 3 模式 0 的波特率计算公式如下表所示：（SYSclk 为系统工作频率）

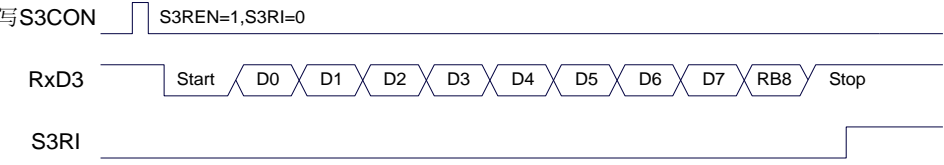
| 选择定时器 | 定时器速度 | 重装值计算公式  | 波特率   |
|-------|-------|--|---|
| 定时器2  | 1T    | $\text{定时器2重载值} = 65536 - \frac{\text{SYSclk}}{4 \times \text{波特率}}$           | $\text{波特率} = \frac{\text{SYSclk}}{4 \times (65536 - \text{定时器重装数})}$           |
|       | 12T   | $\text{定时器2重载值} = 65536 - \frac{\text{SYSclk}}{12 \times 4 \times \text{波特率}}$ | $\text{波特率} = \frac{\text{SYSclk}}{12 \times 4 \times (65536 - \text{定时器重装数})}$ |
| 定时器3  | 1T    | $\text{定时器3重载值} = 65536 - \frac{\text{SYSclk}}{4 \times \text{波特率}}$           | $\text{波特率} = \frac{\text{SYSclk}}{4 \times (65536 - \text{定时器重装数})}$           |
|       | 12T   | $\text{定时器3重载值} = 65536 - \frac{\text{SYSclk}}{12 \times 4 \times \text{波特率}}$ | $\text{波特率} = \frac{\text{SYSclk}}{12 \times 4 \times (65536 - \text{定时器重装数})}$ |

16.5.4 串口 3 模式 1，模式 1 波特率计算公式

串行口 3 的模式 1 为 9 位数据位可变波特率 UART 工作模式。此模式一帧信息为 11 位：1 位起始位，9 位数据位（低位在先）和 1 位停止位。波特率可变，可根据需要进行设置波特率。TxD3 为数据发送口，RxD3 为数据接收口，串行口全双工接受/发送。



发送数据（串口3模式1）



接收数据（串口3模式1）

串口 3 模式 1 的波特率计算公式与模式 0 是完全相同的。请参考模式 0 的波特率计算公式。

## 16.6 串口 4

### 16.6.1 串口 4 控制寄存器 (S4CON)

| 符号    | 地址  | B7    | B6    | B5    | B4    | B3    | B2    | B1   | B0   |
|-------|-----|-------|-------|-------|-------|-------|-------|------|------|
| S4CON | 84H | S4SM0 | S4ST4 | S4SM2 | S4REN | S4TB8 | S4RB8 | S4TI | S4RI |

S4SM0: 指定串口4的通信工作模式, 如下表所示:

| S4SM0 | 串口4工作模式 | 功能说明        |
|-------|---------|-------------|
| 0     | 模式0     | 可变波特率8位数据方式 |
| 1     | 模式1     | 可变波特率9位数据方式 |

S4ST4: 选择串口 4 的波特率发生器

0: 选择定时器 2 为串口 4 的波特率发生器

1: 选择定时器 4 为串口 4 的波特率发生器

S4SM2: 允许串口 4 在模式 1 时允许多机通信控制位。在模式 1 时, 如果 S4SM2 位为 1 且 S4REN 位为 1, 则接收机处于地址帧筛选状态。此时可以利用接收到的第 9 位 (即 S4RB8) 来筛选地址帧: 若 S4RB8=1, 说明该帧是地址帧, 地址信息可以进入 S4BUF, 并使 S4RI 为 1, 进而在中断服务程序中再进行地址号比较; 若 S4RB8=0, 说明该帧不是地址帧, 应丢掉且保持 S4RI=0。在模式 1 中, 如果 S4SM2 位为 0 且 S4REN 位为 1, 接收机处于地址帧筛选被禁止状态。不论收到的 S4RB8 为 0 或 1, 均可使接收到的信息进入 S4BUF, 并使 S4RI=1, 此时 S4RB8 通常为校验位。模式 0 为非多机通信方式, 在这种方式时, 要设置 S4SM2 应为 0。

S4REN: 允许/禁止串口接收控制位

0: 禁止串口接收数据

1: 允许串口接收数据

S4TB8: 当串口 4 使用模式 1 时, S4TB8 为要发送的第 9 位数据, 一般用作校验位或者地址帧/数据帧标志位, 按需要由软件置位或清 0。在模式 0 中, 该位不用。

S4RB8: 当串口 4 使用模式 1 时, S4RB8 为接收到的第 9 位数据, 一般用作校验位或者地址帧/数据帧标志位。在模式 0 中, 该位不用。

S4TI: 串口 4 发送中断请求标志位。在停止位开始发送时由硬件自动将 S4TI 置 1, 向 CPU 发请求中断, 响应中断后 S4TI 必须用软件清零。

S4RI: 串口 4 接收中断请求标志位。串行接收到停止位的中间时刻由硬件自动将 S4RI 置 1, 向 CPU 发中断申请, 响应中断后 S4RI 必须由软件清零。

### 16.6.2 串口 4 数据寄存器 (S4BUF)

| 符号    | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------|-----|----|----|----|----|----|----|----|----|
| S4BUF | 85H |    |    |    |    |    |    |    |    |

S4BUF: 串口 1 数据接收/发送缓冲区。S4BUF 实际是 2 个缓冲器, 读缓冲器和写缓冲器, 两个操作分别对应两个不同的寄存器, 1 个是只写寄存器 (写缓冲器), 1 个是只读寄存器 (读缓冲器)。对 S4BUF 进行读操作, 实际是读取串口接收缓冲区, 对 S4BUF 进行写操作则是触发串口开始发送数据。

### 16.6.3 串口 4 模式 0, 模式 0 波特率计算公式

串行口 4 的模式 0 为 8 位数据位可变波特率 UART 工作模式。此模式一帧信息为 10 位: 1 位起始位, 8 位数据位 (低位在先) 和 1 位停止位。波特率可变, 可根据需要进行设置波特率。TxD4 为数据发送口,

RxD4 为数据接收口，串行口全双工接受/发送。



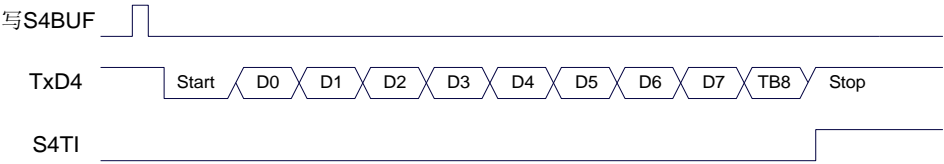
串口 4 的波特率是可变的，其波特率可由定时器 2 或定时器 4 产生。当定时器采用 1T 模式时（12 倍速），相应的波特率的速度也会相应提高 12 倍。

串口 4 模式 0 的波特率计算公式如下表所示：（SYSclk 为系统工作频率）

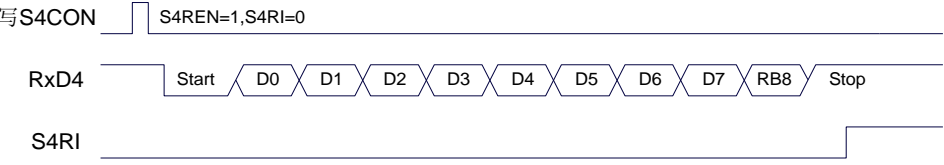
| 选择定时器 | 定时器速度 | 重装值计算公式  | 波特率   |
|-------|-------|--|---|
| 定时器2  | 1T    | 定时器2重载值 = $65536 - \frac{SYSclk}{4 \times \text{波特率}}$           | 波特率 = $\frac{SYSclk}{4 \times (65536 - \text{定时器重装数})}$           |
|       | 12T   | 定时器2重载值 = $65536 - \frac{SYSclk}{12 \times 4 \times \text{波特率}}$ | 波特率 = $\frac{SYSclk}{12 \times 4 \times (65536 - \text{定时器重装数})}$ |
| 定时器4  | 1T    | 定时器4重载值 = $65536 - \frac{SYSclk}{4 \times \text{波特率}}$           | 波特率 = $\frac{SYSclk}{4 \times (65536 - \text{定时器重装数})}$           |
|       | 12T   | 定时器4重载值 = $65536 - \frac{SYSclk}{12 \times 4 \times \text{波特率}}$ | 波特率 = $\frac{SYSclk}{12 \times 4 \times (65536 - \text{定时器重装数})}$ |

16.6.4 串口 4 模式 1，模式 1 波特率计算公式

串行口 4 的模式 1 为 9 位数据位可变波特率 UART 工作模式。此模式一帧信息为 11 位：1 位起始位，9 位数据位（低位在先）和 1 位停止位。波特率可变，可根据需要进行设置波特率。TxD4 为数据发送口，RxD4 为数据接收口，串行口全双工接受/发送。



发送数据（串口4模式1）



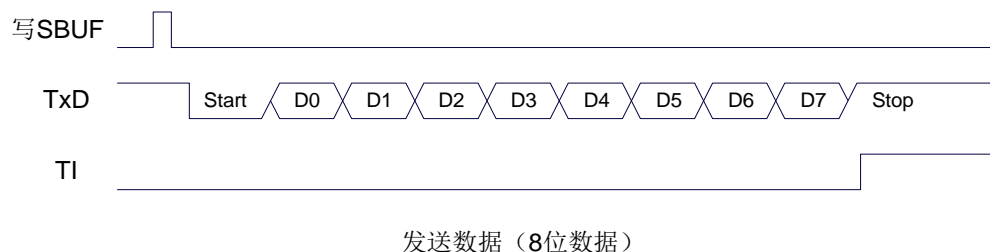
接收数据（串口4模式1）

串口 4 模式 1 的波特率计算公式与模式 0 是完全相同的。请参考模式 0 的波特率计算公式。

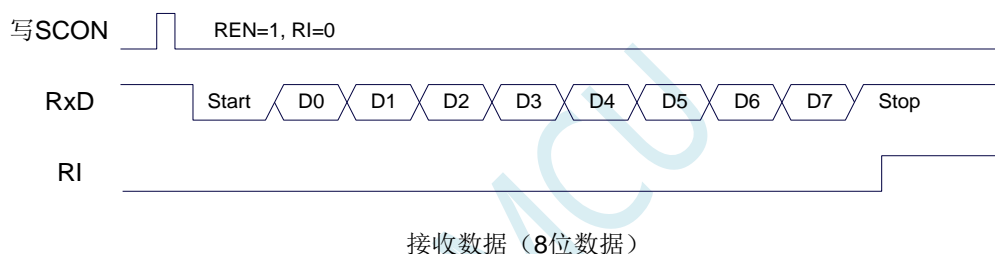
## 16.7 串口注意事项

关于串口中断请求有如下问题需要注意: (串口 1、串口 2、串口 3、串口 4 均类似, 下面以串口 1 为例进行说明)

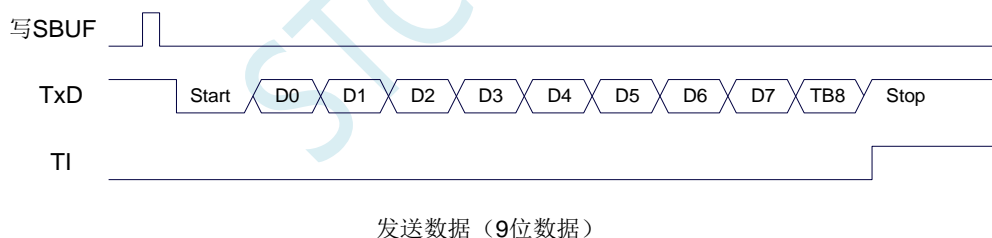
8 位数据模式时, 发送完成约 1/3 个停止位后产生 TI 中断请求, 如下图所示:



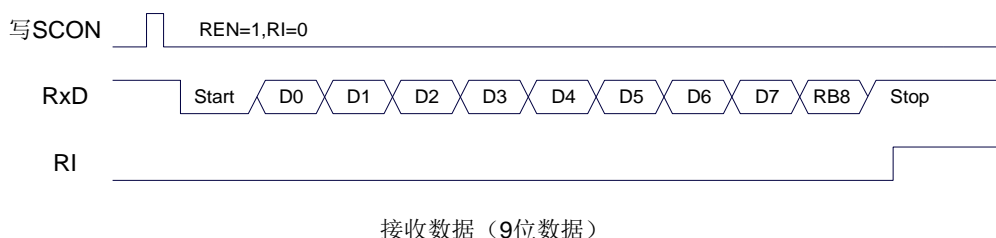
8 位数据模式时, 接收完成一半个停止位后产生 RI 中断请求, 如下图所示:



9 位数据模式时, 发送完成约 1/3 个停止位后产生 TI 中断请求:



9 位数据模式时, 一半个停止位后产生 RI 中断请求, 如下图所示:



## 16.8 范例程序

### 16.8.1 串口 1 使用定时器 2 做波特率发生器

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC 11059200UL
#define BRT (65536 - (FOSC / 115200 + 2) / 4)
```

//加2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
bit busy;
char wptr;
char rptr;
char buffer[16];
```

```
void UartIsr() interrupt 4
```

```
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}
```

```
void UartInit()
```

```
{
    SCON = 0x50;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x15;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}
```

```
void UartSend(char dat)
```

```
{
    while (busy);
    busy = 1;
    SBUF = dat;
}
```

```
void UartSendStr(char *p)
```

```
{
```

```
while (*p)
{
    UartSend(*p++);
}

void main()
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;
    UartSendStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UartSend(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

汇编代码

;测试工作频率为 11.0592MHz

|        |      |       |           |
|--------|------|-------|-----------|
| P_SW2  | DATA | 0BAH  |           |
| AUXR   | DATA | 8EH   |           |
| T2H    | DATA | 0D6H  |           |
| T2L    | DATA | 0D7H  |           |
| BUSY   | BIT  | 20H.0 |           |
| WPTR   | DATA | 21H   |           |
| RPTR   | DATA | 22H   |           |
| BUFFER | DATA | 23H   | ;16 bytes |
| P0M1   | DATA | 093H  |           |
| P0M0   | DATA | 094H  |           |
| P1M1   | DATA | 091H  |           |
| P1M0   | DATA | 092H  |           |
| P2M1   | DATA | 095H  |           |
| P2M0   | DATA | 096H  |           |



```

P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0023H
          LJMP         UART_ISR

          ORG          0100H

UART_ISR:
          PUSH         ACC
          PUSH         PSW
          MOV          PSW,#08H

          JNB          TI,CHKRI
          CLR          TI
          CLR          BUSY

CHKRI:
          JNB          RI,UARTISR_EXIT
          CLR          RI
          MOV          A,WPTR
          ANL          A,#0FH
          ADD          A,#BUFFER
          MOV          R0,A
          MOV          @R0,SBUF
          INC          WPTR

UARTISR_EXIT:
          POP          PSW
          POP          ACC
          RETI

UART_INIT:
          MOV          SCON,#50H
          MOV          T2L,#0E8H
          MOV          T2H,#0FFH
          MOV          AUXR,#15H
          CLR          BUSY
          MOV          WPTR,#00H
          MOV          RPTR,#00H
          RET

UART_SEND:
          JB           BUSY,$
          SETB         BUSY
          MOV          SBUF,A
          RET

UART_SENDSTR:
          CLR          A
          MOVC         A,@A+DPTR
          JZ           SENDEND
          LCALL        UART_SEND
          INC          DPTR
          JMP          UART_SENDSTR

```

;65536-11059200/115200/4=0FFE8H

**SENDEND:****RET****MAIN:**

**MOV SP, #5FH**  
**ORL P\_SW2, #80H** ;使能访问 XFR

**MOV P0M0, #00H**  
**MOV P0M1, #00H**  
**MOV P1M0, #00H**  
**MOV P1M1, #00H**  
**MOV P2M0, #00H**  
**MOV P2M1, #00H**  
**MOV P3M0, #00H**  
**MOV P3M1, #00H**  
**MOV P4M0, #00H**  
**MOV P4M1, #00H**  
**MOV P5M0, #00H**  
**MOV P5M1, #00H**

**LCALL UART\_INIT**  
**SETB ES**  
**SETB EA**

**MOV DPTR, #STRING**  
**LCALL UART\_SENDSTR**

**LOOP:**

**MOV A, RPTR**  
**XRL A, WPTR**  
**ANL A, #0FH**  
**JZ LOOP**  
**MOV A, RPTR**  
**ANL A, #0FH**  
**ADD A, #BUFFER**  
**MOV R0, A**  
**MOV A, @R0**  
**LCALL UART\_SEND**  
**INC RPTR**  
**JMP LOOP**

**STRING: DB 'Uart Test !', 0DH, 0AH, 00H**

**END**

## 16.8.2 串口 1 使用定时器 1（模式 0）做波特率发生器

### C 语言代码

//测试工作频率为 11.0592MHz

**#include "stc8h.h"**  
**#include "intrins.h"**

**#define FOSC 11059200UL**  
**#define BRT (65536 - (FOSC / 115200 + 2) / 4)**

//加2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
bit    busy;
char    wptr;
char    rptr;
char    buffer[16];

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void UartSendStr(char *p)
{
    while (*p)
    {
        UartSend(*p++);
    }
}

void main()
{
    P_SW2 /= 0x80;

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
```

//使能访问 XFR

```

P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

UartInit();
ES = 1;
EA = 1;
UartSendStr("Uart Test !\r\n");

while (1)
{
    if (rptr != wptr)
    {
        UartSend(buffer[rptr++]);
        rptr &= 0x0f;
    }
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|                  |             |                 |           |
|------------------|-------------|-----------------|-----------|
| <i>P_SW2</i>     | <i>DATA</i> | <i>0BAH</i>     |           |
| <i>AUXR</i>      | <i>DATA</i> | <i>8EH</i>      |           |
| <i>BUSY</i>      | <i>BIT</i>  | <i>20H.0</i>    |           |
| <i>WPTR</i>      | <i>DATA</i> | <i>21H</i>      |           |
| <i>RPTR</i>      | <i>DATA</i> | <i>22H</i>      |           |
| <i>BUFFER</i>    | <i>DATA</i> | <i>23H</i>      | ;16 bytes |
| <i>P0M1</i>      | <i>DATA</i> | <i>093H</i>     |           |
| <i>P0M0</i>      | <i>DATA</i> | <i>094H</i>     |           |
| <i>P1M1</i>      | <i>DATA</i> | <i>091H</i>     |           |
| <i>P1M0</i>      | <i>DATA</i> | <i>092H</i>     |           |
| <i>P2M1</i>      | <i>DATA</i> | <i>095H</i>     |           |
| <i>P2M0</i>      | <i>DATA</i> | <i>096H</i>     |           |
| <i>P3M1</i>      | <i>DATA</i> | <i>0B1H</i>     |           |
| <i>P3M0</i>      | <i>DATA</i> | <i>0B2H</i>     |           |
| <i>P4M1</i>      | <i>DATA</i> | <i>0B3H</i>     |           |
| <i>P4M0</i>      | <i>DATA</i> | <i>0B4H</i>     |           |
| <i>P5M1</i>      | <i>DATA</i> | <i>0C9H</i>     |           |
| <i>P5M0</i>      | <i>DATA</i> | <i>0CAH</i>     |           |
|                  | <i>ORG</i>  | <i>0000H</i>    |           |
|                  | <i>LJMP</i> | <i>MAIN</i>     |           |
|                  | <i>ORG</i>  | <i>0023H</i>    |           |
|                  | <i>LJMP</i> | <i>UART_ISR</i> |           |
|                  | <i>ORG</i>  | <i>0100H</i>    |           |
| <i>UART_ISR:</i> | <i>PUSH</i> | <i>ACC</i>      |           |

```

        PUSH        PSW
        MOV         PSW,#08H

        JNB         TI,CHKRI
        CLR         TI
        CLR         BUSY
CHKRI:
        JNB         RI,UARTISR_EXIT
        CLR         RI
        MOV         A,WPTR
        ANL         A,#0FH
        ADD         A,#BUFFER
        MOV         R0,A
        MOV         @R0,SBUF
        INC         WPTR
UARTISR_EXIT:
        POP         PSW
        POP         ACC
        RETI

UART_INIT:
        MOV         SCON,#50H
        MOV         TMOD,#00H
        MOV         TL1,#0E8H           ;65536-11059200/115200/4=0FFE8H
        MOV         TH1,#0FFH
        SETB        TRI
        MOV         AUXR,#40H
        CLR         BUSY
        MOV         WPTR,#00H
        MOV         RPTR,#00H
        RET

UART_SEND:
        JB          BUSY,$
        SETB        BUSY
        MOV         SBUF,A
        RET

UART_SENDSTR:
        CLR         A
        MOVC        A,@A+DPTR
        JZ          SENDEND
        LCALL       UART_SEND
        INC         DPTR
        JMP         UART_SENDSTR
SENDEND:
        RET

MAIN:
        MOV         SP,#5FH
        ORL         P_SW2,#80H         ;使能访问XFR

        MOV         P0M0,#00H
        MOV         P0M1,#00H
        MOV         P1M0,#00H
        MOV         P1M1,#00H
        MOV         P2M0,#00H
        MOV         P2M1,#00H
        MOV         P3M0,#00H

```

```

MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

LCALL    UART_INIT
SETB     ES
SETB     EA

MOV      DPTR, #STRING
LCALL    UART_SENDSTR

LOOP:
MOV      A, RPTR
XRL      A, WPTR
ANL      A, #0FH
JZ       LOOP
MOV      A, RPTR
ANL      A, #0FH
ADD      A, #BUFFER
MOV      R0, A
MOV      A, @R0
LCALL    UART_SEND
INC      RPTR
JMP      LOOP

STRING:   DB      'Uart Test !', 0DH, 0AH, 00H

END

```

### 16.8.3 串口 1 使用定时器 1（模式 2）做波特率发生器

#### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define FOSC      11059200UL
#define BRT       (256 - (FOSC / 115200 + 16) / 32)

```

//加 16 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```

bit      busy;
char     wptr;
char     rptra;
char     buffer[16];

```

```

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
}

```

```
}
if (RI)
{
    RI = 0;
    buffer[wptr++] = SBUF;
    wptr &= 0x0f;
}
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x20;
    TL1 = BRT;
    TH1 = BRT;
    TR1 = 1;
    AUXR = 0x40;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void UartSendStr(char *p)
{
    while (*p)
    {
        UartSend(*p++);
    }
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;
    UartSendStr("Uart Test !\r\n");
}
```

```

    while (1)
    {
        if (rptra != wptra)
        {
            UartSend(buffer[rptra++]);
            rptra &= 0x0f;
        }
    }
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|                  |             |                        |           |
|------------------|-------------|------------------------|-----------|
| <i>P_SW2</i>     | <i>DATA</i> | <i>0BAH</i>            |           |
| <i>AUXR</i>      | <i>DATA</i> | <i>8EH</i>             |           |
| <i>BUSY</i>      | <i>BIT</i>  | <i>20H.0</i>           |           |
| <i>WPTR</i>      | <i>DATA</i> | <i>21H</i>             |           |
| <i>RPTR</i>      | <i>DATA</i> | <i>22H</i>             |           |
| <i>BUFFER</i>    | <i>DATA</i> | <i>23H</i>             | ;16 bytes |
| <i>P0M1</i>      | <i>DATA</i> | <i>093H</i>            |           |
| <i>P0M0</i>      | <i>DATA</i> | <i>094H</i>            |           |
| <i>P1M1</i>      | <i>DATA</i> | <i>091H</i>            |           |
| <i>P1M0</i>      | <i>DATA</i> | <i>092H</i>            |           |
| <i>P2M1</i>      | <i>DATA</i> | <i>095H</i>            |           |
| <i>P2M0</i>      | <i>DATA</i> | <i>096H</i>            |           |
| <i>P3M1</i>      | <i>DATA</i> | <i>0B1H</i>            |           |
| <i>P3M0</i>      | <i>DATA</i> | <i>0B2H</i>            |           |
| <i>P4M1</i>      | <i>DATA</i> | <i>0B3H</i>            |           |
| <i>P4M0</i>      | <i>DATA</i> | <i>0B4H</i>            |           |
| <i>P5M1</i>      | <i>DATA</i> | <i>0C9H</i>            |           |
| <i>P5M0</i>      | <i>DATA</i> | <i>0CAH</i>            |           |
|                  | <i>ORG</i>  | <i>0000H</i>           |           |
|                  | <i>LJMP</i> | <i>MAIN</i>            |           |
|                  | <i>ORG</i>  | <i>0023H</i>           |           |
|                  | <i>LJMP</i> | <i>UART_ISR</i>        |           |
|                  | <i>ORG</i>  | <i>0100H</i>           |           |
| <i>UART_ISR:</i> |             |                        |           |
|                  | <i>PUSH</i> | <i>ACC</i>             |           |
|                  | <i>PUSH</i> | <i>PSW</i>             |           |
|                  | <i>MOV</i>  | <i>PSW,#08H</i>        |           |
|                  | <i>JNB</i>  | <i>TI,CHKRI</i>        |           |
|                  | <i>CLR</i>  | <i>TI</i>              |           |
|                  | <i>CLR</i>  | <i>BUSY</i>            |           |
| <i>CHKRI:</i>    |             |                        |           |
|                  | <i>JNB</i>  | <i>RI,UARTISR_EXIT</i> |           |
|                  | <i>CLR</i>  | <i>RI</i>              |           |
|                  | <i>MOV</i>  | <i>A,WPTR</i>          |           |
|                  | <i>ANL</i>  | <i>A,#0FH</i>          |           |
|                  | <i>ADD</i>  | <i>A,#BUFFER</i>       |           |
|                  | <i>MOV</i>  | <i>R0,A</i>            |           |
|                  | <i>MOV</i>  | <i>@R0,SBUF</i>        |           |



```

        INC            WPTR
UARTISR_EXIT:
        POP            PSW
        POP            ACC
        RETI

UART_INIT:
        MOV            SCON,#50H
        MOV            TMOD,#20H
        MOV            TL1,#0FDH           ;256-11059200/115200/32=0FDH
        MOV            TH1,#0FDH
        SETB           TRI
        MOV            AUXR,#40H
        CLR            BUSY
        MOV            WPTR,#00H
        MOV            RPTR,#00H
        RET

UART_SEND:
        JB             BUSY,$
        SETB           BUSY
        MOV            SBUF,A
        RET

UART_SENDSTR:
        CLR            A
        MOVC           A,@A+DPTR
        JZ             SENDEND
        LCALL          UART_SEND
        INC            DPTR
        JMP            UART_SENDSTR

SENDEND:
        RET

MAIN:
        MOV            SP,#5FH
        ORL            P_SW2,#80H           ;使能访问 XFR

        MOV            P0M0,#00H
        MOV            P0M1,#00H
        MOV            P1M0,#00H
        MOV            P1M1,#00H
        MOV            P2M0,#00H
        MOV            P2M1,#00H
        MOV            P3M0,#00H
        MOV            P3M1,#00H
        MOV            P4M0,#00H
        MOV            P4M1,#00H
        MOV            P5M0,#00H
        MOV            P5M1,#00H

        LCALL          UART_INIT
        SETB           ES
        SETB           EA

        MOV            DPTR,#STRING
        LCALL          UART_SENDSTR

LOOP:

```

```

MOV      A,RPTR
XRL      A,WPTR
ANL      A,#0FH
JZ        LOOP
MOV      A,RPTR
ANL      A,#0FH
ADD      A,#BUFFER
MOV      R0,A
MOV      A,@R0
LCALL    UART_SEND
INC      RPTR
JMP      LOOP

STRING:   DB      'Uart Test !',0DH,0AH,00H

END

```

## 16.8.4 串口 2 使用定时器 2 做波特率发生器

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define FOSC      11059200UL
#define BRT      (65536 - (FOSC / 115200+2) / 4)

```

//加2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```

bit      busy;
char     wptr;
char     rptr;
char     buffer[16];

```

```
void Uart2Isr() interrupt 8
```

```

{
    if (S2CON & 0x02)
    {
        S2CON &= ~0x02;
        busy = 0;
    }
    if (S2CON & 0x01)
    {
        S2CON &= ~0x01;
        buffer[wptr++] = S2BUF;
        wptr &= 0x0f;
    }
}

```

```
void Uart2Init()
```

```

{
    S2CON = 0x10;
    T2L = BRT;
    T2H = BRT >> 8;
}

```

```
AUXR = 0x14;
wptr = 0x00;
rptr = 0x00;
busy = 0;
}

void Uart2Send(char dat)
{
    while (busy);
    busy = 1;
    S2BUF = dat;
}

void Uart2SendStr(char *p)
{
    while (*p)
    {
        Uart2Send(*p++);
    }
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    Uart2Init();
    IE2 = 0x01;
    EA = 1;
    Uart2SendStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            Uart2Send(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

---

## 汇编代码

---

;测试工作频率为11.0592MHz

P\_SW2          DATA          0BAH

|                       |             |                            |  |
|-----------------------|-------------|----------------------------|--|
| <i>AUXR</i>           | <i>DATA</i> | <i>8EH</i>                 |  |
| <i>T2H</i>            | <i>DATA</i> | <i>0D6H</i>                |  |
| <i>T2L</i>            | <i>DATA</i> | <i>0D7H</i>                |  |
| <i>S2CON</i>          | <i>DATA</i> | <i>9AH</i>                 |  |
| <i>S2BUF</i>          | <i>DATA</i> | <i>9BH</i>                 |  |
| <i>IE2</i>            | <i>DATA</i> | <i>0AFH</i>                |  |
|                       |             |                            |  |
| <i>BUSY</i>           | <i>BIT</i>  | <i>20H.0</i>               |  |
| <i>WPTR</i>           | <i>DATA</i> | <i>21H</i>                 |  |
| <i>RPTR</i>           | <i>DATA</i> | <i>22H</i>                 |  |
| <i>BUFFER</i>         | <i>DATA</i> | <i>23H</i>                 | <i>;16 bytes</i>                       |
|                       |             |                            |  |
| <i>P0M1</i>           | <i>DATA</i> | <i>093H</i>                |  |
| <i>P0M0</i>           | <i>DATA</i> | <i>094H</i>                |  |
| <i>P1M1</i>           | <i>DATA</i> | <i>091H</i>                |  |
| <i>P1M0</i>           | <i>DATA</i> | <i>092H</i>                |  |
| <i>P2M1</i>           | <i>DATA</i> | <i>095H</i>                |  |
| <i>P2M0</i>           | <i>DATA</i> | <i>096H</i>                |  |
| <i>P3M1</i>           | <i>DATA</i> | <i>0B1H</i>                |  |
| <i>P3M0</i>           | <i>DATA</i> | <i>0B2H</i>                |  |
| <i>P4M1</i>           | <i>DATA</i> | <i>0B3H</i>                |  |
| <i>P4M0</i>           | <i>DATA</i> | <i>0B4H</i>                |  |
| <i>P5M1</i>           | <i>DATA</i> | <i>0C9H</i>                |  |
| <i>P5M0</i>           | <i>DATA</i> | <i>0CAH</i>                |  |
|                       |             |                            |  |
|                       | <i>ORG</i>  | <i>0000H</i>               |  |
|                       | <i>LJMP</i> | <i>MAIN</i>                |  |
|                       | <i>ORG</i>  | <i>0043H</i>               |  |
|                       | <i>LJMP</i> | <i>UART2_ISR</i>           |  |
|                       |             |                            |  |
|                       | <i>ORG</i>  | <i>0100H</i>               |  |
|                       |             |                            |  |
| <i>UART2_ISR:</i>     | <i>PUSH</i> | <i>ACC</i>                 |  |
|                       | <i>PUSH</i> | <i>PSW</i>                 |  |
|                       | <i>MOV</i>  | <i>PSW,#08H</i>            |  |
|                       |             |                            |  |
|                       | <i>MOV</i>  | <i>A,S2CON</i>             |  |
|                       | <i>JNB</i>  | <i>ACC.1,CHKRI</i>         |  |
|                       | <i>ANL</i>  | <i>S2CON,#NOT 02H</i>      |  |
|                       | <i>CLR</i>  | <i>BUSY</i>                |  |
| <i>CHKRI:</i>         | <i>JNB</i>  | <i>ACC.0,UART2ISR_EXIT</i> |  |
|                       | <i>ANL</i>  | <i>S2CON,#NOT 01H</i>      |  |
|                       | <i>MOV</i>  | <i>A,WPTR</i>              |  |
|                       | <i>ANL</i>  | <i>A,#0FH</i>              |  |
|                       | <i>ADD</i>  | <i>A,#BUFFER</i>           |  |
|                       | <i>MOV</i>  | <i>R0,A</i>                |  |
|                       | <i>MOV</i>  | <i>@R0,S2BUF</i>           |  |
|                       | <i>INC</i>  | <i>WPTR</i>                |  |
| <i>UART2ISR_EXIT:</i> | <i>POP</i>  | <i>PSW</i>                 |  |
|                       | <i>POP</i>  | <i>ACC</i>                 |  |
|                       | <i>RETI</i> |                            |  |
|                       |             |                            |  |
| <i>UART2_INIT:</i>    | <i>MOV</i>  | <i>S2CON,#10H</i>          |  |
|                       | <i>MOV</i>  | <i>T2L,#0E8H</i>           | <i>;65536-11059200/115200/4=0FFE8H</i> |
|                       | <i>MOV</i>  | <i>T2H,#0FFH</i>           |  |

```

MOV    AUXR,#14H
CLR    BUSY
MOV    WPTR,#00H
MOV    RPTR,#00H
RET

UART2_SEND:
JB     BUSY,$
SETB   BUSY
MOV    S2BUF,A
RET

UART2_SENDSTR:
CLR    A
MOVC   A,@A+DPTR
JZ     SEND2END
LCALL  UART2_SEND
INC    DPTR
JMP    UART2_SENDSTR

SEND2END:
RET

MAIN:
MOV    SP,#5FH
ORL    P_SW2,#80H           ;使能访问 XFR

MOV    P0M0,#00H
MOV    P0M1,#00H
MOV    P1M0,#00H
MOV    P1M1,#00H
MOV    P2M0,#00H
MOV    P2M1,#00H
MOV    P3M0,#00H
MOV    P3M1,#00H
MOV    P4M0,#00H
MOV    P4M1,#00H
MOV    P5M0,#00H
MOV    P5M1,#00H

LCALL  UART2_INIT
MOV    IE2,#01H
SETB   EA

MOV    DPTR,#STRING
LCALL  UART2_SENDSTR

LOOP:
MOV    A,RPTR
XRL    A,WPTR
ANL    A,#0FH
JZ     LOOP
MOV    A,RPTR
ANL    A,#0FH
ADD    A,#BUFFER
MOV    R0,A
MOV    A,@R0
LCALL  UART2_SEND
INC    RPTR
JMP    LOOP

```

```
STRING:      DB          'Uart Test !',0DH,0AH,00H

END
```

---

## 16.8.5 串口 3 使用定时器 2 做波特率发生器

### C 语言代码

---

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define  FOSC          11059200UL
#define  BRT           (65536 - (FOSC / 115200+2) / 4)
```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
bit      busy;
char     wptr;
char     rptr;
char     buffer[16];
```

```
void Uart3Isr() interrupt 17
```

```
{
    if (S3CON & 0x02)
    {
        S3CON &= ~0x02;
        busy = 0;
    }
    if (S3CON & 0x01)
    {
        S3CON &= ~0x01;
        buffer[wptr++] = S3BUF;
        wptr &= 0x0f;
    }
}
```

```
void Uart3Init()
```

```
{
    S3CON = 0x10;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x14;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}
```

```
void Uart3Send(char dat)
```

```
{
    while (busy);
    busy = 1;
    S3BUF = dat;
}
```

```
void Uart3SendStr(char *p)
{
    while (*p)
    {
        Uart3Send(*p++);
    }
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    Uart3Init();
    IE2 = 0x08;
    EA = 1;
    Uart3SendStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            Uart3Send(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

汇编代码

;测试工作频率为 11.0592MHz

|        |      |       |           |
|--------|------|-------|-----------|
| P_SW2  | DATA | 0BAH  |           |
| AUXR   | DATA | 8EH   |           |
| T2H    | DATA | 0D6H  |           |
| T2L    | DATA | 0D7H  |           |
| S3CON  | DATA | 0ACH  |           |
| S3BUF  | DATA | 0ADH  |           |
| IE2    | DATA | 0AFH  |           |
| BUSY   | BIT  | 20H.0 |           |
| WPTR   | DATA | 21H   |           |
| RPTR   | DATA | 22H   |           |
| BUFFER | DATA | 23H   | ;16 bytes |

```

P0M1    DATA    093H
P0M0    DATA    094H
P1M1    DATA    091H
P1M0    DATA    092H
P2M1    DATA    095H
P2M0    DATA    096H
P3M1    DATA    0B1H
P3M0    DATA    0B2H
P4M1    DATA    0B3H
P4M0    DATA    0B4H
P5M1    DATA    0C9H
P5M0    DATA    0CAH

```

```

ORG      0000H
LJMP     MAIN
ORG      008BH
LJMP     UART3_ISR

```

```
ORG      0100H
```

**UART3\_ISR:**

```

PUSH     ACC
PUSH     PSW
MOV      PSW,#08H

MOV      A,S3CON
JNB      ACC.1,CHKRI
ANL      S3CON,#NOT 02H
CLR      BUSY

```

**CHKRI:**

```

JNB      ACC.0,UART3ISR_EXIT
ANL      S3CON,#NOT 01H
MOV      A,WPTR
ANL      A,#0FH
ADD      A,#BUFFER
MOV      R0,A
MOV      @R0,S3BUF
INC      WPTR

```

**UART3ISR\_EXIT:**

```

POP      PSW
POP      ACC
RETI

```

**UART3\_INIT:**

```

MOV      S3CON,#10H
MOV      T2L,#0E8H
MOV      T2H,#0FFH
MOV      AUXR,#14H
CLR      BUSY
MOV      WPTR,#00H
MOV      RPTR,#00H
RET

```

;65536-11059200/115200/4=0FFE8H

**UART3\_SEND:**

```

JB       BUSY,$
SETB     BUSY
MOV      S3BUF,A
RET

```



**UART3\_SENDSTR:**

```

CLR      A
MOVC     A,@A+DPTR
JZ       SEND3END
LCALL    UART3_SEND
INC      DPTR
JMP      UART3_SENDSTR

```

**SEND3END:**

```

RET

```

**MAIN:**

```

MOV      SP,#5FH
ORL      P_SW2,#80H      ;使能访问 XFR

MOV      P0M0,#00H
MOV      P0M1,#00H
MOV      P1M0,#00H
MOV      P1M1,#00H
MOV      P2M0,#00H
MOV      P2M1,#00H
MOV      P3M0,#00H
MOV      P3M1,#00H
MOV      P4M0,#00H
MOV      P4M1,#00H
MOV      P5M0,#00H
MOV      P5M1,#00H

LCALL    UART3_INIT
MOV      IE2,#08H
SETB     EA

MOV      DPTR,#STRING
LCALL    UART3_SENDSTR

```

**LOOP:**

```

MOV      A,RPTR
XRL      A,WPTR
ANL      A,#0FH
JZ       LOOP
MOV      A,RPTR
ANL      A,#0FH
ADD      A,#BUFFER
MOV      R0,A
MOV      A,@R0
LCALL    UART3_SEND
INC      RPTR
JMP      LOOP

```

**STRING:**

```

DB       'Uart Test !',0DH,0AH,00H

```

```

END

```

## 16.8.6 串口 3 使用定时器 3 做波特率发生器

C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT      (65536 - (FOSC / 115200+2) / 4)
```

//加2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
bit      busy;
char     wptr;
char     rptr;
char     buffer[16];
```

```
void Uart3Isr() interrupt 17
```

```
{
    if (S3CON & 0x02)
    {
        S3CON &= ~0x02;
        busy = 0;
    }
    if (S3CON & 0x01)
    {
        S3CON &= ~0x01;
        buffer[wptr++] = S3BUF;
        wptr &= 0x0f;
    }
}
```

```
void Uart3Init()
```

```
{
    S3CON = 0x50;
    T3L = BRT;
    T3H = BRT >> 8;
    T4T3M = 0x0a;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}
```

```
void Uart3Send(char dat)
```

```
{
    while (busy);
    busy = 1;
    S3BUF = dat;
}
```

```
void Uart3SendStr(char *p)
```

```
{
    while (*p)
    {
        Uart3Send(*p++);
    }
}
```

```
void main()
```

```
{
    P_SW2 /= 0x80;
    //使能访问 XFR
```

```

P0M0 = 0x00;
P0M1 = 0x00;
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

Uart3Init();
IE2 = 0x08;
EA = 1;
Uart3SendStr("Uart Test !\r\n");

while (1)
{
    if (rptr != wptr)
    {
        Uart3Send(buffer[rptr++]);
        rptr &= 0x0f;
    }
}
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|               |             |              |           |
|---------------|-------------|--------------|-----------|
| <i>P_SW2</i>  | <i>DATA</i> | <i>0BAH</i>  |           |
| <i>T4T3M</i>  | <i>DATA</i> | <i>0D1H</i>  |           |
| <i>T4L</i>    | <i>DATA</i> | <i>0D3H</i>  |           |
| <i>T4H</i>    | <i>DATA</i> | <i>0D2H</i>  |           |
| <i>T3L</i>    | <i>DATA</i> | <i>0D5H</i>  |           |
| <i>T3H</i>    | <i>DATA</i> | <i>0D4H</i>  |           |
| <i>T2L</i>    | <i>DATA</i> | <i>0D7H</i>  |           |
| <i>T2H</i>    | <i>DATA</i> | <i>0D6H</i>  |           |
| <i>S3CON</i>  | <i>DATA</i> | <i>0ACH</i>  |           |
| <i>S3BUF</i>  | <i>DATA</i> | <i>0ADH</i>  |           |
| <i>IE2</i>    | <i>DATA</i> | <i>0AFH</i>  |           |
| <i>BUSY</i>   | <i>BIT</i>  | <i>20H.0</i> |           |
| <i>WPTR</i>   | <i>DATA</i> | <i>21H</i>   |           |
| <i>RPTR</i>   | <i>DATA</i> | <i>22H</i>   |           |
| <i>BUFFER</i> | <i>DATA</i> | <i>23H</i>   | ;16 bytes |
| <i>P0M1</i>   | <i>DATA</i> | <i>093H</i>  |           |
| <i>P0M0</i>   | <i>DATA</i> | <i>094H</i>  |           |
| <i>P1M1</i>   | <i>DATA</i> | <i>091H</i>  |           |
| <i>P1M0</i>   | <i>DATA</i> | <i>092H</i>  |           |
| <i>P2M1</i>   | <i>DATA</i> | <i>095H</i>  |           |
| <i>P2M0</i>   | <i>DATA</i> | <i>096H</i>  |           |
| <i>P3M1</i>   | <i>DATA</i> | <i>0B1H</i>  |           |
| <i>P3M0</i>   | <i>DATA</i> | <i>0B2H</i>  |           |

```

P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          008BH
          LJMP         UART3_ISR

          ORG          0100H

UART3_ISR:
          PUSH         ACC
          PUSH         PSW
          MOV          PSW,#08H

          MOV          A,S3CON
          JNB          ACC.1,CHKRI
          ANL          S3CON,#NOT 02H
          CLR          BUSY

CHKRI:
          JNB          ACC.0,UART3ISR_EXIT
          ANL          S3CON,#NOT 01H
          MOV          A,WPTR
          ANL          A,#0FH
          ADD          A,#BUFFER
          MOV          R0,A
          MOV          @R0,S3BUF
          INC          WPTR

UART3ISR_EXIT:
          POP          PSW
          POP          ACC
          RETI

UART3_INIT:
          MOV          S3CON,#50H
          MOV          T3L,#0E8H
          MOV          T3H,#0FFH
          MOV          T4T3M,#0AH
          CLR          BUSY
          MOV          WPTR,#00H
          MOV          RPTR,#00H
          RET

UART3_SEND:
          JB           BUSY,$
          SETB         BUSY
          MOV          S3BUF,A
          RET

UART3_SENDSTR:
          CLR          A
          MOVC         A,@A+DPTR
          JZ           SEND3END
          LCALL        UART3_SEND
          INC          DPTR
          JMP          UART3_SENDSTR

SEND3END:

```

;65536-11059200/115200/4=0FFE8H

```

    RET

MAIN:
    MOV     SP, #5FH
    ORL     P_SW2, #80H           ;使能访问 XFR

    MOV     P0M0, #00H
    MOV     P0M1, #00H
    MOV     P1M0, #00H
    MOV     P1M1, #00H
    MOV     P2M0, #00H
    MOV     P2M1, #00H
    MOV     P3M0, #00H
    MOV     P3M1, #00H
    MOV     P4M0, #00H
    MOV     P4M1, #00H
    MOV     P5M0, #00H
    MOV     P5M1, #00H

    LCALL   UART3_INIT
    MOV     IE2, #08H
    SETB    EA

    MOV     DPTR, #STRING
    LCALL   UART3_SENDSTR

LOOP:
    MOV     A, RPTR
    XRL     A, WPTR
    ANL     A, #0FH
    JZ      LOOP
    MOV     A, RPTR
    ANL     A, #0FH
    ADD     A, #BUFFER
    MOV     R0, A
    MOV     A, @R0
    LCALL   UART3_SEND
    INC     RPTR
    JMP     LOOP

STRING:    DB      'Uart Test !', 0DH, 0AH, 00H

    END

```

## 16.8.7 串口 4 使用定时器 2 做波特率发生器

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT       (65536 - (FOSC / 115200 + 2) / 4)
```

//加 2 操作是为了让 Keil 编译器

// 自动实现四舍五入运算

```
bit    busy;
char    wptr;
char    rptr;
char    buffer[16];

void Uart4Isr() interrupt 18
{
    if (S4CON & 0x02)
    {
        S4CON &= ~0x02;
        busy = 0;
    }
    if (S4CON & 0x01)
    {
        S4CON &= ~0x01;
        buffer[wptr++] = S4BUF;
        wptr &= 0x0f;
    }
}

void Uart4Init()
{
    S4CON = 0x10;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x14;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart4Send(char dat)
{
    while (busy);
    busy = 1;
    S4BUF = dat;
}

void Uart4SendStr(char *p)
{
    while (*p)
    {
        Uart4Send(*p++);
    }
}

void main()
{
    P_SW2 /= 0x80;

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
```

// 使能访问 XFR

```

P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

Uart4Init();
IE2 = 0x10;
EA = 1;
Uart4SendStr("Uart Test !\r\n");

while (1)
{
    if (rpitr != wpitr)
    {
        Uart4Send(buffer[rpitr++]);
        rpitr &= 0x0f;
    }
}
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|               |             |                  |           |
|---------------|-------------|------------------|-----------|
| <i>P_SW2</i>  | <i>DATA</i> | <i>0BAH</i>      |           |
| <i>AUXR</i>   | <i>DATA</i> | <i>8EH</i>       |           |
| <i>T2H</i>    | <i>DATA</i> | <i>0D6H</i>      |           |
| <i>T2L</i>    | <i>DATA</i> | <i>0D7H</i>      |           |
| <i>S4CON</i>  | <i>DATA</i> | <i>84H</i>       |           |
| <i>S4BUF</i>  | <i>DATA</i> | <i>85H</i>       |           |
| <i>IE2</i>    | <i>DATA</i> | <i>0AFH</i>      |           |
| <i>BUSY</i>   | <i>BIT</i>  | <i>20H.0</i>     |           |
| <i>WPTR</i>   | <i>DATA</i> | <i>21H</i>       |           |
| <i>RPTR</i>   | <i>DATA</i> | <i>22H</i>       |           |
| <i>BUFFER</i> | <i>DATA</i> | <i>23H</i>       | ;16 bytes |
| <i>P0M1</i>   | <i>DATA</i> | <i>093H</i>      |           |
| <i>P0M0</i>   | <i>DATA</i> | <i>094H</i>      |           |
| <i>P1M1</i>   | <i>DATA</i> | <i>091H</i>      |           |
| <i>P1M0</i>   | <i>DATA</i> | <i>092H</i>      |           |
| <i>P2M1</i>   | <i>DATA</i> | <i>095H</i>      |           |
| <i>P2M0</i>   | <i>DATA</i> | <i>096H</i>      |           |
| <i>P3M1</i>   | <i>DATA</i> | <i>0B1H</i>      |           |
| <i>P3M0</i>   | <i>DATA</i> | <i>0B2H</i>      |           |
| <i>P4M1</i>   | <i>DATA</i> | <i>0B3H</i>      |           |
| <i>P4M0</i>   | <i>DATA</i> | <i>0B4H</i>      |           |
| <i>P5M1</i>   | <i>DATA</i> | <i>0C9H</i>      |           |
| <i>P5M0</i>   | <i>DATA</i> | <i>0CAH</i>      |           |
|               | <i>ORG</i>  | <i>0000H</i>     |           |
|               | <i>LJMP</i> | <i>MAIN</i>      |           |
|               | <i>ORG</i>  | <i>0093H</i>     |           |
|               | <i>LJMP</i> | <i>UART4_ISR</i> |           |
|               | <i>ORG</i>  | <i>0100H</i>     |           |

**UART4\_ISR:**

```

PUSH    ACC
PUSH    PSW
MOV      PSW,#08H

MOV      A,S4CON
JNB      ACC.1,CHKRI
ANL      S4CON,#NOT 02H
CLR      BUSY

```

**CHKRI:**

```

JNB      ACC.0,UART4ISR_EXIT
ANL      S4CON,#NOT 01H
MOV      A,WPTR
ANL      A,#0FH
ADD      A,#BUFFER
MOV      R0,A
MOV      @R0,S4BUF
INC      WPTR

```

**UART4ISR\_EXIT:**

```

POP      PSW
POP      ACC
RETI

```

**UART4\_INIT:**

```

MOV      S4CON,#10H
MOV      T2L,#0E8H
MOV      T2H,#0FFH
MOV      AUXR,#14H
CLR      BUSY
MOV      WPTR,#00H
MOV      RPTR,#00H
RET

```

;65536-11059200/115200/4=0FFE8H

**UART4\_SEND:**

```

JB      BUSY,$
SETB    BUSY
MOV      S4BUF,A
RET

```

**UART4\_SENDSTR:**

```

CLR      A
MOVC     A,@A+DPTR
JZ       SEND4END
LCALL    UART4_SEND
INC      DPTR
JMP      UART4_SENDSTR

```

**SEND4END:**

```

RET

```

**MAIN:**

```

MOV      SP,#5FH
ORL      P_SW2,#80H

```

;使能访问 XFR

```

MOV      P0M0,#00H
MOV      P0M1,#00H
MOV      P1M0,#00H
MOV      P1M1,#00H
MOV      P2M0,#00H
MOV      P2M1,#00H

```



```

MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

LCALL    UART4_INIT
MOV      IE2, #10H
SETB     EA

MOV      DPTR, #STRING
LCALL    UART4_SENDSTR

LOOP:
MOV      A, RPTR
XRL      A, WPTR
ANL      A, #0FH
JZ       LOOP
MOV      A, RPTR
ANL      A, #0FH
ADD      A, #BUFFER
MOV      R0, A
MOV      A, @R0
LCALL    UART4_SEND
INC      RPTR
JMP      LOOP

STRING:   DB      'Uart Test !', 0DH, 0AH, 00H

END

```

## 16.8.8 串口 4 使用定时器 4 做波特率发生器

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define FOSC      11059200UL
#define BRT       (65536 - (FOSC / 115200+2) / 4)

```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```

bit      busy;
char      wptr;
char      rptr;
char      buffer[16];

```

```

void Uart4Isr() interrupt 18
{
    if (S4CON & 0x02)
    {
        S4CON &= ~0x02;
    }
}

```

```

        busy = 0;
    }
    if (S4CON & 0x01)
    {
        S4CON &= ~0x01;
        buffer[wptr++] = S4BUF;
        wptr &= 0x0f;
    }
}

void Uart4Init()
{
    S4CON = 0x50;
    T4L = BRT;
    T4H = BRT >> 8;
    T4T3M = 0xa0;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart4Send(char dat)
{
    while (busy);
    busy = 1;
    S4BUF = dat;
}

void Uart4SendStr(char *p)
{
    while (*p)
    {
        Uart4Send(*p++);
    }
}

void main()
{
    P_SW2 /= 0x80;

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    Uart4Init();
    IE2 = 0x10;
    EA = 1;
    Uart4SendStr("Uart Test !\r\n");

    while (1)

```

//使能访问XFR

```
{
    if (rptr != wptr)
    {
        Uart4Send(buffer[rptr++]);
        rptr &= 0x0f;
    }
}
```

汇编代码

;测试工作频率为 11.0592MHz

|                   |             |                       |
|-------------------|-------------|-----------------------|
| <i>P_SW2</i>      | <i>DATA</i> | <i>0BAH</i>           |
| <i>T4T3M</i>      | <i>DATA</i> | <i>0D1H</i>           |
| <i>T4L</i>        | <i>DATA</i> | <i>0D3H</i>           |
| <i>T4H</i>        | <i>DATA</i> | <i>0D2H</i>           |
| <i>T3L</i>        | <i>DATA</i> | <i>0D5H</i>           |
| <i>T3H</i>        | <i>DATA</i> | <i>0D4H</i>           |
| <i>T2L</i>        | <i>DATA</i> | <i>0D7H</i>           |
| <i>T2H</i>        | <i>DATA</i> | <i>0D6H</i>           |
| <i>S4CON</i>      | <i>DATA</i> | <i>84H</i>            |
| <i>S4BUF</i>      | <i>DATA</i> | <i>85H</i>            |
| <i>IE2</i>        | <i>DATA</i> | <i>0AFH</i>           |
| <i>BUSY</i>       | <i>BIT</i>  | <i>20H.0</i>          |
| <i>WPTR</i>       | <i>DATA</i> | <i>21H</i>            |
| <i>RPTR</i>       | <i>DATA</i> | <i>22H</i>            |
| <i>BUFFER</i>     | <i>DATA</i> | <i>23H</i> ;16 bytes  |
| <i>P0M1</i>       | <i>DATA</i> | <i>093H</i>           |
| <i>P0M0</i>       | <i>DATA</i> | <i>094H</i>           |
| <i>P1M1</i>       | <i>DATA</i> | <i>091H</i>           |
| <i>P1M0</i>       | <i>DATA</i> | <i>092H</i>           |
| <i>P2M1</i>       | <i>DATA</i> | <i>095H</i>           |
| <i>P2M0</i>       | <i>DATA</i> | <i>096H</i>           |
| <i>P3M1</i>       | <i>DATA</i> | <i>0B1H</i>           |
| <i>P3M0</i>       | <i>DATA</i> | <i>0B2H</i>           |
| <i>P4M1</i>       | <i>DATA</i> | <i>0B3H</i>           |
| <i>P4M0</i>       | <i>DATA</i> | <i>0B4H</i>           |
| <i>P5M1</i>       | <i>DATA</i> | <i>0C9H</i>           |
| <i>P5M0</i>       | <i>DATA</i> | <i>0CAH</i>           |
|                   | <i>ORG</i>  | <i>0000H</i>          |
|                   | <i>LJMP</i> | <i>MAIN</i>           |
|                   | <i>ORG</i>  | <i>0093H</i>          |
|                   | <i>LJMP</i> | <i>UART4_ISR</i>      |
|                   | <i>ORG</i>  | <i>0100H</i>          |
| <i>UART4_ISR:</i> |             |                       |
|                   | <i>PUSH</i> | <i>ACC</i>            |
|                   | <i>PUSH</i> | <i>PSW</i>            |
|                   | <i>MOV</i>  | <i>PSW,#08H</i>       |
|                   | <i>MOV</i>  | <i>A,S4CON</i>        |
|                   | <i>JNB</i>  | <i>ACC.1,CHKRI</i>    |
|                   | <i>ANL</i>  | <i>S4CON,#NOT 02H</i> |

```

        CLR        BUSY

CHKRI:
        JNB        ACC,0,UART4ISR_EXIT
        ANL        S4CON,#NOT 01H
        MOV        A,WPTR
        ANL        A,#0FH
        ADD        A,#BUFFER
        MOV        R0,A
        MOV        @R0,S4BUF
        INC        WPTR

UART4ISR_EXIT:
        POP        PSW
        POP        ACC
        RETI

UART4_INIT:
        MOV        S4CON,#50H
        MOV        T4L,#0E8H
        MOV        T4H,#0FFH
        MOV        T4T3M,#0A0H
        CLR        BUSY
        MOV        WPTR,#00H
        MOV        RPTR,#00H
        RET

UART4_SEND:
        JB         BUSY,$
        SETB        BUSY
        MOV        S4BUF,A
        RET

UART4_SENDSTR:
        CLR        A
        MOVC        A,@A+DPTR
        JZ         SEND4END
        LCALL        UART4_SEND
        INC        DPTR
        JMP         UART4_SENDSTR

SEND4END:
        RET

MAIN:
        MOV        SP,#5FH
        ORL        P_SW2,#80H
        ;使能访问 XFR

        MOV        P0M0,#00H
        MOV        P0M1,#00H
        MOV        P1M0,#00H
        MOV        P1M1,#00H
        MOV        P2M0,#00H
        MOV        P2M1,#00H
        MOV        P3M0,#00H
        MOV        P3M1,#00H
        MOV        P4M0,#00H
        MOV        P4M1,#00H
        MOV        P5M0,#00H
        MOV        P5M1,#00H

        LCALL        UART4_INIT

```

*MOV*        *IE2,#10H*  
*SETB*       *EA*

*MOV*        *DPTR,#STRING*  
*LCALL*      *UART4\_SENDSTR*

*LOOP:*

*MOV*        *A,RPTR*  
*XRL*        *A,WPTR*  
*ANL*        *A,#0FH*  
*JZ*         *LOOP*  
*MOV*        *A,RPTR*  
*ANL*        *A,#0FH*  
*ADD*        *A,#BUFFER*  
*MOV*        *R0,A*  
*MOV*        *A,@R0*  
*LCALL*      *UART4\_SEND*  
*INC*        *RPTR*  
*JMP*        *LOOP*

*STRING:*    *DB*        *'Uart Test !',0DH,0AH,00H*

*END*

## 16.8.9 串口多机通讯

现参考 STC15 系列数据手册，后续补充

STC MCU

## 16.8.10 串口中断收发—MODBUS 协议

### C 语言代码

---

//测试工作频率为 11.0592MHz

```
#include <reg52.h>
#define MAIN_Fosc      11059200L    //定义主时钟
```

```
/******          功能说明          *****/
```

请先别修改程序, 直接下载"08- 串口 1 中断收发-C 语言-MODBUS 协议"里的"UART1.hex"测试, 主频选择 11.0592MHZ. 测试正常后再修改移植.

串口 1 按 MODBUS-RTU 协议通信. 本例为从机程序, 主机一般是电脑端.

本例程只支持多寄存器读和多寄存器写, 寄存器长度为 64 个, 别的命令用户可以根据需要按 MODBUS-RTU 协议自行添加.

本例子数据使用大端模式(与 C51 一致), CRC16 使用小端模式(与 PC 一致).

默认参数:

串口 1 设置均为 1 位起始位, 8 位数据位, 1 位停止位, 无校验.

串口 1(P3.0 P3.1): 9600bps.

定时器 0 用于超时计时. 串口每收到一个字节都会重置超时计数, 当串口空闲超过 35bit 时间时(9600bps 对应 3.6ms)则接收完成.

用户修改波特率时注意要修改这个超时时间.

本例程只是一个应用例子, 科普 MODBUS-RTU 协议并不在本例子职责范围, 用户可以上网搜索相关协议文本参考.

本例定义了 64 个寄存器, 访问地址为 0x1000~0x103f.

命令例子:

写入 4 个寄存器(8 个字节):

10 10 1000 0004 08 1234 5678 90AB CDEF 4930

返回:

10 10 10 00 00 04 4B C6

读出 4 个寄存器:

10 03 1000 0004 4388

返回:

10 03 08 12 34 56 78 90 AB CD EF 3D D5

命令错误返回信息(自定义):

0x90: 功能码错误 收到了不支持的功能码.

0x91: 命令长度错误.

0x92: 写入或读出寄存器个数或字节数错误.

0x93: 寄存器地址错误.

注意: 收到广播地址 0x00 时要处理信息, 但不返回应答.

\*\*\*\*\*/

```
typedef unsigned char  u8;
```

```
typedef unsigned int    u16;
```

```
typedef unsigned long   u32;
```

```

sfr PIM1    = 0x91;
sfr PIM0    = 0x92;
sfr P3M1    = 0xB1;
sfr P3M0    = 0xB2;
sfr P4M1    = 0xB3;
sfr P4M0    = 0xB4;
sfr P_SW1   = 0xA2;
sfr AUXR    = 0x8E;
sfr IE2     = 0xAF;

```

```

/***** 本地常量声明 *****/
#define RX1_Length 128 /* 接收缓冲长度 */
#define TX1_Length 128 /* 发送缓冲长度 */

```

```

/***** 本地变量声明 *****/
u8 xdataRX1_Buffer[RX1_Length]; /*接收缓冲
u8 xdataTX1_Buffer[TX1_Length]; /*发送缓冲

```

```

u8 RX1_cnt; /*接收字节计数
u8 TX1_cnt; /*发送字节计数
u8 TX1_number; /*要发送的字节数
u8 RX1_TimeOut; /*接收超时计时器

bit B_RX1_OK; /* 接收数据标志
bit B_TX1_Busy; /* 发送忙标志

```

```

/***** 本地函数声明 *****/

```

```

void UART1_config(u32 brt, u8 timer, u8 io); /* brt: 通信波特率, timer=2: 波特率使用定时器 2, 其它值: 使用
Timer1 做波特率 io=0: 串口 1 切换到 P3.0 P3.1, =1: 切换到 P3.6 P3.7, =2: 切换到 P1.6 P1.7, =3: 切换到 P4.3
P4.4.

```

```

u8 Timer0_Config(u8 t, u32 reload); /*t=0: reload 值是主时钟周期数, t=1: reload 值是时间(单位 us), 返回 0 正
确, 返回 1 装载值过大错误

```

```

u16 MODBUS_CRC16(u8 *p, u8 n);
u8 MODBUS_RTU(void);

```

```

#define SL_ADDR 0x10 /* 本从机站号地址 */
#define REG_ADDRESS 0x1000 /* 寄存器首地址 */
#define REG_LENGTH 64 /* 寄存器长度 */
u16 xdata modbus_reg[REG_LENGTH]; /* 寄存器地址 */

```

```

//=====
// 函数: void main(void)

```



```

// 描述: 主函数
// 参数: none.
// 返回: none.
// 版本: VER1.0
// 日期: 2018-4-2
// 备注:
//=====

void main(void)
{
    u8  i;
    u16 crc;

    Timer0_Config(0, MAIN_Fosc / 10000);    //t=0: reload 值是主时钟周期数, (中断频率, 20000 次/秒)
    UART1_config(9600UL, 1, 0); //brt: 通信波特率, timer=2: 波特率使用定时器 2, 其它值: 使用 Timer1 做波特率
    io=0: 串口 1 切换到 P3.0 P3.1, =1: 切换到 P3.6 P3.7, =2: 切换到 P1.6 P1.7, =3: 切换到 P4.3 P4.4.

    EA = 1;

    while (1)
    {
        if(B_RX1_OK && !B_TX1_Busy) //收到数据, 进行MODBUS-RTU 协议解析
        {
            if(MODBUS_CRC16(RX1_Buffer, RX1_cnt) == 0) //首先判断 CRC16 是否正确, 不正确则忽略,
            不处理也不返回信息
            {
                if((RX1_Buffer[0] == 0x00) || (RX1_Buffer[0] == SL_ADDR)) //然后判断站号地址是否正确, 或
                者是否广播地址(不返回信息)
                {
                    if(RX1_cnt > 2) RX1_cnt -= 2; //去掉CRC16 校验字节
                    i = MODBUS_RTU(); //MODBUS-RTU 协议解析
                    if(i != 0) //错误处理
                    {
                        TX1_Buffer[0] = SL_ADDR; //站号地址
                        TX1_Buffer[1] = i; //错误代码
                        crc = MODBUS_CRC16(TX1_Buffer, 2);
                        TX1_Buffer[2] = (u8)(crc >> 8); //CRC 是小端模式
                        TX1_Buffer[3] = (u8)crc;
                        B_TX1_Busy = 1; //标志发送忙
                        TX1_cnt = 0; //发送字节计数
                        TX1_number = 4; //要发送的字节数
                        TI = 1; //启动发送
                    }
                }
            }
            RX1_cnt = 0;
            B_RX1_OK = 0;
        }
    }
}

```

```

    }
}

}

/***** MODBUS_CRC (shift) *****/
计算CRC, 调用方式MODBUS_CRC16(&CRC,8);  &CRC 为首地址, 8 为字节数
CRC-16 for MODBUS
CRC16=X16+X15+X2+1
TEST: ---> ABCDEFGHIJ  CRC16=0x0BEE    1627T
*/

//=====
// 函数: u16  MODBUS_CRC16(u8 *p,  u8 n)
// 描述: 计算CRC16 函数
// 参数: *p: 要计算的数据指针.
//        n: 要计算的字节数
// 返回: CRC16 值
// 版本: V1.0, 2022-3-18 梁工
//=====
u16 MODBUS_CRC16(u8 *p,  u8 n)
{
    u8  i;
    u16 crc16;

    crc16 = 0xffff; //预置16位CRC寄存器为0xffff (即全为1)
    do
    {
        crc16 ^= (u16)*p; //把8位数据与16位CRC寄存器的低位相异或, 把结果放于CRC寄存器
        for(i=0; i<8; i++) //8位数据
        {
            if(crc16 & 1)  crc16 = (crc16 >> 1) ^ 0xA001; //如果最低位为0, 把CRC寄存器的内容右移一位
            //再异或多项式0xA001
        }
        else crc16 >>= 1; //如果最低位为0, 把CRC寄存器的内容右移一位
        //朝低位, 用0填补最高位
    }
    p++;
}while(--n != 0);
return  (crc16);
}

/***** modbus 协议 *****/
/*****
写多寄存器
数据:      地址      功能码      寄存地址  寄存器个数  写入字节数  写入数据  CRC16
偏移:      0        1      2 3      4 5      6      7~      最后2字节

```

字节:    1 byte    1 byte    2 byte 2 byte    1byte    2\*n byte    2 byte  
         addr    0x10    xxxx xxxx    xx    xx.....xx    xxxx

返回

数据:    地址    功能码    寄存地址 寄存器个数    CRC16  
偏移:    0    1    2 3    4 5    6 7  
字节:    1 byte    1 byte    2 byte 2 byte    2 byte  
         addr    0x10    xxxx xxxx    xxxx

读多寄存器

数据: 站号(地址)    功能码    寄存地址 寄存器个数    CRC16  
偏移:    0    1    2 3    4 5    6 7  
字节:    1 byte    1 byte    2 byte 2 byte    2 byte  
         addr    0x03    xxxx xxxx    xxxx

返回

数据: 站号(地址)    功能码    读出字节数    读出数据    CRC16  
偏移:    0    1    2    3~    最后2 字节  
字节:    1 byte    1 byte    1byte    2\*n byte 2 byte  
         addr    0x03    xx    xx.....xx    xxxx

返回错误代码

数据: 站号(地址)    错误码    CRC16  
偏移:    0    1    最后2 字节  
字节:    1 byte    1 byte    2 byte  
         addr    0x03    xxxx

\*\*\*\*\*/

```
u8 MODBUS_RTU(void)
{
    u8 i,j,k;
    u16 reg_addr; //寄存器地址
    u8 reg_len; //写入寄存器个数
    u16 crc;

    if(RX1_Buffer[1]== 0x10)//写多寄存器
    {
        if(RX1_cnt < 9) return 0x91; //命令长度错误
        if((RX1_Buffer[4] != 0) || ((RX1_Buffer[5] *2) != RX1_Buffer[6])) return 0x92; //写入寄存器个数与字节数错误
        if((RX1_Buffer[5]==0) || (RX1_Buffer[5] > REG_LENGTH)) return 0x92; //写入寄存器个数错误

        reg_addr = ((u16)RX1_Buffer[2] << 8) + RX1_Buffer[3]; //寄存器地址
        reg_len = RX1_Buffer[5]; //写入寄存器个数
        if((reg_addr+(u16)RX1_Buffer[5]) > (REG_ADDRESS+REG_LENGTH)) return 0x93; //寄存器地址错误
        if(reg_addr< REG_ADDRESS) return 0x93; //寄存器地址错误
    }
```

```

    if((reg_len*2+7) != RX1_cnt) return 0x91;    //命令长度错误

    j = reg_addr - REG_ADDRESS;    //寄存器数据下标
    for(k=7, i=0; i<reg_len; i++,j++)
    {
        modbus_reg[j] = ((u16)RX1_Buffer[k] << 8) + RX1_Buffer[k+1];    //写入数据, 大端模式
        k += 2;
    }

    if(RX1_Buffer[0] != 0)    //非广播地址则应答
    {
        for(i=0; i<6; i++) TX1_Buffer[i] = RX1_Buffer[i];    //要返回的应答
        crc = MODBUS_CRC16(TX1_Buffer, 6);
        TX1_Buffer[6] = (u8)(crc>>8);    //CRC 是小端模式
        TX1_Buffer[7] = (u8)crc;
        B_TX1_Busy = 1;    //标志发送忙
        TX1_cnt = 0;    //发送字节计数
        TX1_number = 8;    //要发送的字节数
        TI = 1;    //启动发送
    }
}
else if(RX1_Buffer[1] == 0x03)    //读多寄存器
{
    if(RX1_Buffer[0] != 0)    //非广播地址则应答
    {
        if(RX1_cnt != 6) return 0x91;    //命令长度错误
        if(RX1_Buffer[4] != 0) return 0x92;    //读出寄存器个数错误
        if((RX1_Buffer[5]==0) || (RX1_Buffer[5] > REG_LENGTH)) return 0x92;    //读出寄存器个数错误

        reg_addr = ((u16)RX1_Buffer[2] << 8) + RX1_Buffer[3];    //寄存器地址
        reg_len = RX1_Buffer[5];    //读出寄存器个数
        if((reg_addr+(u16)RX1_Buffer[5]) > (REG_ADDRESS+REG_LENGTH)) return 0x93;    //寄存器地址
        错误

        if(reg_addr< REG_ADDRESS) return 0x93;    //寄存器地址错误

        j = reg_addr - REG_ADDRESS;    //寄存器数据下标
        TX1_Buffer[0] = SL_ADDR;    //站号地址
        TX1_Buffer[1] = 0x03;    //读功能码
        TX1_Buffer[2] = reg_len*2;    //返回字节数

        for(k=3, i=0; i<reg_len; i++,j++)
        {
            TX1_Buffer[k++] = (u8)(modbus_reg[j] >> 8);    //数据为大端模式
            TX1_Buffer[k++] = (u8)modbus_reg[j];
        }
        crc = MODBUS_CRC16(TX1_Buffer, k);
    }
}

```

```

        TX1_Buffer[k++] = (u8)(crc>>8); //CRC 是小端模式
        TX1_Buffer[k++] = (u8)crc;
        B_TX1_Busy = 1;           //标志发送忙
        TX1_cnt = 0;             //发送字节计数
        TX1_number = k;          //要发送的字节数
        TI = 1;                  //启动发送
    }
}
else return 0x90; //功能码错误

return 0; //解析正确
}

//=====
// 函数:u8 Timer0_Config(u8 t, u32 reload)
// 描述: timer0 初始化函数.
// 参数:    t: 重装值类型, 0 表示重装的是系统时钟数, 其余值表示重装的是时间(us).
//          reload: 重装值.
// 返回: 0: 初始化正确, 1: 重装值过大, 初始化错误
// 版本: V1.0, 2018-3-5
//=====
u8 Timer0_Config(u8 t, u32 reload) //t=0: reload 值是主时钟周期数, t=1: reload 值是时间(单位 us)
{
    TR0 = 0; //停止计数

    if(t != 0) reload = (u32)(((float)MAIN_Fosc * (float)reload)/1000000UL); //重装的是时间(us), 计算所需要的系统时钟数.
    if(reload >= (65536UL * 12)) return 1; //值过大, 返回错误
    if(reload < 65536UL) AUXR |= 0x80; //1T mode
    else
    {
        AUXR &= ~0x80; //12T mode
        reload = reload / 12;
    }
    reload = 65536UL - reload;
    TH0 = (u8)(reload >> 8);
    TL0 = (u8)(reload);

    ET0 = 1; //允许中断
    TMOD &= 0xf0;
    TMOD |= 0; //工作模式 0: 16 位自动重装, 1: 16 位定时/计数, 2: 8 位自动重装, 3: 16 位自动重装 不可屏蔽中断
    TR0 = 1; //开始运行
    return 0;
}

```

```
//=====
// 函数: void timer0_ISR (void)  interrupt TIMER0_VECTOR
// 描述: timer0 中断函数
// 参数: none.
// 返回: none.
// 版本: V1.0, 2016-5-12
//=====

void timer0_ISR (void) interrupt 1
{
    if(RX1_TimeOut != 0)
    {
        if(--RX1_TimeOut == 0) //超时
        {
            if(RX1_cnt != 0) //接收有数据
            {
                B_RX1_OK = 1; //标志已收到数据块
            }
        }
    }
}

//=====
// 函数: SetTimer2Baudrate(u16 dat)
// 描述: 设置Timer2 做波特率发生器。
// 参数: dat: Timer2 的重装值。
// 返回: none.
// 版本: VER1.0
// 日期: 2018-4-2
// 备注:
//=====

void SetTimer2Baudrate(u16 dat) // 选择波特率, 2: 使用Timer2 做波特率, 其它值: 使用Timer1 做波特率
{
    AUXR &= ~(1<<4); //Timer stop
    AUXR &= ~(1<<3); //Timer2 set As Timer
    AUXR |= (1<<2); //Timer2 set as 1T mode
    TH2 = (u8)(dat >> 8);
    TL2 = (u8)dat;
    IE2 &= ~(1<<2); //禁止中断
    AUXR |= (1<<4); //Timer run enable
}

//=====
// 函数: void UART1_config(u32 brt, u8 timer, u8 io)
```

```

// 描述: UART1 初始化函数。
// 参数: brt: 通信波特率
//      timer: 波特率使用的定时器,timer=2: 波特率使用定时器 2, 其它值: 使用 Timer1 做波特率
//      io: 串口 1 切换到的 IO, io=0: 串口 1 切换到 P3.0 P3.1, =1: 切换到 P3.6 P3.7, =2: 切换到 P1.6 P1.7, =3: 切换到 P4.3 P4.4.
// 返回: none.
// 版本: VER1.0
// 日期: 2018-4-2
// 备注:
//=====
void UART1_config(u32 brt, u8 timer, u8 io) // brt:通信波特率, timer=2: 波特率使用定时器 2, 其它值: 使用
Timer1 做波特率 io=0: 串口 1 切换到 P3.0 P3.1, =1: 切换到 P3.6 P3.7, =2: 切换到 P1.6 P1.7, =3: 切换到 P4.3
P4.4.
{
    brt = 65536UL - (MAIN_Fosc / 4) / brt;
    if(timer == 2) //波特率使用定时器 2
    {
        AUXR |= 0x01; //S1 BRT Use Timer2;
        SetTimer2Baudrate((u16)brt);
    }

    else //波特率使用定时器 1
    {
        TRI = 0;
        AUXR &= ~0x01; //S1 BRT Use Timer1;
        AUXR |= (1<<6); //Timer1 set as IT mode
        TMOD &= ~(1<<6); //Timer1 set As Timer
        TMOD &= ~0x30; //Timer1_16bitAutoReload;
        TH1 = (u8)(brt >> 8);
        TL1 = (u8)brt;
        ET1 = 0; // 禁止 Timer1 中断
        TRI = 1; // 运行 Timer1
    }
    P_SW1 &= ~0xc0; //默认切换到 P3.0 P3.1
    if(io == 1)
    {
        P_SW1 |= 0x40; //切换到 P3.6 P3.7
        P3M1 &= ~0xc0;
        P3M0 &= ~0xc0;
    }
    else if(io == 2)
    {
        P_SW1 |= 0x80; //切换到 P1.6 P1.7
        P1M1 &= ~0xc0;
        P1M0 &= ~0xc0;
    }
}

```

```

else if(io == 3)
{
    P_SW1 |= 0xc0; //切换到 P4.3 P4.4
    P4M1 &= ~0x18;
    P4M0 &= ~0x18;
}
else
{
    P3M1 &= ~0x03;
    P3M0 &= ~0x03;
}

SCON = (SCON & 0x3f) / (1<<6); // 8 位数据, 1 位起始位, 1 位停止位, 无校验
// PS = 1; //高优先级中断
ES = 1; //允许中断
REN = 1; //允许接收
}

//=====
// 函数: void UART1_ISR (void) interrupt UART1_VECTOR
// 描述: 串口1 中断函数
// 参数: none.
// 返回: none.
// 版本: VER1.0
// 日期: 2018-4-2
// 备注:
//=====
void UART1_ISR (void) interrupt 4
{
    if(RI)
    {
        RI = 0;
        if(!B_RX1_OK) //接收缓冲空闲
        {
            if(RX1_cnt >= RX1_Length) RX1_cnt = 0;
            RX1_Buffer[RX1_cnt++] = SBUF;
            RX1_TimeOut = 36; //接收超时计时器, 35 个位时间
        }
    }

    if(TI)
    {
        TI = 0;
        if(TX1_number != 0) //有数据要发
        {

```



```
        SBUF = TX1_Buffer[TX1_cnt++];  
        TX1_number--;  
    }  
    else B_TX1_Busy = 0;  
}  
}
```

STC MCU

## 16.8.11 串口转 LIN 总线

### C 语言代码

---



---

```
//测试工作频率为22.1184MHz
```

```
/****** 功能说明 *****
```

本例程基于 STC8H8K64U 为主控芯片的实验箱 8 进行编写测试, STC8G、STC8H 系列芯片可通用参考。  
通过 UART 接口连接 LIN 收发器实现 LIN 总线信号收发测试例程。

UART1 通过串口工具连接电脑。

UART2 外接 LIN 收发器(TJA1020/1), 连接 LIN 总线。

将电脑串口发送的数据转发到 LIN 总线; 从 LIN 总线接收到的数据转发到电脑串口。

默认传输速率: 9600 波特率, 发送 LIN 数据前切换波特率, 发送 13 个显性间隔信号。

下载时, 选择时钟 22.1184MHz (用户可自行修改频率)。

```
*****/
```

```
#include "stc8h.h"
```

```
#include "intrins.h"
```

```
#define MAIN_Fosc 22118400L
```

```
typedef unsigned char u8;
```

```
typedef unsigned int u16;
```

```
typedef unsigned long u32;
```

```
sbit SLP_N = P2^4; //0: Sleep
```

```
/****** 用户定义宏 *****/
```

```
#define Baudrate1 (65536UL - (MAIN_Fosc / 4) / 9600UL)
```

```
#define Baudrate2 (65536UL - (MAIN_Fosc / 4) / 9600UL)
```

```
#define Baudrate_Break (65536UL - (MAIN_Fosc / 4) / 6647UL) //发送显性间隔信号波特率
```

```
#define UART1_BUF_LENGTH 32
```

```
#define UART2_BUF_LENGTH 32
```

```
#define LIN_ID 0x31
```

```
u8 TX1_Cnt; //发送计数
```

```
u8 RX1_Cnt; //接收计数
```

```
u8 TX2_Cnt; //发送计数
```

```
u8 RX2_Cnt; //接收计数
```

```
bit B_TX1_Busy; //发送忙标志
```

```
bit B_TX2_Busy; //发送忙标志
```

```
u8 RX1_TimeOut;
```

```
u8 RX2_TimeOut;
```

```
u8 xdata RX1_Buffer[UART1_BUF_LENGTH]; //接收缓冲
```

```
u8 xdata RX2_Buffer[UART2_BUF_LENGTH]; //接收缓冲
```

```
void UART1_config(u8 brt);
```

```
void UART2_config(u8 brt);
```

```
void PrintString1(u8 *puts);
```

```
void delay_ms(u8 ms);
```

```
void UART1_TxByte(u8 dat);
```

```
void UART2_TxByte(u8 dat);
```

```
void Lin_Send(u8 *puts);
```

```
void SetTimer2Baudrate(u16 dat);
```

```
//=====
// 函数: void main(void)
// 描述: 主函数。
// 参数: none.
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====
void main(void)
{
    u8 i;

    P0M1 = 0; P0M0 = 0;           // 设置为准双向口
    P1M1 = 0; P1M0 = 0;           // 设置为准双向口
    P2M1 = 0; P2M0 = 0;           // 设置为准双向口
    P3M1 = 0; P3M0 = 0;           // 设置为准双向口
    P4M1 = 0; P4M0 = 0;           // 设置为准双向口
    P5M1 = 0; P5M0 = 0;           // 设置为准双向口
    P6M1 = 0; P6M0 = 0;           // 设置为准双向口
    P7M1 = 0; P7M0 = 0;           // 设置为准双向口

    UART1_config(1);
    UART2_config(2);
    EA = 1;                       // 允许全局中断
    SLP_N = 1;

    PrintString1("STC8H8K64U UART1 Test Programme!\r\n"); // UART1 发送一个字符串

    while (1)
    {
        delay_ms(1);
        if(RX1_TimeOut > 0)
        {
            if(--RX1_TimeOut == 0) // 超时,则串口接收结束
            {
                if(RX1_Cnt > 0)
                {
                    Lin_Send(RX1_Buffer); // 将 UART1 收到的数据发送到 LIN 总线上
                }
                RX1_Cnt = 0;
            }
        }

        if(RX2_TimeOut > 0)
        {
            if(--RX2_TimeOut == 0) // 超时,则串口接收结束
            {
                if(RX2_Cnt > 0)
                {
                    for (i=0; i < RX2_Cnt; i++) // 遇到停止符 0 结束
                    {
                        UART1_TxByte(RX2_Buffer[i]); // 从 LIN 总线收到的数据发送到 UART1
                    }
                }
                RX2_Cnt = 0;
            }
        }
    }
}
```

```

    }
    }
}

//=====================================================
// 函数: void delay_ms(unsigned char ms)
// 描述: 延时函数。
// 参数: ms, 要延时的 ms 数, 这里只支持 1~255ms. 自动适应主时钟.
// 返回: none.
// 版本: VER1.0
// 日期: 2013-4-1
// 备注:
//=====================================================
void delay_ms(u8 ms)
{
    u16 i;
    do{
        i = MAIN_Fosc / 10000;
        while(--i);                //10T per loop
    }while(--ms);
}

//=====================================================
// 函数: u8          Lin_CheckPID(u8 id)
// 描述: ID 码加上校验符, 转成 PID 码。
// 参数: ID 码。
// 返回: PID 码
// 版本: VER1.0
// 日期: 2020-12-2
// 备注:
//=====================================================
u8 Lin_CheckPID(u8 id)
{
    u8 returnpid ;
    u8 P0 ;
    u8 P1 ;

    P0 = (((id)^(id>>1)^(id>>2)^(id>>4))&0x01)<<6;
    P1 = ((~((id>>1)^(id>>3)^(id>>4)^(id>>5)))&0x01)<<7;

    returnpid = id|P0|P1 ;

    return returnpid ;
}

//=====================================================
// 函数: u8 LINCAlcChecksum(u8 *dat)
// 描述: 计算校验码。
// 参数: 数据场传输的数据。
// 返回: 校验码。
// 版本: VER1.0
// 日期: 2020-12-2
// 备注:
//=====================================================
static u8 LINCAlcChecksum(u8 *dat)
{
    u16 sum = 0;
    u8 i;

```

```
for(I = 0; i < 8; i++)
{
    sum += dat[i];
    if(sum & 0xFF00)
    {
        sum = (sum & 0x00FF) + 1;
    }
}
sum ^= 0x00FF;
return (u8)sum;
}

//=====================================================
// 函数: void Lin_SendBreak(void)
// 描述: 发送显性间隔信号。
// 参数: none.
// 返回: none.
// 版本: VER1.0
// 日期: 2020-12-2
// 备注:
//=====================================================
void Lin_SendBreak(void)
{
    SetTimer2Baudrate(Baudrate_Break);
    UART2_TxByte(0);
    SetTimer2Baudrate(Baudrate2);
}

//=====================================================
// 函数: void Lin_Send(u8 *puts)
// 描述: 发送 LIN 总线报文。
// 参数: 待发送的数据场内容
// 返回: none.
// 版本: VER1.0
// 日期: 2020-12-2
// 备注:
//=====================================================
void Lin_Send(u8 *puts)
{
    u8 i;

    Lin_SendBreak();                //Break
    UART2_TxByte(0x55);            //SYNC
    UART2_TxByte(Lin_CheckPID(LIN_ID)); //LIN ID
    for(i=0;i<8;i++)
    {
        UART2_TxByte(puts[i]);
    }
    UART2_TxByte(LINCalcChecksum(puts));
}

//=====================================================
// 函数: void UART1_TxByte(u8 dat)
// 描述: 发送一个字节
// 参数: 无
// 返回: 无
// 版本: V1.0, 2014-6-30
//=====================================================
```

```
void UART1_TxByte(u8 dat)
{
    SBUF = dat;
    B_TX1_Busy = 1;
    while(B_TX1_Busy);
}

//=====================================================
// 函数: void UART2_TxByte(u8 dat)
// 描述: 发送一个字节.
// 参数: 无
// 返回: 无
// 版本: V1.0, 2014-6-30
//=====================================================
void UART2_TxByte(u8 dat)
{
    S2BUF = dat;
    B_TX2_Busy = 1;
    while(B_TX2_Busy);
}

//=====================================================
// 函数: void PrintString1(u8 *puts)
// 描述: 串口1 发送字符串函数。
// 参数: puts:          字符串指针.
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====================================================
void PrintString1(u8 *puts)
{
    for (; *puts != 0; puts++) //遇到停止符0 结束
    {
        SBUF = *puts;
        B_TX1_Busy = 1;
        while(B_TX1_Busy);
    }
}

//=====================================================
// 函数: void PrintString2(u8 *puts)
// 描述: 串口2 发送字符串函数。
// 参数: puts:          字符串指针.
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====================================================
//void PrintString2(u8 *puts)
//{
//    for (; *puts != 0; puts++) //遇到停止符0 结束
//    {
//        S2BUF = *puts;
//        B_TX2_Busy = 1;
//        while(B_TX2_Busy);
//    }
//}
```

```
//=====
// 函数: SetTimer2Baudrate(u16 dat)
// 描述: 设置Timer2 做波特率发生器。
// 参数: dat: Timer2 的重装值。
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====
void SetTimer2Baudrate(u16 dat)
{
    AUXR &= ~(1<<4);           //Timer stop
    AUXR &= ~(1<<3);           //Timer2 set As Timer
    AUXR |= (1<<2);            //Timer2 set as 1T mode
    TH2 = dat / 256;
    TL2 = dat % 256;
    IE2 &= ~(1<<2);           //禁止中断
    AUXR |= (1<<4);           //Timer run enable
}

//=====
// 函数: void UART1_config(u8 brt)
// 描述: UART1 初始化函数。
// 参数: brt: 选择波特率, 2: 使用Timer2 做波特率, 其它值: 使用Timer1 做波特率
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====
void UART1_config(u8 brt)
{
    /******* 波特率使用定时器2 *****/
    if(brt == 2)
    {
        AUXR |= 0x01;           //S1 BRT Use Timer2;
        SetTimer2Baudrate(Baudrate1);
    }

    /******* 波特率使用定时器1 *****/
    else
    {
        TRI = 0;
        AUXR &= ~0x01;           //S1 BRT Use Timer1;
        AUXR |= (1<<6);           //Timer1 set as 1T mode
        TMOD &= ~(1<<6);          //Timer1 set As Timer
        TMOD &= ~0x30;           //Timer1_16bitAutoReload;
        TH1 = (u8)(Baudrate1 / 256);
        TL1 = (u8)(Baudrate1 % 256);
        ET1 = 0;                 //禁止中断
        INT_CLKO &= ~0x02;       //不输出时钟
        TRI = 1;
    }
    /******* */

    SCON = (SCON & 0x3f) / 0x40; //UART1 模式: 0x00: 同步移位输出,
    //                               0x40: 8 位数据,可变波特率,
    //                               0x80: 9 位数据,固定波特率,
    //                               0xc0: 9 位数据,可变波特率
    // PS = 1;                     //高优先级中断

```

```

    ES = 1;                                // 允许中断
    REN = 1;                                // 允许接收
    P_SW1 &= 0x3f;
//    P_SW1 |= 0x80;                        // UART1 switch to: 0x00: P3.0 P3.1,
//                                           0x40: P3.6 P3.7,
//                                           0x80: P1.6 P1.7,
//                                           0xC0: P4.3 P4.4

    B_TX1_Busy = 0;
    TX1_Cnt = 0;
    RX1_Cnt = 0;
}

//=====================================================
// 函数: void UART2_config(u8 brt)
// 描述: UART2 初始化函数。
// 参数: brt: 选择波特率, 2: 使用 Timer2 做波特率, 其它值: 无效
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====================================================
void UART2_config(u8 brt)
{
    if(brt == 2)
    {
        SetTimer2Baudrate(Baudrate2);

        S2CON &= ~(1<<7);                // 8 位数据, 1 位起始位, 1 位停止位, 无校验
        IE2 |= 1;                          // 允许中断
        S2CON |= (1<<4);                  // 允许接收
        P_SW2 &= ~0x01;
//        P_SW2 |= 1;                      // UART2 switch to: 0: P1.0/P1.1, 1: P4.6/P4.7

        B_TX2_Busy = 0;
        TX2_Cnt = 0;
        RX2_Cnt = 0;
    }
}

//=====================================================
// 函数: void UART1_int (void) interrupt UART1_VECTOR
// 描述: UART1 中断函数。
// 参数: none.
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====================================================
void UART1_int (void) interrupt 4
{
    if(RI)
    {
        RI = 0;
        if(RX1_Cnt >= UART1_BUF_LENGTH) RX1_Cnt = 0;
        RX1_Buffer[RX1_Cnt] = SBUF;
        RX1_Cnt++;
        RX1_TimeOut = 5;
    }
}

```



```
    if(TI)
    {
        TI = 0;
        B_TX1_Busy = 0;
    }
}

//=====================================================
// 函数: void UART2_int (void) interrupt UART2_VECTOR
// 描述: UART2 中断函数。
// 参数: none.
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====================================================
void UART2_int (void) interrupt 8
{
    if((S2CON & 1) != 0)
    {
        S2CON &= ~1;                                //Clear Rx flag
        if(RX2_Cnt >= UART2_BUF_LENGTH) RX2_Cnt = 0;
        RX2_Buffer[RX2_Cnt] = S2BUF;
        RX2_Cnt++;
        RX2_TimeOut = 5;
    }

    if((S2CON & 2) != 0)
    {
        S2CON &= ~2;                                //Clear Tx flag
        B_TX2_Busy = 0;
    }
}
```

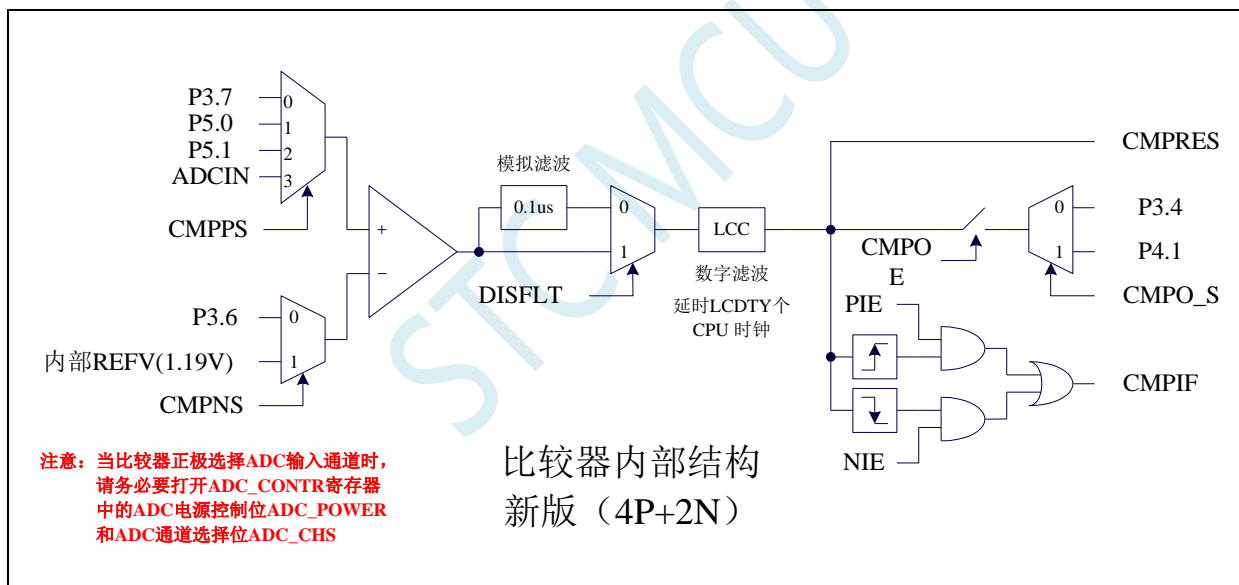
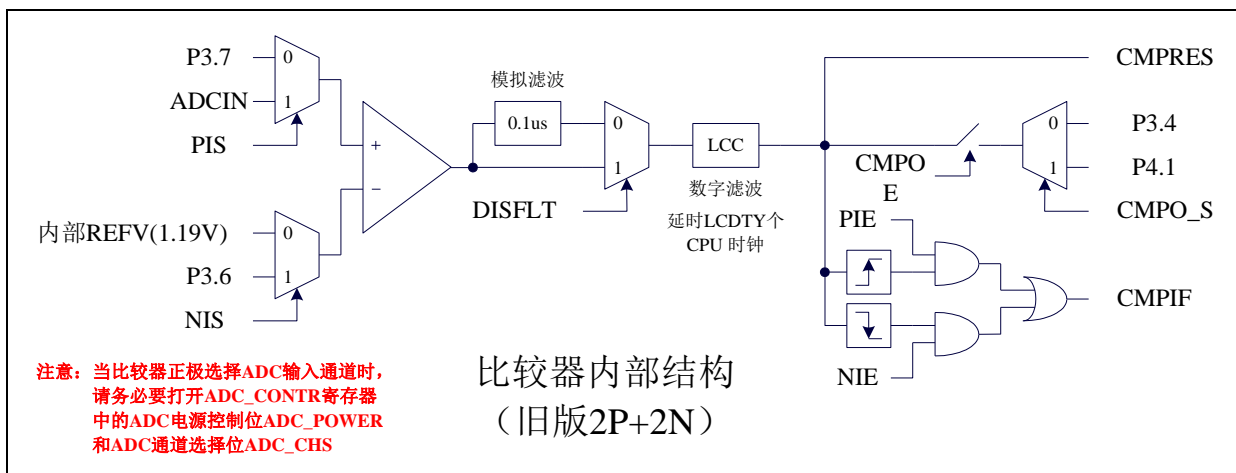
# 17 比较器，掉电检测，内部 1.19V 参考信号源 (BGV)

| 产品线                    | 比较器           |               | CMPEXCFG |
|------------------------|---------------|---------------|----------|
|                        | 旧版<br>(2P+2N) | 新版<br>(4P+2N) |          |
| STC8H1K08 系列           | ●             |               |          |
| STC8H1K28 系列           | ●             |               |          |
| STC8H3K64S4 系列         | ●             |               |          |
| STC8H3K64S2 系列         | ●             |               |          |
| STC8H8K64U 系列 A 版本     | ●             |               |          |
| STC8H8K64U 系列 B/C/D 版本 |               | ●             | ●        |
| STC8H4K64TL 系列         |               | ●             | ●        |
| STC8H4K64TLCD 系列       |               | ●             | ●        |
| STC8H1K08T 系列          |               | ●             | ●        |

STC8H 系列单片机内部集成了一个比较器。比较器的正极可以是 P3.7 端口或者 ADC 的模拟输入通道（新版比较器的正极可以是 P3.7 端口、P5.0 端口、P5.1 端口或者 ADC 的模拟输入通道），而负极可以 P3.6 端口或者是内部 BandGap 经过 OP 后的 REFB 电压（内部固定比较电压）。通过多路选择器和分时复用可实现多个比较器的应用。

比较器内部有可程序控制的两级滤波：模拟滤波和数字滤波。模拟滤波可以过滤掉比较输入信号中的毛刺信号，数字滤波可以等待输入信号更加稳定后再进行比较。比较结果可直接通过读取内部寄存器位获得，也可将比较器结果正向或反向输出到外部端口。将比较结果输出到外部端口可用作外部事件的触发信号和反馈信号，可扩大比较的应用范围。

## 17.1 比较器内部结构图



## 17.2 比较器输出功能脚切换

| 符号    | 地址  | B7    | B6 | B5         | B4     | B3   | B2   | B1   | B0 |
|-------|-----|-------|----|------------|--------|------|------|------|----|
| P_SW2 | BAH | EAXFR | -  | I2C_S[1:0] | CMPO_S | S4_S | S3_S | S2_S |    |

CMPO\_S: 比较器输出脚选择位

| CMPO_S | CMPO |
|--------|------|
| 0      | P3.4 |
| 1      | P4.1 |

## 17.3 比较器相关的寄存器

| 符号       | 描述         | 地址    | 位地址与符号    |        |            |     |     |         |              |         | 复位值       |
|----------|------------|-------|-----------|--------|------------|-----|-----|---------|--------------|---------|-----------|
|          |            |       | B7        | B6     | B5         | B4  | B3  | B2      | B1           | B0      |           |
| CMPCR1   | 比较器控制寄存器 1 | E6H   | CM PEN    | CMPIF  | PIE        | NIE | PIS | NIS     | CMPOE        | CM PRES | 0000,0000 |
| CMPCR2   | 比较器控制寄存器 2 | E7H   | INVCMP0   | DISFLT | LCDTY[5:0] |     |     |         |              |         | 0000,0000 |
| CMPEXCFG | 比较器扩展配置寄存器 | FEAEH | CHYS[1:0] |        | -          | -   | -   | CM P NS | CM P PS[1:0] |         | 00xx,x000 |

### 17.3.1 比较器控制寄存器 1 (CMPCR1)

| 符号     | 地址  | B7     | B6    | B5  | B4  | B3  | B2  | B1    | B0      |
|--------|-----|--------|-------|-----|-----|-----|-----|-------|---------|
| CMPCR1 | E6H | CM PEN | CMPIF | PIE | NIE | PIS | NIS | CMPOE | CM PRES |

CM PEN: 比较器模块使能位

- 0: 关闭比较功能
- 1: 使能比较功能

CMPIF: 比较器中断标志位。当 PIE 或 NIE 被使能后, 若产生相应的中断信号, 硬件自动将 CMPIF 置 1, 并向 CPU 提出中断请求。此标志位必须用户软件清零。

(注意: 没有使能比较器中断时, 硬件不会设置此中断标志, 即使用查询方式访问比较器时, 不能查询此中断标志)

PIE: 比较器上升沿中断使能位。

- 0: 禁止比较器上升沿中断。
- 1: 使能比较器上升沿中断。使能比较器的比较结果由 0 变成 1 时产生中断请求。

NIE: 比较器下降沿中断使能位。

- 0: 禁止比较器下降沿中断。
- 1: 使能比较器下降沿中断。使能比较器的比较结果由 1 变成 0 时产生中断请求。

PIS: 比较器的正极选择位 (适用于旧版比较器, 新版比较器使用 CMPEXCFG 中的 CMPPS 进行选择)

- 0: 选择外部端口 P3.7 为比较器正极输入源。
- 1: 通过 ADC\_CONTR 中的 ADC\_CHS 位选择 ADC 的模拟输入端作为比较器正极输入源。

(注意 1: 当比较器正极选择 ADC 输入通道时, 请务必打开 ADC\_CONTR 寄存器中的 ADC 电源控制位 ADC\_POWER 和 ADC 通道选择位 ADC\_CHS)

(注意 2: 当需要使用比较器中断唤醒掉电模式/时钟停振模式时, 比较器正极必须选择 P3.7, 不能使用 ADC 输入通道)

NIS: 比较器的负极选择位 (适用于旧版比较器, 新版比较器使用 CMPEXCFG 中的 CMPNS 进行选择)

- 0: 选择内部 BandGap 经过 OP 后的电压 REFV 作为比较器负极输入源。 (芯片在出厂时, 内部参考信号源调整为 1.19V)
- 1: 选择外部端口 P3.6 为比较器负极输入源。

CMPOE: 比较器结果输出控制位

- 0: 禁止比较器结果输出
- 1: 使能比较器结果输出。比较器结果输出到 P3.4 或者 P4.1 (由 P\_SW2 中的 CMPO\_S 进行设定)

CM PRES: 比较器的比较结果。此位为只读。

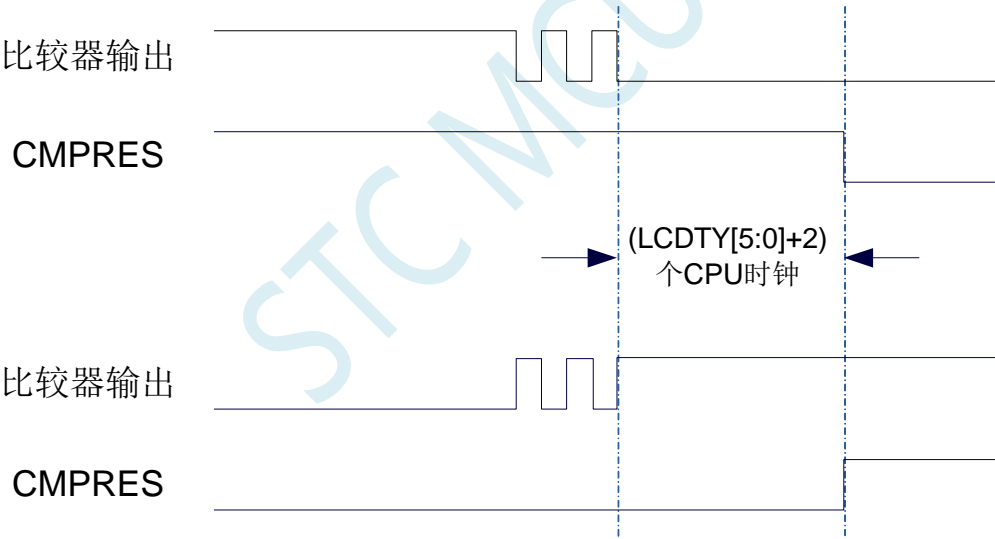
- 0: 表示 CMP+ 的电平低于 CMP- 的电平
- 1: 表示 CMP+ 的电平高于 CMP- 的电平

CM PRES 是经过数字滤波后的输出信号, 而不是比较器的直接输出结果。

### 17.3.2 比较器控制寄存器 2（CMPCR2）

| 符号     | 地址  | B7      | B6     | B5         | B4 | B3 | B2 | B1 | B0 |
|--------|-----|---------|--------|------------|----|----|----|----|----|
| CMPCR2 | E7H | INVCMP0 | DISFLT | LCDTY[5:0] |    |    |    |    |    |

- INVCMP0：比较器结果输出控制
- 0：比较器结果正向输出。若 CMPRES 为 0，则 P3.4/P4.1 输出低电平，反之输出高电平。
  - 1：比较器结果反向输出。若 CMPRES 为 0，则 P3.4/P4.1 输出高电平，反之输出低电平。
- DISFLT：模拟滤波功能控制
- 0：使能 0.1us 模拟滤波功能
  - 1：关闭 0.1us 模拟滤波功能，可略微提高比较器的比较速度。
- LCDTY[5:0]：数字滤波功能控制
- 数字滤波功能即为数字信号去抖动功能。当比较结果发生上升沿或者下降沿变化时，比较器侦测变化后的信号必须维持 LCDTY 所设置的 CPU 时钟数不发生变化，才认为数据变化是有效的；否则将视同信号无变化。
- 注意：**当使能数字滤波功能后，芯片内部实际的等待时钟需额外增加两个状态机切换时间，即若 LCDTY 设置为 0 时，为关闭数字滤波功能；若 LCDTY 设置为非 0 值 n（n=1~63）时，则实际的数字滤波时间为（n+2）个系统时钟



### 17.3.3 比较器扩展配置寄存器（CMPEXCFG）

| 符号       | 地址    | B7        | B6 | B5 | B4 | B3 | B2    | B1         | B0 |
|----------|-------|-----------|----|----|----|----|-------|------------|----|
| CMPEXCFG | FEAEH | CHYS[1:0] |    | -  | -  | -  | CMPNS | CMPPS[1:0] |    |

CHYS[1:0]：比较器 DC 迟滞输入选择

| CHYS [1:0] | 比较器 DC 迟滞输入选择 |
|------------|---------------|
| 00         | 0mV           |
| 01         | 10mV          |
| 10         | 20mV          |
| 11         | 30mV          |

CMPNS：比较器负端输入选择

0: P3.6

1: 选择内部 BandGap 经过 OP 后的电压 REFV 作为比较器负极输入源 (芯片在出厂时, 内部参考电压调整为 **1.19V**)

CMPPS[1:0]: 比较器正端输入选择

| CMPPS[1:0] | 比较器正端 |
|------------|-------|
| 00         | P3.7  |
| 01         | P5.0  |
| 10         | P5.1  |
| 11         | ADCIN |

## 17.4 范例程序

### 17.4.1 旧版比较器的使用（中断方式）

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void CMP_Isr() interrupt 21
```

```
{
    CMPCR1 &= ~0x40;           //清中断标志
    if (CMPCR1 & 0x01)
    {
        P10 = !P10;           //上升沿中断测试端口
    }
    else
    {
        P11 = !P11;           //下降沿中断测试端口
    }
}
```

```
void main()
```

```
{
    P_SW2 /= 0x80;           //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    CMPCR2 = 0x00;
    CMPCR2 &= ~0x80;         //比较器正向输出
    // CMPCR2 /= 0x80;         //比较器反向输出
    CMPCR2 &= ~0x40;         //使能 0.1us 滤波
    // CMPCR2 /= 0x40;         //禁止 0.1us 滤波
    // CMPCR2 &= ~0x3f;         //比较器结果直接输出
    CMPCR2 /= 0x10;          //比较器结果经过 16 个去抖时钟后输出
    CMPCR1 = 0x00;
    CMPCR1 /= 0x30;          //使能比较器边沿中断
    // CMPCR1 &= ~0x20;         //禁止比较器上升沿中断
    // CMPCR1 /= 0x20;         //使能比较器上升沿中断
    // CMPCR1 &= ~0x10;         //禁止比较器下降沿中断
    // CMPCR1 /= 0x10;         //使能比较器下降沿中断
    CMPCR1 &= ~0x08;         //P3.7 为 CMP+ 输入脚
    // CMPCR1 /= 0x08;         //ADC 输入脚为 CMP+ 输入脚
}
```

```
// CMPCR1 &= ~0x04;
CMPCR1 |= 0x04;
// CMPCR1 &= ~0x02;
CMPCR1 |= 0x02;
CMPCR1 |= 0x80;

EA = 1;

while (1);
}
```

## 汇编代码

;测试工作频率为 11.0592MHz

```
P_SW2      DATA      0BAH

CMPCR1      DATA      0E6H
CMPCR2      DATA      0E7H

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

ORG         0000H
LJMP        MAIN
ORG         00ABH
LJMP        CMPISR

CMPISR:     ORG         0100H

            PUSH        ACC
            ANL          CMPCR1,#NOT 40H      ;清中断标志
            MOV          A,CMPCR1
            JB           ACC.0,RSING

FALLING:    CPL          P1.0                ;下降沿中断测试端口
            POP          ACC
            RETI

RSING:      CPL          P1.1                ;上升沿中断测试端口
            POP          ACC
            RETI

MAIN:       MOV          SP,#5FH
            ORL          P_SW2,#80H          ;使能访问 XFR

            MOV          P0M0,#00H
            MOV          P0M1,#00H
```



```

MOV      P1M0, #00H
MOV      P1M1, #00H
MOV      P2M0, #00H
MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      CMPCR2, #00H
ANL      CMPCR2, #NOT 80H      ;比较器正向输出
; ORL      CMPCR2, #80H      ;比较器反向输出
; ANL      CMPCR2, #NOT 40H    ;使能 0.1us 滤波
; ORL      CMPCR2, #40H      ;禁止 0.1us 滤波
; ANL      CMPCR2, #NOT 3FH    ;比较器结果直接输出
; ORL      CMPCR2, #10H      ;比较器结果经过 16 个去抖时钟后输出
MOV      CMPCR1, #00H
ORL      CMPCR1, #30H      ;使能比较器边沿中断
; ANL      CMPCR1, #NOT 20H    ;禁止比较器上升沿中断
; ORL      CMPCR1, #20H      ;使能比较器上升沿中断
; ANL      CMPCR1, #NOT 10H    ;禁止比较器下降沿中断
; ORL      CMPCR1, #10H      ;使能比较器下降沿中断
; ANL      CMPCR1, #NOT 08H    ;P3.7 为 CMP+ 输入脚
; ORL      CMPCR1, #08H      ;ADC 输入脚为 CMP+ 输入脚
; ANL      CMPCR1, #NOT 04H    ;内部 1.19V 参考信号源为 CMP- 输入脚
; ORL      CMPCR1, #04H      ;P3.6 为 CMP- 输入脚
; ANL      CMPCR1, #NOT 02H    ;禁止比较器输出
; ORL      CMPCR1, #02H      ;使能比较器输出
; ORL      CMPCR1, #80H      ;使能比较器模块
SETB     EA

JMP      $

END

```

## 17.4.2 旧版比较器的使用（查询方式）

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```
void main()
```

```

{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;

```

```

P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

CMPCR2 = 0x00;
CMPCR2 &= ~0x80;           // 比较器正向输出
// CMPCR2 |= 0x80;         // 比较器反向输出
CMPCR2 &= ~0x40;           // 使能 0.1us 滤波
// CMPCR2 |= 0x40;         // 禁止 0.1us 滤波
// CMPCR2 &= ~0x3f;        // 比较器结果直接输出
CMPCR2 |= 0x10;             // 比较器结果经过 16 个去抖时钟后输出
CMPCR1 = 0x00;
CMPCR1 |= 0x30;             // 使能比较器边沿中断
// CMPCR1 &= ~0x20;        // 禁止比较器上升沿中断
// CMPCR1 |= 0x20;         // 使能比较器上升沿中断
// CMPCR1 &= ~0x10;        // 禁止比较器下降沿中断
// CMPCR1 |= 0x10;         // 使能比较器下降沿中断
CMPCR1 &= ~0x08;           // P3.7 为 CMP+ 输入脚
// CMPCR1 |= 0x08;         // ADC 输入脚为 CMP+ 输入脚
// CMPCR1 &= ~0x04;        // 内部 1.19V 参考信号源为 CMP- 输入脚
CMPCR1 |= 0x04;            // P3.6 为 CMP- 输入脚
// CMPCR1 &= ~0x02;        // 禁止比较器输出
CMPCR1 |= 0x02;            // 使能比较器输出
CMPCR1 |= 0x80;            // 使能比较器模块

while (1)
{
    P10 = CMPCR1 & 0x01;    // 读取比较器比较结果
}

```

## 汇编代码

; 测试工作频率为 11.0592MHz

|        |      |       |
|--------|------|-------|
| P_SW2  | DATA | 0BAH  |
| CMPCR1 | DATA | 0E6H  |
| CMPCR2 | DATA | 0E7H  |
| P1M1   | DATA | 091H  |
| P1M0   | DATA | 092H  |
| P0M1   | DATA | 093H  |
| P0M0   | DATA | 094H  |
| P2M1   | DATA | 095H  |
| P2M0   | DATA | 096H  |
| P3M1   | DATA | 0B1H  |
| P3M0   | DATA | 0B2H  |
| P4M1   | DATA | 0B3H  |
| P4M0   | DATA | 0B4H  |
| P5M1   | DATA | 0C9H  |
| P5M0   | DATA | 0CAH  |
| ORG    |      | 0000H |
| LJMP   |      | MAIN  |

```

    ORG      0100H

MAIN:

    MOV     SP, #5FH
    ORL     P_SW2, #80H           ;使能访问 XFR

    MOV     P0M0, #00H
    MOV     P0M1, #00H
    MOV     P1M0, #00H
    MOV     P1M1, #00H
    MOV     P2M0, #00H
    MOV     P2M1, #00H
    MOV     P3M0, #00H
    MOV     P3M1, #00H
    MOV     P4M0, #00H
    MOV     P4M1, #00H
    MOV     P5M0, #00H
    MOV     P5M1, #00H

    MOV     CMPCR2, #00H
    ANL     CMPCR2, #NOT 80H      ;比较器正向输出
    ; ORL     CMPCR2, #80H        ;比较器反向输出
    ; ANL     CMPCR2, #NOT 40H    ;使能 0.1us 滤波
    ; ORL     CMPCR2, #40H       ;禁止 0.1us 滤波
    ; ANL     CMPCR2, #NOT 3FH   ;比较器结果直接输出
    ; ORL     CMPCR2, #10H      ;比较器结果经过 16 个去抖时钟后输出
    MOV     CMPCR1, #00H
    ORL     CMPCR1, #30H          ;使能比较器边沿中断
    ; ANL     CMPCR1, #NOT 20H   ;禁止比较器上升沿中断
    ; ORL     CMPCR1, #20H      ;使能比较器上升沿中断
    ; ANL     CMPCR1, #NOT 10H  ;禁止比较器下降沿中断
    ; ORL     CMPCR1, #10H     ;使能比较器下降沿中断
    ; ANL     CMPCR1, #NOT 08H  ;P3.7 为 CMP+ 输入脚
    ; ORL     CMPCR1, #08H     ;ADC 输入脚为 CMP+ 输入脚
    ; ANL     CMPCR1, #NOT 04H  ;内部 1.19V 参考信号源为 CMP- 输入脚
    ; ORL     CMPCR1, #04H     ;P3.6 为 CMP- 输入脚
    ; ANL     CMPCR1, #NOT 02H  ;禁止比较器输出
    ; ORL     CMPCR1, #02H     ;使能比较器输出
    ORL     CMPCR1, #80H         ;使能比较器模块

LOOP:

    MOV     A, CMPCR1
    MOV     C, ACC.0
    MOV     P1.0, C              ;读取比较器比较结果
    JMP     LOOP

END

```

### 17.4.3 新版比较器的使用（中断方式）

#### C 语言代码

```
//测试工作频率为 11.0592MHz
```

```
#include "stc8h.h"
#include "intrins.h"
```

```
void CMP_Isr() interrupt 21
```

```

{
    CMPCR1 &= ~0x40;                //清中断标志
    if (CMPCR1 & 0x01)
    {
        P10 = !P10;                //上升沿中断测试端口
    }
    else
    {
        P11 = !P11;                //下降沿中断测试端口
    }
}

void main()
{
    P_SW2 /= 0x80;                  //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    CMPEXCFG = 0x00;
    // CMPEXCFG /= 0x40;              //比较器 DC 迟滞输入选择
    //                                //0:0mV; 0x40:10mV; 0x80:20mV; 0xc0:30mV

    CMPEXCFG &= ~0x03;              //P3.7 为 CMP+ 输入脚
    // CMPEXCFG /= 0x01;              //P5.0 为 CMP+ 输入脚
    // CMPEXCFG /= 0x02;              //P5.1 为 CMP+ 输入脚
    // CMPEXCFG /= 0x03;              //ADC 输入脚为 CMP+ 输入脚
    CMPEXCFG &= ~0x04;              //P3.6 为 CMP- 输入脚
    // CMPEXCFG /= 0x04;              //内部 1.19V 参考电压为 CMP- 输入脚

    CMPCR2 = 0x00;
    CMPCR2 &= ~0x80;                //比较器正向输出
    // CMPCR2 /= 0x80;                //比较器反向输出
    CMPCR2 &= ~0x40;                //使能 0.1us 滤波
    // CMPCR2 /= 0x40;                //禁止 0.1us 滤波
    // CMPCR2 &= ~0x3f;                //比较器结果直接输出
    // CMPCR2 /= 0x10;                //比较器结果经过 16 个去抖时钟后输出
    CMPCR1 = 0x00;
    CMPCR1 /= 0x30;                  //使能比较器边沿中断
    // CMPCR1 &= ~0x20;                //禁止比较器上升沿中断
    // CMPCR1 /= 0x20;                //使能比较器上升沿中断
    // CMPCR1 &= ~0x10;                //禁止比较器下降沿中断
    // CMPCR1 /= 0x10;                //使能比较器下降沿中断
    // CMPCR1 &= ~0x02;                //禁止比较器输出
    CMPCR1 /= 0x02;                  //使能比较器输出
    CMPCR1 /= 0x80;                  //使能比较器模块

    EA = 1;

```

```

    while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

```

P_SW2      DATA      0BAH
CMPCR1     DATA      0E6H
CMPCR2     DATA      0E7H

P1M1       DATA      091H
P1M0       DATA      092H
P0M1       DATA      093H
P0M0       DATA      094H
P2M1       DATA      095H
P2M0       DATA      096H
P3M1       DATA      0B1H
P3M0       DATA      0B2H
P4M1       DATA      0B3H
P4M0       DATA      0B4H
P5M1       DATA      0C9H
P5M0       DATA      0CAH

CMPEXCFG   XDATA      0FEAEH

            ORG        0000H
            LJMP       MAIN
            ORG        00ABH
            LJMP       CMPISR

            ORG        0100H
CMPISR:
    PUSH     ACC
    ANL     CMPCR1,#NOT 40H      ;清中断标志
    MOV     A,CMPCR1
    JB      ACC.0,RSING

FALLING:
    CPL     P1.0                ;下降沿中断测试端口
    POP     ACC
    RETI

RSING:
    CPL     P1.1                ;上升沿中断测试端口
    POP     ACC
    RETI

MAIN:
    MOV     SP,#5FH
    ORL     P_SW2,#80H          ;使能访问 XFR

    MOV     P0M0,#00H
    MOV     P0M1,#00H
    MOV     P1M0,#00H
    MOV     P1M1,#00H
    MOV     P2M0,#00H
    MOV     P2M1,#00H
    MOV     P3M0,#00H
    MOV     P3M1,#00H
    MOV     P4M0,#00H

```

```

MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      DPTR, #CMPEXCFG
CLR      A
ANL      A, #NOT 03H      ; P3.7 为 CMP+ 输入脚
; ORL     A, #01H          ; P5.0 为 CMP+ 输入脚
; ORL     A, #02H          ; P5.1 为 CMP+ 输入脚
; ORL     A, #03H          ; ADC 输入脚为 CMP+ 输入脚
ANL      A, #NOT 04H      ; P3.6 为 CMP- 输入脚
; ORL     A, #04H          ; 内部 1.19V 参考信号源为 CMP- 输入脚
MOVX     @DPTR, A

MOV      CMPCR2, #00H
ANL      CMPCR2, #NOT 80H ; 比较器正向输出
; ORL     CMPCR2, #80H    ; 比较器反向输出
ANL      CMPCR2, #NOT 40H ; 使能 0.1us 滤波
; ORL     CMPCR2, #40H    ; 禁止 0.1us 滤波
; ANL     CMPCR2, #NOT 3FH ; 比较器结果直接输出
ORL      CMPCR2, #10H     ; 比较器结果经过 16 个去抖时钟后输出
MOV      CMPCR1, #00H
ORL      CMPCR1, #30H     ; 使能比较器边沿中断
; ANL     CMPCR1, #NOT 20H ; 禁止比较器上升沿中断
; ORL     CMPCR1, #20H    ; 使能比较器上升沿中断
; ANL     CMPCR1, #NOT 10H ; 禁止比较器下降沿中断
; ORL     CMPCR1, #10H    ; 使能比较器下降沿中断
; ANL     CMPCR1, #NOT 02H ; 禁止比较器输出
ORL      CMPCR1, #02H     ; 使能比较器输出
ORL      CMPCR1, #80H     ; 使能比较器模块
SETB     EA

JMP      $

END

```

## 17.4.4 新版比较器的使用（查询方式）

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```
void main()
```

```

{
    P_SW2 /= 0x80;          //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;

```

```

P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

CMPEXCFG = 0x00;
// CMPEXCFG |= 0x40;           //比较器DC 迟滞输入选择, 0:0mV; 0x40:10mV; 0x80:20mV; 0xc0:30mV

CMPEXCFG &= ~0x03;           //P3.7 为 CMP+ 输入脚
// CMPEXCFG |= 0x01;           //P5.0 为 CMP+ 输入脚
// CMPEXCFG |= 0x02;           //P5.1 为 CMP+ 输入脚
// CMPEXCFG |= 0x03;           //ADC 输入脚为 CMP+ 输入脚
CMPEXCFG &= ~0x04;           //P3.6 为 CMP- 输入脚
// CMPEXCFG |= 0x04;           //内部1.19V 参考电压为 CMP- 输入脚

CMPCR2 = 0x00;
CMPCR2 &= ~0x80;           //比较器正向输出
// CMPCR2 |= 0x80;           //比较器反向输出
CMPCR2 &= ~0x40;           //使能0.1us 滤波
// CMPCR2 |= 0x40;           //禁止0.1us 滤波
// CMPCR2 &= ~0x3f;           //比较器结果直接输出
CMPCR2 |= 0x10;           //比较器结果经过16 个去抖时钟后输出
CMPCR1 = 0x00;
CMPCR1 |= 0x30;           //使能比较器边沿中断
// CMPCR1 &= ~0x20;           //禁止比较器上升沿中断
// CMPCR1 |= 0x20;           //使能比较器上升沿中断
// CMPCR1 &= ~0x10;           //禁止比较器下降沿中断
// CMPCR1 |= 0x10;           //使能比较器下降沿中断
// CMPCR1 &= ~0x02;           //禁止比较器输出
CMPCR1 |= 0x02;           //使能比较器输出
CMPCR1 |= 0x80;           //使能比较器模块

while (1)
{
    P10 = CMPCR1 & 0x01;     //读取比较器比较结果
}

```

## 汇编代码

;测试工作频率为11.0592MHz

|        |      |      |
|--------|------|------|
| P_SW2  | DATA | 0BAH |
| CMPCR1 | DATA | 0E6H |
| CMPCR2 | DATA | 0E7H |
| P1M1   | DATA | 091H |
| P1M0   | DATA | 092H |
| P0M1   | DATA | 093H |
| P0M0   | DATA | 094H |
| P2M1   | DATA | 095H |
| P2M0   | DATA | 096H |
| P3M1   | DATA | 0B1H |
| P3M0   | DATA | 0B2H |
| P4M1   | DATA | 0B3H |
| P4M0   | DATA | 0B4H |
| P5M1   | DATA | 0C9H |
| P5M0   | DATA | 0CAH |

```

CMPEXCFG  XDATA  0FEAEH

                ORG      0000H
                LJMP     MAIN

MAIN:          ORG      0100H

                MOV      SP, #5FH
                ORL      P_SW2, #80H                ;使能访问 XFR

                MOV      P0M0, #00H
                MOV      P0M1, #00H
                MOV      P1M0, #00H
                MOV      P1M1, #00H
                MOV      P2M0, #00H
                MOV      P2M1, #00H
                MOV      P3M0, #00H
                MOV      P3M1, #00H
                MOV      P4M0, #00H
                MOV      P4M1, #00H
                MOV      P5M0, #00H
                MOV      P5M1, #00H

                MOV      DPTR, #CMPEXCFG
                CLR      A
                ANL      A, #NOT 03H                ; P3.7 为 CMP+ 输入脚
                ORL      A, #01H                    ; P5.0 为 CMP+ 输入脚
                ORL      A, #02H                    ; P5.1 为 CMP+ 输入脚
                ORL      A, #03H                    ; ADC 输入脚为 CMP+ 输入脚
                ANL      A, #NOT 04H                ; P3.6 为 CMP- 输入脚
                ORL      A, #04H                    ; 内部 1.19V 参考信号源为 CMP- 输入脚
                MOVX     @DPTR, A

                MOV      CMPCR2, #00H
                ANL      CMPCR2, #NOT 80H            ; 比较器正向输出
                ORL      CMPCR2, #80H                ; 比较器反向输出
                ANL      CMPCR2, #NOT 40H            ; 使能 0.1us 滤波
                ORL      CMPCR2, #40H                ; 禁止 0.1us 滤波
                ANL      CMPCR2, #NOT 3FH            ; 比较器结果直接输出
                ORL      CMPCR2, #10H                ; 比较器结果经过 16 个去抖时钟后输出
                MOV      CMPCR1, #00H
                ORL      CMPCR1, #30H                ; 使能比较器边沿中断
                ANL      CMPCR1, #NOT 20H            ; 禁止比较器上升沿中断
                ORL      CMPCR1, #20H                ; 使能比较器上升沿中断
                ANL      CMPCR1, #NOT 10H            ; 禁止比较器下降沿中断
                ORL      CMPCR1, #10H                ; 使能比较器下降沿中断
                ANL      CMPCR1, #NOT 02H            ; 禁止比较器输出
                ORL      CMPCR1, #02H                ; 使能比较器输出
                ORL      CMPCR1, #80H                ; 使能比较器模块

LOOP:          MOV      A, CMPCR1
                MOV      C, ACC.0
                MOV      P1.0, C                    ; 读取比较器比较结果
                JMP      LOOP

                END

```



## 17.4.5 旧版比较器的多路复用应用（比较器+ADC 输入通道）

由于比较器的正极可以选择 ADC 的模拟输入通道，因此可以通过多路选择器和分时复用可实现多个比较器的应用。

**注意：**当比较器正极选择 ADC 输入通道时，请务必打开 ADC\_CONTR 寄存器中的 ADC 电源控制位 **ADC\_POWER** 和 ADC 通道选择位 **ADC\_CHS**

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void main()
```

```
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P1M0 &= 0xfe;                //设置 P1.0 为输入口
    P1M1 /= 0x01;
    ADC_CONTR = 0x80;            //使能 ADC 模块并选择 P1.0 为 ADC 输入脚

    CMPCR2 = 0x00;
    CMPCR1 = 0x00;

    CMPCR1 /= 0x08;                //ADC 输入脚为 CMP+ 输入脚
    CMPCR1 /= 0x04;                //P3.6 为 CMP- 输入脚
    CMPCR1 /= 0x02;                //使能比较器输出
    CMPCR1 /= 0x80;                //使能比较器模块

    while (1);
}
```

### 汇编代码

;测试工作频率为 11.0592MHz

|           |      |      |
|-----------|------|------|
| P_SW2     | DATA | 0BAH |
| CMPCR1    | DATA | 0E6H |
| CMPCR2    | DATA | 0E7H |
| ADC_CONTR | DATA | 0BCH |

```

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN

MAIN:     ORG          0100H

          MOV          SP, #5FH
          ORL          P_SW2, #80H          ;使能访问 XFR

          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          ANL          P1M0, #0FEH          ;设置 P1.0 为输入口
          ORL          P1M1, #01H
          MOV          ADC_CONTR, #80H      ;使能 ADC 模块并选择 P1.0 为 ADC 输入脚

          MOV          CMPCR2, #00H
          MOV          CMPCR1, #00H

          ORL          CMPCR1, #08H          ;ADC 输入脚为 CMP+ 输入脚
          ORL          CMPCR1, #04H          ;P3.6 为 CMP- 输入脚
          ORL          CMPCR1, #02H          ;使能比较器输出
          ORL          CMPCR1, #80H          ;使能比较器模块

LOOP:     JMP          LOOP

          END

```

## 17.4.6 新版比较器的多路复用应用（比较器+ADC 输入通道）

由于比较器的正极可以选择 ADC 的模拟输入通道，因此可以通过多路选择器和分时复用可实现多个比较器的应用。

**注意：**当比较器正极选择 ADC 输入通道时，请务必打开 ADC\_CONTR 寄存器中的 ADC 电源控制位 **ADC\_POWER** 和 ADC 通道选择位 **ADC\_CHS**

## C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void main()
```

```
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P1M0 &= 0xfe;                //设置 P1.0 为输入口
    P1M1 /= 0x01;
    ADC_CONTR = 0x80;            //使能 ADC 模块并选择 P1.0 为 ADC 输入脚

    CMPEXCFG = 0x00;
    // CMPEXCFG &= ~0x03;        //P3.7 为 CMP+ 输入脚
    // CMPEXCFG /= 0x01;        //P5.0 为 CMP+ 输入脚
    // CMPEXCFG /= 0x02;        //P5.1 为 CMP+ 输入脚
    CMPEXCFG /= 0x03;            //ADC 输入脚为 CMP+ 输入脚
    CMPEXCFG &= ~0x04;        //P3.6 为 CMP- 输入脚
    // CMPEXCFG /= 0x04;        //内部 1.19V 参考电压为 CMP- 输入脚

    CMPCR2 = 0x00;
    CMPCR1 = 0x00;
    CMPCR1 /= 0x02;              //使能比较器输出
    CMPCR1 /= 0x80;              //使能比较器模块

    while (1);
}
```

## 汇编代码

;测试工作频率为 11.0592MHz

|           |      |      |
|-----------|------|------|
| P_SW2     | DATA | 0BAH |
| CMPCR1    | DATA | 0E6H |
| CMPCR2    | DATA | 0E7H |
| ADC_CONTR | DATA | 0BCH |
|           |      |      |
| P1M1      | DATA | 091H |
| P1M0      | DATA | 092H |
| P0M1      | DATA | 093H |
| P0M0      | DATA | 094H |
| P2M1      | DATA | 095H |
| P2M0      | DATA | 096H |

```

P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

CMPEXCFG  XDATA      0FEAEH

          ORG          0000H
          LJMP         MAIN

          ORG          0100H
MAIN:
          MOV          SP, #5FH
          ORL          P_SW2, #80H          ;使能访问 XFR

          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          ANL          P1M0, #0FEH          ;设置 P1.0 为输入口
          ORL          P1M1, #01H
          MOV          ADC_CONTR, #80H      ;使能 ADC 模块并选择 P1.0 为 ADC 输入脚

          MOV          DPTR, #CMPEXCFG
          CLR          A
;          ANL          A, #NOT 03H          ; P3.7 为 CMP+ 输入脚
;          ORL          A, #01H            ; P5.0 为 CMP+ 输入脚
;          ORL          A, #02H            ; P5.1 为 CMP+ 输入脚
          ORL          A, #03H            ; ADC 输入脚为 CMP+ 输入脚
          ANL          A, #NOT 04H          ; P3.6 为 CMP- 输入脚
;          ORL          A, # 04H            ; 内部 1.19V 参考信号源为 CMP- 输入脚
          MOVX         @DPTR, A

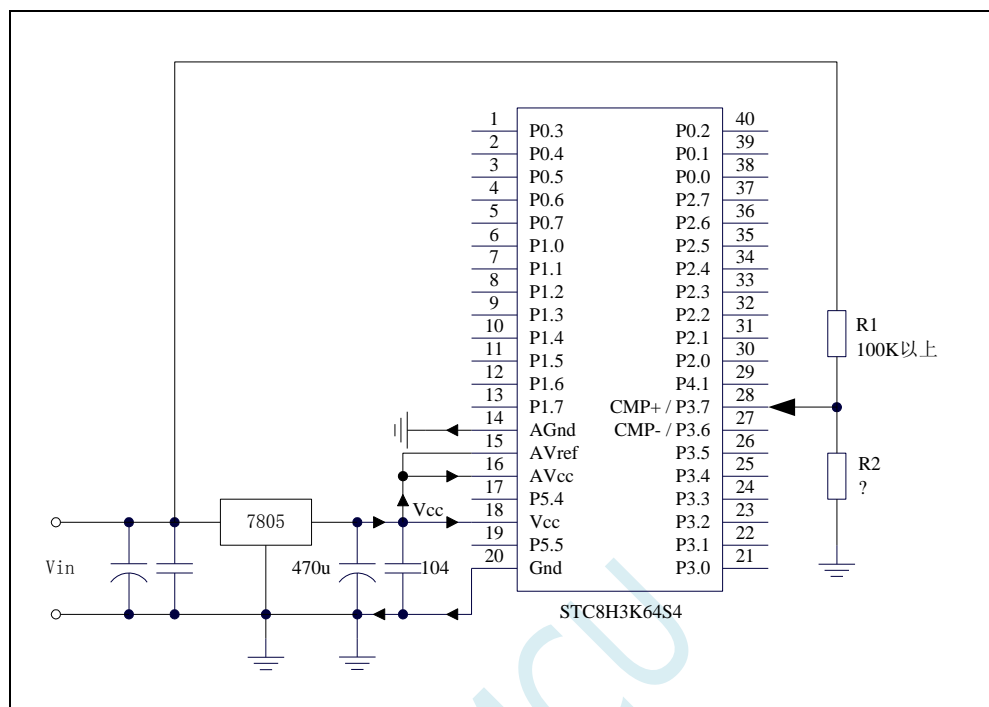
          MOV          CMPCR2, #00H
          MOV          CMPCR1, #00H
          ORL          CMPCR1, #02H          ;使能比较器输出
          ORL          CMPCR1, #80H          ;使能比较器模块

LOOP:
          JMP          LOOP

          END

```

## 17.4.7 比较器作外部掉电检测 (掉电过程中应及时保存用户数据到 EEPROM 中)

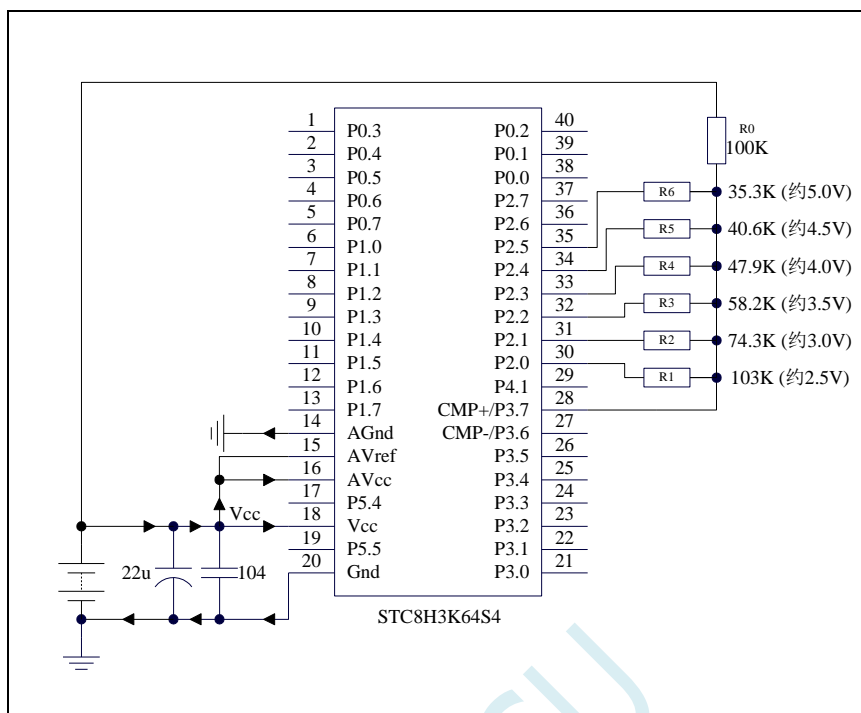


上图中电阻 R1 和 R2 对稳压块 7805 的前端电压进行分压，分压后的电压作为比较器 CMP+ 的外部输入与内部 1.19V 参考信号源进行比较。

一般当交流电在 220V 时，稳压块 7805 前端的直流电压为 11V，但当交流电压降到 160V 时，稳压块 7805 前端的直流电压为 8.5V。当稳压块 7805 前端的直流电压低于或等于 8.5V 时，该前端输入的直流电压被电阻 R1 和 R2 分压到比较器正极输入端 CMP+，CMP+ 端输入电压低于内部 1.19V 参考信号源，此时可产生比较器中断，这样在掉电检测时就有充足的时间将数据保存到 EEPROM 中。当稳压块 7805 前端的直流电压高于 8.5V 时，该前端输入的直流电压被电阻 R1 和 R2 分压到比较器正极输入端 CMP+，CMP+ 端输入电压高于内部 1.19V 参考信号源，此时 CPU 可继续正常工作。

内部 1.19V 参考信号源即为内部 BandGap 经过 OP 后的电压 REFV (芯片在出厂时，内部参考信号源调整为 1.19V)。具体的数值要通过读取内部 1.19V 参考信号源在内部 RAM 区或者 Flash 程序存储器 (ROM) 区所占用的地址的值获得。对于 STC8 系列，内部 1.19V 参考信号源值在 RAM 和 Flash 程序存储器 (ROM) 中的存储地址请参考 “[存储器中的特殊参数](#)” 章节

## 17.4.8 比较器检测工作电压（电池电压）



上图中，利用电阻分压的原理可以近似的测量出 MCU 的工作电压（选通的通道，MCU 的 IO 口输出低电平，端口电压值接近 Gnd，未选通的通道，MCU 的 IO 口输出开漏模式的高，不影响其他通道）。

比较器的负端选择内部 1.19V 参考信号源，正端选择通过电阻分压后输入到 CMP+管脚的电压值。

初始化时 P2.5~P2.0 口均设置为开漏模式，并输出高。首先 P2.0 口输出低电平，此时若 Vcc 电压低于 2.5V 则比较器的比较值为 0，反之若 Vcc 电压高于 2.5V 则比较器的比较值为 1；

若确定 Vcc 高于 2.5V，则将 P2.0 口输出高，P2.1 口输出低电平，此时若 Vcc 电压低于 3.0V 则比较器的比较值为 0，反之若 Vcc 电压高于 3.0V 则比较器的比较值为 1；

若确定 Vcc 高于 3.0V，则将 P2.1 口输出高，P2.2 口输出低电平，此时若 Vcc 电压低于 3.5V 则比较器的比较值为 0，反之若 Vcc 电压高于 3.5V 则比较器的比较值为 1；

若确定 Vcc 高于 3.5V，则将 P2.2 口输出高，P2.3 口输出低电平，此时若 Vcc 电压低于 4.0V 则比较器的比较值为 0，反之若 Vcc 电压高于 4.0V 则比较器的比较值为 1；

若确定 Vcc 高于 4.0V，则将 P2.3 口输出高，P2.4 口输出低电平，此时若 Vcc 电压低于 4.5V 则比较器的比较值为 0，反之若 Vcc 电压高于 4.5V 则比较器的比较值为 1；

若确定 Vcc 高于 4.5V，则将 P2.4 口输出高，P2.5 口输出低电平，此时若 Vcc 电压低于 5.0V 则比较器的比较值为 0，反之若 Vcc 电压高于 5.0V 则比较器的比较值为 1。

### C 语言代码

```
//测试工作频率为11.0592MHz
```

```
#include "stc8h.h"
#include "intrins.h"
```

```
void delay ()
{
```

```

    char i;

    for (i=0; i<20; i++);
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    unsigned char v;

    P2M0 = 0x3f; //P2.5~P2.0 初始化为开漏模式
    P2M1 = 0x3f;
    P2 = 0xff;

    CMPCR2 = 0x10; //比较器结果经过 16 个去抖时钟后输出
    CMPCR1 = 0x00;
    CMPCR1 &= ~0x08; //P3.7 为 CMP+ 输入脚
    CMPCR1 &= ~0x04; //内部 1.19V 参考信号源为 CMP- 输入脚
    CMPCR1 &= ~0x02; //禁止比较器输出
    CMPCR1 |= 0x80; //使能比较器模块

    while (1)
    {
        v = 0x00; //电压<2.5V
        P2 = 0xfe; //P2.0 输出 0
        delay();
        if (!(CMPCR1 & 0x01)) goto ShowVol;
        v = 0x01; //电压>2.5V
        P2 = 0xfd; //P2.1 输出 0
        delay();
        if (!(CMPCR1 & 0x01)) goto ShowVol;
        v = 0x03; //电压>3.0V
        P2 = 0xfb; //P2.2 输出 0
        delay();
        if (!(CMPCR1 & 0x01)) goto ShowVol;
        v = 0x07; //电压>3.5V
        P2 = 0xf7; //P2.3 输出 0
        delay();
        if (!(CMPCR1 & 0x01)) goto ShowVol;
        v = 0x0f; //电压>4.0V
        P2 = 0xef; //P2.4 输出 0
        delay();
        if (!(CMPCR1 & 0x01)) goto ShowVol;
        v = 0x1f; //电压>4.5V
        P2 = 0xdf; //P2.5 输出 0
    }
}

```

```
    delay();
    if (!(CMPCR1 & 0x01)) goto ShowVol;
    v = 0x3f; //电压>5.0V
ShowVol:
    P2 = 0xff;
    P0 = ~v;
}
}
```

汇编代码

;测试工作频率为 11.0592MHz

|        |      |                                      |
|--------|------|--------------------------------------|
| P_SW2  | DATA | 0BAH                                 |
| CMPCR1 | DATA | 0E6H                                 |
| CMPCR2 | DATA | 0E7H                                 |
| P2M0   | DATA | 096H                                 |
| P2M1   | DATA | 095H                                 |
| P1M1   | DATA | 091H                                 |
| P1M0   | DATA | 092H                                 |
| P0M1   | DATA | 093H                                 |
| P0M0   | DATA | 094H                                 |
| P2M1   | DATA | 095H                                 |
| P2M0   | DATA | 096H                                 |
| P3M1   | DATA | 0B1H                                 |
| P3M0   | DATA | 0B2H                                 |
| P4M1   | DATA | 0B3H                                 |
| P4M0   | DATA | 0B4H                                 |
| P5M1   | DATA | 0C9H                                 |
| P5M0   | DATA | 0CAH                                 |
|        | ORG  | 0000H                                |
|        | LJMP | MAIN                                 |
| MAIN:  | ORG  | 0100H                                |
|        | MOV  | SP, #5FH                             |
|        | ORL  | P_SW2, #80H ;使能访问 XFR                |
|        | MOV  | P0M0, #00H                           |
|        | MOV  | P0M1, #00H                           |
|        | MOV  | P1M0, #00H                           |
|        | MOV  | P1M1, #00H                           |
|        | MOV  | P2M0, #00H                           |
|        | MOV  | P2M1, #00H                           |
|        | MOV  | P3M0, #00H                           |
|        | MOV  | P3M1, #00H                           |
|        | MOV  | P4M0, #00H                           |
|        | MOV  | P4M1, #00H                           |
|        | MOV  | P5M0, #00H                           |
|        | MOV  | P5M1, #00H                           |
|        | MOV  | P2M0, #00111111B ;P2.5~P2.0 初始化为开漏模式 |
|        | MOV  | P2M1, #00111111B                     |
|        | MOV  | P2, #0FFH                            |
|        | MOV  | CMPCR2, #10H ;比较器结果经过 16 个去抖时钟后输出    |
|        | MOV  | CMPCR1, #00H                         |



```

        ANL      CMPCR1,#NOT 08H      ;P3.7 为CMP+输入脚
        ANL      CMPCR1,#NOT 04H      ;内部1.19V 参考信号源为CMP-输入脚
        ANL      CMPCR1,#NOT 02H      ;禁止比较器输出
        ORL      CMPCR1,#80H          ;使能比较器模块

LOOP:
        MOV      R0,#00000000B        ;电压<2.5V
        MOV      P2,#11111110B        ;P2.0 输出0
        CALL     DELAY
        MOV      A,CMPCR1
        JNB      ACC.0,SKIP
        MOV      R0,#00000001B        ;电压>2.5V
        MOV      P2,#11111101B        ;P2.1 输出0
        CALL     DELAY
        MOV      A,CMPCR1
        JNB      ACC.0,SKIP
        MOV      R0,#00000011B        ;电压>3.0V
        MOV      P2,#11111011B        P2.2 输出0
        CALL     DELAY
        MOV      A,CMPCR1
        JNB      ACC.0,SKIP
        MOV      R0,#00000111B        ;电压>3.5V
        MOV      P2,#11111011B        ;P2.3 输出0
        CALL     DELAY
        MOV      A,CMPCR1
        JNB      ACC.0,SKIP
        MOV      R0,#00001111B        ;电压>4.0V
        MOV      P2,#11110111B        ;P2.4 输出0
        CALL     DELAY
        MOV      A,CMPCR1
        JNB      ACC.0,SKIP
        MOV      R0,#00011111B        ;电压>4.5V
        MOV      P2,#11011111B        ;P2.5 输出0
        CALL     DELAY
        MOV      A,CMPCR1
        JNB      ACC.0,SKIP
        MOV      R0,#00111111B        ;电压>5.0V

SKIP:
        MOV      P2,#11111111B
        MOV      A,R0
        CPL      A
        MOV      P0,A                ;P0.5~P0.0 口显示电压
        JMP      LOOP

DELAY:
        MOV      R0,#20
        DJNZ     R0,$
        RET

END

```

## 18 IAP/EEPROM/DATA-FLASH

STC8H 系列单片机内部集成了大容量的 EEPROM。利用 ISP/IAP 技术可将内部 Data Flash 当 EEPROM，擦写次数在 10 万次以上。EEPROM 可分为若干个扇区，每个扇区包含 512 字节。

注意：EEPROM 的写操作只能将字节中的 1 写为 0，当需要将字节中的 0 写为 1，则必须执行扇区擦除操作。EEPROM 的读/写操作是以 1 字节为单位进行，而 EEPROM 擦除操作是以 1 扇区（512 字节）为单位进行，在执行擦除操作时，如果目标扇区中有需要保留的数据，则必须预先将这些数据读取到 RAM 中暂存，待擦除完成后再将保存的数据和需要更新的数据一起再写回 EEPROM/DATA-FLASH。

所以在使用 EEPROM 时，建议同一次修改的数据放在同一个扇区，不是同一次修改的数据放在不同的扇区，不一定要用满。数据存储器的擦除操作是按扇区进行的（每扇区 512 字节）。

EEPROM 可用于保存一些需要在应用过程中修改并且掉电不丢失的参数数据。在用户程序中，可以对 EEPROM 进行字节读/字节编程/扇区擦除操作。在工作电压偏低时，建议不要进行 EEPROM 操作，以免发送数据丢失的情况。

### 18.1 EEPROM 操作时间

- 读取 1 字节：4 个系统时钟（使用 MOVC 指令读取更方便快捷）
- 编程 1 字节：约 30~40us（实际的编程时间为 6~7.5us，但还需要加上状态转换时间和各种控制信号的 SETUP 和 HOLD 时间）
- 擦除 1 扇区（512 字节）：约 4~6ms

EEPROM 操作所需时间是硬件自动控制的，用户只需要正确设置 IAP\_TPS 寄存器即可。

**IAP\_TPS = 系统工作频率 / 1000000（小数部分四舍五入进行取整）**

例如：系统工作频率为 12MHz，则 IAP\_TPS 设置为 12

又例如：系统工作频率为 22.1184MHz，则 IAP\_TPS 设置为 22

再例如：系统工作频率为 5.5296MHz，则 IAP\_TPS 设置为 6

### 18.2 EEPROM 相关的寄存器

| 符号        | 描述            | 地址  | 位地址与符号 |      |             |          |    |    |          |    | 复位值       |
|-----------|---------------|-----|--------|------|-------------|----------|----|----|----------|----|-----------|
|           |               |     | B7     | B6   | B5          | B4       | B3 | B2 | B1       | B0 |           |
| IAP_DATA  | IAP 数据寄存器     | C2H |        |      |             |          |    |    |          |    | 1111,1111 |
| IAP_ADDRH | IAP 高地址寄存器    | C3H |        |      |             |          |    |    |          |    | 0000,0000 |
| IAP_ADDRL | IAP 低地址寄存器    | C4H |        |      |             |          |    |    |          |    | 0000,0000 |
| IAP_CMD   | IAP 命令寄存器     | C5H | -      | -    | -           | -        | -  | -  | CMD[1:0] |    | xxxx,xx00 |
| IAP_TRIG  | IAP 触发寄存器     | C6H |        |      |             |          |    |    |          |    | 0000,0000 |
| IAP_CONTR | IAP 控制寄存器     | C7H | IAPEN  | SWBS | SWRST       | CMD_FAIL | -  | -  | -        | -  | 0000,xxxx |
| IAP_TPS   | IAP 等待时间控制寄存器 | F5H | -      | -    | IAPTPS[5:0] |          |    |    |          |    | xx00,0000 |

#### 18.2.1 EEPROM 数据寄存器（IAP\_DATA）

| 符号       | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----------|-----|----|----|----|----|----|----|----|----|
| IAP_DATA | C2H |    |    |    |    |    |    |    |    |

在进行 EEPROM 的读操作时，命令执行完成后读出的 EEPROM 数据保存在 IAP\_DATA 寄存器中。

在进行 EEPROM 的写操作时, 在执行写命令前, 必须将待写入的数据存放在 IAP\_DATA 寄存器中, 再发送写命令。擦除 EEPROM 命令与 IAP\_DATA 寄存器无关。

## 18.2.2 EEPROM 地址寄存器 (IAP\_ADDR)

| 符号        | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----------|-----|----|----|----|----|----|----|----|----|
| IAP_ADDRH | C3H |    |    |    |    |    |    |    |    |
| IAP_ADDRL | C4H |    |    |    |    |    |    |    |    |

EEPROM 进行读、写、擦除操作的目标地址寄存器。IAP\_ADDRH 保存地址的高字节, IAP\_ADDRL 保存地址的低字节

## 18.2.3 EEPROM 命令寄存器 (IAP\_CMD)

| 符号      | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1       | B0 |
|---------|-----|----|----|----|----|----|----|----------|----|
| IAP_CMD | C5H | -  | -  | -  | -  | -  | -  | CMD[1:0] |    |

CMD[1:0]: 发送EEPROM操作命令

00: 空操作

01: 读 EEPROM 命令。读取目标地址所在的 1 字节。

10: 写 EEPROM 命令。写目标地址所在的 1 字节。**注意: 写操作只能将目标字节中的 1 写为 0, 而不能将 0 写为 1。一般当目标字节不为 FFH 时, 必须先擦除。**

11: 擦除 EEPROM。擦除目标地址所在的 1 页 (1 扇区/512 字节)。**注意: 擦除操作会一次擦除 1 个扇区 (512 字节), 整个扇区的内容全部变成 FFH。**

## 18.2.4 EEPROM 触发寄存器 (IAP\_TRIG)

| 符号       | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----------|-----|----|----|----|----|----|----|----|----|
| IAP_TRIG | C6H |    |    |    |    |    |    |    |    |

设置完成 EEPROM 读、写、擦除的命令寄存器、地址寄存器、数据寄存器以及控制寄存器后, 需要向触发寄存器 IAP\_TRIG 依次写入 5AH、A5H (顺序不能交换) 两个触发命令来触发相应的读、写、擦除操作。操作完成后, EEPROM 地址寄存器 IAP\_ADDRH、IAP\_ADDRL 和 EEPROM 命令寄存器 IAP\_CMD 的内容不变。如果接下来要对下一个地址的数据进行操作, 需手动更新地址寄存器 IAP\_ADDRH 和寄存器 IAP\_ADDRL 的值。

**注意:** 每次 EEPROM 操作时, 都要对 IAP\_TRIG 先写入 5AH, 再写入 A5H, 相应的命令才会生效。写完触发命令后, CPU 会处于 IDLE 等待状态, 直到相应的 IAP 操作执行完成后 CPU 才会从 IDLE 状态返回正常状态继续执行 CPU 指令。

## 18.2.5 EEPROM 控制寄存器 (IAP\_CONTR)

| 符号        | 地址  | B7    | B6   | B5    | B4       | B3 | B2 | B1 | B0 |
|-----------|-----|-------|------|-------|----------|----|----|----|----|
| IAP_CONTR | C7H | IAPEN | SWBS | SWRST | CMD_FAIL | -  | -  | -  | -  |

IAPEN: EEPROM操作使能控制位

0: 禁止 EEPROM 操作

1: 使能 EEPROM 操作

SWBS: 软件复位选择控制位, (需要与SWRST配合使用)

0: 软件复位后从用户代码开始执行程序

1: 软件复位后从系统 ISP 监控代码区开始执行程序

SWRST: 软件复位控制位

0: 无动作

1: 产生软件复位

CMD\_FAIL: EEPROM操作失败状态位, 需要软件清零

0: EEPROM 操作正确

1: EEPROM 操作失败

## 18.2.6 EEPROM 等待时间控制寄存器 (IAP\_TPS)

| 符号      | 地址  | B7 | B6 | B5          | B4 | B3 | B2 | B1 | B0 |
|---------|-----|----|----|-------------|----|----|----|----|----|
| IAP_TPS | F5H | -  | -  | IAPTPS[5:0] |    |    |    |    |    |

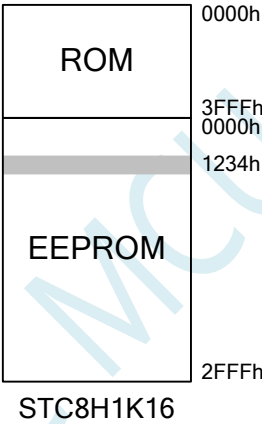
需要根据工作频率进行设置

若工作频率为12MHz, 则需要将IAP\_TPS设置为12; 若工作频率为24MHz, 则需要将IAP\_TPS设置为24, 其他频率以此类推。

### 18.3 EEPROM 大小及地址

STC8H 系列单片机内部均有助于保存用户数据的 EEPROM。内部的 EEPROM 有 3 操作方式：读、写和擦除，其中擦除操作是以扇区为单位进行操作，每扇区为 512 字节，即每执行一次擦除命令就会擦除一个扇区，而读数据和写数据都是以字节为单位进行操作的，即每执行一次读或者写命令时只能读出或者写入一个字节。

STC8H 系列单片机内部的 EEPROM 的访问方式有两种：IAP 方式和 MOV C 方式。IAP 方式可对 EEPROM 执行读、写、擦除操作，但 MOV C 只能对 EEPROM 进行读操作，而不能进行写和擦除操作。无论是使用 IAP 方式还是使用 MOV C 方式访问 EEPROM，首先都需要设置正确的目标地址。IAP 方式时，目标地址与 EEPROM 实际的物理地址是一致的，均是从地址 0000H 开始访问，但若使用 MOV C 指令进行读取 EEPROM 数据时，目标地址必须是在 EEPROM 实际的物理地址的基础上还有加上程序大小的偏移。下面以 STC8H1K16 这个型号为例，对目标地址进行详细说明：



STC8H1K16 的程序空间为 16K 字节（0000h~3FFFh），EEPROM 空间为 12K（0000h~2FFFh）。当需要对 EEPROM 物理地址 1234h 的单元进行读、写、擦除时，若使用 IAP 方式进行访问时，设置的目标地址为 1234h，即 IAP\_ADDRH 设置 12h，IAP\_ADDRL 设置 34h，然后设置相应的触发命令即可对 1234h 单元进行正确操作了。但若是使用 MOV C 方式读取 EEPROM 的 1234h 单元，则必须在 1234h 的基础上还有加上 ROM 空间的大小 4000h，即必须将 DPTR 设置为 5234h，然后才能使用 MOV C 指令进行读取。

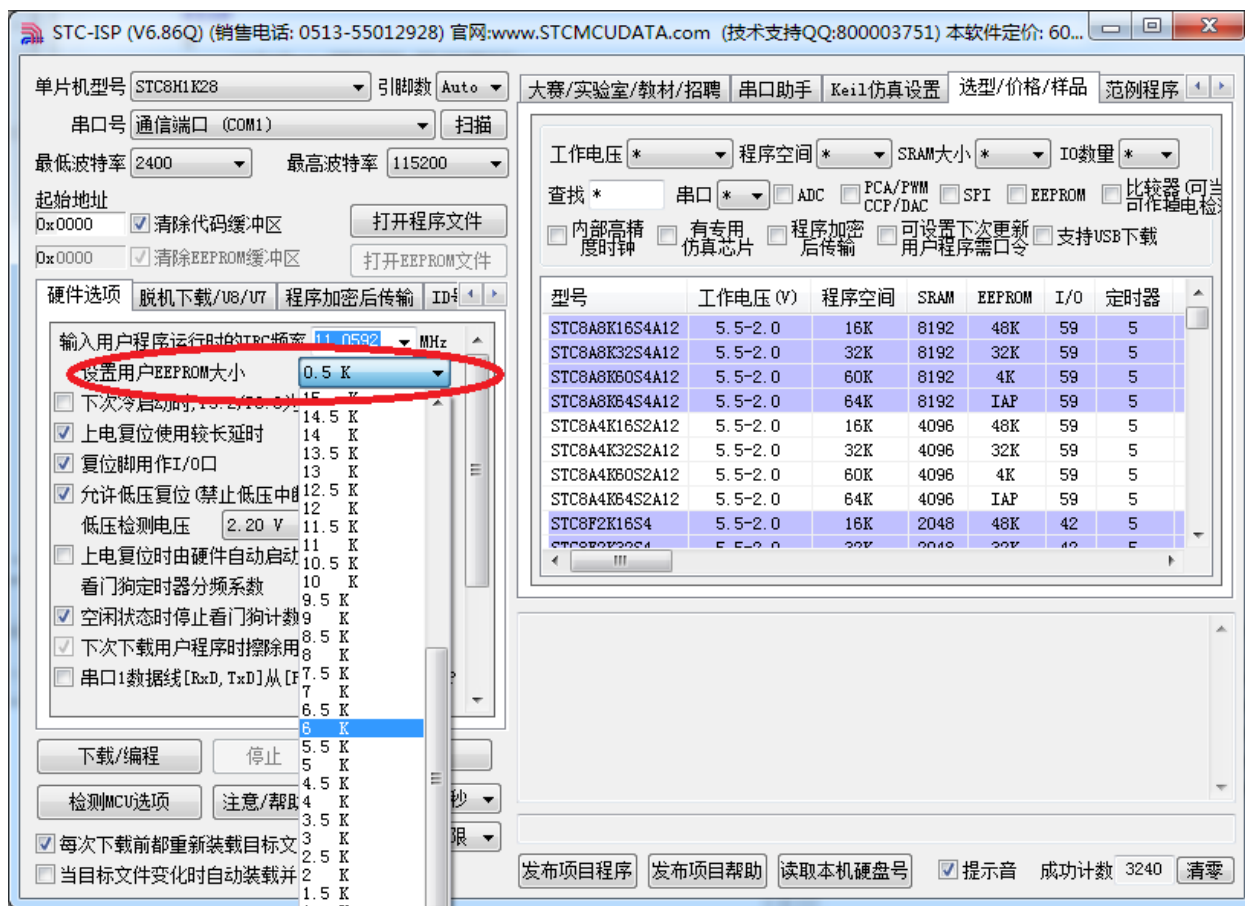
注意：由于擦除是以 512 字节为单位进行操作的，所以执行擦除操作时所设置的目标地址的低 9 位是无意义的。例如：执行擦除命令时，设置地址 1234H/1200H/1300H/13FFH，最终执行擦除的动作都是相同的，都是擦除 1200H~13FFH 这 512 字节。

不同型号内部 EEPROM 的大小及访问地址会存在差异，针对各个型号 EEPROM 的详细大小和地址请参考下表

| 型号        | 大小                   | 扇区 | IAP方式读/写/擦除 |       | MOV C读取 |       |
|-----------|----------------------|----|-------------|-------|---------|-------|
|           |                      |    | 起始地址        | 结束地址  | 起始地址    | 结束地址  |
| STC8H1K16 | 12K                  | 24 | 0000h       | 2FFFh | 4000h   | 6FFFh |
| STC8H1K24 | 4K                   | 8  | 0000h       | 0FFFh | 6000h   | 6FFFh |
| STC8H1K28 | 用户自定义 <sup>[1]</sup> |    |             |       |         |       |
| STC8H1K33 | 用户自定义 <sup>[1]</sup> |    |             |       |         |       |

|               |                      |    |       |       |       |       |
|---------------|----------------------|----|-------|-------|-------|-------|
| STC8H1K08     | 4K                   | 8  | 0000h | 0FFFh | 2000h | 2FFFh |
| STC8H1K12     | 用户自定义 <sup>[1]</sup> |    |       |       |       |       |
| STC8H1K17     | 用户自定义 <sup>[1]</sup> |    |       |       |       |       |
| STC8H3K32S4   | 32K                  | 64 | 0000h | 7FFFh | 8000h | FFFFh |
| STC8H3K48S4   | 16K                  | 32 | 0000h | 3FFFh | C000h | FFFFh |
| STC8H3K60S4   | 4K                   | 8  | 0000h | 0FFFh | F000h | FFFFh |
| STC8H3K64S4   | 用户自定义 <sup>[1]</sup> |    |       |       |       |       |
| STC8H3K32S2   | 32K                  | 64 | 0000h | 7FFFh | 8000h | FFFFh |
| STC8H3K48S2   | 16K                  | 32 | 0000h | 3FFFh | C000h | FFFFh |
| STC8H3K60S2   | 4K                   | 8  | 0000h | 0FFFh | F000h | FFFFh |
| STC8H3K64S2   | 用户自定义 <sup>[1]</sup> |    |       |       |       |       |
| STC8H8K32U    | 32K                  | 64 | 0000h | 7FFFh | 8000h | FFFFh |
| STC8H8K48U    | 16K                  | 32 | 0000h | 3FFFh | C000h | FFFFh |
| STC8H8K60U    | 4K                   | 8  | 0000h | 0FFFh | F000h | FFFFh |
| STC8H8K64U    | 用户自定义 <sup>[1]</sup> |    |       |       |       |       |
| STC8H4K32TL   | 32K                  | 64 | 0000h | 7FFFh | 8000h | FFFFh |
| STC8H4K48TL   | 16K                  | 32 | 0000h | 3FFFh | C000h | FFFFh |
| STC8H4K60TL   | 4K                   | 8  | 0000h | 0FFFh | F000h | FFFFh |
| STC8H4K64TL   | 用户自定义 <sup>[1]</sup> |    |       |       |       |       |
| STC8H4K32TLCD | 32K                  | 64 | 0000h | 7FFFh | 8000h | FFFFh |
| STC8H4K48TLCD | 16K                  | 32 | 0000h | 3FFFh | C000h | FFFFh |
| STC8H4K60TLCD | 4K                   | 8  | 0000h | 0FFFh | F000h | FFFFh |
| STC8H4K64TLCD | 用户自定义 <sup>[1]</sup> |    |       |       |       |       |
| STC8H1K08T    | 4K                   | 8  | 0000h | 0FFFh | 2000h | 2FFFh |
| STC8H1K12T    | 用户自定义 <sup>[1]</sup> |    |       |       |       |       |
| STC8H1K17T    | 用户自定义 <sup>[1]</sup> |    |       |       |       |       |

<sup>[1]</sup>: 这个为特殊型号, 这个型号的 EEPROM 大小是可用在 ISP 下载时用户自己设置的。如下图所示:



用户可用根据自己的需要在整个 FLASH 空间中规划出任意不超过 FLASH 大小的 EEPROM 空间，但需要注意：**EEPROM 总是从后向前进行规划的。**

例如：STC8H1K28 这个型号的 FLASH 为 28K，此时若用户想分出其中的 8K 作为 EEPROM 使用，则 EEPROM 的物理地址则为 28K 的最后 8K，物理地址为 5000h~6FFFh，当然，用户若使用 IAP 的方式进行访问，目标地址仍然从 0000h 开始，到 1FFFh 结束，当使用 MOVC 读取则需要从 5000h 开始，到 6FFFh 结束。**注意：STC8H1K28 这个型号的程序空间为 28K，即整个 28K 的范围均可运行程序，即使在 ISP 下载时将 28K 最后的 8K 设置为 EEPROM 使用，但这分配的 8K 空间仍然可以运行程序。**其它可自定义 EEPROM 大小的型号与此类似。

## 18.4 范例程序

### 18.4.1 EEPROM 基本操作

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void IapIdle()
```

```
{
    IAP_CONTR = 0;           //关闭 IAP 功能
    IAP_CMD = 0;             //清除命令寄存器
    IAP_TRIG = 0;            //清除触发寄存器
    IAP_ADDRH = 0x80;         //将地址设置到非 IAP 区域
    IAP_ADDRL = 0;
}
```

```
char IapRead(int addr)
```

```
{
    char dat;

    IAP_CONTR = 0x80;        //使能 IAP
    IAP_TPS = 12;             //设置等待参数 12MHz
    IAP_CMD = 1;              //设置 IAP 读命令
    IAP_ADDRL = addr;         //设置 IAP 低地址
    IAP_ADDRH = addr >> 8;    //设置 IAP 高地址
    IAP_TRIG = 0x5a;          //写触发命令(0x5a)
    IAP_TRIG = 0xa5;          //写触发命令(0xa5)
    _nop_();
    dat = IAP_DATA;           //读 IAP 数据
    IapIdle();                //关闭 IAP 功能

    return dat;
}
```

```
void IapProgram(int addr, char dat)
```

```
{
    IAP_CONTR = 0x80;        //使能 IAP
    IAP_TPS = 12;             //设置等待参数 12MHz
    IAP_CMD = 2;              //设置 IAP 写命令
    IAP_ADDRL = addr;         //设置 IAP 低地址
    IAP_ADDRH = addr >> 8;    //设置 IAP 高地址
    IAP_DATA = dat;           //写 IAP 数据
    IAP_TRIG = 0x5a;          //写触发命令(0x5a)
    IAP_TRIG = 0xa5;          //写触发命令(0xa5)
    _nop_();
    IapIdle();                //关闭 IAP 功能
}
```

```
void IapErase(int addr)
```

```
{
    IAP_CONTR = 0x80;        //使能 IAP
    IAP_TPS = 12;             //设置等待参数 12MHz
```



```

    IAP_CMD = 3; //设置IAP 擦除命令
    IAP_ADDRL = addr; //设置IAP 低地址
    IAP_ADDRH = addr >> 8; //设置IAP 高地址
    IAP_TRIG = 0x5a; //写触发命令(0x5a)
    IAP_TRIG = 0xa5; //写触发命令(0xa5)
    _nop_(); //
    IapIdle(); //关闭IAP 功能
}

void main()
{
    P_SW2 /= 0x80; //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    IapErase(0x0400);
    P0 = IapRead(0x0400); //P0=0xff
    IapProgram(0x0400, 0x12);
    P1 = IapRead(0x0400); //P1=0x12

    while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

|                  |             |             |
|------------------|-------------|-------------|
| <i>P_SW2</i>     | <i>DATA</i> | <i>0BAH</i> |
| <i>IAP_DATA</i>  | <i>DATA</i> | <i>0C2H</i> |
| <i>IAP_ADDRH</i> | <i>DATA</i> | <i>0C3H</i> |
| <i>IAP_ADDRL</i> | <i>DATA</i> | <i>0C4H</i> |
| <i>IAP_CMD</i>   | <i>DATA</i> | <i>0C5H</i> |
| <i>IAP_TRIG</i>  | <i>DATA</i> | <i>0C6H</i> |
| <i>IAP_CONTR</i> | <i>DATA</i> | <i>0C7H</i> |
| <i>IAP_TPS</i>   | <i>DATA</i> | <i>0F5H</i> |
| <i>P1M1</i>      | <i>DATA</i> | <i>091H</i> |
| <i>P1M0</i>      | <i>DATA</i> | <i>092H</i> |
| <i>P0M1</i>      | <i>DATA</i> | <i>093H</i> |
| <i>P0M0</i>      | <i>DATA</i> | <i>094H</i> |
| <i>P2M1</i>      | <i>DATA</i> | <i>095H</i> |
| <i>P2M0</i>      | <i>DATA</i> | <i>096H</i> |
| <i>P3M1</i>      | <i>DATA</i> | <i>0B1H</i> |
| <i>P3M0</i>      | <i>DATA</i> | <i>0B2H</i> |
| <i>P4M1</i>      | <i>DATA</i> | <i>0B3H</i> |
| <i>P4M0</i>      | <i>DATA</i> | <i>0B4H</i> |
| <i>P5M1</i>      | <i>DATA</i> | <i>0C9H</i> |

*P5M0 DATA 0CAH*

*ORG 0000H  
LJMP MAIN*

*ORG 0100H*

#### *IAP\_IDLE:*

*MOV IAP\_CONTR,#0 ;关闭 IAP 功能  
MOV IAP\_CMD,#0 ;清除命令寄存器  
MOV IAP\_TRIG,#0 ;清除触发寄存器  
MOV IAP\_ADDRH,#80H ;将地址设置到非 IAP 区域  
MOV IAP\_ADDRL,#0  
RET*

#### *IAP\_READ:*

*MOV IAP\_CONTR,#80H ;使能 IAP  
MOV IAP\_TPS,#12 ;设置等待参数 12MHz  
MOV IAP\_CMD,#1 ;设置 IAP 读命令  
MOV IAP\_ADDRL,DPL ;设置 IAP 低地址  
MOV IAP\_ADDRH,DPH ;设置 IAP 高地址  
MOV IAP\_TRIG,#5AH ;写触发命令(0x5a)  
MOV IAP\_TRIG,#0A5H ;写触发命令(0xa5)  
NOP  
MOV A,IAP\_DATA ;读取 IAP 数据  
LCALL IAP\_IDLE ;关闭 IAP 功能  
RET*

#### *IAP\_PROGRAM:*

*MOV IAP\_CONTR,#80H ;使能 IAP  
MOV IAP\_TPS,#12 ;设置等待参数 12MHz  
MOV IAP\_CMD,#2 ;设置 IAP 写命令  
MOV IAP\_ADDRL,DPL ;设置 IAP 低地址  
MOV IAP\_ADDRH,DPH ;设置 IAP 高地址  
MOV IAP\_DATA,A ;写 IAP 数据  
MOV IAP\_TRIG,#5AH ;写触发命令(0x5a)  
MOV IAP\_TRIG,#0A5H ;写触发命令(0xa5)  
NOP  
LCALL IAP\_IDLE ;关闭 IAP 功能  
RET*

#### *IAP\_ERASE:*

*MOV IAP\_CONTR,#80H ;使能 IAP  
MOV IAP\_TPS,#12 ;设置等待参数 12MHz  
MOV IAP\_CMD,#3 ;设置 IAP 擦除命令  
MOV IAP\_ADDRL,DPL ;设置 IAP 低地址  
MOV IAP\_ADDRH,DPH ;设置 IAP 高地址  
MOV IAP\_TRIG,#5AH ;写触发命令(0x5a)  
MOV IAP\_TRIG,#0A5H ;写触发命令(0xa5)  
NOP  
LCALL IAP\_IDLE ;关闭 IAP 功能  
RET*

#### *MAIN:*

*MOV SP,#5FH  
ORL P\_SW2,#80H ;使能访问 XFR  
  
MOV P0M0,#00H  
MOV P0M1,#00H*

```

MOV      P1M0, #00H
MOV      P1M1, #00H
MOV      P2M0, #00H
MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      DPTR, #0400H
LCALL    IAP_ERASE
MOV      DPTR, #0400H
LCALL    IAP_READ
MOV      P0, A                      ;P0=0FFH
MOV      DPTR, #0400H
MOV      A, #12H
LCALL    IAP_PROGRAM
MOV      DPTR, #0400H
LCALL    IAP_READ
MOV      P1, A                      ;P1=12H

SJMP     $

END

```

## 18.4.2 使用 MOVC 读取 EEPROM

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define IAP_OFFSET      0x4000H           //STC8H1K16

```

```

void IapIdle()
{

```

```

    IAP_CONTR = 0;           //关闭 IAP 功能
    IAP_CMD = 0;             //清除命令寄存器
    IAP_TRIG = 0;            //清除触发寄存器
    IAP_ADDRH = 0x80;         //将地址设置到非 IAP 区域
    IAP_ADDRL = 0;
}

```

```

char IapRead(int addr)
{

```

```

    addr += IAP_OFFSET;      //使用 MOVC 读取 EEPROM 需要加上相应的偏移
    return *(char code *) (addr); //使用 MOVC 读取数据
}

```

```

void IapProgram(int addr, char dat)
{

```

```

    IAP_CONTR = 0x80;         //使能 IAP
    IAP_TPS = 12;             //设置等待参数 12MHz
}

```

```

    IAP_CMD = 2;                //设置IAP 写命令
    IAP_ADDRL = addr;           //设置IAP 低地址
    IAP_ADDRH = addr >> 8;      //设置IAP 高地址
    IAP_DATA = dat;             //写IAP 数据
    IAP_TRIG = 0x5a;            //写触发命令(0x5a)
    IAP_TRIG = 0xa5;            //写触发命令(0xa5)
    _nop_();
    IapIdle();                  //关闭IAP 功能
}

void IapErase(int addr)
{
    IAP_CONTR = 0x80;          //使能IAP
    IAP_TPS = 12;              //设置等待参数 12MHz
    IAP_CMD = 3;               //设置IAP 擦除命令
    IAP_ADDRL = addr;          //设置IAP 低地址
    IAP_ADDRH = addr >> 8;     //设置IAP 高地址
    IAP_TRIG = 0x5a;           //写触发命令(0x5a)
    IAP_TRIG = 0xa5;           //写触发命令(0xa5)
    _nop_();                   //
    IapIdle();                 //关闭IAP 功能
}

void main()
{
    P_SW2 /= 0x80; //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    IapErase(0x0400);
    P0 = IapRead(0x0400);      //P0=0xff
    IapProgram(0x0400, 0x12);
    P1 = IapRead(0x0400);      //P1=0x12

    while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|           |      |      |
|-----------|------|------|
| P_SW2     | DATA | 0BAH |
| IAP_DATA  | DATA | 0C2H |
| IAP_ADDRH | DATA | 0C3H |
| IAP_ADDRL | DATA | 0C4H |
| IAP_CMD   | DATA | 0C5H |
| IAP_TRIG  | DATA | 0C6H |

```

IAP_CONTR  DATA    0C7H
IAP_TPS     DATA    0F5H

IAP_OFFSET EQU      4000H                ;STC8H1K16

P1M1        DATA    091H
P1M0        DATA    092H
P0M1        DATA    093H
P0M0        DATA    094H
P2M1        DATA    095H
P2M0        DATA    096H
P3M1        DATA    0B1H
P3M0        DATA    0B2H
P4M1        DATA    0B3H
P4M0        DATA    0B4H
P5M1        DATA    0C9H
P5M0        DATA    0CAH

                ORG      0000H
                LJMP     MAIN

                ORG      0100H

IAP_IDLE:
    MOV        IAP_CONTR,#0                ;关闭 IAP 功能
    MOV        IAP_CMD,#0                  ;清除命令寄存器
    MOV        IAP_TRIG,#0                 ;清除触发寄存器
    MOV        IAP_ADDRH,#80H              ;将地址设置到非 IAP 区域
    MOV        IAP_ADDRL,#0
    RET

IAP_READ:
    MOV        A,#LOW IAP_OFFSET            ;使用 MOVC 读取 EEPROM 需要加上相应的偏移
    ADD        A,DPL
    MOV        DPL,A
    MOV        A,@HIGH IAP_OFFSET
    ADDC       A,DPH
    MOV        DPH,A
    CLR        A
    MOVC       A,@A+DPTR                    ;使用 MOVC 读取数据
    RET

IAP_PROGRAM:
    MOV        IAP_CONTR,#80H                ;使能 IAP
    MOV        IAP_TPS,#12                    ;设置等待参数 12MHz
    MOV        IAP_CMD,#2                    ;设置 IAP 写命令
    MOV        IAP_ADDRL,DPL                  ;设置 IAP 低地址
    MOV        IAP_ADDRH,DPH                  ;设置 IAP 高地址
    MOV        IAP_DATA,A                    ;写 IAP 数据
    MOV        IAP_TRIG,#5AH                  ;写触发命令(0x5a)
    MOV        IAP_TRIG,#0A5H                 ;写触发命令(0xa5)
    NOP
    LCALL      IAP_IDLE                      ;关闭 IAP 功能
    RET

IAP_ERASE:
    MOV        IAP_CONTR,#80H                ;使能 IAP
    MOV        IAP_TPS,#12                    ;设置等待参数 12MHz
    MOV        IAP_CMD,#3                    ;设置 IAP 擦除命令

```

```

MOV      IAP_ADDRL,DPL      ;设置 IAP 低地址
MOV      IAP_ADDRH,DPH      ;设置 IAP 高地址
MOV      IAP_TRIG,#5AH      ;写触发命令(0x5a)
MOV      IAP_TRIG,#0A5H     ;写触发命令(0xa5)
NOP
LCALL    IAP_IDLE            ;关闭 IAP 功能
RET

```

MAIN:

```

MOV      SP,#5FH
ORL      P_SW2,#80H          ;使能访问 XFR

MOV      P0M0,#00H
MOV      P0M1,#00H
MOV      P1M0,#00H
MOV      P1M1,#00H
MOV      P2M0,#00H
MOV      P2M1,#00H
MOV      P3M0,#00H
MOV      P3M1,#00H
MOV      P4M0,#00H
MOV      P4M1,#00H
MOV      P5M0,#00H
MOV      P5M1,#00H

MOV      DPTR,#0400H
LCALL    IAP_ERASE
MOV      DPTR,#0400H
LCALL    IAP_READ
MOV      P0,A                ;P0=0FFH
MOV      DPTR,#0400H
MOV      A,#12H
LCALL    IAP_PROGRAM
MOV      DPTR,#0400H
LCALL    IAP_READ
MOV      P1,A                ;P1=12H

SJMP     $

END

```

### 18.4.3 使用串口送出 EEPROM 数据

#### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define FOSC      11059200UL
#define BRT       (65536 - (FOSC / 115200+2) / 4)

```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```

void UartInit()
{

```

```

    SCON = 0x5a;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x15;
}

void UartSend(char dat)
{
    while (!TI);
    TI = 0;
    SBUF = dat;
}

void IapIdle()
{
    IAP_CONTR = 0;           //关闭IAP 功能
    IAP_CMD = 0;             //清除命令寄存器
    IAP_TRIG = 0;            //清除触发寄存器
    IAP_ADDRH = 0x80;        //将地址设置到非IAP 区域
    IAP_ADDRL = 0;
}

char IapRead(int addr)
{
    char dat;

    IAP_CONTR = 0x80;        //使能IAP
    IAP_TPS = 12;             //设置等待参数 12MHz
    IAP_CMD = 1;              //设置IAP 读命令
    IAP_ADDRL = addr;         //设置IAP 低地址
    IAP_ADDRH = addr >> 8;    //设置IAP 高地址
    IAP_TRIG = 0x5a;          //写触发命令(0x5a)
    IAP_TRIG = 0xa5;          //写触发命令(0xa5)
    _nop_();
    dat = IAP_DATA;           //读IAP 数据
    IapIdle();                //关闭IAP 功能

    return dat;
}

void IapProgram(int addr, char dat)
{
    IAP_CONTR = 0x80;        //使能IAP
    IAP_TPS = 12;             //设置等待参数 12MHz
    IAP_CMD = 2;              //设置IAP 写命令
    IAP_ADDRL = addr;         //设置IAP 低地址
    IAP_ADDRH = addr >> 8;    //设置IAP 高地址
    IAP_DATA = dat;           //写IAP 数据
    IAP_TRIG = 0x5a;          //写触发命令(0x5a)
    IAP_TRIG = 0xa5;          //写触发命令(0xa5)
    _nop_();
    IapIdle();                //关闭IAP 功能
}

void IapErase(int addr)
{
    IAP_CONTR = 0x80;        //使能IAP
    IAP_TPS = 12;             //设置等待参数 12MHz
    IAP_CMD = 3;              //设置IAP 擦除命令

```

```

    IAP_ADDRL = addr;
    IAP_ADDRH = addr >> 8;
    IAP_TRIG = 0x5a;
    IAP_TRIG = 0xa5;
    _nop_();
    IapIdle();
}

void main()
{
    P_SW2 /= 0x80;

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    IapErase(0x0400);
    UartSend(IapRead(0x0400));
    IapProgram(0x0400, 0x12);
    UartSend(IapRead(0x0400));

    while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

|                  |             |             |
|------------------|-------------|-------------|
| <i>P_SW2</i>     | <i>DATA</i> | <i>0BAH</i> |
| <i>AUXR</i>      | <i>DATA</i> | <i>8EH</i>  |
| <i>T2H</i>       | <i>DATA</i> | <i>0D6H</i> |
| <i>T2L</i>       | <i>DATA</i> | <i>0D7H</i> |
| <i>IAP_DATA</i>  | <i>DATA</i> | <i>0C2H</i> |
| <i>IAP_ADDRH</i> | <i>DATA</i> | <i>0C3H</i> |
| <i>IAP_ADDRL</i> | <i>DATA</i> | <i>0C4H</i> |
| <i>IAP_CMD</i>   | <i>DATA</i> | <i>0C5H</i> |
| <i>IAP_TRIG</i>  | <i>DATA</i> | <i>0C6H</i> |
| <i>IAP_CONTR</i> | <i>DATA</i> | <i>0C7H</i> |
| <i>IAP_TPS</i>   | <i>DATA</i> | <i>0F5H</i> |
| <i>P1M1</i>      | <i>DATA</i> | <i>091H</i> |
| <i>P1M0</i>      | <i>DATA</i> | <i>092H</i> |
| <i>P0M1</i>      | <i>DATA</i> | <i>093H</i> |
| <i>P0M0</i>      | <i>DATA</i> | <i>094H</i> |
| <i>P2M1</i>      | <i>DATA</i> | <i>095H</i> |
| <i>P2M0</i>      | <i>DATA</i> | <i>096H</i> |
| <i>P3M1</i>      | <i>DATA</i> | <i>0B1H</i> |



```

P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN

          ORG          0100H

UART_INIT:
          MOV          SCON,#5AH
          MOV          T2L,#0E8H          ;65536-11059200/115200/4=0FFE8H
          MOV          T2H,#0FFH
          MOV          AUXR,#15H
          RET

UART_SEND:
          JNB          TI,$
          CLR          TI
          MOV          SBUF,A
          RET

IAP_IDLE:
          MOV          IAP_CONTR,#0          ;关闭 IAP 功能
          MOV          IAP_CMD,#0          ;清除命令寄存器
          MOV          IAP_TRIG,#0          ;清除触发寄存器
          MOV          IAP_ADDRH,#80H          ;将地址设置到非 IAP 区域
          MOV          IAP_ADDRL,#0
          RET

IAP_READ:
          MOV          IAP_CONTR,#80H          ;使能 IAP
          MOV          IAP_TPS,#12          ;设置等待参数 12MHz
          MOV          IAP_CMD,#1          ;设置 IAP 读命令
          MOV          IAP_ADDRL,DPL          ;设置 IAP 低地址
          MOV          IAP_ADDRH,DPH          ;设置 IAP 高地址
          MOV          IAP_TRIG,#5AH          ;写触发命令(0x5a)
          MOV          IAP_TRIG,#0A5H          ;写触发命令(0xa5)
          NOP
          MOV          A,IAP_DATA          ;读取 IAP 数据
          LCALL         IAP_IDLE          ;关闭 IAP 功能
          RET

IAP_PROGRAM:
          MOV          IAP_CONTR,#80H          ;使能 IAP
          MOV          IAP_TPS,#12          ;设置等待参数 12MHz
          MOV          IAP_CMD,#2          ;设置 IAP 写命令
          MOV          IAP_ADDRL,DPL          ;设置 IAP 低地址
          MOV          IAP_ADDRH,DPH          ;设置 IAP 高地址
          MOV          IAP_DATA,A          ;写 IAP 数据
          MOV          IAP_TRIG,#5AH          ;写触发命令(0x5a)
          MOV          IAP_TRIG,#0A5H          ;写触发命令(0xa5)
          NOP
          LCALL         IAP_IDLE          ;关闭 IAP 功能
          RET

IAP_ERASE:

```

```

MOV      IAP_CONTR,#80H      ;使能 IAP
MOV      IAP_TPS,#12         ;设置等待参数 12MHz
MOV      IAP_CMD,#3          ;设置 IAP 擦除命令
MOV      IAP_ADDRL,DPL       ;设置 IAP 低地址
MOV      IAP_ADDRH,DPH       ;设置 IAP 高地址
MOV      IAP_TRIG,#5AH       ;写触发命令(0x5a)
MOV      IAP_TRIG,#0A5H      ;写触发命令(0xa5)
NOP
LCALL    IAP_IDLE            ;关闭 IAP 功能
RET

```

MAIN:

```

MOV      SP,#5FH
ORL      P_SW2,#80H          ;使能访问 XFR

MOV      P0M0,#00H
MOV      P0M1,#00H
MOV      P1M0,#00H
MOV      P1M1,#00H
MOV      P2M0,#00H
MOV      P2M1,#00H
MOV      P3M0,#00H
MOV      P3M1,#00H
MOV      P4M0,#00H
MOV      P4M1,#00H
MOV      P5M0,#00H
MOV      P5M1,#00H

LCALL    UART_INIT
MOV      DPTR,#0400H
LCALL    IAP_ERASE
MOV      DPTR,#0400H
LCALL    IAP_READ
LCALL    UART_SEND
MOV      DPTR,#0400H
MOV      A,#12H
LCALL    IAP_PROGRAM
MOV      DPTR,#0400H
LCALL    IAP_READ
LCALL    UART_SEND

SJMP     $

END

```

## 18.4.4 串口 1 读写 EEPROM-带 MOVC 读

### C 语言代码 (main.c)

//测试工作频率为 11.0592MHz

/\* 本程序经过测试完全正常, 不提供电话技术支持, 如不能理解, 请自行补充相关基础 \*/

/\*\*\*\*\*\* 本程序功能说明 \*\*\*\*\*

STC8G 系列EEPROM 通用测试程序

请先别修改程序, 直接下载"02-串口 1 读写 EEPROM-带 MOVC 读"里面的"UART-EEPROM.hex"测试, 下载时选择主频 11.0592MHZ.

PC 串口设置: 波特率 115200, 8, n, 1.

对EEPROM 做扇区擦除、写入 64 字节、读出 64 字节的操作。

命令例子:

E 0 对EEPROM 进行扇区擦除操作, E 表示擦除, 数字 0 为 0 扇区(十进制, 0~126, 看具体 IC).

W 0 对EEPROM 进行写入操作, W 表示写入, 数字 0 为 0 扇区(十进制, 0~126, 看具体 IC). 从扇区的开始地址连续写 64 字节

R 0 对EEPROM 进行IAP 读出操作, R 表示读出, 数字 0 为 0 扇区(十进制, 0~126, 看具体 IC). 从扇区的开始地址连续读 64 字节

M 0 对EEPROM 进行MOVC 读出操作(操作地址为扇区\*512+偏移地址) 数字 0 为 0 扇区(十进制, 0~126, 看具体 IC). 从扇区的开始地址连续读 64 字节

注意: 为了通用, 程序不识别扇区是否有效, 用户自己根据具体的型号来决定。

日期: 2019-6-10

\*\*\*\*\*/

```
#include "config.H"
```

```
#include "EEPROM.h"
```

```
#define Baudrate1 115200L
```

```
#define UART1_BUF_LENGTH 10
```

```
#define EEADDR_OFFSET (8 * 1024) //定义EEPROM 用MOVC 访问时加的偏移量,
//等于FLASH ROM 的大小对于IAP 或IRC 开头的,
//则偏移量必须为0
```

```
#define TimeOutSet1 5
```

\*\*\*\*\* 本地常量声明 \*\*\*\*\*/

```
u8 code T_Strings[]={"去年今日此门中，人面桃花相映红。人面不知何处去，桃花依旧笑春风。"};
```

\*\*\*\*\* 本地变量声明 \*\*\*\*\*/

```
u8 xdatatmp[70];
```

```
u8 xdataRX1_Buffer[UART1_BUF_LENGTH];
```

```
u8 RX1_Cnt;
```

```
u8 RX1_TimeOut;
```

```
bit B_TX1_Busy;
```

\*\*\*\*\* 本地函数声明 \*\*\*\*\*/

```
void UART1_config(void);
```

```
void TX1_write2buff(u8 dat); //写入发送缓冲
```

```
void PrintString1(u8 *puts); //发送一个字符串
```

\*\*\*\*\* 外部函数和变量声明 \*\*\*\*\*/

\*\*\*\*\*/

```
u8 CheckData(u8 dat)
```

```
{
    if((dat >= '0') && (dat <= '9')) return (dat-'0');
    if((dat >= 'A') && (dat <= 'F')) return (dat-'A'+10);
    if((dat >= 'a') && (dat <= 'f')) return (dat-'a'+10);
    return 0xff;
}
```

```
u16 GetAddress(void)
```

```
{
    u16 address;
    u8 i;
```

```

address = 0;
if(RX1_Cnt < 3)    return 65535;           //error
if(RX1_Cnt <= 5)   //5 个字节以内是扇区操作，十进制
                  //支持命令:  E 0, E 12, E 120
                  //           W 0, W 12, W 120
                  //           R 0, R 12, R 120

{
    for(i=2; i<RX1_Cnt; i++)
    {
        if(CheckData(RX1_Buffer[i]) > 9)
            return 65535;           //error
        address = address * 10 + CheckData(RX1_Buffer[i]);
    }
    if(address < 124)               //限制在0~123 扇区
    {
        address <<= 9;
        return (address);
    }
}
else if(RX1_Cnt == 8)              //8 个字节直接地址操作，十六进制
                                  //支持命令: E 0x1234, W 0x12b3, R 0x0A00
{
    if((RX1_Buffer[2] == '0') && ((RX1_Buffer[3] == 'x') || (RX1_Buffer[3] == 'X')))
    {
        for(i=4; i<8; i++)
        {
            if(CheckData(RX1_Buffer[i]) > 0x0F)
                return 65535;       //error
            address = (address << 4) + CheckData(RX1_Buffer[i]);
        }
        if(address < 63488)
            return (address);       //限制在0~123 扇区
    }
}

return 65535;                     //error
}

//=====
// 函数: void  delay_ms(u8 ms)
// 描述: 延时函数。
// 参数: ms,要延时的ms 数, 这里只支持1~255ms. 自动适应主时钟
// 返回: none.
// 版本: VER1.0
// 日期: 2013-4-1
// 备注:
//=====
void delay_ms(u8 ms)
{
    u16 i;
    do
    {
        i = MAIN_Fosc / 10000;
        while(--i) ;
    }while(--ms);
}

```

**//使用MOVC 读EEPROM**

```

void EEPROM_MOVC_read_n(u16 EE_address, u8 *DataAddress, u16 number)
{
    u8 code *pc;

    pc = EE_address + EEADDR_OFFSET;
    do
    {
        *DataAddress = *pc;           //读出的数据
        DataAddress++;
        pc++;
    }while(--number);
}

/***** 主函数 *****/
void main(void)
{
    u8 i;
    u16 addr;

    UART1_config();                 // 选择波特率 2: 使用 Timer2 做波特率
                                    //其它值: 使用 Timer1 做波特率
    EA = 1;                         //允许总中断

    PrintString1("STC8 系列MCU 用串口1 测试EEPROM 程序!\r\n"); //UART1 发送一个字符串

    while(1)
    {
        delay_ms(1);
        if(RX1_TimeOut > 0)        //超时计数
        {
            if(--RX1_TimeOut == 0)
            {
                if(RX1_Buffer[1] == ' ')
                {
                    addr = GetAddress();
                    if(addr < 63488) //限制在 0~123 扇区
                    {
                        if(RX1_Buffer[0] == 'E') //PC 请求擦除一个扇区
                        {
                            EEPROM_SectorErase(addr);
                            PrintString1("扇区擦除完成!\r\n");
                        }

                        else if(RX1_Buffer[0] == 'W') //PC 请求写入 EEPROM 64 字节数据
                        {
                            EEPROM_write_n(addr,T_Strings,64);
                            PrintString1("写入操作完成!\r\n");
                        }

                        else if(RX1_Buffer[0] == 'R') //PC 请求返回 64 字节EEPROM 数据
                        {
                            PrintString1("IAP 读出的数据如下:\r\n");
                            EEPROM_read_n(addr,tmp,64);
                            for(i=0; i<64; i++)
                                TX1_write2buff(tmp[i]); //将数据返回给串口
                            TX1_write2buff(0x0d);
                            TX1_write2buff(0x0a);
                        }

                        else if(RX1_Buffer[0] == 'M') //PC 请求返回 64 字节EEPROM 数据

```

```

        {
            PrintString1("MOVC 读出的数据如下:\r\n");
            EEPROM_MOVC_read_n(addr,tmp,64);
            for(i=0; i<64; i++)
                TX1_write2buff(tmp[i]);    //将数据返回给串口
            TX1_write2buff(0x0d);
            TX1_write2buff(0x0a);
        }
        else PrintString1("命令错误!\r\n");
    }
    else PrintString1("命令错误!\r\n");
}

RX1_Cnt = 0;
}
}
}
}

/***** 发送一个字节 *****/
void TX1_write2buff(u8 dat)                //写入发送缓冲
{
    B_TX1_Busy = 1;                        //标志发送忙
    SBUF = dat;                            //发送一个字节
    while(B_TX1_Busy);                    //等待发送完毕
}

//=====
// 函数: void PrintString1(u8 *puts)
// 描述: 串口1 发送字符串函数。
// 参数: puts: 字符串指针。
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====
void PrintString1(u8 *puts)                //发送一个字符串
{
    for (; *puts != 0; puts++)            //遇到停止符0 结束
    {
        TX1_write2buff(*puts);
    }
}

//=====
// 函数: void UART1_config(void)
// 描述: UART1 初始化函数。
// 参数: none.
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====
void UART1_config(void)
{
    TRI = 0;
    AUXR &= ~0x01;                        //S1 BRT Use Timer1;
    AUXR /= (1<<6);                       //Timer1 set as 1T mode
}

```

```

    TMOD &= ~(1<<6);                //Timer1 set As Timer
    TMOD &= ~0x30;                  //Timer1_16bitAutoReload;
    TH1 = (u8)((65536L-(MAIN_Fosc / 4) / Baudrate1) >> 8);
    TL1 = (u8)(65536L-(MAIN_Fosc / 4) / Baudrate1);
    ET1 = 0;                        // 禁止 Timer1 中断
    INT_CLKO &= ~0x02;              // Timer1 不输出高速时钟
    TRI = 1;                        // 运行 Timer1

    SI_USE_P30P31(); P3n_standard(0x03); //切换到 P3.0 P3.1
    //SI_USE_P36P37(); P3n_standard(0xc0); //切换到 P3.6 P3.7
    //SI_USE_P16P17(); P1n_standard(0xc0); //切换到 P1.6 P1.7

    SCON = (SCON & 0x3f) / 0x40;    //UART1 模式: 0x00: 同步移位输出,
    //                                0x40: 8 位数据,可变波特率
    //                                0x80: 9 位数据,固定波特率
    //                                0xc0: 9 位数据,可变波特率

// PS = 1;                        //高优先级中断
ES = 1;                          //允许中断
REN = 1;                         //允许接收

    B_TX1_Busy = 0;
    RX1_Cnt = 0;
}

//=====
// 函数: void UART1_int (void) interrupt UART1_VECTOR
// 描述: UART1 中断函数。
// 参数: none.
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====
void UART1_int (void) interrupt 4
{
    if(RI)
    {
        RI = 0;
        if(RX1_Cnt >= UART1_BUF_LENGTH)
            RX1_Cnt = 0;                //防溢出
        RX1_Buffer[RX1_Cnt++] = SBUF;
        RX1_TimeOut = TimeOutSet1;
    }

    if(TI)
    {
        TI = 0;
        B_TX1_Busy = 0;
    }
}

```

## C 语言代码 (EEPROM.c)

//测试工作频率为 11.0592MHz

// 本程序是 STC 系列的内置 EEPROM 读写程序。

```

#include "config.h"
#include "eeprom.h"

```

```
//=====
// 函数: void IAP_Disable(void)
// 描述: 禁止访问ISP/IAP.
// 参数: non.
// 返回: non.
// 版本: V1.0, 2012-10-22
//=====
void DisableEEPROM(void)
{
    IAP_CONTR = 0;                //禁止 ISP/IAP 操作
    IAP_TPS = 0;
    IAP_CMD = 0;                  //去除 ISP/IAP 命令
    IAP_TRIG = 0;                  //防止 ISP/IAP 命令误触发
    IAP_ADDRH = 0xff;              //清0 地址高字节
    IAP_ADDRL = 0xff;              //清0 地址低字节, 指向非EEPROM 区, 防止误操作
}

//=====
// 函数: void EEPROM_read_n(u16 EE_address,u8 *DataAddress,u16 number)
// 描述: 从指定EEPROM 首地址读出n 个字节放指定的缓冲.
// 参数: EE_address: 读出EEPROM 的首地址.
//       DataAddress: 读出数据放缓冲的首地址.
//       number: 读出的字节长度.
// 返回: non.
// 版本: V1.0, 2012-10-22
//=====
void EEPROM_read_n(u16 EE_address,u8 *DataAddress,u16 number)
{
    EA = 0;                        //禁止中断
    IAP_CONTR = IAP_EN;            //允许 ISP/IAP 操作
    IAP_TPS = (u8)(MAIN_Fosc / 1000000L); //工作频率设置
    IAP_READ();                    //送字节读命令, 命令不需改变时, 不需重新送命令
    do
    {
        IAP_ADDRH = EE_address / 256; //送地址高字节 (地址需要改变时才需重新送地址)
        IAP_ADDRL = EE_address % 256; //送地址低字节
        IAP_TRIG();                    //先送 5AH, 再送 A5H 到ISP/IAP 触发寄存器,
        //每次都需要如此
        //送完 A5H 后, ISP/IAP 命令立即被触发启动
        //CPU 等待 IAP 完成后, 才会继续执行程序。

        _nop_();
        _nop_();
        _nop_();
        *DataAddress = IAP_DATA;        //读出的数据送往
        EE_address++;
        DataAddress++;
    }while(--number);

    DisableEEPROM();
    EA = 1;                            //重新允许中断
}

/***** 扇区擦除函数 *****/
//=====
// 函数: void EEPROM_SectorErase(u16 EE_address)
// 描述: 把指定地址的EEPROM 扇区擦除.
// 参数: EE_address: 要擦除的扇区EEPROM 的地址.
// 返回: non.
```



// 版本: V1.0, 2013-5-10

//=====

void EEPROM\_SectorErase(u16 EE\_address)

```

{
    EA = 0;                                     //禁止中断
                                                //只有扇区擦除，没有字节擦除，512 字节/扇区。
                                                //扇区中任意一个字节地址都是扇区地址。
    IAP_ADDRH = EE_address / 256;               //送扇区地址高字节（地址需要改变时才需重新送地址）
    IAP_ADDRL = EE_address % 256;               //送扇区地址低字节
    IAP_CONTR = IAP_EN;                         //允许 ISP/IAP 操作
    IAP_TPS = (u8)(MAIN_Fosc / 1000000L);       //工作频率设置
    IAP_ERASE();                                //送扇区擦除命令，命令不需改变时，不需重新送命令
    IAP_TRIG();
    _nop_();
    _nop_();
    _nop_();
    DisableEEPROM();
    EA = 1;                                     //重新允许中断
}

```

//=====

// 函数: void EEPROM\_write\_n(u16 EE\_address,u8 \*DataAddress,u16 number)

// 描述: 把缓冲的n 个字节写入指定首地址的EEPROM.

// 参数: EE\_address: 写入 EEPROM 的首地址.

// DataAddress: 写入源数据的缓冲的首地址.

// number: 写入的字节长度.

// 返回: non.

// 版本: V1.0, 2012-10-22

//=====

void EEPROM\_write\_n(u16 EE\_address,u8 \*DataAddress,u16 number)

```

{
    EA = 0;                                     //禁止中断

    IAP_CONTR = IAP_EN;                         //允许 ISP/IAP 操作
    IAP_TPS = (u8)(MAIN_Fosc / 1000000L);       //工作频率设置
    IAP_WRITE();                                //送字节写命令，命令不需改变时，不需重新送命令
    do
    {
        IAP_ADDRH = EE_address / 256;           //送地址高字节（地址需要改变时才需重新送地址）
        IAP_ADDRL = EE_address % 256;           //送地址低字节
        IAP_DATA = *DataAddress;                //送数据到 IAP_DATA，只有数据改变时才需重新送
        IAP_TRIG();
        _nop_();
        _nop_();
        _nop_();
        EE_address++;
        DataAddress++;
    }while(--number);

    DisableEEPROM();
    EA = 1;                                     //重新允许中断
}

```

## 18.4.5 口令擦除写入-多扇区备份-串口 1 操作

C 语言代码 (main.c)

//测试工作频率为 11.0592MHz

/\* 本程序经过测试完全正常, 不提供电话技术支持, 如不能理解, 请自行补充相关基础 \*/

/\*\*\*\*\*\* 本程序功能说明 \*\*\*\*\*\*/

STC8G 系列 STC8H 系列 STC8C 系列 EEPROM 通用测试程序, 演示多扇区备份、有扇区错误则用正确扇区数据写入、全部扇区错误(比如第一次运行程序)则写入默认值。

每次写都写入 3 个扇区, 即冗余备份。

每个扇区写一条记录, 写入完成后读出保存的数据和校验值跟源数据和校验值比较, 并从串口 1(P3.0 P3.1)返回结果(正确或错误提示)。

每条记录自校验, 64 字节数据, 2 字节校验值, 校验值 = 64 字节数据累加和 ^ 0x5555, ^0x5555 是为了保证写入的 66 个数据不全部为 0。

如果有扇区错误, 则将正确扇区的数据写入错误扇区, 如果 3 个扇区都错误, 则均写入默认值。

擦除、写入、读出操作前均需要设置口令, 如果口令不对则退出操作, 并且每次退出操作都会清除口令。

下载时选择主频 11.0592MHZ。

PC 串口设置: 波特率 115200,8,n,1。

对 EEPROM 做扇区擦除、写入 64 字节、读出 64 字节的操作。

命令例子:

使用串口助手发单个字符, 大小写均可。

发 E 或 e: 对 EEPROM 进行扇区擦除操作, E 表示擦除, 会擦除扇区 0、1、2。

发 W 或 w: 对 EEPROM 进行写入操作, W 表示写入, 会写入扇区 0、1、2, 每个扇区连续写 64 字节, 扇区 0 写入 0x0000~0x003f, 扇区 1 写入 0x0200~0x023f, 扇区 2 写入 0x0400~0x043f。

发 R 或 r: 对 EEPROM 进行读出操作, R 表示读出, 会读出扇区 0、1、2, 每个扇区连续读 64 字节, 扇区 0 读出 0x0000~0x003f, 扇区 1 读出 0x0200~0x023f, 扇区 2 读出 0x0400~0x043f。

注意: 为了通用, 程序不识别扇区是否有效, 用户自己根据具体的型号来决定。

日期: 2021-11-5

\*\*\*\*\*/

#include "config.H"

#include "EEPROM.h"

#define Baudrate1 115200L

/\*\*\*\*\*\* 本地常量声明 \*\*\*\*\*\*/

u8 code T\_StringD[]={"去年今日此门中, 人面桃花相映红。人面不知何处去, 桃花依旧笑春风。"};

u8 code T\_StringW[]={"横看成岭侧成峰, 远近高低各不同。不识庐山真面目, 只缘身在此山中。"};

/\*\*\*\*\*\* 本地变量声明 \*\*\*\*\*\*/

u8 xdatatmp[70]; //通用数据

u8 xdataSaveTmp[70]; //要写入的数组

bit B\_TX1\_Busy;

u8 cmd; //串口单字符命令

/\*\*\*\*\*\* 本地函数声明 \*\*\*\*\*\*/

void UART1\_config(void);

void TX1\_write2buff(u8 dat); //写入发送缓冲

void PrintString1(u8 \*puts); //发送一个字符串

```
/****** 外部函数和变量声明 *****/
```

```
/****** 读取EEPROM 记录, 并且校验, 返回校验结果 0 为正确, 1 为错误 *****/
```

```
u8 ReadRecord(u16 addr)
{
    u8 i;
    u16 ChckSum;                //计算的累加和
    u16 j;                      //读取的累加和

    for(i=0; i<66; i++) tmp[i] = 0; //清除缓冲
    PassWord = D_PASSWORD;        //给定口令
    EEPROM_read_n(addr,tmp,66);    //读出扇区0
    for(ChckSum=0, i=0; i<64; i++)
        ChckSum += tmp[i];        //计算累加和
    j = ((u16)tmp[64]<<8) + (u16)tmp[65]; //读取记录的累加和
    j ^= 0x5555;                  //隔位取反, 避免全0
    if(ChckSum != j) return 1;    //累加和错误, 返回1
    return 0;                    //累加和正确, 返回0
}
```

```
/****** 写入EEPROM 记录, 并且校验, 返回校验结果 0 为正确, 1 为错误 *****/
```

```
u8 SaveRecord(u16 addr)
{
    u8 i;
    u16 ChckSum;                //计算的累加和

    for(ChckSum=0, i=0; i<64; i++)
        ChckSum += SaveTmp[i];    //计算累加和
    ChckSum ^= 0x5555;            //隔位取反, 避免全0
    SaveTmp[64] = (u8)(ChckSum >> 8);
    SaveTmp[65] = (u8)ChckSum;

    PassWord = D_PASSWORD;        //给定口令
    EEPROM_SectorErase(addr);      //擦除一个扇区
    PassWord = D_PASSWORD;        //给定口令
    EEPROM_write_n(addr, SaveTmp, 66); //写入扇区

    for(i=0; i<66; i++)
        tmp[i] = 0;              //清除缓冲
    PassWord = D_PASSWORD;        //给定口令
    EEPROM_read_n(addr,tmp,66);    //读出扇区0
    for(i=0; i<66; i++)            //数据比较
    {
        if(SaveTmp[i] != tmp[i])
            return 1;            //数据有错误, 返回1
    }
    return 0;                    //累加和正确, 返回0
}
```

```
/****** 主函数 *****/
```

```
void main(void)
{
    u8 i;
    u8 status;                  //状态

    UART1_config();             //选择波特率 2: 使用 Timer2 做波特率
                                //其它值: 使用 Timer1 做波特率
    EA = 1;                    //允许总中断
}
```

```
PrintStringI("STC8G-8H-8C 系列MCU 用串口1 测试EEPROM 程序!\r\n"); //UART1 发送一个字符串

//上电读取 3 个扇区并校验 如果有扇区错误则将正确的
//扇区写入错误扇区, 如果3 个扇区都错误 则写入默认值

status = 0;
if(ReadRecord(0x0000) == 0) //读扇区0
{
    status |= 0x01; //正确则标记 status.0=1
    for(i=0; i<64; i++)
        SaveTmp[i] = tmp[i]; //保存在写缓冲
}
if(ReadRecord(0x0200) == 0) //读扇区1
{
    status |= 0x02; //正确则标记 status.1=1
    for(i=0; i<64; i++)
        SaveTmp[i] = tmp[i]; //保存在写缓冲
}
if(ReadRecord(0x0400) == 0) //读扇区2
{
    status |= 0x04; //正确则标记 status.2=1
    for(i=0; i<64; i++)
        SaveTmp[i] = tmp[i]; //保存在写缓冲
}

if(status == 0) //所有扇区都错误 则写入默认值
{
    for(i=0; i<64; i++)
        SaveTmp[i] = T_StringD[i]; //读取默认值
}
else PrintStringI("上电读取 3 个扇区数据均正确!\r\n"); //UART1 发送一个字符串提示

if((status & 0x01) == 0) //扇区0 错误 则写入默认值
{
    if(SaveRecord(0x0000) == 0)
        PrintStringI("写入扇区0 正确!\r\n"); //写入记录0 扇区正确
    else
        PrintStringI("写入扇区0 错误!\r\n"); //写入记录0 扇区错误
}
if((status & 0x02) == 0) //扇区1 错误 则写入默认值
{
    if(SaveRecord(0x0200) == 0)
        PrintStringI("写入扇区1 正确!\r\n"); //写入记录1 扇区正确
    else
        PrintStringI("写入扇区1 错误!\r\n"); //写入记录1 扇区错误
}
if((status & 0x04) == 0) //扇区2 错误 则写入默认值
{
    if(SaveRecord(0x0400) == 0)
        PrintStringI("写入扇区2 正确!\r\n"); //写入记录2 扇区正确
    else
        PrintStringI("写入扇区2 错误!\r\n"); //写入记录2 扇区错误
}

while(1)
{
    if(cmd != 0) //有串口命令
    {
```

```

if((cmd >= 'a') && (cmd <= 'z'))
    cmd -= 0x20; //小写转大写

if(cmd == 'E') //PC 请求擦除一个扇区
{
    PassWord = D_PASSWORD; //给定口令
    EEPROM_SectorErase(0x0000); //擦除一个扇区
    PassWord = D_PASSWORD; //给定口令
    EEPROM_SectorErase(0x0200); //擦除一个扇区
    PassWord = D_PASSWORD; //给定口令
    EEPROM_SectorErase(0x0400); //擦除一个扇区
    PrintString1("扇区擦除完成!\r\n");
}

else if(cmd == 'W') //PC 请求写入 EEPROM 64 字节数据
{
    for(i=0; i<64; i++)
        SaveTmp[i] = T_StringW[i]; //写入数值
    if(SaveRecord(0x0000) == 0)
        PrintString1("写入扇区0 正确!\r\n"); //写入记录0 扇区正确
    else
        PrintString1("写入扇区0 错误!\r\n"); //写入记录0 扇区错误
    if(SaveRecord(0x0200) == 0)
        PrintString1("写入扇区1 正确!\r\n"); //写入记录1 扇区正确
    else
        PrintString1("写入扇区1 错误!\r\n"); //写入记录1 扇区错误
    if(SaveRecord(0x0400) == 0)
        PrintString1("写入扇区2 正确!\r\n"); //写入记录2 扇区正确
    else
        PrintString1("写入扇区2 错误!\r\n"); //写入记录2 扇区错误
}

else if(cmd == 'R') //PC 请求返回 64 字节EEPROM 数据
{
    if(ReadRecord(0x0000) == 0) //读出扇区0 的数据
    {
        PrintString1("读出扇区0 的数据如下:\r\n");
        for(i=0; i<64; i++)
            TX1_write2buff(tmp[i]); //将数据返回给串口
        TX1_write2buff(0x0d); //回车换行
        TX1_write2buff(0x0a);
    }
    else PrintString1("读出扇区0 的数据错误!\r\n");

    if(ReadRecord(0x0200) == 0) //读出扇区1 的数据
    {
        PrintString1("读出扇区1 的数据如下:\r\n");
        for(i=0; i<64; i++)
            TX1_write2buff(tmp[i]); //将数据返回给串口
        TX1_write2buff(0x0d); //回车换行
        TX1_write2buff(0x0a);
    }
    else PrintString1("读出扇区1 的数据错误!\r\n");

    if(ReadRecord(0x0400) == 0) //读出扇区2 的数据
    {
        PrintString1("读出扇区2 的数据如下:\r\n");
        for(i=0; i<64; i++)
            TX1_write2buff(tmp[i]); //将数据返回给串口
    }
}

```

```

        TX1_write2buff(0x0d);           //回车换行
        TX1_write2buff(0x0a);
    }
    else PrintString1("读出扇区2 的数据错误!\r\n");
}
else PrintString1("命令错误!\r\n");
cmd = 0;
}
}
}
}
}

/***** 发送一个字节 *****/
void TX1_write2buff(u8 dat)             //写入发送缓冲
{
    B_TX1_Busy = 1;                     //标志发送忙
    SBUF = dat;                          //发送一个字节
    while(B_TX1_Busy);                  //等待发送完毕
}

//=====
// 函数: void PrintString1(u8 *puts)
// 描述: 串口1 发送字符串函数。
// 参数: puts: 字符串指针
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====
void PrintString1(u8 *puts)             //发送一个字符串
{
    for (; *puts != 0; puts++)          //遇到停止符0 结束
    {
        TX1_write2buff(*puts);
    }
}

//=====
// 函数: void UART1_config(void)
// 描述: UART1 初始化函数。
// 参数: none.
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====
void UART1_config(void)
{
    TR1 = 0;
    AUXR &= ~0x01;                      //S1 BRT Use Timer1;
    AUXR |= (1<<6);                     //Timer1 set as IT mode
    TMOD &= ~(1<<6);                     //Timer1 set As Timer
    TMOD &= ~0x30;                       //Timer1_16bitAutoReload;
    TH1 = (u8)((65536L-(MAIN_Fosc / 4) / Baudrate1) >> 8);
    TL1 = (u8)(65536L-(MAIN_Fosc / 4) / Baudrate1);
    ET1 = 0;                             // 禁止 Timer1 中断
    INT_CLKO &= ~0x02;                  // Timer1 不输出高速时钟
    TR1 = 1;                             // 运行 Timer1
}

```

```

    SI_USE_P30P31(); P3n_standard(0x03); //切换到 P3.0 P3.1
    //SI_USE_P36P37(); P3n_standard(0xc0); //切换到 P3.6 P3.7
    //SI_USE_P16P17(); P1n_standard(0xc0); //切换到 P1.6 P1.7

    SCON = (SCON & 0x3f) / 0x40; //UART1 模式 0x00: 同步移位输出
                                //          0x40: 8 位数据,可变波特率
                                //          0x80: 9 位数据,固定波特率
                                //          0xc0: 9 位数据,可变波特率

// PS = 1; //高优先级中断
// ES = 1; //允许中断
// REN = 1; //允许接收

    B_TX1_Busy = 0;
}

//=====
// 函数: void UART1_int (void) interrupt UART1_VECTOR
// 描述: UART1 中断函数。
// 参数: none.
// 返回: none.
// 版本: VER1.0
// 日期: 2014-11-28
// 备注:
//=====
void UART1_int (void) interrupt 4
{
    if(RI)
    {
        RI = 0;
        cmd = SBUF;
    }

    if(TI)
    {
        TI = 0;
        B_TX1_Busy = 0;
    }
}

```

## C 语言代码 (EEPROM.c)

//测试工作频率为 11.0592MHz

// 本程序是 STC 系列的内置 EEPROM 读写程序。

```

#include "config.h"
#include "EEPROM.h"

```

```

u32      PassWord; //擦除 写入时需要的口令

```

```

//=====
// 函数: void IAP_Disable(void)
// 描述: 禁止访问 ISP/IAP.
// 参数: non.
// 返回: non.
// 版本: V1.0, 2012-10-22
//=====
void DisableEEPROM(void)
{

```



```

    IAP_CONTR = 0;                                //禁止 ISP/IAP 操作
    IAP_TPS    = 0;
    IAP_CMD    = 0;                                //去除 ISP/IAP 命令
    IAP_TRIG   = 0;                                //防止 ISP/IAP 命令误触发
    IAP_ADDRH  = 0xff;                             //清0 地址高字节
    IAP_ADDRL  = 0xff;                             //清0 地址低字节, 指向非EEPROM 区, 防止误操作
}

//=====================================================
// 函数: void EEPROM_read_n(u16 EE_address,u8 *DataAddress,u16 number)
// 描述: 从指定EEPROM 首地址读出n 个字节放指定的缓冲.
// 参数: EE_address: 读出EEPROM 的首地址.
//       DataAddress: 读出数据放缓冲的首地址.
//       number:      读出的字节长度.
// 返回: non.
// 版本: V1.0, 2012-10-22
//=====================================================
void EEPROM_read_n(u16 EE_address,u8 *DataAddress,u16 number)
{
    if(PassWord == D_PASSWORD)                    //口令正确才会操作EEPROM
    {
        EA = 0;                                  //禁止中断
        IAP_CONTR = IAP_EN;                       //允许 ISP/IAP 操作
        IAP_TPS = (u8)(MAIN_Fosc / 1000000L);     //工作频率设置
        IAP_READ();                               //送字节读命令, 命令不需改变时, 不需重新送命令
        do
        {
            IAP_ADDRH = EE_address / 256;         //送地址高字节 (地址需要改变时才需重新送地址)
            IAP_ADDRL = EE_address % 256;         //送地址低字节
            if(PassWord == D_PASSWORD)             //口令正确才触发操作
            {
                IAP_TRIG = 0x5A;                   //先送 5AH, 再送 55H 到ISP/IAP 触发寄存器,
                                                    //每次都需要如此
                IAP_TRIG = 0x55;                   //送完 55H 后, ISP/IAP 命令立即被触发启动
                                                    //CPU 等待 IAP 完成后, 才会继续执行程序。
            }
            _nop_();
            _nop_();
            _nop_();
            *DataAddress = IAP_DATA;               //读出的数据送往
            EE_address++;
            DataAddress++;
        }while(--number);

        DisableEEPROM();
        EA = 1;                                   //重新允许中断
    }
    PassWord = 0;                                //清除口令
}

/***** 扇区擦除函数 *****/
//=====================================================
// 函数: void EEPROM_SectorErase(u16 EE_address)
// 描述: 把指定地址的EEPROM 扇区擦除.
// 参数: EE_address: 要擦除的扇区EEPROM 的地址.
// 返回: non.
// 版本: V1.0, 2013-5-10
//=====================================================
void EEPROM_SectorErase(u16 EE_address)
{

```



```

if(PassWord == D_PASSWORD)                                //口令正确才会操作EEPROM
{
    EA = 0;                                                //禁止中断
                                                            //只有扇区擦除，没有字节擦除，512 字节/扇区。
                                                            //扇区中任意一个字节地址都是扇区地址。
    IAP_ADDRH = EE_address / 256;                          //送扇区地址高字节（地址需要改变时才需重新送地址）
    IAP_ADDRL = EE_address % 256;                          //送扇区地址低字节
    IAP_CONTR = IAP_EN;                                    //允许 ISP/IAP 操作
    IAP_TPS = (u8)(MAIN_Fosc / 1000000L);                  //工作频率设置
    IAP_ERASE();                                            //送扇区擦除命令，命令不需改变时，不需重新送命令
    if(PassWord == D_PASSWORD)                              //口令正确才触发操作
    {
        IAP_TRIG = 0x5A;                                  //先送 5AH，再送 A5H 到 ISP/IAP 触发寄存器，
                                                            //每次都需要如此
        IAP_TRIG = 0xA5;                                  //送完 A5H 后，ISP/IAP 命令立即被触发启动
                                                            //CPU 等待 IAP 完成后，才会继续执行程序。
    }
    _nop_();
    _nop_();
    _nop_();
    DisableEEPROM();
    EA = 1;                                                //重新允许中断
}
PassWord = 0;                                              //清除口令
}

//=====================================================
// 函数: void EEPROM_write_n(u16 EE_address,u8 *DataAddress,u16 number)
// 描述: 把缓冲的 n 个字节写入指定首地址的 EEPROM.
// 参数: EE_address: 写入 EEPROM 的首地址.
//       DataAddress: 写入源数据的缓冲的首地址.
//       number:      写入的字节长度.
// 返回: non.
// 版本: V1.0, 2012-10-22
//=====================================================
void EEPROM_write_n(u16 EE_address,u8 *DataAddress,u16 number)
{
    if(PassWord == D_PASSWORD)                              //口令正确才会操作EEPROM
    {
        EA = 0;                                            //禁止中断

        IAP_CONTR = IAP_EN;                                //允许 ISP/IAP 操作
        IAP_TPS = (u8)(MAIN_Fosc / 1000000L);              //工作频率设置
        IAP_WRITE();                                       //送字节写命令，命令不需改变时，不需重新送命令
        do
        {
            IAP_ADDRH = EE_address / 256;                  //送地址高字节（地址需要改变时才需重新送地址）
            IAP_ADDRL = EE_address % 256;                  //送地址低字节
            IAP_DATA = *DataAddress;                        //送数据到 IAP_DATA，只有数据改变时才需重新送
            if(PassWord == D_PASSWORD)                      //口令正确才触发操作
            {
                IAP_TRIG = 0x5A;                            //先送 5AH，再送 A5H 到 ISP/IAP 触发寄存器，
                                                            //每次都需要如此
                IAP_TRIG = 0xA5;                            //送完 A5H 后，ISP/IAP 命令立即被触发启动
                                                            //CPU 等待 IAP 完成后，才会继续执行程序。
            }
            _nop_();
            _nop_();
            _nop_();
            EE_address++;
            DataAddress++;
        } while (number--);
    }
}

```

```
    }while(--number);

    DisableEEPROM();
    EA = 1;                //重新允许中断
}
PassWord = 0;            //清除口令
}
```

---

STC MCU

## 19 ADC 模数转换, 内部 1.19V 参考信号源(BGV)

| 产品线                    | ADC 分辨率 | ADC 通道数 | ADCEXCFG |
|------------------------|---------|---------|----------|
| STC8H1K08 系列           | 10 位    | 9 通道    |          |
| STC8H1K28 系列           | 10 位    | 12 通道   |          |
| STC8H3K64S4 系列         | 12 位    | 12 通道   |          |
| STC8H3K64S2 系列         | 12 位    | 12 通道   |          |
| STC8H8K64U 系列 A 版本     | 12 位    | 15 通道   |          |
| STC8H8K64U 系列 B/C/D 版本 | 12 位    | 15 通道   | ●        |
| STC8H4K64TL 系列         | 12 位    | 15 通道   | ●        |
| STC8H4K64TLCD 系列       | 12 位    | 15 通道   | ●        |
| STC8H1K08T 系列          | 12 位    | 15 通道   | ●        |

STC8H 系列单片机内部集成了一个 10 位/12 位高速 A/D 转换器。ADC 的时钟频率为系统频率 2 分频再经过用户设置的分频系数进行再次分频（ADC 的工作时钟频率范围为 SYSclk/2/1 到 SYSclk/2/16）。

STC8H 系列的 ADC 最快速度：**12 位 ADC 为 800K（每秒进行 80 万次 ADC 转换），10 位 ADC 为 500K（每秒进行 50 万次 ADC 转换）**

ADC 转换结果的数据格式有两种：左对齐和右对齐。可方便用户程序进行读取和引用。

**注意：**ADC 的第 15 通道是专门测量内部 1.19V 参考信号源的通道，参考信号源值出厂时校准为 1.19V，由于制造误差以及测量误差，导致实际的内部参考信号源相比 1.19V，大约有±1%的误差。如果用户需要知道每一颗芯片的准确内部参考信号源值，可外接精准参考信号源，然后利用 ADC 的第 15 通道进行测量标定。ADC\_VRef+脚外接参考电源时，可利用 ADC 的第 15 通道可以反推 ADC\_VRef+脚外接参考电源的电压；如将 ADC\_VREF+短接到 MCU-VCC，就可以反推 MCU-VCC 的电压。

**如果芯片有 ADC 的外部参考电源管脚 ADC\_VRef+，则一定不能浮空，必须接外部参考电源或者直接连到 VCC**

### 19.1 ADC 相关的寄存器

| 符号        | 描述            | 地址  | 位地址与符号    |           |          |           |              |    |    |    | 复位值       |
|-----------|---------------|-----|-----------|-----------|----------|-----------|--------------|----|----|----|-----------|
|           |               |     | B7        | B6        | B5       | B4        | B3           | B2 | B1 | B0 |           |
| ADC_CONTR | ADC 控制寄存器     | BCH | ADC_POWER | ADC_START | ADC_FLAG | ADC_EPWMT | ADC_CHS[3:0] |    |    |    | 000x,0000 |
| ADC_RES   | ADC 转换结果高位寄存器 | BDH |           |           |          |           |              |    |    |    | 0000,0000 |
| ADC_RES_L | ADC 转换结果低位寄存器 | BEH |           |           |          |           |              |    |    |    | 0000,0000 |
| ADCCFG    | ADC 配置寄存器     | DEH | -         | -         | RESFMT   | -         | SPEED[3:0]   |    |    |    | xx0x,0000 |

| 符号     | 描述          | 地址    | 位地址与符号  |             |    |              |    |    |    |    | 复位值       |
|--------|-------------|-------|---------|-------------|----|--------------|----|----|----|----|-----------|
|        |             |       | B7      | B6          | B5 | B4           | B3 | B2 | B1 | B0 |           |
| ADCTIM | ADC 时序控制寄存器 | FEA8H | CSSETUP | CSHOLD[1:0] |    | SMPDUTY[4:0] |    |    |    |    | 0010,1010 |

## 19.1.1 ADC 控制寄存器 (ADC\_CONTR), PWM 触发 ADC 控制

| 符号        | 地址  | B7        | B6        | B5       | B4        | B3           | B2 | B1 | B0 |
|-----------|-----|-----------|-----------|----------|-----------|--------------|----|----|----|
| ADC_CONTR | BCH | ADC_POWER | ADC_START | ADC_FLAG | ADC_EPWMT | ADC_CHS[3:0] |    |    |    |

ADC\_POWER: ADC 电源控制位

0: 关闭 ADC 电源

1: 打开 ADC 电源。

建议进入空闲模式和掉电模式前将 ADC 电源关闭, 以降低功耗

**特别注意:**

1、给 MCU 的内部 ADC 模块电源打开后, 需等待约 1ms, 等 MCU 内部的 ADC 电源稳定后再让 ADC 工作;

2、适当加长对外部信号的采样时间, 就是对 ADC 内部采样保持电容的充电或放电时间, 时间够, 内部才能和外部电势相等。

ADC\_START: ADC 转换启动控制位。写入 1 后开始 ADC 转换, 转换完成后硬件自动将此位清零。

0: 无影响。即使 ADC 已经开始转换工作, 写 0 也不会停止 A/D 转换。

1: 开始 ADC 转换, 转换完成后硬件自动将此位清零。

ADC\_FLAG: ADC 转换结束标志位。当 ADC 完成一次转换后, 硬件会自动将此位置 1, 并向 CPU 提出中断请求。此标志位必须软件清零。

**ADC\_EPWMT: 使能 PWM 实时触发 ADC 功能。详情请参考 16 位高级 PWM 定时器章节**

ADC\_CHS[3:0]: ADC 模拟通道选择位

(注意: 被选择为 ADC 输入通道的 I/O 口, 必须设置 PxM0/PxM1 寄存器将 I/O 口模式设置为高阻输入模式。另外如果 MCU 进入掉电模式/时钟停振模式后, 仍需要使能 ADC 通道, 则需要设置 PxIE 寄存器关闭数字输入通道, 以防止外部模拟输入信号忽高忽低而产生额外的功耗)

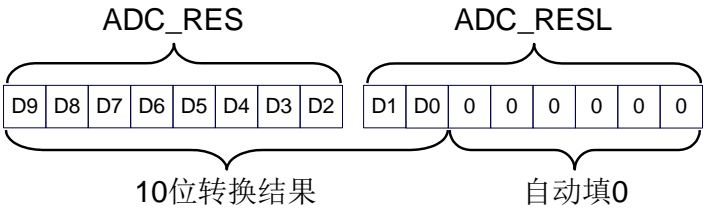
| ADC_CHS | ADC 通道     | STC8H1K28 系列 | STC8H1K08 系列 | STC8H3K64S4<br>STC8H3K64S2 系列 | STC8H8K64U<br>STC8H4K64TL 系列 | STC8H4K64TLCD 系列 | STC8H1K08T 系列 |
|---------|------------|--------------|--------------|-------------------------------|------------------------------|------------------|---------------|
| 0000    | ADC0       | P1.0         | P1.0         | P1.0                          | P1.0                         | P1.0             | P1.0          |
| 0001    | ADC1       | P1.1         | P1.1         | P1.1                          | P1.1                         | P1.1             | P1.1          |
| 0010    | ADC2       | P1.2         | 无此通道         | P1.2                          | P5.4                         | P5.4             | P5.4          |
| 0011    | ADC3       | P1.3         | 无此通道         | 无此通道                          | P1.3                         | P1.3             | P1.3          |
| 0100    | ADC4       | P1.4         | 无此通道         | 无此通道                          | P1.4                         | P1.4             | P1.4          |
| 0101    | ADC5       | P1.5         | 无此通道         | 无此通道                          | P1.5                         | P1.5             | P1.5          |
| 0110    | ADC6       | P1.6         | 无此通道         | P1.6                          | P1.6                         | P6.2             | P1.6          |
| 0111    | ADC7       | P1.7         | 无此通道         | P1.7                          | P1.7                         | P6.3             | P1.7          |
| 1000    | ADC8       | P0.0         | P3.0         | P0.0                          | P0.0                         | P0.0             | P3.0          |
| 1001    | ADC9       | P0.1         | P3.1         | P0.1                          | P0.1                         | P0.1             | P3.1          |
| 1010    | ADC10      | P0.2         | P3.2         | P0.2                          | P0.2                         | P0.2             | P3.2          |
| 1011    | ADC11      | P0.3         | P3.3         | P0.3                          | P0.3                         | P0.3             | P3.3          |
| 1100    | ADC12      | 无此通道         | P3.4         | P0.4                          | P0.4                         | P0.4             | P3.4          |
| 1101    | ADC13      | 无此通道         | P3.5         | P0.5                          | P0.5                         | P0.5             | P3.5          |
| 1110    | ADC14      | 无此通道         | P3.6         | P0.6                          | P0.6                         | P0.6             | P3.6          |
| 1111    | 测试内部 1.19V | 有            | 有            | 有                             | 有                            | 有                | 有             |

19.1.2 ADC 配置寄存器 (ADCCFG)

| 符号     | 地址  | B7 | B6 | B5     | B4 | B3         | B2 | B1 | B0 |
|--------|-----|----|----|--------|----|------------|----|----|----|
| ADCCFG | DEH | -  | -  | RESFMT | -  | SPEED[3:0] |    |    |    |

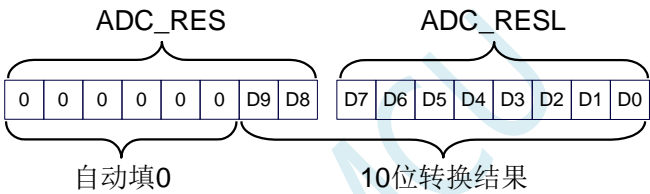
RESFMT: ADC 转换结果格式控制位 (STC8H1K28 系列、STC8H1K08 系列)

0: 转换结果左对齐。ADC\_RES 保存结果的高 8 位, ADC\_RESL 保存结果的低 2 位。格式如下:



RESFMT=0

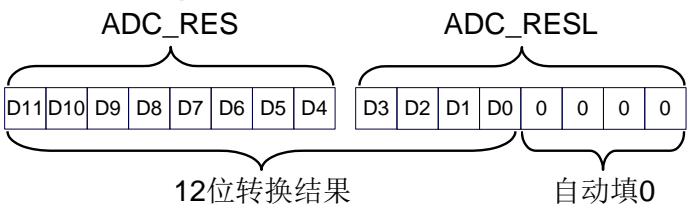
1: 转换结果右对齐。ADC\_RES 保存结果的高 2 位, ADC\_RESL 保存结果的低 8 位。格式如下:



RESFMT=1

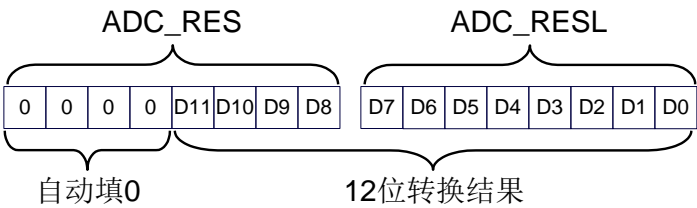
RESFMT: ADC 转换结果格式控制位 (STC8H3K64S4 系列、STC8H3K64S2 系列、STC8H8K64U 系列、STC8H4K64TL 系列、STC8H4K64TLCD 系列、STC8H1K08T 系列)

0: 转换结果左对齐。ADC\_RES 保存结果的高 8 位, ADC\_RESL 保存结果的低 4 位。格式如下:



RESFMT=0

1: 转换结果右对齐。ADC\_RES 保存结果的高 4 位, ADC\_RESL 保存结果的低 8 位。格式如下:



RESFMT=1

SPEED[3:0]: 设置 ADC 工作时钟频率 { $F_{ADC}=SYSclk/2/(SPEED+1)$ }

| SPEED[3:0] | 给 ADC 的工作时钟频率 |
|------------|---------------|
| 0000       | $SYSclk/2/1$  |
| 0001       | $SYSclk/2/2$  |
| 0010       | $SYSclk/2/3$  |
| ...        | ...           |
| 1101       | $SYSclk/2/14$ |
| 1110       | $SYSclk/2/15$ |
| 1111       | $SYSclk/2/16$ |

### 19.1.3 ADC 转换结果寄存器 (ADC\_RES, ADC\_RESL)

| 符号       | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----------|-----|----|----|----|----|----|----|----|----|
| ADC_RES  | BDH |    |    |    |    |    |    |    |    |
| ADC_RESL | BEH |    |    |    |    |    |    |    |    |

当 A/D 转换完成后, 10 位/12 位的转换结果会自动保存到 ADC\_RES 和 ADC\_RESL 中。保存结果的数据格式请参考 ADC\_CFG 寄存器中的 RESFMT 设置。

19.1.4 ADC 时序控制寄存器 (ADCTIM)

| 符号     | 地址    | B7      | B6          | B5 | B4           | B3 | B2 | B1 | B0 |
|--------|-------|---------|-------------|----|--------------|----|----|----|----|
| ADCTIM | FEA8H | CSSETUP | CSHOLD[1:0] |    | SMPDUTY[4:0] |    |    |    |    |

CSSETUP: ADC 通道选择时间控制  $T_{\text{setup}}$

| CSSETUP | 占用 ADC 工作时钟数 |
|---------|--------------|
| 0       | 1 (默认值)      |
| 1       | 2            |

CSHOLD[1:0]: ADC 通道选择保持时间控制  $T_{\text{hold}}$

| CSHOLD[1:0] | 占用 ADC 工作时钟数 |
|-------------|--------------|
| 00          | 1            |
| 01          | 2 (默认值)      |
| 10          | 3            |
| 11          | 4            |

SMPDUTY[4:0]: ADC 模拟信号采样时间控制  $T_{\text{duty}}$  (注意: SMPDUTY 一定不能设置小于 01010B)

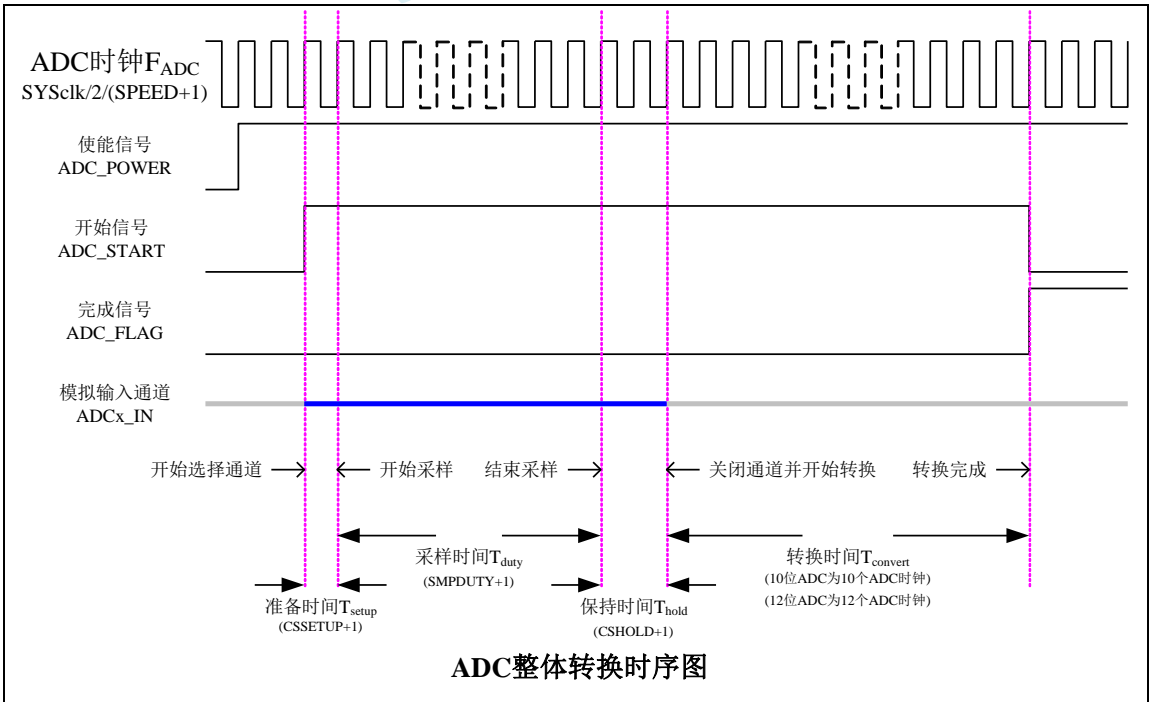
| SMPDUTY[4:0] | 占用 ADC 工作时钟数 |
|--------------|--------------|
| 00000        | 1            |
| 00001        | 2            |
| ...          | ...          |
| 01010        | 11 (默认值)     |
| ...          | ...          |
| 11110        | 31           |
| 11111        | 32           |

ADC 数模转换时间:  $T_{\text{convert}}$

10 位 ADC 的转换时间固定为 10 个 ADC 工作时钟

12 位 ADC 的转换时间固定为 12 个 ADC 工作时钟

一个完整的 ADC 转换时间为:  $T_{\text{setup}} + T_{\text{duty}} + T_{\text{hold}} + T_{\text{convert}}$ , 如下图所示



### 19.1.5 ADC 扩展配置寄存器 (ADCEXCFG)

| 符号       | 地址    | B7 | B6 | B5           | B4 | B3 | B2             | B1 | B0 |
|----------|-------|----|----|--------------|----|----|----------------|----|----|
| ADCEXCFG | FEADH | -  | -  | ADCETRS[1:0] |    | -  | CVTIMESEL[2:0] |    |    |

ADCETRS[1:0]: ADC 外部触发脚 ADC\_ETR 控制位

| ADCETRS[1:0] | ADC_ETR 设置            |
|--------------|-----------------------|
| 0x           | 禁止 ETR 功能             |
| 10           | 使能 ADC_ETR 的上升沿触发 ADC |
| 11           | 使能 ADC_ETR 的下降沿触发 ADC |

**注: 使用此功能前, 必须打开 ADC\_CONTR 中的 ADC 电源开关, 并设置好相应的 ADC 通道**

CVTIMESEL[2:0]: ADC 自动转换次数选择

| CVTIMESEL [2:0] | ADC 自动转换次数   |
|-----------------|--------------|
| 0xx             | 转换 1 次       |
| 100             | 转换 2 次并取平均值  |
| 101             | 转换 4 次并取平均值  |
| 110             | 转换 8 次并取平均值  |
| 111             | 转换 16 次并取平均值 |

**注: 当使能 ADC 自动转换多次功能后, ADC 中断标志只会在 ADC 自动转换到设置的次数后, 才会被置 1 (例如: 设置 CVTIMESEL 为 101B, 即 ADC 自动转换 4 次并取平均值, 则 ADC 中断标志位每完成 4 次 ADC 转换才会被置 1)**



## 19.2 ADC 相关计算公式

### 19.2.1 ADC 速度计算公式

ADC 的转换速度由 ADCCFG 寄存器中的 SPEED 和 ADCTIM 寄存器共同控制。转换速度的计算公式如下所示:

$$10\text{位ADC转换速度} = \frac{\text{MCU工作频率SYSclk}}{2 \times (\text{SPEED}[3:0] + 1) \times [(\text{CSSETUP} + 1) + (\text{CSHOLD} + 1) + (\text{SMPDUTY} + 1) + 10]}$$

$$12\text{位ADC转换速度} = \frac{\text{MCU工作频率SYSclk}}{2 \times (\text{SPEED}[3:0] + 1) \times [(\text{CSSETUP} + 1) + (\text{CSHOLD} + 1) + (\text{SMPDUTY} + 1) + 12]}$$

**注意:**

- 10 位 ADC 的速度不能高于 500KHz
- 12 位 ADC 的速度不能高于 800KHz
- SMPDUTY 的值不能小于 10, 建议设置为 15
- CSSETUP 可使用上电默认值 0
- CHOLD 可使用上电默认值 1 (ADCTIM 建议设置为 3FH)

### 19.2.2 ADC 转换结果计算公式

$$10\text{位ADC转换结果} = 1024 \times \frac{\text{ADC被转换通道的输入电压Vin}}{\text{MCU工作电压Vcc}} \quad (\text{无独立ADC\_Vref+管脚})$$

$$10\text{位ADC转换结果} = 1024 \times \frac{\text{ADC被转换通道的输入电压Vin}}{\text{ADC外部参考源的电压}} \quad (\text{有独立ADC\_Vref+管脚})$$

$$12\text{位ADC转换结果} = 4096 \times \frac{\text{ADC被转换通道的输入电压Vin}}{\text{MCU工作电压Vcc}} \quad (\text{无独立ADC\_Vref+管脚})$$

$$12\text{位ADC转换结果} = 4096 \times \frac{\text{ADC被转换通道的输入电压Vin}}{\text{ADC外部参考源的电压}} \quad (\text{有独立ADC\_Vref+管脚})$$

### 19.2.3 反推 ADC 输入电压计算公式

$$\text{ADC被转换通道的输入电压 } V_{in} = \text{MCU工作电压 } V_{cc} \times \frac{\text{10位ADC转换结果}}{1024} \quad (\text{无独立ADC\_Vref+管脚})$$

$$\text{ADC被转换通道的输入电压 } V_{in} = \text{ADC外部参考源的电压} \times \frac{\text{10位ADC转换结果}}{1024} \quad (\text{有独立ADC\_Vref+管脚})$$

$$\text{ADC被转换通道的输入电压 } V_{in} = \text{MCU工作电压 } V_{cc} \times \frac{\text{12位ADC转换结果}}{4096} \quad (\text{无独立ADC\_Vref+管脚})$$

$$\text{ADC被转换通道的输入电压 } V_{in} = \text{ADC外部参考源的电压} \times \frac{\text{12位ADC转换结果}}{4096} \quad (\text{有独立ADC\_Vref+管脚})$$

### 19.2.4 反推工作电压计算公式

当需要使用 ADC 输入电压和 ADC 转换结果反推工作电压时，若目标芯片无独立的 ADC\_Vref+管脚，则可直接测量并使用下面公式，若目标芯片有独立 ADC\_Vref+管脚时，则必须将 ADC\_Vref+管脚连接到 Vcc 管脚。

$$\text{MCU工作电压Vcc} = 1024 \times \frac{\text{ADC被转换通道的输入电压Vin}}{\text{10位ADC转换结果}}$$

$$\text{MCU工作电压Vcc} = 4096 \times \frac{\text{ADC被转换通道的输入电压Vin}}{\text{12位ADC转换结果}}$$

### 19.3 10 位 ADC 静态特性

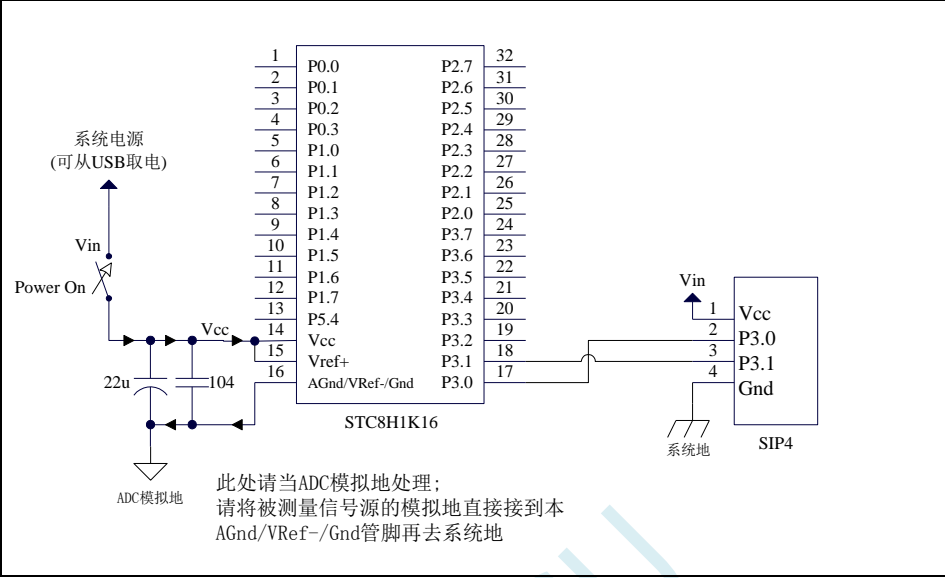
| 符号               | 描述              | 最小值 | 典型值  | 最大值 | 单位   |
|------------------|-----------------|-----|------|-----|------|
| RES              | 分辨率             | -   | 10   | -   | Bits |
| E <sub>T</sub>   | 整体误差            | -   | 1.3  | 3   | LSB  |
| E <sub>O</sub>   | 偏移误差            | -   | 0.3  | 1   | LSB  |
| E <sub>G</sub>   | 增益误差            | -   | 0    | 1   | LSB  |
| E <sub>D</sub>   | 微分非线性误差         | -   | 0.7  | 1.5 | LSB  |
| E <sub>I</sub>   | 积分非线性误差         | -   | 1    | 2   | LSB  |
| R <sub>AIN</sub> | 通道等效电阻          | -   | ∞    | -   | ohm  |
| R <sub>ESD</sub> | 采样保持电容前串接的抗静电电阻 | -   | 700  | -   | ohm  |
| C <sub>ADC</sub> | 内部采样保持电容        | -   | 16.5 | -   | pF   |

### 19.4 12 位 ADC 静态特性

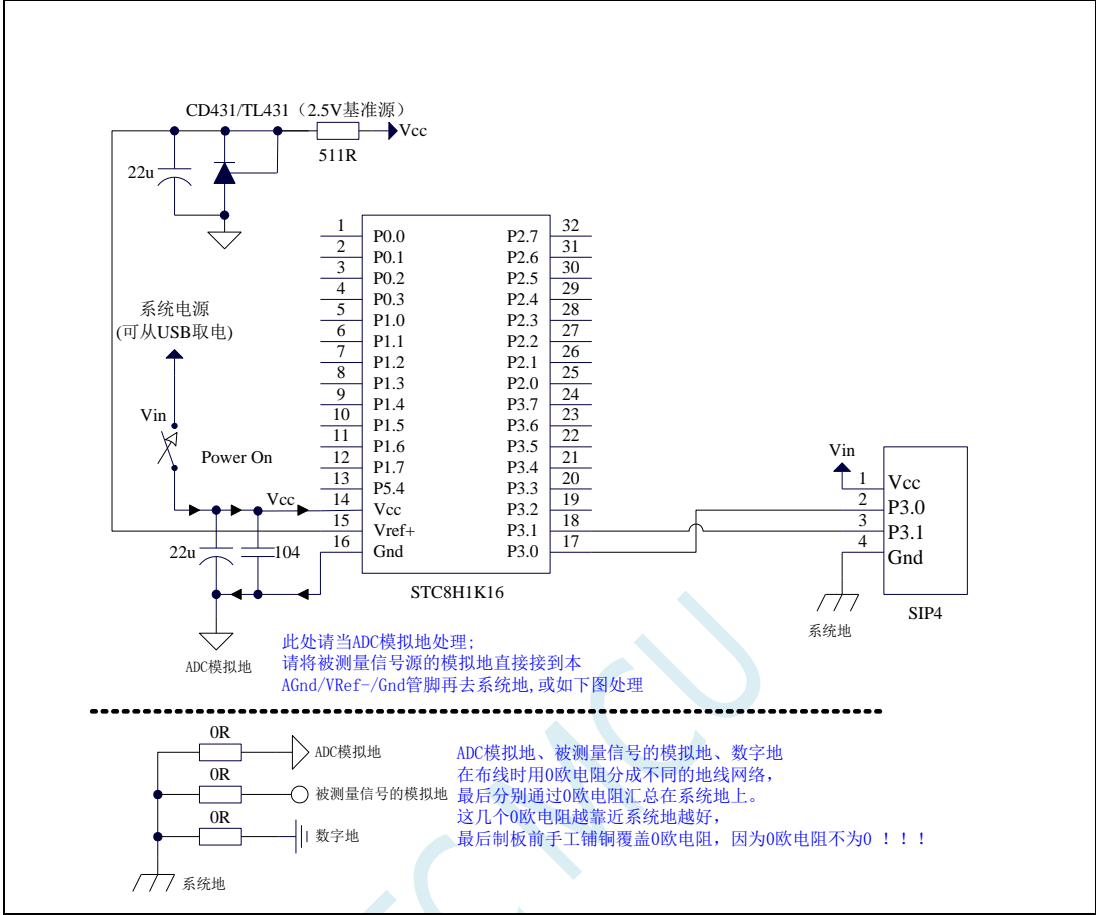
| 符号               | 描述              | 最小值 | 典型值  | 最大值 | 单位   |
|------------------|-----------------|-----|------|-----|------|
| RES              | 分辨率             | -   | 12   | -   | Bits |
| E <sub>T</sub>   | 整体误差            | -   | 0.5  | 1   | LSB  |
| E <sub>O</sub>   | 偏移误差            | -   | -0.1 | 1   | LSB  |
| E <sub>G</sub>   | 增益误差            | -   | 0    | 1   | LSB  |
| E <sub>D</sub>   | 微分非线性误差         | -   | 0.7  | 1.5 | LSB  |
| E <sub>I</sub>   | 积分非线性误差         | -   | 1    | 2   | LSB  |
| R <sub>AIN</sub> | 通道等效电阻          | -   | ∞    | -   | ohm  |
| R <sub>ESD</sub> | 采样保持电容前串接的抗静电电阻 | -   | 700  | -   | ohm  |
| C <sub>ADC</sub> | 内部采样保持电容        | -   | 16.5 | -   | pF   |

## 19.5 ADC 应用参考线路图

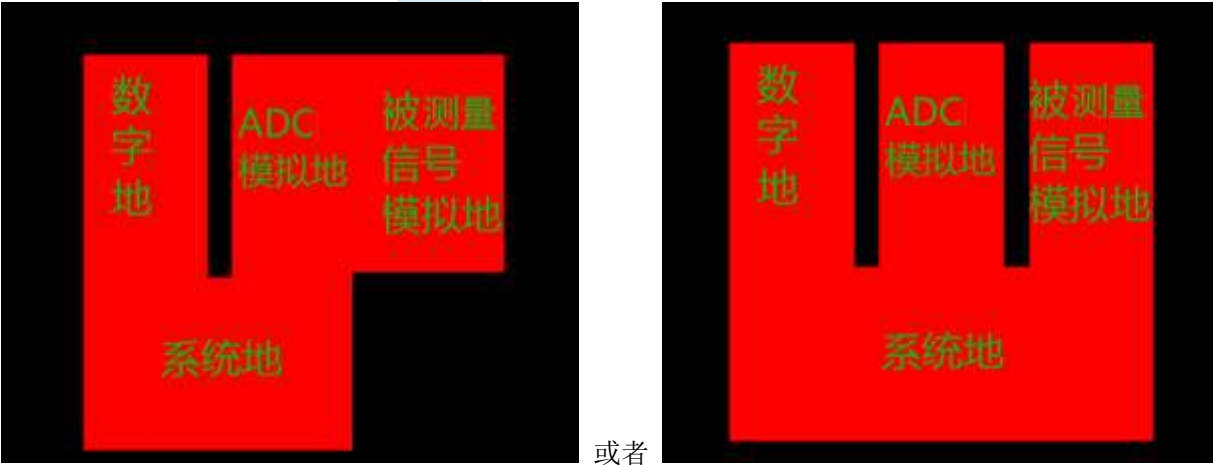
### 19.5.1 一般精度 ADC 参考线路图



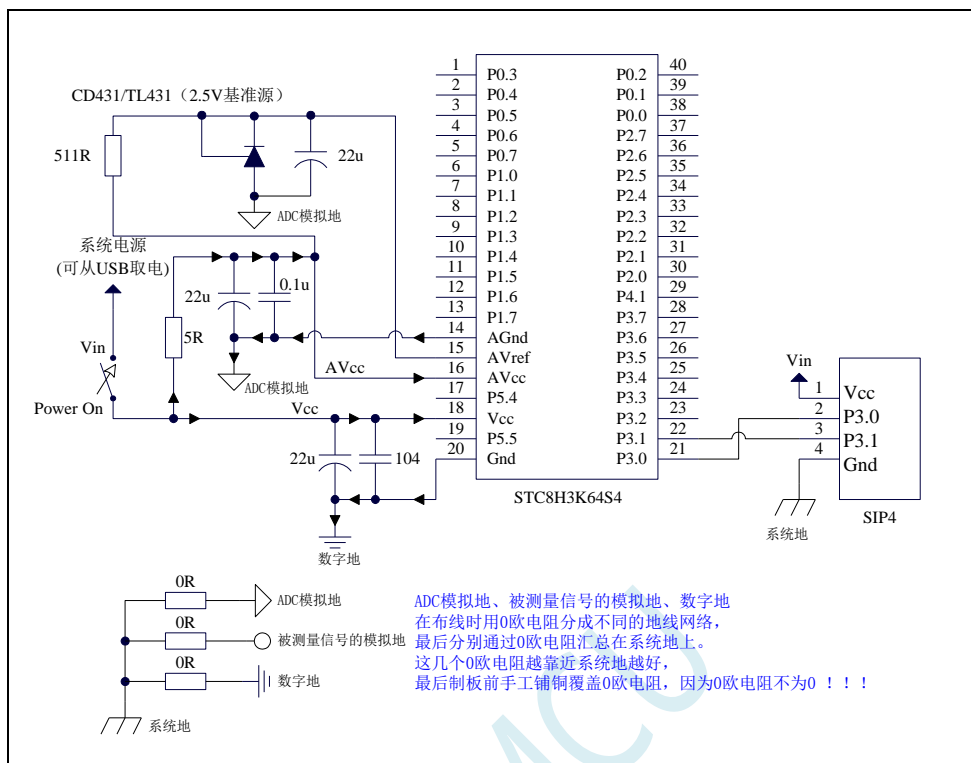
19.5.2 高精度 ADC 参考线路图



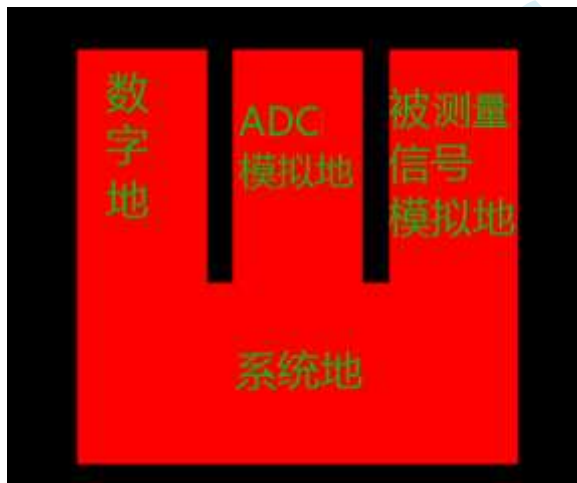
PCB 布线示意图



### 19.5.3 高精度 ADC 参考线路图（有独立 AVcc 和 AGnd）



### PCB 布线示意图



## 19.6 范例程序

### 19.6.1 ADC 基本操作（查询方式）

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void main()
```

```
{
    P_SW2 /= 0x80;                //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P1M0 = 0x00;                //设置P1.0 为ADC 口
    P1M1 = 0x01;

    ADCTIM = 0x3f;              //设置ADC 内部时序
    ADCCFG = 0x0f;              //设置ADC 时钟为系统时钟/16
    ADC_CONTR = 0x80;          //使能ADC 模块

    while (1)
    {
        ADC_CONTR /= 0x40;      //启动AD 转换
        _nop_();
        _nop_();
        while (!(ADC_CONTR & 0x20)); //查询ADC 完成标志
        ADC_CONTR &= ~0x20;      //清完成标志
        P2 = ADC_RES;           //读取ADC 结果
    }
}
```

#### 汇编代码

;测试工作频率为 11.0592MHz

```
ADC_CONTR    DATA    0BCH
ADC_RES      DATA    0BDH
ADC_RESL     DATA    0BEH
ADCCFG       DATA    0DEH

P_SW2        DATA    0BAH
```

```

ADCTIM      XDATA      0FEA8H

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

            ORG          0000H
            LJMP         MAIN

MAIN:        ORG          0100H

            MOV          SP, #5FH
            ORL          P_SW2, #80H                ;使能访问 XFR

            MOV          P0M0, #00H
            MOV          P0M1, #00H
            MOV          P1M0, #00H
            MOV          P1M1, #00H
            MOV          P2M0, #00H
            MOV          P2M1, #00H
            MOV          P3M0, #00H
            MOV          P3M1, #00H
            MOV          P4M0, #00H
            MOV          P4M1, #00H
            MOV          P5M0, #00H
            MOV          P5M1, #00H

            MOV          P1M0, #00H                ;设置 P1.0 为 ADC 口
            MOV          P1M1, #01H
            MOV          DPTR, #ADCTIM              ;设置 ADC 内部时序
            MOV          A, #3FH
            MOVX         @DPTR, A
            MOV          ADCCFG, #0FH                ;设置 ADC 时钟为系统时钟/2/16
            MOV          ADC_CONTR, #80H            ;使能 ADC 模块

LOOP:        ORL          ADC_CONTR, #40H            ;启动 AD 转换
            NOP
            NOP
            MOV          A, ADC_CONTR                ;查询 ADC 完成标志
            JNB          ACC.5, $-2
            ANL          ADC_CONTR, #NOT 20H        ;清完成标志
            MOV          P2, ADC_RES                ;读取 ADC 结果

            SJMP         LOOP

            END

```



## 19.6.2 ADC 基本操作（中断方式）

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void ADC_Isr() interrupt 5
```

```
{
    ADC_CONTR &= ~0x20;           //清中断标志
    P2 = ADC_RES;                 //读取ADC 结果
    ADC_CONTR |= 0x40;           //继续AD 转换
}
```

```
void main()
```

```
{
    P_SW2 /= 0x80;                //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P1M0 = 0x00;                 //设置P1.0 为ADC 口
    P1M1 = 0x01;

    ADCTIM = 0x3f;               //设置ADC 内部时序
    ADCCFG = 0x0f;               //设置ADC 时钟为系统时钟/2/16
    ADC_CONTR = 0x80;           //使能ADC 模块
    EADC = 1;                    //使能ADC 中断
    EA = 1;
    ADC_CONTR |= 0x40;           //启动AD 转换

    while (1);
}
```

### 汇编代码

;测试工作频率为 11.0592MHz

```
ADC_CONTR    DATA    0BCH
ADC_RES      DATA    0BDH
ADC_RESL     DATA    0BEH
ADCCFG       DATA    0DEH

P_SW2        DATA    0BAH
ADCTIM       XDATA    0FEA8H
```

```

EADC      BIT      IE.5

P1M1      DATA    091H
P1M0      DATA    092H
P0M1      DATA    093H
P0M0      DATA    094H
P2M1      DATA    095H
P2M0      DATA    096H
P3M1      DATA    0B1H
P3M0      DATA    0B2H
P4M1      DATA    0B3H
P4M0      DATA    0B4H
P5M1      DATA    0C9H
P5M0      DATA    0CAH

      ORG      0000H
      LJMP     MAIN
      ORG      002BH
      LJMP     ADCISR

      ORG      0100H
ADCISR:
      ANL      ADC_CONTR,#NOT 20H      ;清完成标志
      MOV      P2,ADC_RES              ;读取ADC 结果
      ORL      ADC_CONTR,#40H          ;继续AD 转换
      RETI

MAIN:
      MOV      SP,#5FH
      ORL      P_SW2,#80H              ;使能访问 XFR

      MOV      P0M0,#00H
      MOV      P0M1,#00H
      MOV      P1M0,#00H
      MOV      P1M1,#00H
      MOV      P2M0,#00H
      MOV      P2M1,#00H
      MOV      P3M0,#00H
      MOV      P3M1,#00H
      MOV      P4M0,#00H
      MOV      P4M1,#00H
      MOV      P5M0,#00H
      MOV      P5M1,#00H

      MOV      P1M0,#00H                ;设置P1.0 为ADC 口
      MOV      P1M1,#01H

      MOV      DPTR,#ADCTIM              ;设置ADC 内部时序
      MOV      A,#3FH
      MOVX     @DPTR,A
      MOV      ADCCFG,#0FH              ;设置ADC 时钟为系统时钟/2/16
      MOV      ADC_CONTR,#80H            ;使能ADC 模块
      SETB     EADC                      ;使能ADC 中断
      SETB     EA
      ORL      ADC_CONTR,#40H            ;启动AD 转换

      SJMP     $

      END

```

## 19.6.3 格式化 ADC 转换结果

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void main()
```

```
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P1M0 = 0x00;                //设置 P1.0 为 ADC 口
    P1M1 = 0x01;
    ADCTIM = 0x3f;              //设置 ADC 内部时序
    ADCCFG = 0x0f;              //设置 ADC 时钟为系统时钟/2/16
    ADC_CONTR = 0x80;           //使能 ADC 模块
    ADC_CONTR /= 0x40;          //启动 AD 转换
    _nop_();
    _nop_();
    while (!(ADC_CONTR & 0x20)); //查询 ADC 完成标志
    ADC_CONTR &= ~0x20;         //清完成标志

    ADCCFG = 0x00;              //设置结果左对齐
    ACC = ADC_RES;              //A 存储 ADC 的 10 位结果的高 8 位
    B = ADC_RES;                //B[7:6] 存储 ADC 的 10 位结果的低 2 位, B[5:0] 为 0

    // ADCCFG = 0x20;           //设置结果右对齐
    // ACC = ADC_RES;           //A[1:0] 存储 ADC 的 10 位结果的高 2 位, A[7:2] 为 0
    // B = ADC_RES;             //B 存储 ADC 的 10 位结果的低 8 位

    while (1);
}
```

### 汇编代码

;测试工作频率为 11.0592MHz

```
ADC_CONTR    DATA    0BCH
ADC_RES      DATA    0BDH
ADC_RES      DATA    0BEH
ADCCFG       DATA    0DEH
```

```

P_SW2      DATA      0BAH
ADCTIM      XDATA      0FEA8H

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

                ORG      0000H
                LJMP     MAIN

MAIN:         ORG      0100H

                MOV      SP, #5FH
                ORL      P_SW2, #80H                ;使能访问 XFR

                MOV      P0M0, #00H
                MOV      P0M1, #00H
                MOV      P1M0, #00H
                MOV      P1M1, #00H
                MOV      P2M0, #00H
                MOV      P2M1, #00H
                MOV      P3M0, #00H
                MOV      P3M1, #00H
                MOV      P4M0, #00H
                MOV      P4M1, #00H
                MOV      P5M0, #00H
                MOV      P5M1, #00H

                MOV      P1M0, #00H                ;设置 P1.0 为 ADC 口
                MOV      P1M1, #01H
                MOV      DPTR, #ADCTIM                ;设置 ADC 内部时序
                MOV      A, #3FH
                MOVX     @DPTR, A
                MOV      ADCCFG, #0FH                ;设置 ADC 时钟为系统时钟/2/16
                MOV      ADC_CONTR, #80H                ;使能 ADC 模块

                ORL      ADC_CONTR, #40H                ;启动 AD 转换
                NOP
                NOP
                MOV      A, ADC_CONTR                ;查询 ADC 完成标志
                JNB      ACC.5, $-2
                ANL      ADC_CONTR, #NOT 20H                ;清完成标志

                MOV      ADCCFG, #00H                ;设置结果左对齐
                MOV      A, ADC_RES                ;A 存储 ADC 的 10 位结果的高 8 位
                MOV      B, ADC_RES                ;B[7:6] 存储 ADC 的 10 位结果的低 2 位, B[5:0] 为 0

;                MOV      ADCCFG, #20H                ;设置结果右对齐
;                MOV      A, ADC_RES                ;A[3:0] 存储 ADC 的 10 位结果的高 2 位, A[7:2] 为 0

```

```
;          MOV          B,ADC_RESL          ;B 存储ADC 的10 位结果的低8 位

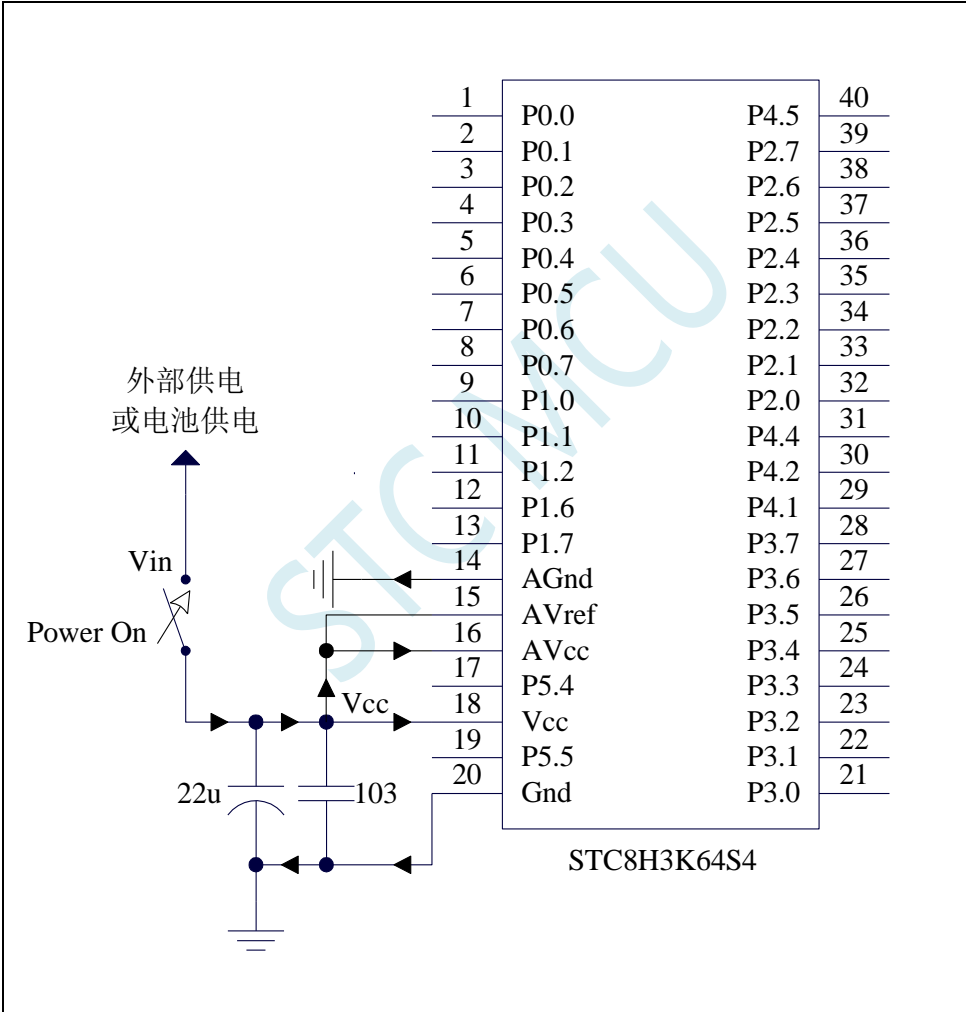
          SJMP          $

          END
```

### 19.6.4 利用 ADC 第 15 通道测量外部电压或电池电压

STC8H 系列 ADC 的第 15 通道用于测量内部参考信号源, 由于内部参考信号源很稳定, 约为 1.19V, 且不会随芯片的工作电压的改变而变化, 所以可以通过测量内部 1.19V 参考信号源, 然后通过 ADC 的值便可反推出外部电压或外部电池电压。

下图为参考线路图:



#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"

#define FOSC 11059200UL
#define BRT (65536 - (FOSC / 115200+2) / 4)
```

```

//加2 操作是为了让 Keil 编译器
//自动实现四舍五入运算

int      *BGV;

//内部参考信号源值存放在 idata 中
//idata 的 EFH 地址存放高字节
//idata 的 F0H 地址存放低字节
//电压单位为毫伏(mV)

bit      busy;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void ADCInit()
{
    ADCTIM = 0x3f;
    ADCCFG = 0x2f;
    ADC_CONTR = 0x8f;
}

int  ADCRead()
{
    int res;

    ADC_CONTR |= 0x40;
    _nop_();
    _nop_();
    while (!(ADC_CONTR & 0x20));
    ADC_CONTR &= ~0x20;
    res = (ADC_RES << 8) / ADC_RES1;
}

```

//设置ADC 内部时序

//设置ADC 时钟为系统时钟/2/16

//使能ADC 模块,并选择第15 通道

//启动AD 转换

//查询ADC 完成标志

//清完成标志

//读取ADC 结果

```

    return res;
}

void main()
{
    int res;
    int vcc;
    int i;

    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    BGV = (int idata *)0xef;
    ADCInit(); //ADC 初始化
    UartInit(); //串口初始化

    ES = 1;
    EA = 1;

    // ADCRead();
    // ADCRead(); //前两个数据建议丢弃

    res = 0;
    for (i=0; i<8; i++)
    {
        res += ADCRead(); //读取 8 次数据
    }
    res >>= 3; //取平均值

    vcc = (int)(4096L * *BGV / res); // (12 位 ADC 算法) 计算 VREF 管脚电压, 即电池电压
    // vcc = (int)(1024L * *BGV / res); // (10 位 ADC 算法) 计算 VREF 管脚电压, 即电池电压
    //注意, 此电压的单位为毫伏(mV)

    UartSend(vcc >> 8); //输出电压值到串口
    UartSend(vcc);

    while (1);
}

```

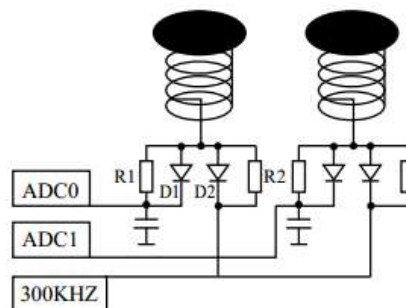
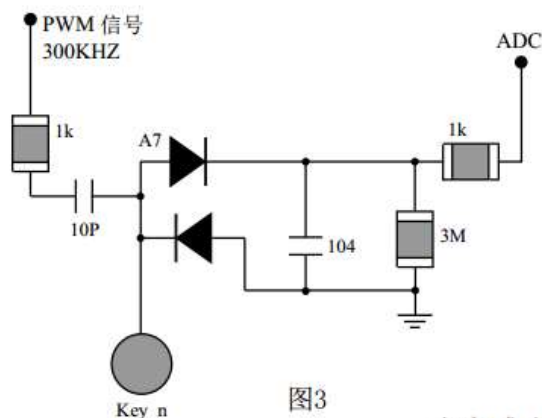
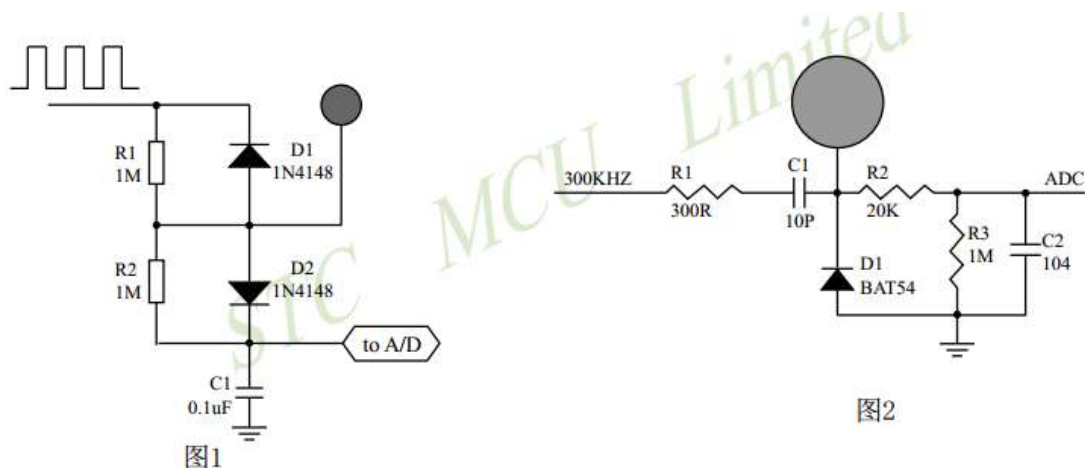
上面的方法是使用 ADC 的第 15 通道反推外部电池电压的。在 ADC 测量范围内, ADC 的外部测量电压与 ADC 的测量值是成正比例的, 所以也可以使用 ADC 的第 15 通道反推外部通道输入电压, 假设当前已获取了内部参考信号源电压为 BGV, 内部参考信号源的 ADC 测量值为  $res_{bg}$ , 外部通道输入电压的 ADC 测量值为  $res_x$ , 则外部通道输入电压  $V_x = BGV / res_{bg} * res_x$ ;

## 19.6.5 ADC 做电容感应触摸按键

按键是电路最常用的零件之一，是人机界面重要的输入方式，我们最熟悉的是机械式按键，但是机械按键有一个缺点（特别是便宜的按键），触点有寿命，很容易出现接触不良而失效。而非接触的按键则没有机械触点，寿命长，使用方便。

非接触的按键有多种方案，而电容感应按键则是低成本方案，多年前一般是使用专门的 IC 来实现，随着 MCU 功能的加强，以及广大用户的实践经验，直接使用 MCU 来做电容感应按键的技术已经成熟，其中最典型最可靠的是使用 ADC 做的方案。

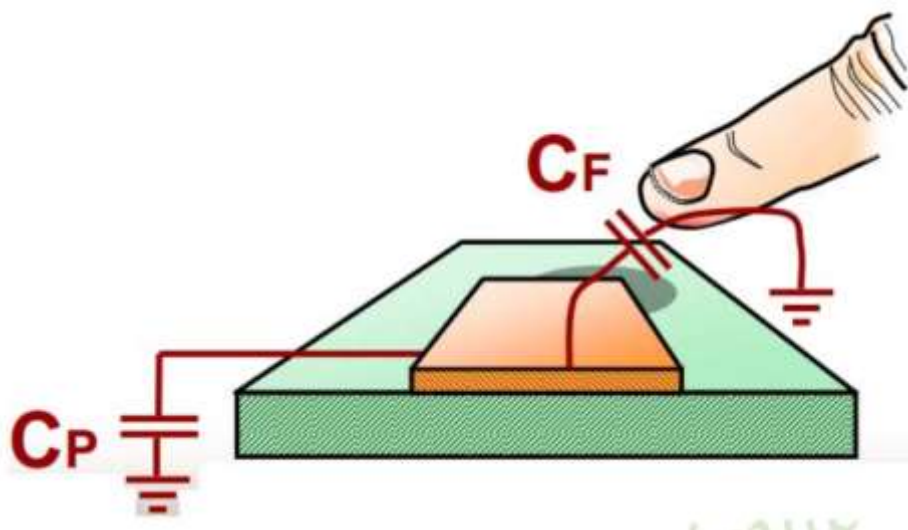
本文档详述使用 STC 带 ADC 的系列 MCU 做的方案，可以使用任何带 ADC 功能的 MCU 来实现。下面前 3 个图是用得最多的方式，原理都一样，本文使用第 2 个图。



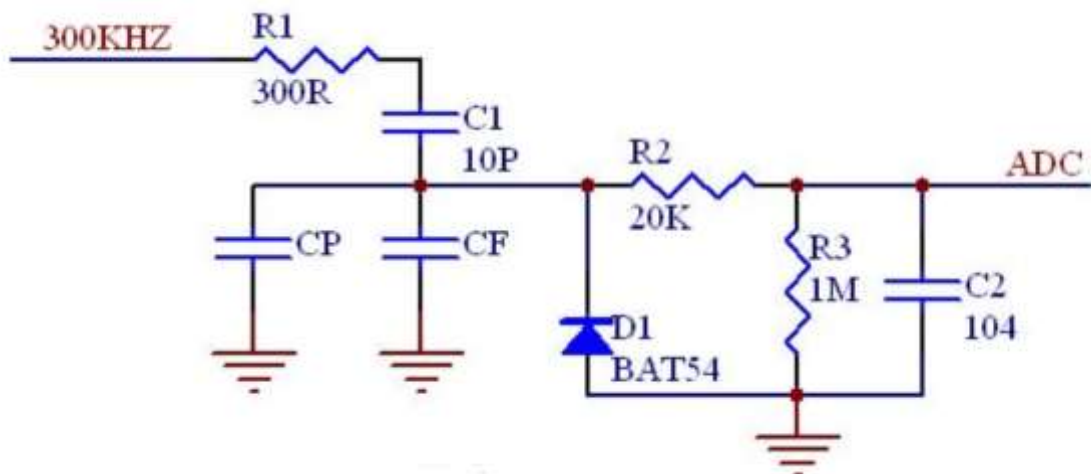
电容感应按键取样电路

一般实际应用时，都使用图 4 所示的感应弹簧来加大手指按下的面积。感应弹簧等效一块对地的金属板，对地有一个电容  $C_P$ ，而手指按下后，则再并联一个对地的电容  $C_F$ ，如下图所示。





下面为电路图的说明, CP 为金属板和分布电容, CF 为手指电容, 并联在一起与 C1 对输入的 300KHZ 方波进行分压, 经过 D1 整流, R2、C2 滤波后送 ADC, 当手指压上去后, 送去 ADC 的电压降低, 程序就可以检测出按键动作。



## C 语言代码

//测试工作频率为24MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define MAIN_Fosc 24000000UL //定义主时钟
#define Timer0_Reload (65536UL -(MAIN_Fosc / 600000)) //Timer 0 重装值, 对应300KHZ
```

```
typedef unsigned char u8;
typedef unsigned int u16;
typedef unsigned long u32;
```

```
#define CHANNEL 8 //ADC 通道数
#define ADC_90T (3<<5) //ADC 时间 90T
#define ADC_180T (2<<5) //ADC 时间 180T
#define ADC_360T (1<<5) //ADC 时间 360T
#define ADC_540T 0 //ADC 时间 540T
#define ADC_FLAG (1<<4) //软件清0
```

```

#define    ADC_START    (1<<3)                                //自动清0

sbit      P_LED7       =    P2^7;
sbit      P_LED6       =    P2^6;
sbit      P_LED5       =    P2^5;
sbit      P_LED4       =    P2^4;
sbit      P_LED3       =    P2^3;
sbit      P_LED2       =    P2^2;
sbit      P_LED1       =    P2^1;
sbit      P_LED0       =    P2^0;

u16 idata adc[TOUCH_CHANNEL];                                //当前ADC 值
u16 idata adc_prev[TOUCH_CHANNEL];                          //上一个ADC 值
u16 idata TouchZero[TOUCH_CHANNEL];                        //0 点ADC 值
u8 idata TouchZeroCnt[TOUCH_CHANNEL];                      //0 点自动跟踪计数
u8 cnt_250ms;

void delay_ms(u8 ms);
void ADC_init(void);
u16 Get_ADC10bitResult(u8 channel);
void AutoZero(void);
u8 check_adc(u8 index);
void ShowLED(void);

void main(void)
{
    u8 i;

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    delay_ms(50);
    ET0 = 0;                                                  //初始化 Timer0 输出一个 300KHZ 时钟
    TR0 = 0;
    AUXR /= 0x80;                                             //Timer0 set as 1T mode
    AUXR2 /= 0x01;                                           //允许输出时钟
    TMOD = 0;                                                 //Timer0 set as Timer, 16 bits Auto Reload.
    TH0 = (u8)(Timer0_Reload >> 8);
    TL0 = (u8)Timer0_Reload;
    TR0 = 1;
    ADC_init();                                              //ADC 初始化
    delay_ms(50);                                           //延时 50ms
    for (i=0; i<TOUCH_CHANNEL; i++)                        //初始化 0 点和上一个值和 0 点自动跟踪计数
    {
        adc_prev[i] = 1023;
        TouchZero[i] = 1023;
        TouchZeroCnt[i] = 0;
    }
    cnt_250ms = 0;
}

```

```

while (1)
{
    delay_ms(50);                //每隔 50ms 处理一次按键
    ShowLED();
    if (++cnt_250ms >= 5)
    {
        cnt_250ms = 0;
        AutoZero();              //每隔 250ms 处理一次 0 点自动跟踪
    }
}

void delay_ms(u8 ms)
{
    unsigned int i;

    do
    {
        i = MAIN_Fosc / 10000;
        while(--i);
    } while(--ms);
}

void ADC_init(void)
{
    P1M0 = 0x00;                  //8 路 ADC
    P1M1 = 0xff;
    ADC_CONTR = 0x80;             //允许 ADC
}

u16 Get_ADC10bitResult(u8 channel)
{
    ADC_RES = 0;
    ADC_RESL = 0;
    ADC_CONTR = 0x80 | ADC_90T | ADC_START | channel; //触发 ADC
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    while((ADC_CONTR & ADC_FLAG) == 0); //等待 ADC 转换结束
    ADC_CONTR = 0x80; //清除标志
    return(((u16)ADC_RES << 2) | ((u16)ADC_RESL & 3)); //返回 ADC 结果
}

void AutoZero(void) //250ms 调用一次
//这是使用相邻 2 个采样的差的绝对值之和来检测。
{
    u8 i;
    u16 j,k;

    for(i=0; i<TOUCH_CHANNEL; i++) //处理 8 个通道
    {
        j = adc[i];
        k = j - adc_prev[i]; //减前一个读数
        F0 = 0; //按下
        if(k & 0x8000) F0 = 1, k = 0 - k; //释放 求出两次采样的差值
        if(k >= 20) //变化比较大
        {
            TouchZeroCnt[i] = 0; //如果变化比较大, 则清 0 计数器
        }
    }
}

```

```

        if(F0) TouchZero[i] = j; //如果是释放, 并且变化比较大, 则直接替代
    }
    else //变化比较小, 则蠕动, 自动0点跟踪
    {
        if(++TouchZeroCnt[i] >= 20) //连续检测到小变化 20 次/4 = 5 秒.
        {
            TouchZeroCnt[i] = 0;
            TouchZero[i] = adc_prev[i]; //变化缓慢的值作为0点
        }
        adc_prev[i] = j; //保存这一次的采样值
    }
}

u8 check_adc(u8 index) //获取触摸信息函数 50ms 调用1次
                        //判断键按下或释放, 有回差控制
{
    u16 delta;

    adc[index] = 1023 - Get_ADC10bitResult(index); //获取ADC值, 转成按下键, ADC值增加
    if(adc[index] < TouchZero[index]) return 0; //比0点还小的值, 则认为是键释放
    delta = adc[index] - TouchZero[index];
    if(delta >= 40) return 1; //键按下
    if(delta <= 20) return 0; //键释放
    return 2; //保持原状态
}

void ShowLED(void)
{
    u8 i;

    i = check_adc(0);
    if(i == 0) P_LED0 = 1; //指示灯灭
    if(i == 1) P_LED0 = 0; //指示灯亮
    i = check_adc(1);
    if(i == 0) P_LED1 = 1; //指示灯灭
    if(i == 1) P_LED1 = 0; //指示灯亮
    i = check_adc(2);
    if(i == 0) P_LED2 = 1; //指示灯灭
    if(i == 1) P_LED2 = 0; //指示灯亮
    i = check_adc(3);
    if(i == 0) P_LED3 = 1; //指示灯灭
    if(i == 1) P_LED3 = 0; //指示灯亮
    i = check_adc(4);
    if(i == 0) P_LED4 = 1; //指示灯灭
    if(i == 1) P_LED4 = 0; //指示灯亮
    i = check_adc(5);
    if(i == 0) P_LED5 = 1; //指示灯灭
    if(i == 1) P_LED5 = 0; //指示灯亮
    i = check_adc(6);
    if(i == 0) P_LED6 = 1; //指示灯灭
    if(i == 1) P_LED6 = 0; //指示灯亮
    i = check_adc(7);
    if(i == 0) P_LED7 = 1; //指示灯灭
    if(i == 1) P_LED7 = 0; //指示灯亮
}

```

## 汇编代码

;测试工作频率为24MHz

|                     |             |                                |                          |
|---------------------|-------------|--------------------------------|--------------------------|
| <i>Fosc_KHZ</i>     | <i>EQU</i>  | 24000                          | ;定义主时钟 KHZ               |
| <i>Reload</i>       | <i>EQU</i>  | (65536 - <i>Fosc_KHZ</i> /600) | ;Timer 0 重装值,对应300KHZ    |
| <i>ADC_CONTR</i>    | <i>DATA</i> | 0xBC                           | ;带AD 系列                  |
| <i>ADC_RES</i>      | <i>DATA</i> | 0xBD                           | ;带AD 系列                  |
| <i>ADC_RESL</i>     | <i>DATA</i> | 0xBE                           | ;带AD 系列                  |
| <i>AUXR</i>         | <i>DATA</i> | 0x8E                           |                          |
| <i>AUXR2</i>        | <i>DATA</i> | 0x8F                           |                          |
| <i>P0M1</i>         | <i>DATA</i> | 093H                           |                          |
| <i>P0M0</i>         | <i>DATA</i> | 094H                           |                          |
| <i>P1M1</i>         | <i>DATA</i> | 091H                           |                          |
| <i>P1M0</i>         | <i>DATA</i> | 092H                           |                          |
| <i>P2M1</i>         | <i>DATA</i> | 095H                           |                          |
| <i>P2M0</i>         | <i>DATA</i> | 096H                           |                          |
| <i>P3M1</i>         | <i>DATA</i> | 0B1H                           |                          |
| <i>P3M0</i>         | <i>DATA</i> | 0B2H                           |                          |
| <i>P4M1</i>         | <i>DATA</i> | 0B3H                           |                          |
| <i>P4M0</i>         | <i>DATA</i> | 0B4H                           |                          |
| <i>P5M1</i>         | <i>DATA</i> | 0C9H                           |                          |
| <i>P5M0</i>         | <i>DATA</i> | 0CAH                           |                          |
| <i>CHANNEL</i>      | <i>EQU</i>  | 8                              | ;ADC 通道数                 |
| <i>ADC_90T</i>      | <i>EQU</i>  | (3 SHL 5)                      | ;ADC 时间 90T              |
| <i>ADC_180T</i>     | <i>EQU</i>  | (2 SHL 5)                      | ;ADC 时间 180T             |
| <i>ADC_360T</i>     | <i>EQU</i>  | (1 SHL 5)                      | ;ADC 时间 360T             |
| <i>ADC_540T</i>     | <i>EQU</i>  | 0                              | ;ADC 时间 540T             |
| <i>ADC_FLAG</i>     | <i>EQU</i>  | (1 SHL 4)                      | ;软件清0                    |
| <i>ADC_START</i>    | <i>EQU</i>  | (1 SHL 3)                      | ;自动清0                    |
| <i>P_LED7</i>       | <i>BIT</i>  | P2.7;                          |                          |
| <i>P_LED6</i>       | <i>BIT</i>  | P2.6;                          |                          |
| <i>P_LED5</i>       | <i>BIT</i>  | P2.5;                          |                          |
| <i>P_LED4</i>       | <i>BIT</i>  | P2.4;                          |                          |
| <i>P_LED3</i>       | <i>BIT</i>  | P2.3;                          |                          |
| <i>P_LED2</i>       | <i>BIT</i>  | P2.2;                          |                          |
| <i>P_LED1</i>       | <i>BIT</i>  | P2.1;                          |                          |
| <i>P_LED0</i>       | <i>BIT</i>  | P2.0;                          |                          |
| <i>adc</i>          | <i>EQU</i>  | 30H                            | ;当前ADC 值 30H~3FH,两字节一个值  |
| <i>adc_prev</i>     | <i>EQU</i>  | 40H                            | ;上一个ADC 值 40H~4FH,两字节一个值 |
| <i>TouchZero</i>    | <i>EQU</i>  | 50H                            | ;0 点ADC 值 50H~5FH,两字节一个值 |
| <i>TouchZeroCnt</i> | <i>EQU</i>  | 60H                            | ;0 点自动跟踪计数 60H~67H       |
| <i>cnt_250ms</i>    | <i>DATA</i> | 68H                            |                          |
|                     | <i>ORG</i>  | 0000H                          |                          |
|                     | <i>LJMP</i> | MAIN                           |                          |
|                     | <i>ORG</i>  | 0100H                          |                          |
| MAIN:               |             |                                |                          |
|                     | <i>MOV</i>  | SP,#0D0H                       |                          |
|                     | <i>MOV</i>  | P0M0,#00H                      |                          |
|                     | <i>MOV</i>  | P0M1,#00H                      |                          |
|                     | <i>MOV</i>  | P1M0,#00H                      |                          |
|                     | <i>MOV</i>  | P1M1,#00H                      |                          |
|                     | <i>MOV</i>  | P2M0,#00H                      |                          |
|                     | <i>MOV</i>  | P2M1,#00H                      |                          |
|                     | <i>MOV</i>  | P3M0,#00H                      |                          |

```

MOV      P3M1,#00H
MOV      P4M0,#00H
MOV      P4M1,#00H
MOV      P5M0,#00H
MOV      P5M1,#00H

MOV      R7,#50
LCALL    F_delay_ms
CLR      ET0                      ;初始化 Timer0 输出一个 300KHZ 时钟
CLR      TR0
ORL      AUXR,#080H              ;Timer0 set as 1T mode
ORL      AUXR2,#01H              ;允许输出时钟
MOV      TMOD,#0                 ;Timer0 set as Timer,16 bits Auto Reload.
MOV      TH0,#HIGH Reload
MOV      TL0,#LOW Reload
SETB     TR0
LCALL    F_ADC_init
MOV      R7,#50
LCALL    F_delay_ms
MOV      R0,#adc_prev            ;初始化上一个 ADC 值

L_Init_Loop1:
MOV      @R0,#03H
INC      R0
MOV      @R0,#0FFH
INC      R0
MOV      A,R0
CJNE     A,#(adc_prev + CHANNEL * 2),L_Init_Loop1
MOV      R0,#TouchZero          ;初始化 0 点 ADC 值

L_Init_Loop2:
MOV      @R0,#03H
INC      R0
MOV      @R0,#0FFH
INC      R0
MOV      A,R0
CJNE     A,#(TouchZero+CHANNEL * 2),L_Init_Loop2
MOV      R0,#TouchZeroCnt       ;初始化自动跟踪计数值

L_Init_Loop3:
MOV      @R0,#0
INC      R0
MOV      A,R0
CJNE     A,#(TouchZeroCnt + CHANNEL),L_Init_Loop3
MOV      cnt_250ms,#5

L_MainLoop:
MOV      R7,#50                  ;延时 50ms
LCALL    F_delay_ms
LCALL    F_ShowLED               ;处理一次触摸键值
DJNZ     cnt_250ms,L_MainLoop
MOV      cnt_250ms,#5            ;250ms 处理一次 0 点自动跟踪
LCALL    F_AutoZero              ;自动跟踪零点
SJMP     L_MainLoop

F_ADC_init:
MOV      P1M0,#00H              ;8 路 ADC
MOV      P1M1,#0FFH
MOV      ADC_CONTR,#080H        ;允许 ADC
RET

F_Get_ADC10bitResult:
MOV      ADC_RES,#0

```

```

MOV     ADC_RESL,#0
MOV     A,R7
ORL     A,#0E8H           ;触发ADC
MOV     ADC_CONTR,A
NOP
NOP
NOP
NOP
L_10bitADC_Loop1:
MOV     A,ADC_CONTR
JNB     ACC.4,L_10bitADC_Loop1 ;等待ADC 转换结束
MOV     ADC_CONTR,#080H    ;清除标志
MOV     A,ADC_RES
MOV     B,#04H
MUL     AB
MOV     R7,A
MOV     R6,B
MOV     A,ADC_RESL
ANL     A,#03H
ORL     A,R7
MOV     R7,A
RET

F_AutoZero:                ;250ms 调用一次
                                ;这是使用相邻2 个采样的差的绝对值之和来检测。
CLR     A
MOV     R5,A
L_AutoZero_Loop:
MOV     A,R5
ADD     A,ACC
ADD     A,#LOW (adc)
MOV     R0,A
MOV     A,@R0
MOV     R6,A
INC     R0
MOV     A,@R0
MOV     R7,A
MOV     A,R5
ADD     A,ACC
ADD     A,#LOW (adc_prev+01H)
MOV     R0,A
CLR     C
MOV     A,R7
SUBB    A,@R0
MOV     R3,A
MOV     A,R6
DEC     R0
SUBB    A,@R0
MOV     R2,A
CLR     F0 ;按下
JNB     ACC.7,L_AutoZero_I
SETB    F0
CLR     C
CLR     A
SUBB    A,R3
MOV     R3,A
MOV     A,R3
CLR     A
SUBB    A,R2

```

```

MOV R2,A
L_AutoZero_1:
CLR C ;计算 [R2 R3] - #20,if(k >= 20)
MOV A,R3
SUBB A,#20
MOV A,R2
SUBB A,#00H
JC L_AutoZero_2 ;[R2 R3],20,转
MOV A,#LOW (TouchZeroCnt) ;如果变化比较大, 则清0 计数器 TouchZeroCnt[i] = 0;
ADD A,R5
MOV R0,A
MOV @R0,#0
JNB F0,L_AutoZero_3
MOV A,R5
ADD A,ACC
ADD A,#LOW (TouchZero)
MOV R0,A
MOV @R0,6
INC R0
MOV @R0,7
SJMP L_AutoZero_3

L_AutoZero_2: ;变化比较小, 则蠕动, 自动0 点跟踪
;连续检测到小变化 20 次/4 = 5 秒.
MOV A,#LOW (TouchZeroCnt)
ADD A,R5
MOV R0,A
INC @R0
MOV A,@R0
CLR C
SUBB A,#20
JC L_AutoZero_3 ;if(TouchZeroCnt[i] < 20), 转
MOV @R0,#0 ;TouchZeroCnt[i] = 0;
MOV A,R5 ;变化缓慢的值作为0 点
ADD A,ACC
ADD A,#LOW (adc_prev)
MOV R0,A
MOV A,@R0
MOV R2,A
INC R0
MOV A,@R0
MOV R3,A
MOV A,R5
ADD A,ACC
ADD A,#LOW (TouchZero)
MOV R0,A
MOV @R0,2
INC R0
MOV @R0,3

L_AutoZero_3: ;保存采样值 adc_prev[i] = j;
MOV A,R5
ADD A,ACC
ADD A,#LOW (adc_prev)
MOV R0,A
MOV @R0,6
INC R0
MOV @R0,7
INC R5
MOV A,R5
XRL A,#08H

```



*JZ*                *\$ + 5H*  
*LJMP*            *L\_AutoZero\_Loop*  
*RET*

*F\_check\_adc:*

;判断键按下或释放,有回差控制

*MOV R4,7*  
*LCALL F\_Get\_ADC10bitResult*        ;返回的ADC 值在 [R6 R7]  
*CLR C*  
*MOV A,#0FFH*  
*SUBB A,R7*  
*MOV R7,A*  
*MOV A,#03H*  
*SUBB A,R6*  
*MOV R6,A*  
*MOV A,R4*                                ;保存 adc[index]  
*ADD A,ACC*  
*ADD A,#LOW (adc)*  
*MOV R0,A*  
*MOV @R0,6*  
*INC R0*  
*MOV @R0,7*  
*MOV A,R4*  
*ADD A,ACC*  
*ADD A,#LOW (TouchZero+01H)*  
*MOV RI,A*  
*MOV A,R4*  
*ADD A,ACC*  
*ADD A,#LOW (adc)*  
*MOV R0,A*  
*MOV A,@R0*  
*MOV R6,A*  
*INC R0*  
*MOV A,@R0*  
*CLR C*  
*SUBB A,@R1*                             ;计算 adc[index] - TouchZero[index]  
*MOV A,R6*  
*DEC RI*  
*SUBB A,@R1*  
*JNC L\_check\_adc\_1*  
*MOV R7,#00H*  
*RET*

*L\_check\_adc\_1:*

*MOV A,R4*  
*ADD A,ACC*  
*ADD A,#LOW (TouchZero+01H)*  
*MOV RI,A*  
*MOV A,R4*  
*ADD A,ACC*  
*ADD A,#LOW (adc+01H)*  
*MOV R0,A*  
*CLR C*  
*MOV A,@R0*  
*SUBB A,@R1*  
*MOV R7,A*  
*DEC R0*  
*MOV A,@R0*  
*DEC RI*  
*SUBB A,@R1*  
*MOV R6,A*

```

    CLR        C
    MOV        A,R7
    SUBB       A,#40
    MOV        A,R6
    SUBB       A,#00H
    JC         L_check_adc_2      ;if(delta < 40), 转
    MOV        R7,#1              ;if(delta >= 40) return 1; // 键按下 返回 1
    RET

L_check_adc_2:
    SETB       C
    MOV        A,R7
    SUBB       A,#20
    MOV        A,R6
    SUBB       A,#00H
    JNC        L_check_adc_3
    MOV        R7,#0
    RET

L_check_adc_3:
    MOV        R7,#2
    RET

F_ShowLED:
    MOV        R7,#0
    LCALL      F_check_adc
    MOV        A,R7
    ANL        A,#0FEH
    JNZ        L_QuitCheck0
    MOV        A,R7
    MOV        C,ACC.0
    CPL        C
    MOV        P_LED0,C

L_QuitCheck0:
    MOV        R7,#1
    LCALL      F_check_adc
    MOV        A,R7
    ANL        A,#0FEH
    JNZ        L_QuitCheck1
    MOV        A,R7
    MOV        C,ACC.0
    CPL        C
    MOV        P_LED1,C

L_QuitCheck1:
    MOV        R7,#2
    LCALL      F_check_adc
    MOV        A,R7
    ANL        A,#0FEH
    JNZ        L_QuitCheck2
    MOV        A,R7
    MOV        C,ACC.0
    CPL        C
    MOV        P_LED2,C

L_QuitCheck2:
    MOV        R7,#3
    LCALL      F_check_adc
    MOV        A,R7
    ANL        A,#0FEH
    JNZ        L_QuitCheck3
    MOV        A,R7
    MOV        C,ACC.0

```

```

        CPL                C
        MOV                P_LED3,C
L_QuitCheck3:
        MOV                R7,#4
        LCALL              F_check_adc
        MOV                A,R7
        ANL                A,#0FEH
        JNZ                L_QuitCheck4
        MOV                A,R7
        MOV                C,ACC.0
        CPL                C
        MOV                P_LED4,C
L_QuitCheck4:
        MOV                R7,#5
        LCALL              F_check_adc
        MOV                A,R7
        ANL                A,#0FEH
        JNZ                L_QuitCheck5
        MOV                A,R7
        MOV                C,ACC.0
        CPL                C
        MOV                P_LED5,C
L_QuitCheck5:
        MOV                R7,#6
        LCALL              F_check_adc
        MOV                A,R7
        ANL                A,#0FEH
        JNZ                L_QuitCheck6
        MOV                A,R7
        MOV                C,ACC.0
        CPL                C
        MOV                P_LED6,C
L_QuitCheck6:
        MOV                R7,#7
        LCALL              F_check_adc
        MOV                A,R7
        ANL                A,#0FEH
        JNZ                L_QuitCheck7
        MOV                A,R7
        MOV                C,ACC.0
        CPL                C
        MOV                P_LED7,C
L_QuitCheck7:
        RET

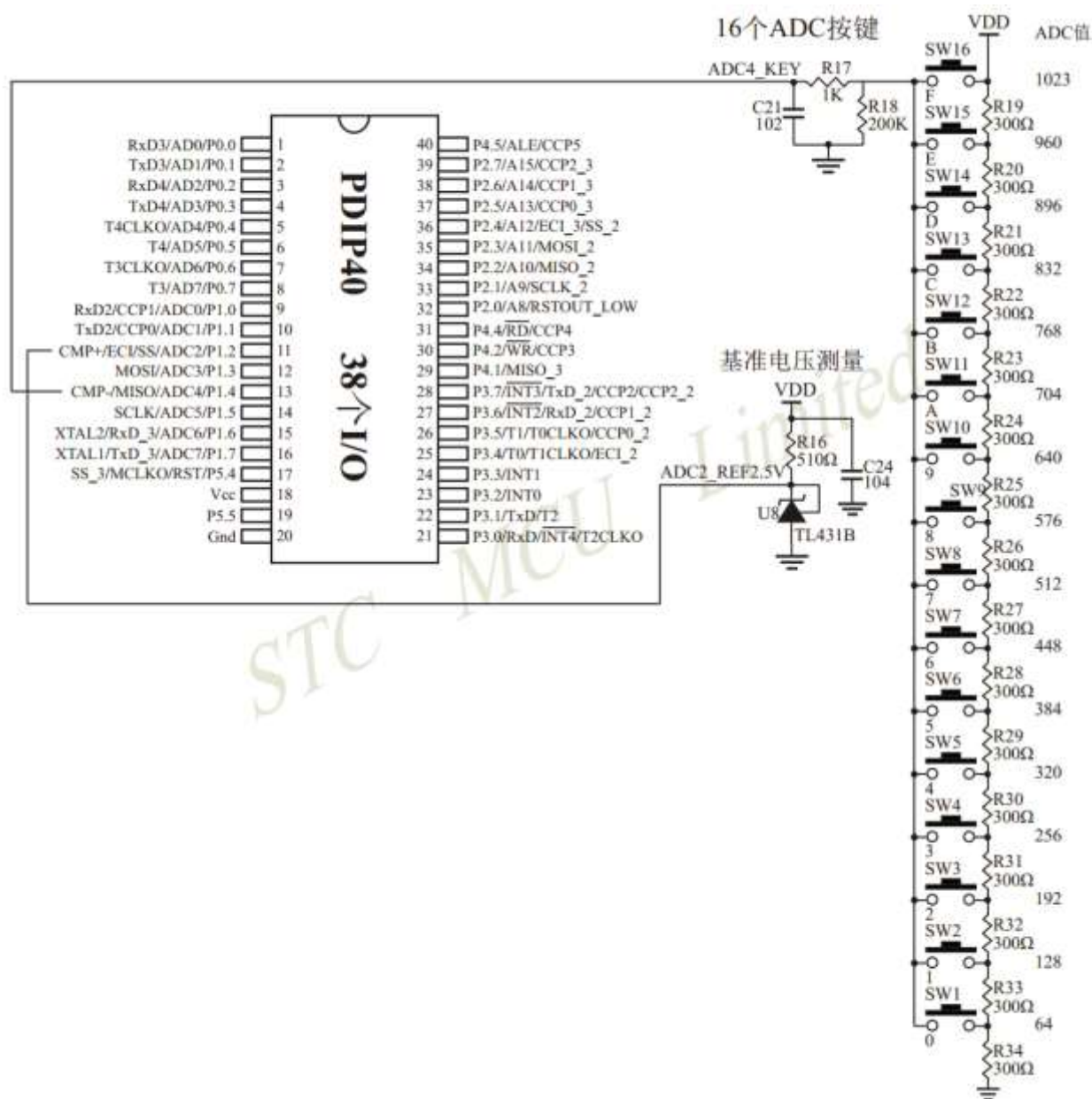
F_delay_ms:
        PUSH                3
        PUSH                4
L_delay_ms_1:
        MOV                R3,#HIGH (Fosc_KHZ / 13)
        MOV                R4,#LOW (Fosc_KHZ / 13)
L_delay_ms_2:
        MOV                A,R4
        DEC                R4
        JNZ                L_delay_ms_3
        DEC                R3
L_delay_ms_3:
        DEC                A
        ORL                A,R3

```

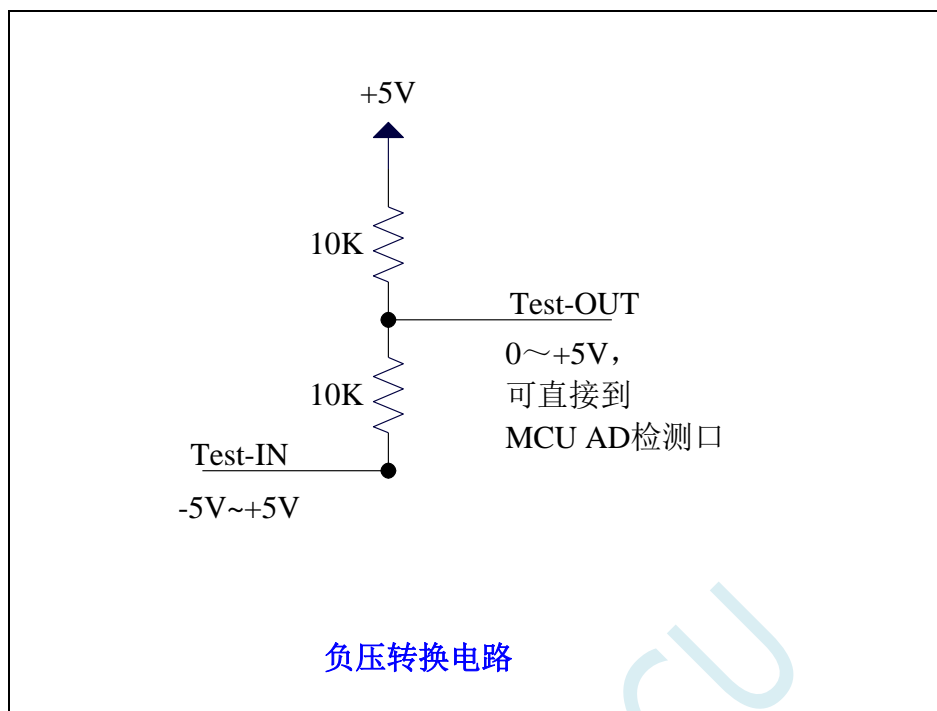
|      |                 |
|------|-----------------|
| JNZ  | L_delay_ms_2    |
| DJNZ | R7,L_delay_ms_1 |
| POP  | 4               |
| POP  | 3               |
| RET  |                 |
| END  |                 |

## 19.6.6 ADC 作按键扫描应用线路图

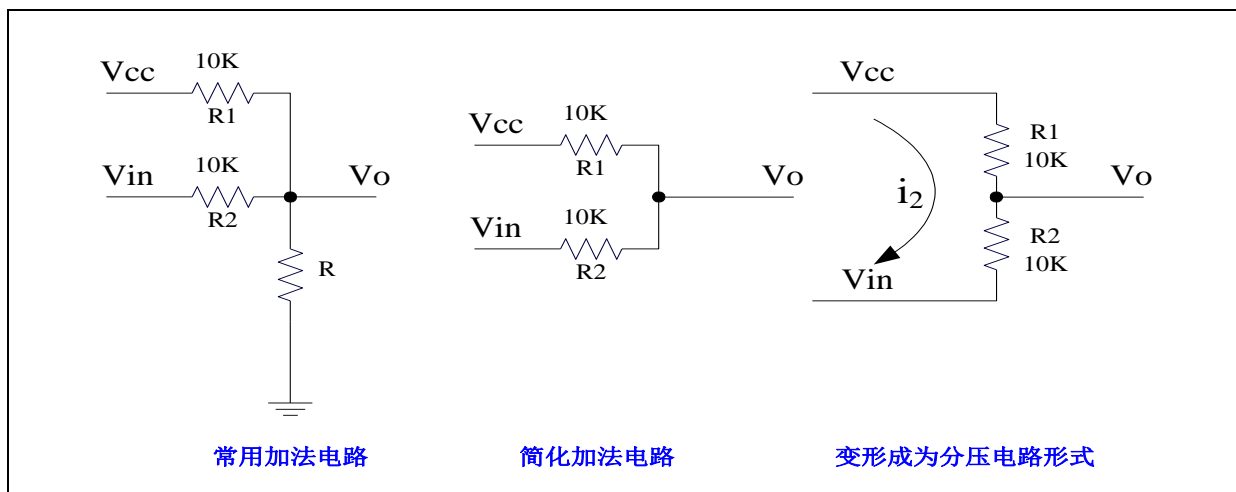
读 ADC 键的方法：每隔 10ms 左右读一次 ADC 值，并且保存最后 3 次的读数，其变化比较小时再判断键。判断键有效时，允许一定的偏差，比如±16 个字的偏差。



### 19.6.7 检测负电压参考线路图



## 19.6.8 常用加法电路在 ADC 中的应用



参照分压电路得到公式 1

公式 1:  $V_o = V_{in} + i_2 * R_2$

公式 2:  $i_2 = (V_{cc} - V_{in}) / (R_1 + R_2)$  { 条件: 流向  $V_o$  的电流  $\approx 0$  }

将  $R_1=R_2$  代入公式 2 得公式 3

公式 3:  $i_2 = (V_{cc} - V_{in}) / 2R_2$

将公式 3 代入公式 1 得公式 4

公式 4:  $V_o = (V_{cc} + V_{in}) / 2$

根据公式 4, 可以将以上电路看成加法电路。

在单片机的模数转换测量中, 要求被测电压大于 0 并且小于  $V_{CC}$ 。如果被测电压小于 0V, 可以利用加法电路将被测电压提升到 0V 以上。此时对被测电压的变化范围有一定的要求:

把上述条件代入公式 4 可得到下面 2 式

$(V_{cc} + V_{in}) / 2 > 0$  即  $V_{in} > -V_{cc}$

$(V_{cc} + V_{in}) / 2 < V_{cc}$  即  $V_{in} < V_{cc}$

上面 2 式可以合起来:  **$-V_{cc} < V_{in} < V_{cc}$**

## 20 同步串行外设接口 SPI

| 产品线                    | SPI | 快速 SPI<br>(SPR=11B 时, SPI 时钟为系统时钟/2) |
|------------------------|-----|--------------------------------------|
| STC8H1K08 系列           | ●   |                                      |
| STC8H1K28 系列           | ●   |                                      |
| STC8H3K64S4 系列 A 版本    | ●   |                                      |
| STC8H3K64S4 系列 A 版本    | ●   |                                      |
| STC8H3K64S2 系列 B 版本    |     | ●                                    |
| STC8H3K64S4 系列 B 版本    |     | ●                                    |
| STC8H8K64U 系列 A 版本     | ●   |                                      |
| STC8H8K64U 系列 B/C/D 版本 |     | ●                                    |
| STC8H4K64TL 系列         |     | ●                                    |
| STC8H4K64TLCD 系列       |     | ●                                    |
| STC8H1K08T 系列          |     | ●                                    |

STC8H 系列单片机内部集成了一种高速串行通信接口——SPI 接口。SPI 是一种全双工的高速同步通信总线。STC8H 系列集成的 SPI 接口提供了两种操作模式：主模式和从模式。

### 20.1 SPI 功能脚切换

| 符号    | 地址  | B7        | B6 | B5 | B4 | B3         | B2 | B1 | B0 |
|-------|-----|-----------|----|----|----|------------|----|----|----|
| P_SW1 | A2H | S1_S[1:0] |    | -  | -  | SPI_S[1:0] |    | 0  | -  |

SPI\_S[1:0]: SPI 功能脚选择位

| SPI_S[1:0] | SS                                      | MOSI | MISO | SCLK |
|------------|---|------|------|------|
| 00         | <a href="#">P1.2/P5.4<sup>[1]</sup></a> | P1.3 | P1.4 | P1.5 |
| 01         | P2.2                                    | P2.3 | P2.4 | P2.5 |
| 10         | P5.4                                    | P4.0 | P4.1 | P4.3 |
| 11         | P3.5                                    | P3.4 | P3.3 | P3.2 |

注<sup>[1]</sup>：对于部分没有 P1.2 口的单片机型号，此功能在 P5.4 口上

### 20.2 SPI 相关的寄存器

| 符号     | 描述        | 地址  | 位地址与符号 |      |      |      |      |      |          |    | 复位值       |
|--------|-----------|-----|--------|------|------|------|------|------|----------|----|-----------|
|        |           |     | B7     | B6   | B5   | B4   | B3   | B2   | B1       | B0 |           |
| SPSTAT | SPI 状态寄存器 | CDH | SPIF   | WCOL | -    | -    | -    | -    | -        | -  | 00xx,xxxx |
| SPCTL  | SPI 控制寄存器 | CEH | SSIG   | SPEN | DORD | MSTR | CPOL | CPHA | SPR[1:0] |    | 0000,0100 |
| SPDAT  | SPI 数据寄存器 | CFH |        |      |      |      |      |      |          |    | 0000,0000 |

#### 20.2.1 SPI 状态寄存器（SPSTAT）

| 符号     | 地址  | B7   | B6   | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-----|------|------|----|----|----|----|----|----|
| SPSTAT | CDH | SPIF | WCOL | -  | -  | -  | -  | -  | -  |

**SPIF: SPI 中断标志位。**

当发送/接收完成 1 字节的数据后, 硬件自动将此位置 1, 并向 CPU 提出中断请求。当 SPI 处于主机模式且 SSIG 位被设置为 0 时, 如果 SS 管脚的输入电平被驱动为低电平时, 此标志位也会被硬件自动置 1, 以标志设备模式发生变化。

注意: 此标志位必须用户通过软件方式向此位写 1 进行清零。

**WCOL: SPI 写冲突标志位。**

当 SPI 在进行数据传输的过程中写 SPDAT 寄存器时, 硬件将此位置 1。

注意: 此标志位必须用户通过软件方式向此位写 1 进行清零。

## 20.2.2 SPI 控制寄存器 (SPCTL), SPI 速度控制

| 符号    | 地址  | B7   | B6   | B5   | B4   | B3   | B2   | B1       | B0 |
|-------|-----|------|------|------|------|------|------|----------|----|
| SPCTL | CEH | SSIG | SPEN | DORD | MSTR | CPOL | CPHA | SPR[1:0] |    |

**SSIG: SS 引脚功能控制位**

0: SS 引脚确定器件是主机还是从机

1: 忽略 SS 引脚功能, 使用 MSTR 确定器件是主机还是从机

**SPEN: SPI 使能控制位**

0: 关闭 SPI 功能

1: 使能 SPI 功能

**DORD: SPI 数据位发送/接收的顺序**

0: 先发送/接收数据的高位 (MSB)

1: 先发送/接收数据的低位 (LSB)

**MSTR: 器件主/从模式选择位**

设置主机模式:

若 SSIG=0, 则 SS 管脚必须为高电平且设置 MSTR 为 1

若 SSIG=1, 则只需要设置 MSTR 为 1 (忽略 SS 管脚的电平)

设置从机模式:

若 SSIG=0, 则 SS 管脚必须为低电平 (与 MSTR 位无关)

若 SSIG=1, 则只需要设置 MSTR 为 0 (忽略 SS 管脚的电平)

**CPOL: SPI 时钟极性控制**

0: SCLK 空闲时为低电平, SCLK 的前时钟沿为上升沿, 后时钟沿为下降沿

1: SCLK 空闲时为高电平, SCLK 的前时钟沿为下降沿, 后时钟沿为上升沿

**CPHA: SPI 时钟相位控制**

0: 数据 SS 管脚为低电平驱动第一位数据并在 SCLK 的后时钟沿改变数据, 前时钟沿采样数据 (必须 SSIG=0)

1: 数据在 SCLK 的前时钟沿驱动, 后时钟沿采样

**SPR[1:0]: SPI 时钟频率选择**

| SPR[1:0] | SCLK 频率   | 快速 SPI 的 SCLK 频率 |
|----------|-----------|------------------|
| 00       | SYSClk/4  | SYSClk/4         |
| 01       | SYSClk/8  | SYSClk/8         |
| 10       | SYSClk/16 | SYSClk/16        |
| 11       | SYSClk/32 | SYSClk/2         |



### 20.2.3 SPI 数据寄存器 (SPDAT)

| 符号    | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------|-----|----|----|----|----|----|----|----|----|
| SPDAT | CFH |    |    |    |    |    |    |    |    |

SPI 发送/接收数据缓冲器。

STC MCU

## 20.3 SPI 通信方式

SPI 的通信方式通常有 3 种: 单主单从 (一个主机设备连接一个从机设备)、互为主从 (两个设备连接, 设备和互为主机和从机)、单主多从 (一个主机设备连接多个从机设备)

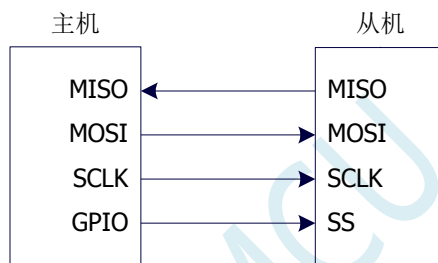
### 20.3.1 单主单从

两个设备相连, 其中一个设备固定作为主机, 另外一个固定作为从机。

主机设置: SSIG 设置为 1, MSTR 设置为 1, 固定为主机模式。主机可以使用任意端口连接从机的 SS 管脚, 拉低从机的 SS 脚即可使能从机

从机设置: SSIG 设置为 0, SS 管脚作为从机的片选信号。

单主单从连接配置图如下所示:



单主单从配置

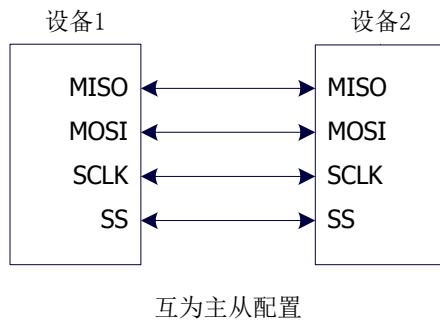
### 20.3.2 互为主从

两个设备相连, 主机和从机不固定。

设置方法 1: 两个设备初始化时都设置为 SSIG 设置为 0, MSTR 设置为 1, 且将 SS 脚设置为双向口模式输出高电平。此时两个设备都是不忽略 SS 的主机模式。当其中一个设备需要启动传输时, 可将自己的 SS 脚设置为输出模式并输出低电平, 拉低对方的 SS 脚, 这样另一个设备就被强行设置为从机模式了。

设置方法 2: 两个设备初始化时都将自己设置成忽略 SS 的从机模式, 即将 SSIG 设置为 1, MSTR 设置为 0。当其中一个设备需要启动传输时, 先检测 SS 管脚的电平, 如果时候高电平, 就将自己设置成忽略 SS 的主模式, 即可进行数据传输了。

互为主从连接配置图如下所示:



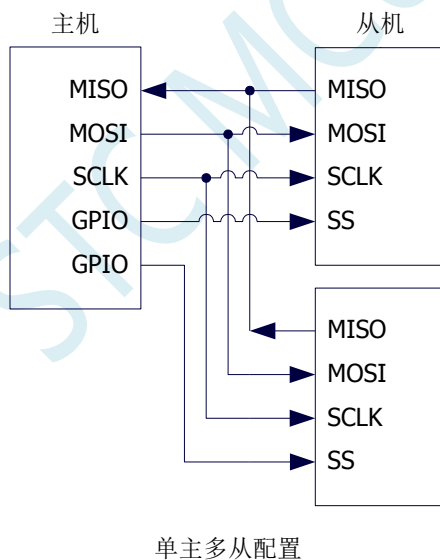
### 20.3.3 单主多从

多个设备相连，其中一个设备固定作为主机，其他设备固定作为从机。

主机设置：SSIG 设置为 1，MSTR 设置为 1，固定为主机模式。主机可以使用任意端口分别连接各个从机的 SS 管脚，拉低其中一个从机的 SS 脚即使能相应的从机设备

从机设置：SSIG 设置为 0，SS 管脚作为从机的片选信号。

单主多从连接配置图如下所示：



## 20.4 配置 SPI

| 控制位  |      |      | 通信端口 |      |      |      | 说明   |
|------|------|------|------|------|------|------|--|
| SPEN | SSIG | MSTR | SS   | MISO | MOSI | SCLK |  |
| 0    | x    | x    | x    | 输入   | 输入   | 输入   | 关闭 SPI 功能, SS/MOSI/MISO/SCLK 均为普通 IO                                       |
| 1    | 0    | 0    | 0    | 输出   | 输入   | 输入   | 从机模式, 且被选中   |
| 1    | 0    | 0    | 1    | 高阻   | 输入   | 输入   | 从机模式, 但未被选中  |
| 1    | 0    | 1→0  | 0    | 输出   | 输入   | 输入   | 从机模式, 不忽略 SS 且 MSTR 为 1 的主机模式, 当 SS 管脚被拉低时, MSTR 将被硬件自动清零, 工作模式将被被动设置为从机模式 |
| 1    | 0    | 1    | 1    | 输入   | 高阻   | 高阻   | 主机模式, 空闲状态   |
|      |      |      |      |      | 输出   | 输出   | 主机模式, 激活状态   |
| 1    | 1    | 0    | x    | 输出   | 输入   | 输入   | 从机模式   |
| 1    | 1    | 1    | x    | 输入   | 输出   | 输出   | 主机模式   |

### 从机模式的注意事项:

当 CPHA=0 时, SSIG 必须为 0 (即不能忽略 SS 脚)。在每次串行字节开始还发送前 SS 脚必须拉低, 并且在串行字节发送完后须重新设置为高电平。SS 管脚为低电平时不能对 SPDAT 寄存器执行写操作, 否则将导致一个写冲突错误。CPHA=0 且 SSIG=1 时的操作未定义。

当 CPHA=1 时, SSIG 可以置 1 (即可以忽略脚)。如果 SSIG=0, SS 脚可在连续传输之间保持低有效 (即一直固定为低电平)。这种方式适用于固定单主单从的系统。

### 主机模式的注意事项:

在 SPI 中, 传输总是由主机启动的。如果 SPI 使能 (SPEN=1) 并选择作为主机时, 主机对 SPI 数据寄存器 SPDAT 的写操作将启动 SPI 时钟发生器和数据的传输。在数据写入 SPDAT 之后的半个到一个 SPI 位时间后, 数据将出现在 MOSI 脚。写入主机 SPDAT 寄存器的数据从 MOSI 脚移出发送到从机的 MOSI 脚。同时从机 SPDAT 寄存器的数据从 MISO 脚移出发送到主机的 MISO 脚。

传输完一个字节后, SPI 时钟发生器停止, 传输完成标志 (SPIF) 置位, 如果 SPI 中断使能则会产生一个 SPI 中断。主机和从机 CPU 的两个移位寄存器可以看作是一个 16 位循环移位寄存器。当数据从主机移位传送到从机的同时, 数据也以相反的方向移入。这意味着在一个移位周期中, 主机和从机的数据相互交换。

### 通过 SS 改变模式

如果 SPEN=1, SSIG=0 且 MSTR=1, SPI 使能为主机模式, 并将 SS 脚可配置为输入模式或准双向口模式。这种情况下, 另外一个主机可将该脚驱动为低电平, 从而将该器件选择为 SPI 从机并向其发送数据。为了避免争夺总线, SPI 系统将该从机的 MSTR 清零, MOSI 和 SCLK 强制变为输入模式, 而 MISO 则变为输出模式, 同时 SPSTAT 的 SPIF 标志位置 1。

用户软件必须一直对 MSTR 位进行检测, 如果该位被一个从机选择动作而被动清零, 而用户想继续将 SPI 作为主机, 则必须重新设置 MSTR 位, 否则将一直处于从机模式。

### 写冲突

SPI 在发送时为单缓冲，在接收时为双缓冲。这样在前一次发送尚未完成之前，不能将新的数据写入移位寄存器。当发送过程中对数据寄存器 SPDAT 进行写操作时，WCOL 位将被置 1 以指示发生数据写冲突错误。在这种情况下，当前发送的数据继续发送，而新写入的数据将丢失。

当对主机或从机进行写冲突检测时，主机发生写冲突的情况是很罕见的，因为主机拥有数据传输的完全控制权。但从机有可能发生写冲突，因为当主机启动传输时，从机无法进行控制。

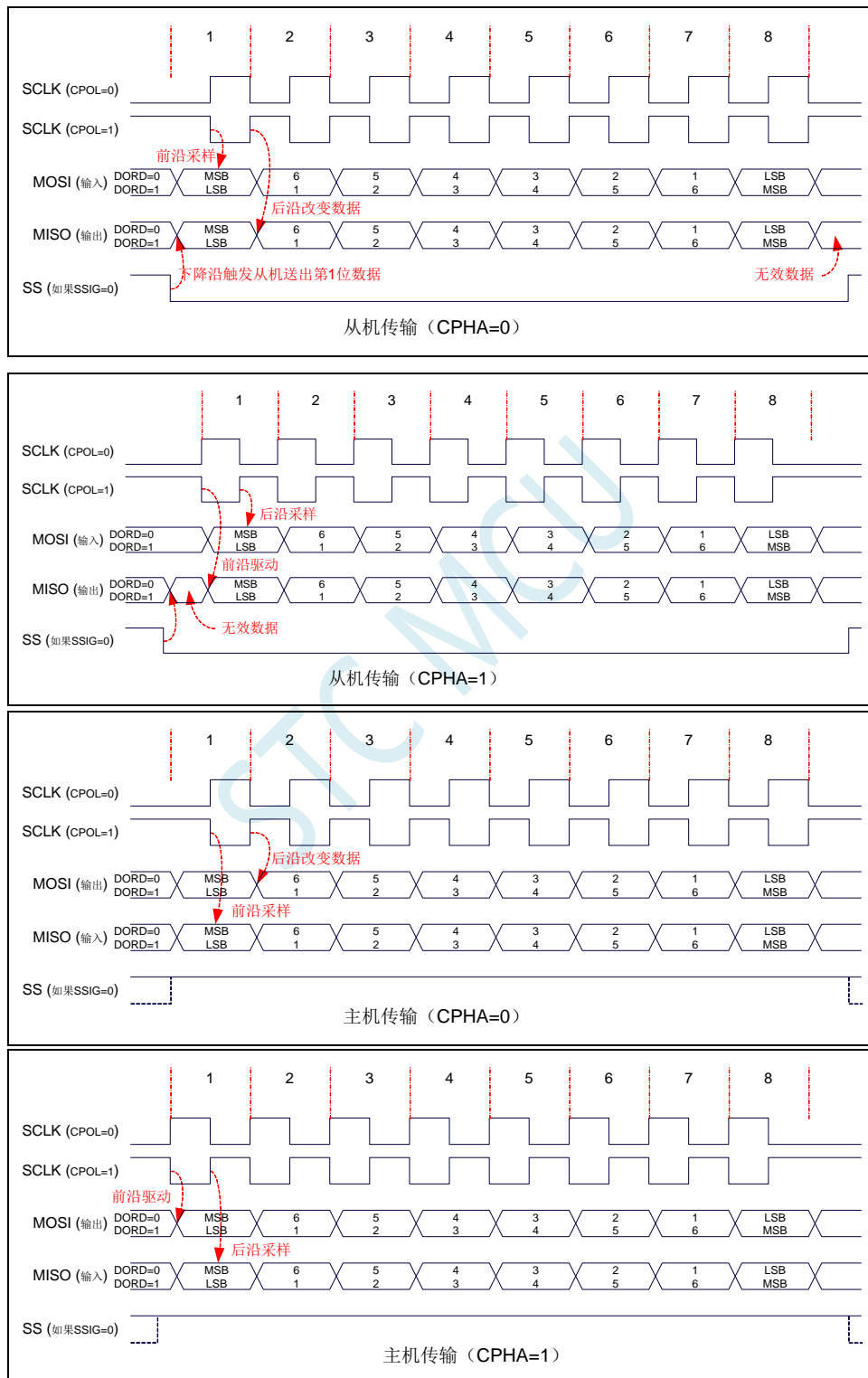
接收数据时，接收到的数据传送到一个并行读数据缓冲区，这样将释放移位寄存器以进行下一个数据的接收。但必须在下个字符完全移入之前从数据寄存器中读出接收到的数据，否则，前一个接收数据将丢失。

WCOL 可通过软件向其写入“1”清零。

STC MCU

## 20.5 数据模式

SPI 的时钟相位控制位 **CPHA** 可以让用户设定数据采样和改变时的时钟沿。时钟极性位 **CPOL** 可以让用户设定时钟极性。下面图例显示了不同时钟相位、极性设置下 **SPI** 通讯时序。



## 20.6 范例程序

### 20.6.1 SPI 单主单从系统主机程序（中断方式）

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
sbit    SS          =   P1^0;
sbit    LED         =   P1^1;
```

```
bit      busy;
```

```
void SPI_Isr() interrupt 9
```

```
{
    SPSTAT = 0xc0;           //清中断标志
    SS = 1;                  //拉高从机的 SS 管脚
    busy = 0;
    LED = !LED;              //测试端口
}
```

```
void main()
```

```
{
    P_SW2 /= 0x80;           //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    LED = 1;
    SS = 1;
    busy = 0;

    SPCTL = 0x50;            //使能 SPI 主机模式
    SPSTAT = 0xc0;          //清中断标志
    IE2 = ESPI;              //使能 SPI 中断
    EA = 1;

    while (1)
    {
        while (busy);
        busy = 1;
        SS = 0;              //拉低从机 SS 管脚
        SPDAT = 0x5a;        //发送测试数据
    }
}
```

```

    }
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

*P\_SW2          DATA          0BAH*

*SPSTAT        DATA          0CDH*

*SPCTL         DATA          0CEH*

*SPDAT         DATA          0CFH*

*IE2            DATA          0AFH*

*ESPI          EQU            02H*

*BUSY          BIT            20H.0*

*SS            BIT            P1.0*

*LED          BIT            P1.1*

*P1M1         DATA          091H*

*P1M0         DATA          092H*

*P0M1         DATA          093H*

*P0M0         DATA          094H*

*P2M1         DATA          095H*

*P2M0         DATA          096H*

*P3M1         DATA          0B1H*

*P3M0         DATA          0B2H*

*P4M1         DATA          0B3H*

*P4M0         DATA          0B4H*

*P5M1         DATA          0C9H*

*P5M0         DATA          0CAH*

*ORG            0000H*

*LJMP           MAIN*

*ORG            004BH*

*LJMP           SPIISR*

*ORG            0100H*

*SPIISR:*

*MOV           SPSTAT,#0C0H*

;清中断标志

*SETB          SS*

;拉高从机的 SS 管脚

*CLR           BUSY*

*CPL           LED*

*RETI*

*MAIN:*

*MOV           SP, #5FH*

*ORL           P\_SW2,#80H*

;使能访问 XFR

*MOV           P0M0, #00H*

*MOV           P0M1, #00H*

*MOV           P1M0, #00H*

*MOV           P1M1, #00H*

*MOV           P2M0, #00H*

*MOV           P2M1, #00H*

*MOV           P3M0, #00H*

*MOV           P3M1, #00H*

*MOV           P4M0, #00H*

*MOV           P4M1, #00H*



```

MOV     P5M0, #00H
MOV     P5M1, #00H

SETB    LED
SETB    SS
CLR     BUSY

MOV     SPCTL, #50H           ;使能 SPI 主机模式
MOV     SPSTAT, #0C0H        ;清中断标志
MOV     IE2, #ESPI           ;使能 SPI 中断
SETB    EA

LOOP:
JB      BUSY, $
SETB    BUSY
CLR     SS                   ;拉低从机 SS 管脚
MOV     SPDAT, #5AH          ;发送测试数据
JMP     LOOP

END

```

## 20.6.2 SPI 单主单从系统从机程序（中断方式）

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

sbit    LED      =    P1^1;

```

```

void SPI_Isr() interrupt 9
{
    SPSTAT = 0xc0;
    SPDAT = SPDAT;
    LED = !LED;
}

```

```

//清中断标志
//将接收到的数据回传给主机
//测试端口

```

```

void main()
{

```

```

    P_SW2 /= 0x80;           //使能访问 XFR

```

```

P0M0 = 0x00;
P0M1 = 0x00;
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

```

```
SPCTL = 0x40; //使能SPI 从机模式
SPSTAT = 0xc0; //清中断标志
IE2 = ESPI; //使能SPI 中断
EA = 1;

while (1);
}
```

汇编代码

;测试工作频率为 11.0592MHz

```
P_SW2      DATA      0BAH

SPSTAT      DATA      0CDH
SPCTL       DATA      0CEH
SPDAT       DATA      0CFH
IE2         DATA      0AFH
ESPI        EQU        02H

LED         BIT        P1.1

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

ORG         0000H
LJMP        MAIN
ORG         004BH
LJMP        SPIISR

SPIISR:     ORG         0100H

MOV         SPSTAT,#0C0H ;清中断标志
MOV         SPDAT,SPDAT ;将接收到的数据回传给主机
CPL         LED
RETI

MAIN:      MOV         SP,#5FH
ORL         P_SW2,#80H ;使能访问 XFR

MOV         P0M0,#00H
MOV         P0M1,#00H
MOV         P1M0,#00H
MOV         P1M1,#00H
MOV         P2M0,#00H
MOV         P2M1,#00H
MOV         P3M0,#00H
MOV         P3M1,#00H
```

```

MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      SPCTL, #40H      ;使能 SPI 从机模式
MOV      SPSTAT, #0C0H    ;清中断标志
MOV      IE2, #ESPI      ;使能 SPI 中断
SETB     EA

JMP      $

END

```

## 20.6.3 SPI 单主单从系统主机程序（查询方式）

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

sbit      SS          =   P1^0;
sbit      LED         =   P1^1;

void main()
{
    P_SW2 /= 0x80;      //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    LED = 1;
    SS = 1;

    SPCTL = 0x50;      //使能 SPI 主机模式
    SPSTAT = 0xc0;     //清中断标志

    while (1)
    {
        SS = 0;      //拉低从机 SS 管脚
        SPDAT = 0x5a; //发送测试数据
        while (!(SPSTAT & 0x80)); //查询完成标志
        SPSTAT = 0xc0; //清中断标志
        SS = 1;      //拉高从机的 SS 管脚
    }
}

```

```

        LED = !LED;
    }
}
//测试端口

```

## 汇编代码

;测试工作频率为 11.0592MHz

```

P_SW2      DATA      0BAH

SPSTAT      DATA      0CDH
SPCTL       DATA      0CEH
SPDAT       DATA      0CFH
IE2         DATA      0AFH
ESPI        EQU        02H

SS          BIT        P1.0
LED         BIT        P1.1

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

            ORG         0000H
            LJMP        MAIN

MAIN:       ORG         0100H

            MOV         SP, #5FH
            ORL         P_SW2, #80H           ;使能访问 XFR

            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            SETB        LED
            SETB        SS

            MOV         SPCTL, #50H          ;使能 SPI 主机模式
            MOV         SPSTAT, #0C0H        ;清中断标志

```

LOOP:

```

        CLR        SS                ;拉低从机 SS 管脚
        MOV        SPDAT,#5AH       ;发送测试数据
        MOV        A,SPSTAT          ;查询完成标志
        JNB        ACC.7,$-2
        MOV        SPSTAT,#0C0H     ;清中断标志
        SETB       SS
        CPL        LED
        JMP        LOOP

```

END

## 20.6.4 SPI 单主单从系统从机程序（查询方式）

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

sbit      LED      =    P1^1;

```

```

void SPI_Isr() interrupt 9
{
    SPSTAT = 0xc0;
}

```

//清中断标志

```

void main()
{

```

```

    P_SW2 /= 0x80;

```

//使能访问 XFR

```

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

```

```

    SPCTL = 0x40;
    SPSTAT = 0xc0;

```

//使能 SPI 从机模式

//清中断标志

```

    while (1)
    {

```

```

        while (!(SPSTAT & 0x80));
        SPSTAT = 0xc0;
        SPDAT = SPDAT;
        LED = !LED;
    }
}

```

//查询完成标志

//清中断标志

//将接收到的数据回传给主机

//测试端口

## 汇编代码

;测试工作频率为 11.0592MHz

```

P_SW2      DATA      0BAH

SPSTAT      DATA      0CDH
SPCTL       DATA      0CEH
SPDAT       DATA      0CFH
IE2         DATA      0AFH
ESPI        EQU        02H

LED         BIT        P1.1

P1M1        DATA      091H
P1M0        DATA      092H
P0M1        DATA      093H
P0M0        DATA      094H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

                ORG      0000H
                LJMP     MAIN

MAIN:         ORG      0100H

                MOV      SP, #5FH
                ORL      P_SW2, #80H                ;使能访问 XFR

                MOV      P0M0, #00H
                MOV      P0M1, #00H
                MOV      P1M0, #00H
                MOV      P1M1, #00H
                MOV      P2M0, #00H
                MOV      P2M1, #00H
                MOV      P3M0, #00H
                MOV      P3M1, #00H
                MOV      P4M0, #00H
                MOV      P4M1, #00H
                MOV      P5M0, #00H
                MOV      P5M1, #00H

                MOV      SPCTL, #40H                ;使能 SPI 从机模式
                MOV      SPSTAT, #0C0H                ;清中断标志

LOOP:        MOV      A, SPSTAT                ;查询完成标志
                JNB      ACC.7, $-2
                MOV      SPSTAT, #0C0H                ;清中断标志
                MOV      SPDAT, SPDAT                ;将接收到的数据回传给主机
                CPL      LED
                JMP      LOOP

```

---

---

END

---

---

## 20.6.5 SPI 互为主从系统程序（中断方式）

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
sbit    SS        = P1^0;
sbit    LED       = P1^1;
sbit    KEY       = P0^0;
```

```
void SPI_Isr() interrupt 9
```

```
{
    SPSTAT = 0xc0;                                //清中断标志
    if (SPCTL & 0x10)
    {
        SS = 1;                                    //主机模式
        SPCTL = 0x40;                              //拉高从机的 SS 管脚
                                                    //重新设置为从机待机
    }
    else
    {
        SPDAT = SPDAT;                             //从机模式
                                                    //将接收到的数据回传给主机
    }
    LED = !LED;                                     //测试端口
}
```

```
void main()
```

```
{
    P_SW2 |= 0x80; //使能访问 XFR
```

```
P0M0 = 0x00;
P0M1 = 0x00;
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;
```

```
LED = 1;
KEY = 1;
SS = 1;
```

```
SPCTL = 0x40;
SPSTAT = 0xc0;
IE2 = ESPI;
EA = 1;
```

```
//使能 SPI 从机模式进行待机
//清中断标志
//使能 SPI 中断
```

```
while (1)
{
    if (!KEY)                //等待按键触发
    {
        SPCTL = 0x50;        //使能SPI 主机模式
        SS = 0;              //拉低从机SS 管脚
        SPDAT = 0x5a;        //发送测试数据
        while (!KEY);        //等待按键释放
    }
}
```

汇编代码

;测试工作频率为11.0592MHz

|         |      |              |               |
|---------|------|--------------|---------------|
| P_SW2   | DATA | 0BAH         |               |
| SPSTAT  | DATA | 0CDH         |               |
| SPCTL   | DATA | 0CEH         |               |
| SPDAT   | DATA | 0CFH         |               |
| IE2     | DATA | 0AFH         |               |
| ESPI    | EQU  | 02H          |               |
| SS      | BIT  | P1.0         |               |
| LED     | BIT  | P1.1         |               |
| KEY     | BIT  | P0.0         |               |
| P1M1    | DATA | 091H         |               |
| P1M0    | DATA | 092H         |               |
| P0M1    | DATA | 093H         |               |
| P0M0    | DATA | 094H         |               |
| P2M1    | DATA | 095H         |               |
| P2M0    | DATA | 096H         |               |
| P3M1    | DATA | 0B1H         |               |
| P3M0    | DATA | 0B2H         |               |
| P4M1    | DATA | 0B3H         |               |
| P4M0    | DATA | 0B4H         |               |
| P5M1    | DATA | 0C9H         |               |
| P5M0    | DATA | 0CAH         |               |
|         | ORG  | 0000H        |               |
|         | LJMP | MAIN         |               |
|         | ORG  | 004BH        |               |
|         | LJMP | SPIISR       |               |
|         | ORG  | 0100H        |               |
| SPIISR: |      |              |               |
|         | PUSH | ACC          |               |
|         | MOV  | SPSTAT,#0C0H | ;清中断标志        |
|         | MOV  | A,SPCTL      |               |
|         | JB   | ACC.4,MASTER |               |
| SLAVE:  |      |              |               |
|         | MOV  | SPDAT,SPDAT  | ;将接收到的数据回传给主机 |
|         | JMP  | ISREXIT      |               |
| MASTER: |      |              |               |
|         | SETB | SS           | ;拉高从机的SS 管脚   |
|         | MOV  | SPCTL,#40H   | ;重新设置为从机待机    |



```
ISREXIT:
    CPL        LED
    POP        ACC
    RETI

MAIN:
    MOV        SP, #5FH
    ORL        P_SW2, #80H        ;使能访问 XFR

    MOV        P0M0, #00H
    MOV        P0M1, #00H
    MOV        P1M0, #00H
    MOV        P1M1, #00H
    MOV        P2M0, #00H
    MOV        P2M1, #00H
    MOV        P3M0, #00H
    MOV        P3M1, #00H
    MOV        P4M0, #00H
    MOV        P4M1, #00H
    MOV        P5M0, #00H
    MOV        P5M1, #00H

    SETB       SS
    SETB       LED
    SETB       KEY

    MOV        SPCTL, #40H        ;使能 SPI 从机模式进行待机
    MOV        SPSTAT, #0C0H      ;清中断标志
    MOV        IE2, #ESPI        ;使能 SPI 中断
    SETB       EA

LOOP:
    JB         KEY, LOOP          ;等待按键触发
    MOV        SPCTL, #50H        ;使能 SPI 主机模式
    CLR        SS                ;拉低从机 SS 管脚
    MOV        SPDAT, #5AH        ;发送测试数据
    JNB        KEY, $             ;等待按键释放
    JMP        LOOP

END
```

### 20.6.6 SPI 互为主从系统程序（查询方式）

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

sbit    SS        =    P1^0;
sbit    LED       =    P1^1;
sbit    KEY       =    P0^0;

void main()
{
```

```

P_SW2 |= 0x80; //使能访问 XFR

P0M0 = 0x00;
P0M1 = 0x00;
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

LED = 1;
KEY = 1;
SS = 1;

SPCTL = 0x40; //使能 SPI 从机模式进行待机
SPSTAT = 0xc0; //清中断标志

while (1)
{
    if (!KEY) //等待按键触发
    {
        SPCTL = 0x50; //使能 SPI 主机模式
        SS = 0; //拉低从机 SS 管脚
        SPDAT = 0x5a; //发送测试数据
        while (!KEY); //等待按键释放
    }
    if (SPSTAT & 0x80)
    {
        SPSTAT = 0xc0; //清中断标志
        if (SPCTL & 0x10)
        {
            SS = 1; //主机模式
            SPCTL = 0x40; //拉高从机的 SS 管脚
            //重新设置为从机待机
        }
        else
        {
            SPDAT = SPDAT; //从机模式
            //将接收到的数据回传给主机
        }
        LED = !LED; //测试端口
    }
}
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|        |      |      |
|--------|------|------|
| P_SW2  | DATA | 0BAH |
| SPSTAT | DATA | 0CDH |
| SPCTL  | DATA | 0CEH |
| SPDAT  | DATA | 0CFH |
| IE2    | DATA | 0AFH |
| ESPI   | EQU  | 02H  |

```

SS      BIT      P1.0
LED      BIT      P1.1
KEY      BIT      P0.0

P1M1     DATA    091H
P1M0     DATA    092H
P0M1     DATA    093H
P0M0     DATA    094H
P2M1     DATA    095H
P2M0     DATA    096H
P3M1     DATA    0B1H
P3M0     DATA    0B2H
P4M1     DATA    0B3H
P4M0     DATA    0B4H
P5M1     DATA    0C9H
P5M0     DATA    0CAH

      ORG      0000H
      LJMP     MAIN

MAIN:      ORG      0100H

      MOV      SP, #5FH
      ORL      P_SW2, #80H      ;使能访问 XFR

      MOV      P0M0, #00H
      MOV      P0M1, #00H
      MOV      P1M0, #00H
      MOV      P1M1, #00H
      MOV      P2M0, #00H
      MOV      P2M1, #00H
      MOV      P3M0, #00H
      MOV      P3M1, #00H
      MOV      P4M0, #00H
      MOV      P4M1, #00H
      MOV      P5M0, #00H
      MOV      P5M1, #00H

      SETB     SS
      SETB     LED
      SETB     KEY

      MOV      SPCTL, #40H      ;使能 SPI 从机模式进行待机
      MOV      SPSTAT, #0C0H    ;清中断标志

LOOP:      JB      KEY, SKIP      ;等待按键触发
      MOV      SPCTL, #50H      ;使能 SPI 主机模式
      CLR      SS                ;拉低从机 SS 管脚
      MOV      SPDAT, #5AH      ;发送测试数据
      JNB      KEY, $            ;等待按键释放

SKIP:      MOV      A, SPSTAT
      JNB      ACC.7, LOOP
      MOV      SPSTAT, #0C0H    ;清中断标志
      MOV      A, SPCTL
      JB      ACC.4, MASTER

SLAVE:

```

---

|                |             |                    |                      |
|----------------|-------------|--------------------|----------------------|
|                | <i>MOV</i>  | <i>SPDAT,SPDAT</i> | <i>;将接收到的数据回传给主机</i> |
|                | <i>CPL</i>  | <i>LED</i>         |                      |
|                | <i>JMP</i>  | <i>LOOP</i>        |                      |
| <i>MASTER:</i> |             |                    |                      |
|                | <i>SETB</i> | <i>SS</i>          | <i>;拉高从机的SS 管脚</i>   |
|                | <i>MOV</i>  | <i>SPCTL,#40H</i>  | <i>;重新设置为从机待机</i>    |
|                | <i>CPL</i>  | <i>LED</i>         |                      |
|                | <i>JMP</i>  | <i>LOOP</i>        |                      |
|                | <i>END</i>  |                    |                      |

---

## 21 I<sup>2</sup>C 总线

| 产品线              | I2C |
|------------------|-----|
| STC8H1K08 系列     | ●   |
| STC8H1K28 系列     | ●   |
| STC8H3K64S4 系列   | ●   |
| STC8H3K64S2 系列   | ●   |
| STC8H8K64U 系列    | ●   |
| STC8H4K64TL 系列   | ●   |
| STC8H4K64TLCD 系列 | ●   |
| STC8H1K08T 系列    | ●   |

STC8H 系列的单片机内部集成了一个 I<sup>2</sup>C 串行总线控制器。I<sup>2</sup>C 是一种高速同步通讯总线，通讯使用 SCL（时钟线）和 SDA（数据线）两线进行同步通讯。对于 SCL 和 SDA 的端口分配，STC8 系列的单片机提供了切换模式，可将 SCL 和 SDA 切换到不同的 I/O 口上，以方便用户将一组 I<sup>2</sup>C 总线当作多组进行分时复用。

- 与标准 I<sup>2</sup>C 协议相比较，忽略了如下两种机制：
- 发送起始信号（START）后不进行仲裁
  - 时钟信号（SCL）停留在低电平时不进行超时检测

STC8 系列的 I<sup>2</sup>C 总线提供了两种操作模式：主机模式（SCL 为输出口，发送同步时钟信号）和从机模式（SCL 为输入口，接收同步时钟信号）

**STC 创新：**STC 的 I<sup>2</sup>C 串行总线控制器工作在从机模式时，SDA 管脚的下降沿信号可以唤醒进入掉电模式的 MCU。（注意：由于 I<sup>2</sup>C 传输速度比较快，MCU 唤醒后第一包数据一般是不正确的）

### 21.1 I2C 功能脚切换

| 符号    | 地址  | B7    | B6 | B5         | B4 | B3     | B2   | B1   | B0   |
|-------|-----|-------|----|------------|----|--------|------|------|------|
| P_SW2 | BAH | EAXFR | -  | I2C_S[1:0] |    | CMPO_S | S4_S | S3_S | S2_S |

I2C\_S[1:0]：I<sup>2</sup>C 功能脚选择位

| I2C_S[1:0] | SCL  | SDA  |
|------------|------|------|
| 00         | P1.5 | P1.4 |
| 01         | P2.5 | P2.4 |
| 10         | P7.7 | P7.6 |
| 11         | P3.2 | P3.3 |

### 21.2 I<sup>2</sup>C 相关的寄存器

| 符号 | 描述 | 地址 | 位地址与符号 |    |    |    |    |    |    |    | 复位值 |
|----|----|----|--------|----|----|----|----|----|----|----|-----|
|    |    |    | B7     | B6 | B5 | B4 | B3 | B2 | B1 | B0 |     |

|          |                            |       |               |       |              |      |            |       |        |        |           |
|----------|----------------------------|-------|---------------|-------|--------------|------|------------|-------|--------|--------|-----------|
| I2CCFG   | I <sup>2</sup> C 配置寄存器     | FE80H | ENI2C         | MSSL  | MSSPEED[5:0] |      |            |       |        |        | 0000,0000 |
| I2CMSCR  | I <sup>2</sup> C 主机控制寄存器   | FE81H | EMSI          | -     | -            | -    | MSCMD[3:0] |       |        |        | 0xxx,0000 |
| I2CMSST  | I <sup>2</sup> C 主机状态寄存器   | FE82H | MSBUSY        | MSIF  | -            | -    | -          | -     | MSACKI | MSACKO | 00xx,xx10 |
| I2CSLCR  | I <sup>2</sup> C 从机控制寄存器   | FE83H | -             | ESTAI | ERXI         | ETXI | ESTOI      | -     | -      | SLRST  | x000,0xx0 |
| I2CSLST  | I <sup>2</sup> C 从机状态寄存器   | FE84H | SLBUSY        | STAIF | RXIF         | TXIF | STOIF      | TXING | SLACKI | SLACKO | 0000,0000 |
| I2CSLADR | I <sup>2</sup> C 从机地址寄存器   | FE85H | I2CSLADR[7:1] |       |              |      |            |       |        | MA     | 0000,0000 |
| I2CTXD   | I <sup>2</sup> C 数据发送寄存器   | FE86H |               |       |              |      |            |       |        |        | 0000,0000 |
| I2CRXD   | I <sup>2</sup> C 数据接收寄存器   | FE87H |               |       |              |      |            |       |        |        | 0000,0000 |
| I2CMSAUX | I <sup>2</sup> C 主机辅助控制寄存器 | FE88H | -             | -     | -            | -    | -          | -     | -      | WDTA   | xxxx,xxx0 |

## 21.3 I<sup>2</sup>C 主机模式

### 21.3.1 I2C 配置寄存器 (I2CCFG)，总线速度控制

| 符号     | 地址    | B7    | B6   | B5           | B4 | B3 | B2 | B1 | B0 |
|--------|-------|-------|------|--------------|----|----|----|----|----|
| I2CCFG | FE80H | ENI2C | MSSL | MSSPEED[5:0] |    |    |    |    |    |

ENI2C: I<sup>2</sup>C 功能使能控制位

- 0: 禁止 I<sup>2</sup>C 功能
- 1: 允许 I<sup>2</sup>C 功能

MSSL: I<sup>2</sup>C 工作模式选择位

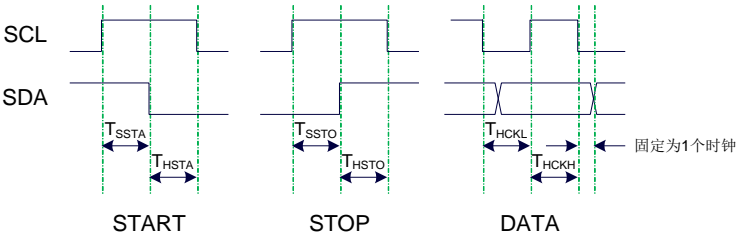
- 0: 从机模式
- 1: 主机模式

MSSPEED[5:0]: I<sup>2</sup>C 总线速度（等待时钟数）控制，I2C 总线速度 =  $F_{OSC} / 2 / (MSSPEED * 2 + 4)$

| MSSPEED[5:0] | 对应的时钟数 |
|--------------|--------|
| 0            | 4      |
| 1            | 6      |
| 2            | 8      |
| ...          | ...    |
| x            | 2x+4   |
| ...          | ...    |
| 62           | 128    |
| 63           | 130    |

只有当 I<sup>2</sup>C 模块工作在主机模式时，MSSPEED 参数设置的等待参数才有效。此等待参数主要用于主机模式的以下几个信号：

- T<sub>SSTA</sub>: 起始信号的建立时间（Setup Time of START）
- T<sub>HSTA</sub>: 起始信号的保持时间（Hold Time of START）
- T<sub>SSTO</sub>: 停止信号的建立时间（Setup Time of STOP）
- T<sub>HSTO</sub>: 停止信号的保持时间（Hold Time of STOP）
- T<sub>HCKL</sub>: 时钟信号的低电平保持时间（Hold Time of SCL Low）
- T<sub>HCKH</sub>: 时钟信号的高电平保持时间（Hold Time of SCL High）



例 1: 当 MSSPEED=10 时， $T_{SSTA}=T_{HSTA}=T_{SSTO}=T_{HSTO}=T_{HCKL}=24/F_{OSC}$

例 2: 当 24MHz 的工作频率下需要 400K 的 I2C 总线速度时，  
 $MSSPEED=(24M / 400K / 2 - 4) / 2=13$

### 21.3.2 I2C 主机控制寄存器 (I2CMSCR)

| 符号      | 地址    | B7   | B6 | B5 | B4 | B3         | B2 | B1 | B0 |
|---------|-------|------|----|----|----|------------|----|----|----|
| I2CMSCR | FE81H | EMSI | -  | -  | -  | MSCMD[3:0] |    |    |    |

EMSI: 主机模式中断使能控制位

- 0: 关闭主机模式的中断
- 1: 允许主机模式的中断

MSCMD[3:0]: 主机命令

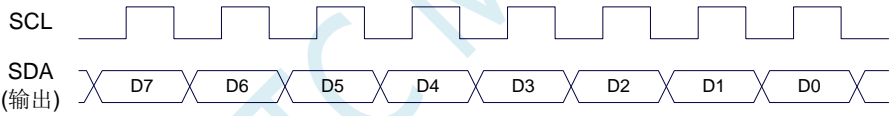
- 0000: 待机, 无动作。
- 0001: 起始命令。

发送 START 信号。如果当前 I<sup>2</sup>C 控制器处于空闲状态, 即 MSBUSY (I2CMSST.7) 为 0 时, 写此命令会使控制器进入忙状态, 硬件自动将 MSBUSY 状态位置 1, 并开始发送 START 信号; 若当前 I<sup>2</sup>C 控制器处于忙状态, 写此命令可触发发送 START 信号。发送 START 信号的波形如下图所示:



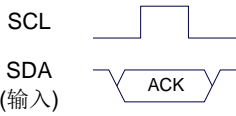
0010: 发送数据命令。

写此命令后, I<sup>2</sup>C 总线控制器会在 SCL 管脚上产生 8 个时钟, 并将 I2CTXD 寄存器里面数据按位送到 SDA 管脚上 (先发送高位数据)。发送数据的波形如下图所示:



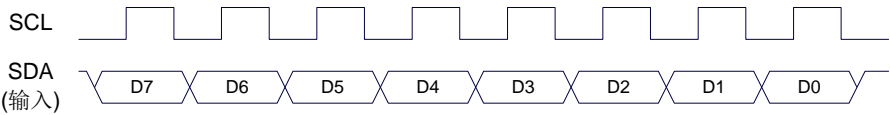
0011: 接收 ACK 命令。

写此命令后, I<sup>2</sup>C 总线控制器会在 SCL 管脚上产生 1 个时钟, 并将从 SDA 端口上读取的数据保存到 MSACKI (I2CMSST.1)。接收 ACK 的波形如下图所示:



0100: 接收数据命令。

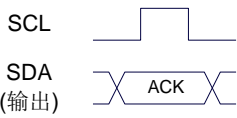
写此命令后, I<sup>2</sup>C 总线控制器会在 SCL 管脚上产生 8 个时钟, 并将从 SDA 端口上读取的数据依次左移到 I2CRXD 寄存器 (先接收高位数据)。接收数据的波形如下图所示:



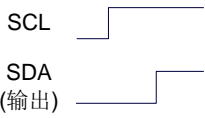
0101: 发送 ACK 命令。

写此命令后, I<sup>2</sup>C 总线控制器会在 SCL 管脚上产生 1 个时钟, 并将 MSACKO (I2CMSST.0) 中的数据发送到 SDA 端口。发送 ACK 的波形如下图所示:





0110: 停止命令。  
发送 STOP 信号。写此命令后，I<sup>2</sup>C 总线控制器开始发送 STOP 信号。信号发送完成后，硬件自动将 MSBUSY 状态位清零。STOP 信号的波形如下图所示：



- 0111: 保留。  
1000: 保留。  
1001: 起始命令+发送数据命令+接收 ACK 命令。  
此命令为命令 0001、命令 0010、命令 0011 三个命令的组合，下此命令后控制器会依次执行这三个命令。  
1010: 发送数据命令+接收 ACK 命令。  
此命令为命令 0010、命令 0011 两个命令的组合，下此命令后控制器会依次执行这两个命令。  
1011: 接收数据命令+发送 ACK(0)命令。  
此命令为命令 0100、命令 0101 两个命令的组合，下此命令后控制器会依次执行这两个命令。  
注意：此命令所返回的应答信号固定为 ACK (0)，不受 MSACKO 位的影响。  
1100: 接收数据命令+发送 NAK(1)命令。  
此命令为命令 0100、命令 0101 两个命令的组合，下此命令后控制器会依次执行这两个命令。  
注意：此命令所返回的应答信号固定为 NAK (1)，不受 MSACKO 位的影响。

21.3.3 I2C 主机辅助控制寄存器（I2CMSAUX）

| 符号       | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0   |
|----------|-------|----|----|----|----|----|----|----|------|
| I2CMSAUX | FE88H | -  | -  | -  | -  | -  | -  | -  | WDTA |

WDTA: 主机模式时 I<sup>2</sup>C 数据自动发送允许位  
0: 禁止自动发送  
1: 使能自动发送  
若自动发送功能被使能，当 MCU 执行完成对 I2CTXD 数据寄存器的写操作后，I<sup>2</sup>C 控制器会自动触发“1010”命令，即自动发送数据并接收 ACK 信号。

21.3.4 I2C 主机状态寄存器（I2CMSST）

| 符号      | 地址    | B7     | B6   | B5 | B4 | B3 | B2 | B1     | B0     |
|---------|-------|--------|------|----|----|----|----|--------|--------|
| I2CMSST | FE82H | MSBUSY | MSIF | -  | -  | -  | -  | MSACKI | MSACKO |

MSBUSY: 主机模式时 I<sup>2</sup>C 控制器状态位（只读位）  
0: 控制器处于空闲状态  
1: 控制器处于忙碌状态  
当 I<sup>2</sup>C 控制器处于主机模式时，在空闲状态下，发送完成 START 信号后，控制器便进入到忙碌状态，忙碌状态会一直维持到成功发送完成 STOP 信号，之后状态会再次恢复到空闲状态。  
MSIF: 主机模式的中断请求位（中断标志位）。当处于主机模式的 I<sup>2</sup>C 控制器执行完成寄存器 I2CMSCR

中 MSCMD 命令后产生中断信号，硬件自动将此位 1，向 CPU 发请求中断，响应中断后 MSIF 位必须用软件清零。

**MSACKI:** 主机模式时，发送“0011”命令到 I2CMSCR 的 MSCMD 位后所接收到的 ACK 数据。（只读位）

**MSACKO:** 主机模式时，准备将要发送出去的 ACK 信号。当发送“0101”命令到 I2CMSCR 的 MSCMD 位后，控制器会自动读取此位的数据当作 ACK 发送到 SDA。

STC MCU

## 21.4 I<sup>2</sup>C 从机模式

### 21.4.1 I<sup>2</sup>C 从机控制寄存器 (I2CSLCR)

| 符号      | 地址    | B7 | B6    | B5   | B4   | B3    | B2 | B1 | B0    |
|---------|-------|----|-------|------|------|-------|----|----|-------|
| I2CSLCR | FE83H | -  | ESTAI | ERXI | ETXI | ESTOI | -  | -  | SLRST |

ESTAI: 从机模式时接收到 START 信号中断允许位

- 0: 禁止从机模式时接收到 START 信号时发生中断
- 1: 使能从机模式时接收到 START 信号时发生中断

ERXI: 从机模式时接收到 1 字节数据后中断允许位

- 0: 禁止从机模式时接收到数据后发生中断
- 1: 使能从机模式时接收到 1 字节数据后发生中断

ETXI: 从机模式时发送完成 1 字节数据后中断允许位

- 0: 禁止从机模式时发送完成数据后发生中断
- 1: 使能从机模式时发送完成 1 字节数据后发生中断

ESTOI: 从机模式时接收到 STOP 信号中断允许位

- 0: 禁止从机模式时接收到 STOP 信号时发生中断
- 1: 使能从机模式时接收到 STOP 信号时发生中断

SLRST: 复位从机模式

### 21.4.2 I<sup>2</sup>C 从机状态寄存器 (I2CSLST)

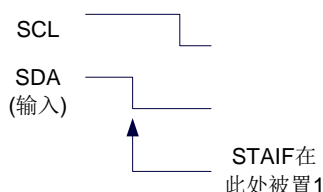
| 符号      | 地址    | B7     | B6    | B5   | B4   | B3    | B2 | B1     | B0     |
|---------|-------|--------|-------|------|------|-------|----|--------|--------|
| I2CSLST | FE84H | SLBUSY | STAIF | RXIF | TXIF | STOIF | -  | SLACKI | SLACKO |

SLBUSY: 从机模式时 I<sup>2</sup>C 控制器状态位 (只读位)

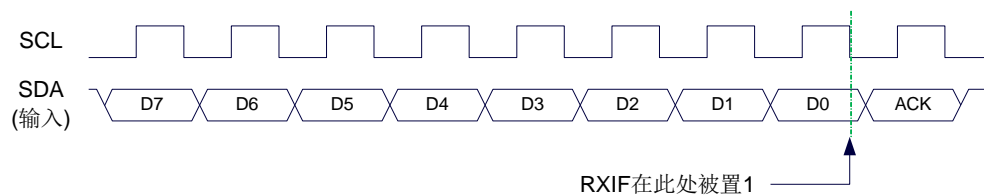
- 0: 控制器处于空闲状态
- 1: 控制器处于忙碌状态

当 I<sup>2</sup>C 控制器处于从机模式时, 在空闲状态下, 接收到主机发送 START 信号后, 控制器会继续检测之后的设备地址数据, 若设备地址与当前 I2CSLADR 寄存器中所设置的从机地址相同时, 控制器便进入到忙碌状态, 忙碌状态会一直维持到成功接收到主机发送 STOP 信号, 之后状态会再次恢复到空闲状态。

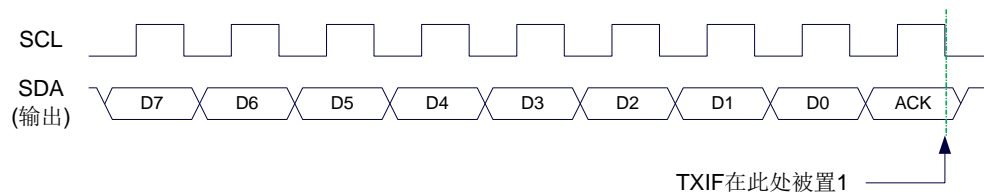
STAIF: 从机模式时接收到 START 信号后的中断请求位。从机模式的 I<sup>2</sup>C 控制器接收到 START 信号后, 硬件会自动将此位置 1, 并向 CPU 发请求中断, 响应中断后 STAIF 位必须用软件清零。STAIF 被置 1 的时间点如下图所示:



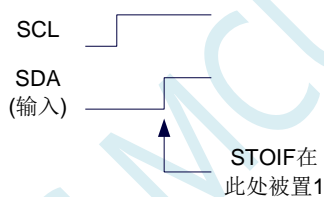
RXIF: 从机模式时接收到 1 字节的数据后的中断请求位。从机模式的 I<sup>2</sup>C 控制器接收到 1 字节的数据后, 在第 8 个时钟的下降沿时硬件会自动将此位置 1, 并向 CPU 发请求中断, 响应中断后 RXIF 位必须用软件清零。RXIF 被置 1 的时间点如下图所示:



**TXIF:** 从机模式时发送完成 1 字节的数据后的中断请求位。从机模式的 I<sup>2</sup>C 控制器发送完成 1 字节的数据并成功接收到 1 位 ACK 信号后，在第 9 个时钟的下降沿时硬件会自动将此位置 1，并向 CPU 发请求中断，响应中断后 TXIF 位必须用软件清零。TXIF 被置 1 的时间点如下图所示：

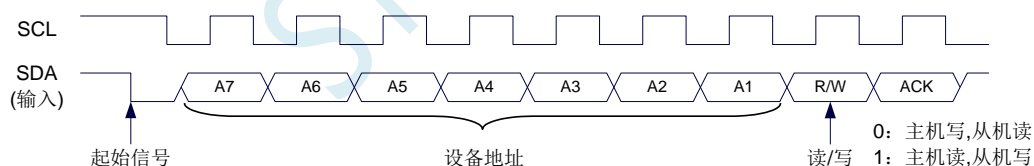


**STOIF:** 从机模式时接收到 STOP 信号后的中断请求位。从机模式的 I<sup>2</sup>C 控制器接收到 STOP 信号后，硬件会自动将此位置 1，并向 CPU 发请求中断，响应中断后 STOIF 位必须用软件清零。STOIF 被置 1 的时间点如下图所示：



**SLACKI:** 从机模式时，接收到的 ACK 数据。

**SLACKO:** 从机模式时，准备将要发送出去的 ACK 信号。



### 21.4.3 I2C 从机地址寄存器 (I2CSLADR)

| 符号       | 地址    | B7            | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----------|-------|---------------|----|----|----|----|----|----|----|
| I2CSLADR | FE85H | I2CSLADR[7:1] |    |    |    |    |    |    | MA |

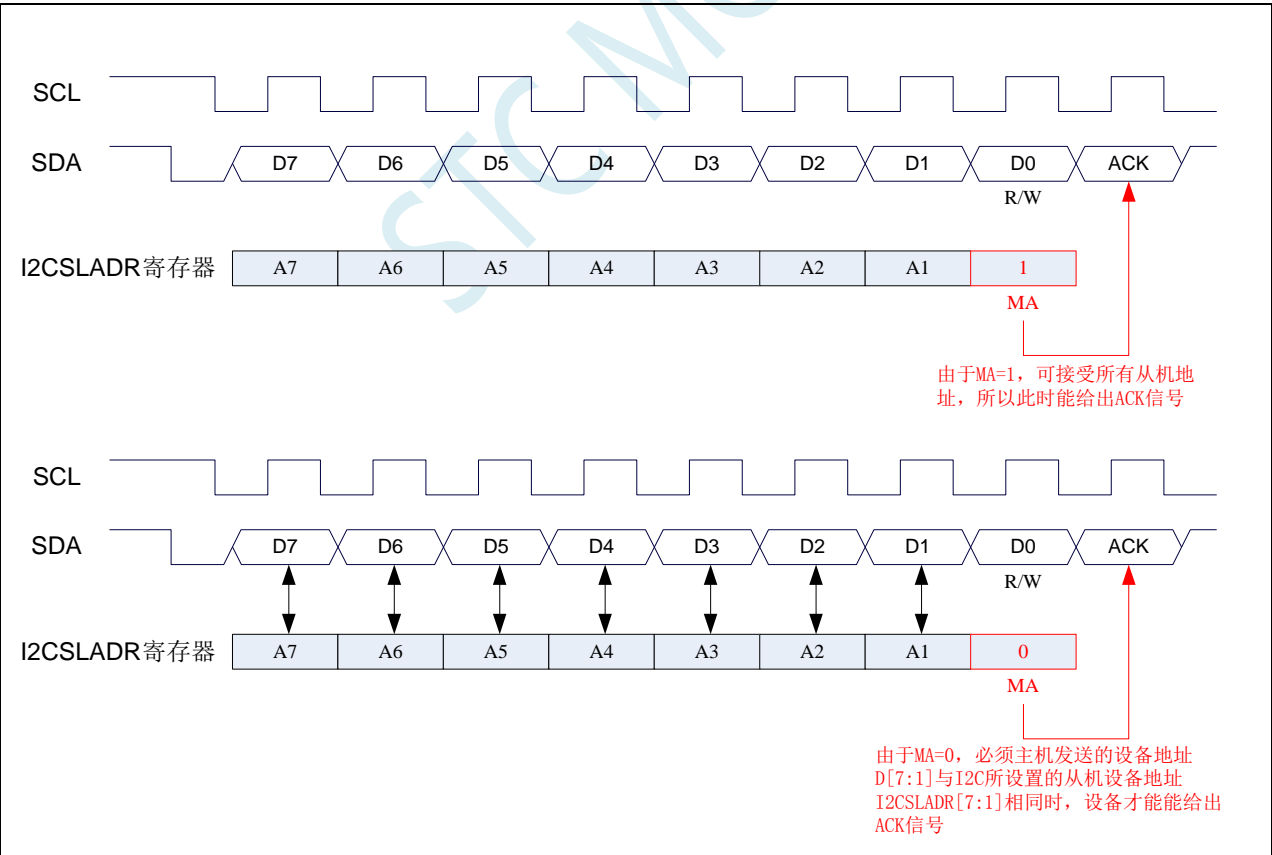
I2CSLADR[7:1]: 从机设备地址

当 I<sup>2</sup>C 控制器处于从机模式时, 控制器在接收到 START 信号后, 会继续检测接下来主机发送出的设备地址数据以及读/写信号。当主机发送出的设备地址与 I2CSLADR[7:1]中所设置的从机设备地址相同时, 控制器才会向 CPU 发出中断求, 请求 CPU 处理 I<sup>2</sup>C 事件; 否则若设备地址不同, I<sup>2</sup>C 控制器继续监控, 等待下一个起始信号, 对下一个设备地址继续比较。

MA: 从机设备地址比较控制

- 0: 设备地址必须与 I2CSLADR[7:1]相同
- 1: 忽略 I2CSLADR[7:1]中的设置, 接受所有的设备地址

**说明:** I2C 总线协议规定 I2C 总线上最多可挂载 128 个 I2C 设备 (理论值), 不同的 I2C 设备用不同的 I2C 从机设备地址进行识别。I2C 主机发送完成起始信号后, 发送的第一个数据 (DATA0) 的高 7 位即为从机设备地址 (DATA0[7:1]为 I2C 设备地址), 最低位为读写信号。当 I2C 设备从机地址寄存器 MA (I2CSLADR.0) 为 1 时, 表示 I2C 从机能够接受所有的设备地址, 此时主机发送的任何设备地址, 即 DATA0[7:1]为任何值, 从机都能响应。当 I2C 设备从机地址寄存器 MA (I2CSLADR.0) 为 0 时, 主机发送的设备地址 DATA0[7:1]必须与从机的设备地址 I2CSLADR[7:1]相同时才能访问此从机设备



#### 21.4.4 I2C 数据寄存器 (I2CTXD, I2CRXD)

| 符号     | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-------|----|----|----|----|----|----|----|----|
| I2CTXD | FE86H |    |    |    |    |    |    |    |    |
| I2CRXD | FE87H |    |    |    |    |    |    |    |    |

I2CTXD 是 I<sup>2</sup>C 发送数据寄存器，存放将要发送的 I<sup>2</sup>C 数据

I2CRXD 是 I<sup>2</sup>C 接收数据寄存器，存放接收完成的 I<sup>2</sup>C 数据

## 21.5 范例程序

### 21.5.1 I<sup>2</sup>C 主机模式访问 AT24C256（中断方式）

#### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
sbit    SDA        =    P1^4;
sbit    SCL        =    P1^5;
```

```
bit      busy;
```

```
void I2C_Isr() interrupt 24
```

```
{
    if (I2CMSST & 0x40)
    {
        I2CMSST &= ~0x40;           //清中断标志
        busy = 0;
    }
}
```

```
void Start()
```

```
{
    busy = 1;
    I2CMSCR = 0x81;                 //发送 START 命令
    while (busy);
}
```

```
void SendData(char dat)
```

```
{
    I2CTXD = dat;                   //写数据到数据缓冲区
    busy = 1;
    I2CMSCR = 0x82;                 //发送 SEND 命令
    while (busy);
}
```

```
void RecvACK()
```

```
{
    busy = 1;
    I2CMSCR = 0x83;                 //发送读 ACK 命令
    while (busy);
}
```

```
char RecvData()
```

```
{
    busy = 1;
    I2CMSCR = 0x84;                 //发送 RECV 命令
    while (busy);
    return I2CRXD;
}
```

```
void SendACK()
```

```

{
    I2CMSST = 0x00;                                //设置ACK 信号
    busy = 1;
    I2CMSCR = 0x85;                                //发送ACK 命令
    while (busy);
}

void SendNAK()
{
    I2CMSST = 0x01;                                //设置NAK 信号
    busy = 1;
    I2CMSCR = 0x85;                                //发送ACK 命令
    while (busy);
}

void Stop()
{
    busy = 1;
    I2CMSCR = 0x86;                                //发送STOP 命令
    while (busy);
}

void Delay()
{
    int i;

    for (i=0; i<3000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P_SW2 /= 0x80;                                //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    I2CCFG = 0xe0;                                //使能I2C 主机模式
    I2CMSST = 0x00;
    EA = 1;

    Start();    //发送起始命令
    SendData(0xa0);                                //发送设备地址+ 写命令
    RecvACK();
}

```



```

    SendData(0x00);                //发送存储地址高字节
    RecvACK();
    SendData(0x00);                //发送存储地址低字节
    RecvACK();
    SendData(0x12);                //写测试数据1
    RecvACK();
    SendData(0x78);                //写测试数据2
    RecvACK();
    Stop();                        //发送停止命令

    Delay();                        //等待设备写数据

    Start();                        //发送起始命令
    SendData(0xa0);                //发送设备地址+写命令
    RecvACK();
    SendData(0x00);                //发送存储地址高字节
    RecvACK();
    SendData(0x00);                //发送存储地址低字节
    RecvACK();
    Start();                        //发送起始命令
    SendData(0xa1);                //发送设备地址+读命令
    RecvACK();
    P0 = RecvData();                //读取数据1
    SendACK();
    P2 = RecvData();                //读取数据2
    SendNAK();
    Stop();                        //发送停止命令

    while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|                 |              |               |
|-----------------|--------------|---------------|
| <b>P_SW2</b>    | <b>DATA</b>  | <b>0BAH</b>   |
|                 |              |               |
| <b>I2CCFG</b>   | <b>XDATA</b> | <b>0FE80H</b> |
| <b>I2CMSCR</b>  | <b>XDATA</b> | <b>0FE81H</b> |
| <b>I2CMSST</b>  | <b>XDATA</b> | <b>0FE82H</b> |
| <b>I2CSLCR</b>  | <b>XDATA</b> | <b>0FE83H</b> |
| <b>I2CSLST</b>  | <b>XDATA</b> | <b>0FE84H</b> |
| <b>I2CSLADR</b> | <b>XDATA</b> | <b>0FE85H</b> |
| <b>I2CTXD</b>   | <b>XDATA</b> | <b>0FE86H</b> |
| <b>I2CRXD</b>   | <b>XDATA</b> | <b>0FE87H</b> |
|                 |              |               |
| <b>SDA</b>      | <b>BIT</b>   | <b>P1.4</b>   |
| <b>SCL</b>      | <b>BIT</b>   | <b>P1.5</b>   |
|                 |              |               |
| <b>BUSY</b>     | <b>BIT</b>   | <b>20H.0</b>  |
|                 |              |               |
| <b>P1M1</b>     | <b>DATA</b>  | <b>091H</b>   |
| <b>P1M0</b>     | <b>DATA</b>  | <b>092H</b>   |
| <b>P0M1</b>     | <b>DATA</b>  | <b>093H</b>   |
| <b>P0M0</b>     | <b>DATA</b>  | <b>094H</b>   |
| <b>P2M1</b>     | <b>DATA</b>  | <b>095H</b>   |
| <b>P2M0</b>     | <b>DATA</b>  | <b>096H</b>   |
| <b>P3M1</b>     | <b>DATA</b>  | <b>0B1H</b>   |
| <b>P3M0</b>     | <b>DATA</b>  | <b>0B2H</b>   |

```

P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          00C3H
          LJMP         I2CISR

I2CISR:   ORG          0100H

          PUSH         ACC
          PUSH         DPL
          PUSH         DPH

          MOV          DPTR,#I2CMSST      ;清中断标志
          MOVX         A,@DPTR
          ANL          A,#NOT 40H
          MOV          DPTR,#I2CMSST
          MOVX         @DPTR,A
          CLR          BUSY              ;复位忙标志

          POP          DPH
          POP          DPL
          POP          ACC
          RETI

START:

          SETB         BUSY
          MOV          A,#10000001B      ;发送 START 命令
          MOV          DPTR,#I2CMSCR
          MOVX         @DPTR,A
          JMP          WAIT

SENDDATA: MOV          DPTR,#I2CTXD      ;写数据到数据缓冲区
          MOVX         @DPTR,A
          SETB         BUSY
          MOV          A,#10000010B      ;发送 SEND 命令
          MOV          DPTR,#I2CMSCR
          MOVX         @DPTR,A
          JMP          WAIT

RECVACK:  SETB         BUSY
          MOV          A,#10000011B      ;发送读 ACK 命令
          MOV          DPTR,#I2CMSCR
          MOVX         @DPTR,A
          JMP          WAIT

RECVDATA: SETB         BUSY
          MOV          A,#10000100B      ;发送 RECV 命令
          MOV          DPTR,#I2CMSCR
          MOVX         @DPTR,A
          CALL         WAIT
          MOV          DPTR,#I2CRXD      ;从数据缓冲区读取数据
          MOVX         A,@DPTR
          RET

SENDACK:  MOV          A,#00000000B      ;设置 ACK 信号

```

```

MOV DPTR,#I2CMSST
MOVX @DPTR,A
SETB BUSY
MOV A,#10000101B ;发送ACK 命令
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

SENDNAK:
MOV A,#00000001B ;设置NAK 信号
MOV DPTR,#I2CMSST
MOVX @DPTR,A
SETB BUSY
MOV A,#10000101B ;发送ACK 命令
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

STOP:
SETB BUSY
MOV A,#10000110B ;发送STOP 命令
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

WAIT:
JB BUSY,$ ;等待命令发送完成
RET

DELAY:
MOV R0,#0
MOV R1,#0

DELAY1:
NOP
NOP
NOP
NOP
DJNZ R1,DELAY1
DJNZ R0,DELAY1
RET

MAIN:
MOV SP,#5FH
ORL P_SW2,#80H ;使能访问XFR

MOV P0M0,#00H
MOV P0M1,#00H
MOV P1M0,#00H
MOV P1M1,#00H
MOV P2M0,#00H
MOV P2M1,#00H
MOV P3M0,#00H
MOV P3M1,#00H
MOV P4M0,#00H
MOV P4M1,#00H
MOV P5M0,#00H
MOV P5M1,#00H

MOV A,#11100000B ;设置I2C 模块为主机模式
MOV DPTR,#I2CCFG
MOVX @DPTR,A
MOV A,#00000000B

```

```

MOV      DPTR,#I2CMSST
MOVX     @DPTR,A
SETB     EA

CALL     START                ;发送起始命令
MOV      A,#0A0H
CALL     SENDDATA             ;发送设备地址+ 写命令
CALL     RECVACK
MOV      A,#000H              ;发送存储地址高字节
CALL     SENDDATA
CALL     RECVACK
MOV      A,#000H              ;发送存储地址低字节
CALL     SENDDATA
CALL     RECVACK
MOV      A,#12H               ;写测试数据 1
CALL     SENDDATA
CALL     RECVACK
MOV      A,#78H               ;写测试数据 2
CALL     SENDDATA
CALL     RECVACK
CALL     STOP                 ;发送停止命令

CALL     DELAY                ;等待设备写数据

CALL     START                ;发送起始命令
MOV      A,#0A0H              ;发送设备地址+ 写命令
CALL     SENDDATA
CALL     RECVACK
MOV      A,#000H              ;发送存储地址高字节
CALL     SENDDATA
CALL     RECVACK
MOV      A,#000H              ;发送存储地址低字节
CALL     SENDDATA
CALL     RECVACK
CALL     START                ;发送起始命令
MOV      A,#0A1H              ;发送设备地址+ 读命令
CALL     SENDDATA
CALL     RECVACK
CALL     RECVDATA             ;读取数据 1
MOV      P0,A
CALL     SENDACK
CALL     RECVDATA             ;读取数据 2
MOV      P2,A
CALL     SENDNAK
CALL     STOP                 ;发送停止命令

JMP      $

END

```

## 21.5.2 I<sup>2</sup>C 主机模式访问 AT24C256（查询方式）

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"

sbit      SDA      =  P1^4;
sbit      SCL      =  P1^5;

void Wait()
{
    while (!(I2CMSST & 0x40));
    I2CMSST &= ~0x40;
}

void Start()
{
    I2CMSCR = 0x01;                //发送START 命令
    Wait();
}

void SendData(char dat)
{
    I2CTXD = dat;                  //写数据到数据缓冲区
    I2CMSCR = 0x02;                //发送SEND 命令
    Wait();
}

void RecvACK()
{
    I2CMSCR = 0x03;                //发送读ACK 命令
    Wait();
}

char RecvData()
{
    I2CMSCR = 0x04;                //发送RECV 命令
    Wait();
    return I2CRXD;
}

void SendACK()
{
    I2CMSST = 0x00;                //设置ACK 信号
    I2CMSCR = 0x05;                //发送ACK 命令
    Wait();
}

void SendNAK()
{
    I2CMSST = 0x01;                //设置NAK 信号
    I2CMSCR = 0x05;                //发送ACK 命令
    Wait();
}

void Stop()
{
    I2CMSCR = 0x06;                //发送STOP 命令
    Wait();
}

void Delay()
```

```

{
    int i;

    for (i=0; i<3000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    I2CCFG = 0xe0; //使能 I2C 主机模式
    I2CMSST = 0x00;

    Start(); //发送起始命令
    SendData(0xa0); //发送设备地址+写命令
    RecvACK();
    SendData(0x00); //发送存储地址高字节
    RecvACK();
    SendData(0x00); //发送存储地址低字节
    RecvACK();
    SendData(0x12); //写测试数据 1
    RecvACK();
    SendData(0x78); //写测试数据 2
    RecvACK();
    Stop(); //发送停止命令

    Delay(); //等待设备写数据

    Start(); //发送起始命令
    SendData(0xa0); //发送设备地址+写命令
    RecvACK();
    SendData(0x00); //发送存储地址高字节
    RecvACK();
    SendData(0x00); //发送存储地址低字节
    RecvACK();
    Start(); //发送起始命令
    SendData(0xa1); //发送设备地址+读命令
    RecvACK();
    P0 = RecvData(); //读取数据 1
    SendACK();

```

```

    P2 = RecvData();           //读取数据2
    SendNAK();
    Stop();                   //发送停止命令

    while (1);
}

```

## 汇编代码

;测试工作频率为 11.0592MHz

|           |       |               |              |
|-----------|-------|---------------|--------------|
| P_SW2     | DATA  | 0BAH          |              |
| I2CCFG    | XDATA | 0FE80H        |              |
| I2CMSCR   | XDATA | 0FE81H        |              |
| I2CMSST   | XDATA | 0FE82H        |              |
| I2CSLCR   | XDATA | 0FE83H        |              |
| I2CSLST   | XDATA | 0FE84H        |              |
| I2CSLADR  | XDATA | 0FE85H        |              |
| I2CTXD    | XDATA | 0FE86H        |              |
| I2CRXD    | XDATA | 0FE87H        |              |
| SDA       | BIT   | P1.4          |              |
| SCL       | BIT   | P1.5          |              |
| P1M1      | DATA  | 091H          |              |
| P1M0      | DATA  | 092H          |              |
| P0M1      | DATA  | 093H          |              |
| P0M0      | DATA  | 094H          |              |
| P2M1      | DATA  | 095H          |              |
| P2M0      | DATA  | 096H          |              |
| P3M1      | DATA  | 0B1H          |              |
| P3M0      | DATA  | 0B2H          |              |
| P4M1      | DATA  | 0B3H          |              |
| P4M0      | DATA  | 0B4H          |              |
| P5M1      | DATA  | 0C9H          |              |
| P5M0      | DATA  | 0CAH          |              |
|           | ORG   | 0000H         |              |
|           | LJMP  | MAIN          |              |
|           | ORG   | 0100H         |              |
| START:    |       |               |              |
|           | MOV   | A,#00000001B  | ;发送 START 命令 |
|           | MOV   | DPTR,#I2CMSCR |              |
|           | MOVX  | @DPTR,A       |              |
|           | JMP   | WAIT          |              |
| SENDDATA: |       |               |              |
|           | MOV   | DPTR,#I2CTXD  | ;写数据到数据缓冲区   |
|           | MOVX  | @DPTR,A       |              |
|           | MOV   | A,#00000010B  | ;发送 SEND 命令  |
|           | MOV   | DPTR,#I2CMSCR |              |
|           | MOVX  | @DPTR,A       |              |
|           | JMP   | WAIT          |              |
| RECVACK:  |       |               |              |
|           | MOV   | A,#00000011B  | ;发送读 ACK 命令  |
|           | MOV   | DPTR,#I2CMSCR |              |
|           | MOVX  | @DPTR,A       |              |
|           | JMP   | WAIT          |              |

**RECVDATA:**

```

MOV    A,#00000100B           ;发送 RECV 命令
MOV    DPTR,#I2CMSCR
MOVX   @DPTR,A
CALL   WAIT
MOV    DPTR,#I2CRXD           ;从数据缓冲区读取数据
MOVX   A,@DPTR
RET

```

**SENDACK:**

```

MOV    A,#00000000B           ;设置 ACK 信号
MOV    DPTR,#I2CMSST
MOVX   @DPTR,A
MOV    A,#00000101B           ;发送 ACK 命令
MOV    DPTR,#I2CMSCR
MOVX   @DPTR,A
JMP    WAIT

```

**SENDNAK:**

```

MOV    A,#00000001B           ;设置 NAK 信号
MOV    DPTR,#I2CMSST
MOVX   @DPTR,A
MOV    A,#00000101B           ;发送 ACK 命令
MOV    DPTR,#I2CMSCR
MOVX   @DPTR,A
JMP    WAIT

```

**STOP:**

```

MOV    A,#00000110B           ;发送 STOP 命令
MOV    DPTR,#I2CMSCR
MOVX   @DPTR,A
JMP    WAIT

```

**WAIT:**

```

MOV    DPTR,#I2CMSST           ;清中断标志
MOVX   A,@DPTR
JNB    ACC.6,WAIT
ANL    A,#NOT 40H
MOVX   @DPTR,A
RET

```

**DELAY:**

```

MOV    R0,#0
MOV    R1,#0

```

**DELAY1:**

```

NOP
NOP
NOP
NOP
DJNZ   R1,DELAY1
DJNZ   R0,DELAY1
RET

```

**MAIN:**

```

MOV    SP,#5FH
ORL    P_SW2,#80H             ;使能访问 XFR

MOV    P0M0,#00H
MOV    P0M1,#00H
MOV    P1M0,#00H
MOV    P1M1,#00H
MOV    P2M0,#00H
MOV    P2M1,#00H

```



```

MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      A, #11100000B           ;设置 I2C 模块为主机模式
MOV      DPTR, #I2CCFG
MOVX     @DPTR, A
MOV      A, #00000000B
MOV      DPTR, #I2CMSST
MOVX     @DPTR, A

CALL     START                   ;发送起始命令
MOV      A, #0A0H
CALL     SENDDATA                ;发送设备地址+ 写命令
CALL     RECVACK
MOV      A, #000H                ;发送存储地址高字节
CALL     SENDDATA
CALL     RECVACK
MOV      A, #000H                ;发送存储地址低字节
CALL     SENDDATA
CALL     RECVACK
MOV      A, #12H                 ;写测试数据 1
CALL     SENDDATA
CALL     RECVACK
MOV      A, #78H                 ;写测试数据 2
CALL     SENDDATA
CALL     RECVACK
CALL     STOP                    ;发送停止命令

CALL     DELAY                  ;等待设备写数据

CALL     START                   ;发送起始命令
MOV      A, #0A0H                ;发送设备地址+ 写命令
CALL     SENDDATA
CALL     RECVACK
MOV      A, #000H                ;发送存储地址高字节
CALL     SENDDATA
CALL     RECVACK
MOV      A, #000H                ;发送存储地址低字节
CALL     SENDDATA
CALL     RECVACK
CALL     START                   ;发送起始命令
MOV      A, #0A1H                ;发送设备地址+ 读命令
CALL     SENDDATA
CALL     RECVACK
CALL     RECVDATA                ;读取数据 1
MOV      P0, A
CALL     SENDACK
CALL     RECVDATA                ;读取数据 2
MOV      P2, A
CALL     SENDNAK
CALL     STOP                    ;发送停止命令

JMP      $

END

```

## 21.5.3 I<sup>2</sup>C 主机模式访问 PCF8563

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
sbit    SDA        =  P1^4;
sbit    SCL        =  P1^5;
```

```
void Wait()
{
    while (!(I2CMSST & 0x40));
    I2CMSST &= ~0x40;
}
```

```
void Start()
{
    I2CMSCR = 0x01;           //发送 START 命令
    Wait();
}
```

```
void SendData(char dat)
{
    I2CTXD = dat;             //写数据到数据缓冲区
    I2CMSCR = 0x02;           //发送 SEND 命令
    Wait();
}
```

```
void RecvACK()
{
    I2CMSCR = 0x03;           //发送读 ACK 命令
    Wait();
}
```

```
char RecvData()
{
    I2CMSCR = 0x04;           //发送 RECV 命令
    Wait();
    return I2CRXD;
}
```

```
void SendACK()
{
    I2CMSST = 0x00;           //设置 ACK 信号
    I2CMSCR = 0x05;           //发送 ACK 命令
    Wait();
}
```

```
void SendNAK()
{
    I2CMSST = 0x01;           //设置 NAK 信号
    I2CMSCR = 0x05;           //发送 ACK 命令
}
```

```
    Wait();
}

void Stop()
{
    I2CMSCR = 0x06;           //发送STOP 命令
    Wait();
}

void Delay()
{
    int i;

    for (i=0; i<3000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P_SW2 /= 0x80;           //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    I2CCFG = 0xe0;           //使能 I2C 主机模式
    I2CMSST = 0x00;

    Start();                 //发送起始命令
    SendData(0xa2);          //发送设备地址+写命令
    RecvACK();
    SendData(0x02);          //发送存储地址
    RecvACK();
    SendData(0x00);          //设置秒值
    RecvACK();
    SendData(0x00);          //设置分钟值
    RecvACK();
    SendData(0x12);          //设置小时值
    RecvACK();
    Stop();                  //发送停止命令

    while (1)
    {
        Start();             //发送起始命令
        SendData(0xa2);       //发送设备地址+写命令
```

```
    RecvACK();
    SendData(0x02);           //发送存储地址
    RecvACK();
    Start();                  //发送起始命令
    SendData(0xa3);           //发送设备地址+读命令
    RecvACK();
    P0 = RecvData();           //读取秒值
    SendACK();
    P2 = RecvData();           //读取分钟值
    SendACK();
    P3 = RecvData();           //读取小时值
    SendNAK();
    Stop();                   //发送停止命令

    Delay();
}
}
```

### 汇编代码

;测试工作频率为 11.0592MHz

|                  |              |                                  |
|------------------|--------------|----------------------------------|
| <i>P_SW2</i>     | <i>DATA</i>  | <i>0BAH</i>                      |
| <i>I2CCFG</i>    | <i>XDATA</i> | <i>0FE80H</i>                    |
| <i>I2CMSCR</i>   | <i>XDATA</i> | <i>0FE81H</i>                    |
| <i>I2CMSST</i>   | <i>XDATA</i> | <i>0FE82H</i>                    |
| <i>I2CSLCR</i>   | <i>XDATA</i> | <i>0FE83H</i>                    |
| <i>I2CSLST</i>   | <i>XDATA</i> | <i>0FE84H</i>                    |
| <i>I2CSLADR</i>  | <i>XDATA</i> | <i>0FE85H</i>                    |
| <i>I2CTXD</i>    | <i>XDATA</i> | <i>0FE86H</i>                    |
| <i>I2CRXD</i>    | <i>XDATA</i> | <i>0FE87H</i>                    |
| <i>SDA</i>       | <i>BIT</i>   | <i>P1.4</i>                      |
| <i>SCL</i>       | <i>BIT</i>   | <i>P1.5</i>                      |
| <i>P1M1</i>      | <i>DATA</i>  | <i>091H</i>                      |
| <i>P1M0</i>      | <i>DATA</i>  | <i>092H</i>                      |
| <i>P0M1</i>      | <i>DATA</i>  | <i>093H</i>                      |
| <i>P0M0</i>      | <i>DATA</i>  | <i>094H</i>                      |
| <i>P2M1</i>      | <i>DATA</i>  | <i>095H</i>                      |
| <i>P2M0</i>      | <i>DATA</i>  | <i>096H</i>                      |
| <i>P3M1</i>      | <i>DATA</i>  | <i>0B1H</i>                      |
| <i>P3M0</i>      | <i>DATA</i>  | <i>0B2H</i>                      |
| <i>P4M1</i>      | <i>DATA</i>  | <i>0B3H</i>                      |
| <i>P4M0</i>      | <i>DATA</i>  | <i>0B4H</i>                      |
| <i>P5M1</i>      | <i>DATA</i>  | <i>0C9H</i>                      |
| <i>P5M0</i>      | <i>DATA</i>  | <i>0CAH</i>                      |
|                  | <i>ORG</i>   | <i>0000H</i>                     |
|                  | <i>LJMP</i>  | <i>MAIN</i>                      |
|                  | <i>ORG</i>   | <i>0100H</i>                     |
| <i>START:</i>    |              |                                  |
|                  | <i>MOV</i>   | <i>A,#00000001B</i> ;发送 START 命令 |
|                  | <i>MOV</i>   | <i>DPTR,#I2CMSCR</i>             |
|                  | <i>MOVX</i>  | <i>@DPTR,A</i>                   |
|                  | <i>JMP</i>   | <i>WAIT</i>                      |
| <i>SENDDATA:</i> |              |                                  |

```

MOV DPTR,#I2CTXD ;写数据到数据缓冲区
MOVX @DPTR,A
MOV A,#00000010B ;发送 SEND 命令
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

RECVACK:
MOV A,#00000011B ;发送读 ACK 命令
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

RECVDATA:
MOV A,#00000100B ;发送 RECV 命令
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
CALL WAIT
MOV DPTR,#I2CRXD ;从数据缓冲区读取数据
MOVX A,@DPTR
RET

SENDACK:
MOV A,#00000000B ;设置 ACK 信号
MOV DPTR,#I2CMSST
MOVX @DPTR,A
MOV A,#00000101B ;发送 ACK 命令
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

SENDNAK:
MOV A,#00000001B ;设置 NAK 信号
MOV DPTR,#I2CMSST
MOVX @DPTR,A
MOV A,#00000101B ;发送 ACK 命令
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

STOP:
MOV A,#00000110B ;发送 STOP 命令
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

WAIT:
MOV DPTR,#I2CMSST ;清中断标志
MOVX A,@DPTR
JNB ACC.6,WAIT
ANL A,#NOT 40H
MOVX @DPTR,A
RET

DELAY:
MOV R0,#0
MOV R1,#0

DELAY1:
NOP
NOP
NOP
NOP
DJNZ R1,DELAY1
DJNZ R0,DELAY1
RET

```

**MAIN:**

```

MOV     SP, #5FH
ORL     P_SW2, #80H           ;使能访问 XFR

MOV     P0M0, #00H
MOV     P0M1, #00H
MOV     P1M0, #00H
MOV     P1M1, #00H
MOV     P2M0, #00H
MOV     P2M1, #00H
MOV     P3M0, #00H
MOV     P3M1, #00H
MOV     P4M0, #00H
MOV     P4M1, #00H
MOV     P5M0, #00H
MOV     P5M1, #00H

MOV     A, #11100000B         ;设置 I2C 模块为主机模式
MOV     DPTR, #I2CCFG
MOVX    @DPTR, A
MOV     A, #00000000B
MOV     DPTR, #I2CMSST
MOVX    @DPTR, A

CALL    START                 ;发送起始命令
MOV     A, #0A2H
CALL    SENDDATA              ;发送设备地址+ 写命令
CALL    RECVACK
MOV     A, #002H              ;发送存储地址
CALL    SENDDATA
CALL    RECVACK
MOV     A, #00H               ;设置秒值
CALL    SENDDATA
CALL    RECVACK
MOV     A, #00H               ;设置分钟值
CALL    SENDDATA
CALL    RECVACK
MOV     A, #12H               ;设置小时值
CALL    SENDDATA
CALL    RECVACK
CALL    STOP                  ;发送停止命令

```

**LOOP:**

```

CALL    START                 ;发送起始命令
MOV     A, #0A2H              ;发送设备地址+ 写命令
CALL    SENDDATA
CALL    RECVACK
MOV     A, #002H              ;发送存储地址
CALL    SENDDATA
CALL    RECVACK
CALL    START                 ;发送起始命令
MOV     A, #0A3H              ;发送设备地址+ 读命令
CALL    SENDDATA
CALL    RECVACK
CALL    RECVDATA              ;读取秒值
MOV     P0, A
CALL    SENDACK
CALL    RECVDATA              ;读取分钟值
MOV     P2, A

```

```
CALL    SENDACK
CALL    RECVDATA    ;读取小时值
MOV     P3,A
CALL    SENDNAK
CALL    STOP        ;发送停止命令

CALL    DELAY

JMP     LOOP

END
```

## 21.5.4 I<sup>2</sup>C 从机模式（中断方式）

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
sbit    SDA        =    P1^4;
sbit    SCL        =    P1^5;
```

```
bit      isda;      //设备地址标志
bit      isma;      //存储地址标志
unsigned char      addr;
unsigned char xdata buffer[256];
```

```
void I2C_Isr() interrupt 24
{
```

```
    if (I2CSLST & 0x40)
```

```
    {
```

```
        I2CSLST &= ~0x40;
```

```
        isda = 1;
```

```
    }
```

```
    else if (I2CSLST & 0x20)
```

```
    {
```

```
        I2CSLST &= ~0x20;
```

```
        if (isda)
```

```
        {
```

```
            isda = 0;
```

```
        }
```

```
        else if (isma)
```

```
        {
```

```
            isma = 0;
```

```
            addr = I2CRXD;
```

```
            I2CTXD = buffer[addr];
```

```
        }
```

```
        else
```

```
        {
```

```
            buffer[addr++] = I2CRXD;
```

```
        }
```

```
    }
```

```
    else if (I2CSLST & 0x10)
```

```
    {
```

//处理 START 事件

//若为重复起始信号时必须作此设置

//处理 RECV 事件

//处理 RECV 事件 (RECV DEVICE ADDR)

//处理 RECV 事件 (RECV MEMORY ADDR)

//处理 RECV 事件 (RECV DATA)

```
I2CSLST &= ~0x10; //处理 SEND 事件
if (I2CSLST & 0x02)
{
    I2CTXD = 0xff; //接收到 NAK 则停止读取数据
}
else
{
    I2CTXD = buffer[++addr]; //接收到 ACK 则继续读取数据
}
}
else if (I2CSLST & 0x08)
{
    I2CSLST &= ~0x08; //处理 STOP 事件
    isda = 1;
    isma = 1;
}
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    I2CCFG = 0x81; //使能 I2C 从机模式
    I2CSLADR = 0x5a; //设置从机设备地址寄存器 I2CSLADR=0101_1010B
                    //即 I2CSLADR[7:1]=010_1101B,MA=0B。
                    //由于 MA 为 0,主机发送的设备地址必须与
                    //I2CSLADR[7:1]相同才能访问此 I2C 从机设备。
                    //主机若需要写数据则要发送 5AH(0101_1010B)
                    //主机若需要读数据则要发送 5BH(0101_1011B)

    I2CSLST = 0x00;
    I2CSLCR = 0x78; //使能从机模式中断
    EA = 1;

    isda = 1; //用户变量初始化
    isma = 1;
    addr = 0;
    I2CTXD = buffer[addr];

    while (1);
}
```

## 汇编代码

;测试工作频率为 11.0592MHz

P\_SW2      DATA      0BAH



|                 |              |                      |              |
|-----------------|--------------|----------------------|--------------|
| <i>I2CCFG</i>   | <i>XDATA</i> | <i>0FE80H</i>        |              |
| <i>I2CMSCR</i>  | <i>XDATA</i> | <i>0FE81H</i>        |              |
| <i>I2CMSST</i>  | <i>XDATA</i> | <i>0FE82H</i>        |              |
| <i>I2CSLCR</i>  | <i>XDATA</i> | <i>0FE83H</i>        |              |
| <i>I2CSLST</i>  | <i>XDATA</i> | <i>0FE84H</i>        |              |
| <i>I2CSLADR</i> | <i>XDATA</i> | <i>0FE85H</i>        |              |
| <i>I2CTXD</i>   | <i>XDATA</i> | <i>0FE86H</i>        |              |
| <i>I2CRXD</i>   | <i>XDATA</i> | <i>0FE87H</i>        |              |
| <i>SDA</i>      | <i>BIT</i>   | <i>P1.4</i>          |              |
| <i>SCL</i>      | <i>BIT</i>   | <i>P1.5</i>          |              |
| <i>ISDA</i>     | <i>BIT</i>   | <i>20H.0</i>         | ;设备地址标志      |
| <i>ISMA</i>     | <i>BIT</i>   | <i>20H.1</i>         | ;存储地址标志      |
| <i>ADDR</i>     | <i>DATA</i>  | <i>21H</i>           |              |
| <i>P1M1</i>     | <i>DATA</i>  | <i>091H</i>          |              |
| <i>P1M0</i>     | <i>DATA</i>  | <i>092H</i>          |              |
| <i>P0M1</i>     | <i>DATA</i>  | <i>093H</i>          |              |
| <i>P0M0</i>     | <i>DATA</i>  | <i>094H</i>          |              |
| <i>P2M1</i>     | <i>DATA</i>  | <i>095H</i>          |              |
| <i>P2M0</i>     | <i>DATA</i>  | <i>096H</i>          |              |
| <i>P3M1</i>     | <i>DATA</i>  | <i>0B1H</i>          |              |
| <i>P3M0</i>     | <i>DATA</i>  | <i>0B2H</i>          |              |
| <i>P4M1</i>     | <i>DATA</i>  | <i>0B3H</i>          |              |
| <i>P4M0</i>     | <i>DATA</i>  | <i>0B4H</i>          |              |
| <i>P5M1</i>     | <i>DATA</i>  | <i>0C9H</i>          |              |
| <i>P5M0</i>     | <i>DATA</i>  | <i>0CAH</i>          |              |
|                 | <i>ORG</i>   | <i>0000H</i>         |              |
|                 | <i>LJMP</i>  | <i>MAIN</i>          |              |
|                 | <i>ORG</i>   | <i>00C3H</i>         |              |
|                 | <i>LJMP</i>  | <i>I2CISR</i>        |              |
| <i>I2CISR:</i>  | <i>ORG</i>   | <i>0100H</i>         |              |
|                 | <i>PUSH</i>  | <i>ACC</i>           |              |
|                 | <i>PUSH</i>  | <i>PSW</i>           |              |
|                 | <i>PUSH</i>  | <i>DPL</i>           |              |
|                 | <i>PUSH</i>  | <i>DPH</i>           |              |
|                 | <i>MOV</i>   | <i>DPTR,#I2CSLST</i> | ;检测从机状态      |
|                 | <i>MOVX</i>  | <i>A,@DPTR</i>       |              |
|                 | <i>JB</i>    | <i>ACC.6,STARTIF</i> |              |
|                 | <i>JB</i>    | <i>ACC.5,RXIF</i>    |              |
|                 | <i>JB</i>    | <i>ACC.4,TXIF</i>    |              |
|                 | <i>JB</i>    | <i>ACC.3,STOPIF</i>  |              |
| <i>ISREXIT:</i> | <i>POP</i>   | <i>DPH</i>           |              |
|                 | <i>POP</i>   | <i>DPL</i>           |              |
|                 | <i>POP</i>   | <i>PSW</i>           |              |
|                 | <i>POP</i>   | <i>ACC</i>           |              |
|                 | <i>RETI</i>  |                      |              |
| <i>STARTIF:</i> | <i>ANL</i>   | <i>A,#NOT 40H</i>    | ;处理 START 事件 |
|                 | <i>MOVX</i>  | <i>@DPTR,A</i>       |              |
|                 | <i>SETB</i>  | <i>ISDA</i>          |              |
|                 | <i>JMP</i>   | <i>ISREXIT</i>       |              |
| <i>RXIF:</i>    |              |                      |              |

```

        ANL        A,#NOT 20H                ;处理 RECV 事件
        MOVX       @DPTR,A
        MOV        DPTR,#I2CRXD
        MOVX       A,@DPTR
        JBC        ISDA,RXDA
        JBC        ISMA,RXMA
        MOV        R0,ADDR                    ;处理 RECV 事件 (RECV DATA)
        MOVX       @R0,A
        INC        ADDR
        JMP        ISREXIT

RXDA:
        JMP        ISREXIT                    ;处理 RECV 事件 (RECV DEVICE ADDR)

RXMA:
        MOV        ADDR,A                    ;处理 RECV 事件 (RECV MEMORY ADDR)
        MOV        R0,A
        MOVX       A,@R0
        MOV        DPTR,#I2CTXD
        MOVX       @DPTR,A
        JMP        ISREXIT

TXIF:
        ANL        A,#NOT 10H                ;处理 SEND 事件
        MOVX       @DPTR,A
        JB         ACC.1,RXNAK
        INC        ADDR
        MOV        R0,ADDR
        MOVX       A,@R0
        MOV        DPTR,#I2CTXD
        MOVX       @DPTR,A
        JMP        ISREXIT

RXNAK:
        MOVX       A,#0FFH
        MOV        DPTR,#I2CTXD
        MOVX       @DPTR,A
        JMP        ISREXIT

STOPIF:
        ANL        A,#NOT 08H                ;处理 STOP 事件
        MOVX       @DPTR,A
        SETB       ISDA
        SETB       ISMA
        JMP        ISREXIT

MAIN:
        MOV        SP,#5FH
        ORL        P_SW2,#80H                ;使能访问 XFR

        MOV        P0M0,#00H
        MOV        P0M1,#00H
        MOV        P1M0,#00H
        MOV        P1M1,#00H
        MOV        P2M0,#00H
        MOV        P2M1,#00H
        MOV        P3M0,#00H
        MOV        P3M1,#00H
        MOV        P4M0,#00H
        MOV        P4M1,#00H
        MOV        P5M0,#00H
        MOV        P5M1,#00H

        MOV        A,#10000001B                ;使能 I2C 从机模式

```

```

MOV      DPTR,#I2CCFG
MOVX     @DPTR,A
MOV      A,#01011010B           ;设置从机设备地址寄存器 I2CSLADR=0101_1010B
                                           ;即 I2CSLADR[7:1]=010_1101B,MA=0B。
                                           ;由于 MA 为 0,主机发送的的设备地址必须与
                                           ;I2CSLADR[7:1]相同才能访问此 I2C 从机设备。
                                           ;主机若需要写数据则要发送 5AH(0101_1010B)
                                           ;主机若需要读数据则要发送 5BH(0101_1011B)

MOV      DPTR,#I2CSLADR
MOVX     @DPTR,A
MOV      A,#00000000B
MOV      DPTR,#I2CSLST
MOVX     @DPTR,A
MOV      A,#01111000B           ;使能从机模式中断
MOV      DPTR,#I2CSLCR
MOVX     @DPTR,A

SETB     ISDA                   ;用户变量初始化
SETB     ISMA
CLR      A
MOV      ADDR,A
MOV      R0,A
MOVX     A,@R0
MOV      DPTR,#I2CTXD
MOVX     @DPTR,A

SETB     EA

SJMP     $

END

```

## 21.5.5 I<sup>2</sup>C 从机模式（查询方式）

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

```

```

sbit      SDA      = P1^4;
sbit      SCL      = P1^5;

```

```

bit        isda;           //设备地址标志
bit        isma;           //存储地址标志
unsigned char      addr;
unsigned char xdata buffer[256];

```

```
void main()
```

```

{
    P_SW2 /= 0x80;           //使能访问 XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;

```

```

P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

I2CCFG = 0x81;
I2CSLADR = 0x5a;

I2CSLST = 0x00;
I2SLCR = 0x00;

isda = 1;
isma = 1;
addr = 0;
I2CTXD = buffer[addr];

while (1)
{
    if (I2CSLST & 0x40)
    {
        I2CSLST &= ~0x40;
        isda = 1;
    }
    else if (I2CSLST & 0x20)
    {
        I2CSLST &= ~0x20;
        if (isda)
        {
            isda = 0;
        }
        else if (isma)
        {
            isma = 0;
            addr = I2CRXD;
            I2CTXD = buffer[addr];
        }
        else
        {
            buffer[addr++] = I2CRXD;
        }
    }
    else if (I2CSLST & 0x10)
    {
        I2CSLST &= ~0x10;
        if (I2CSLST & 0x02)
        {
            I2CTXD = 0xff;
        }
        else
        {

```

//使能I2C 从机模式  
 //设置从机设备地址寄存器 I2CSLADR=0101\_1010B  
 //即 I2CSLADR[7:1]=010\_1101B,MA=0B。  
 //由于 MA 为 0,主机发送的设备地址必须与  
 //I2CSLADR[7:1]相同才能访问此 I2C 从机设备。  
 //主机若需要写数据则要发送 5AH(0101\_1010B)  
 //主机若需要读数据则要发送 5BH(0101\_1011B)

//禁止从机模式中断

//用户变量初始化

//处理 START 事件  
 //若为重复起始信号时必须作此设置

//处理 RECV 事件

//处理 RECV 事件 (RECV DEVICE ADDR)

//处理 RECV 事件 (RECV MEMORY ADDR)

//处理 RECV 事件 (RECV DATA)

//处理 SEND 事件

//接收到 NAK 则停止读取数据

```
        I2CTXD = buffer[++addr];           //接收到ACK 则继续读取数据
    }
}
else if (I2CSLST & 0x08)
{
    I2CSLST &= ~0x08;                     //处理STOP 事件
    isda = 1;
    isma = 1;
}
}
}
```

汇编代码

;测试工作频率为 11.0592MHz

|          |       |             |           |
|----------|-------|-------------|-----------|
| P_SW2    | DATA  | 0BAH        |           |
| I2CCFG   | XDATA | 0FE80H      |           |
| I2CMSCR  | XDATA | 0FE81H      |           |
| I2CMSST  | XDATA | 0FE82H      |           |
| I2CSLCR  | XDATA | 0FE83H      |           |
| I2CSLST  | XDATA | 0FE84H      |           |
| I2CSLADR | XDATA | 0FE85H      |           |
| I2CTXD   | XDATA | 0FE86H      |           |
| I2CRXD   | XDATA | 0FE87H      |           |
| SDA      | BIT   | P1.4        |           |
| SCL      | BIT   | P1.5        |           |
| ISDA     | BIT   | 20H.0       | ;设备地址标志   |
| ISMA     | BIT   | 20H.1       | ;存储地址标志   |
| ADDR     | DATA  | 21H         |           |
| P1M1     | DATA  | 091H        |           |
| P1M0     | DATA  | 092H        |           |
| P0M1     | DATA  | 093H        |           |
| P0M0     | DATA  | 094H        |           |
| P2M1     | DATA  | 095H        |           |
| P2M0     | DATA  | 096H        |           |
| P3M1     | DATA  | 0B1H        |           |
| P3M0     | DATA  | 0B2H        |           |
| P4M1     | DATA  | 0B3H        |           |
| P4M0     | DATA  | 0B4H        |           |
| P5M1     | DATA  | 0C9H        |           |
| P5M0     | DATA  | 0CAH        |           |
|          | ORG   | 0000H       |           |
|          | LJMP  | MAIN        |           |
|          | ORG   | 0100H       |           |
| MAIN:    | MOV   | SP, #5FH    |           |
|          | ORL   | P_SW2, #80H | ;使能访问 XFR |
|          | MOV   | P0M0, #00H  |           |
|          | MOV   | P0M1, #00H  |           |
|          | MOV   | P1M0, #00H  |           |
|          | MOV   | P1M1, #00H  |           |

```

MOV      P2M0, #00H
MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      A, #10000001B      ; 使能 I2C 从机模式
MOV      DPTR, #I2CCFG
MOVX     @DPTR, A
MOV      A, #01011010B      ; 设置从机设备地址寄存器 I2CSLADR=0101_1010B
                                ; 即 I2CSLADR[7:1]=010_1101B, MA=0B。
                                ; 由于 MA 为 0, 主机发送的设备地址必须与
                                ; I2CSLADR[7:1] 相同才能访问此 I2C 从机设备。
                                ; 主机若需要写数据则要发送 5AH(0101_1010B)
                                ; 主机若需要读数据则要发送 5BH(0101_1011B)

MOV      DPTR, #I2CSLADR
MOVX     @DPTR, A
MOV      A, #00000000B
MOV      DPTR, #I2CSLST
MOVX     @DPTR, A
MOV      A, #00000000B      ; 禁止从机模式中断
MOV      DPTR, #I2CSLCR
MOVX     @DPTR, A

SETB     ISDA                ; 用户变量初始化
SETB     ISMA
CLR      A
MOV      ADDR, A
MOV      R0, A
MOVX     A, @R0
MOV      DPTR, #I2CTXD
MOVX     @DPTR, A

LOOP:
MOV      DPTR, #I2CSLST      ; 检测从机状态
MOVX     A, @DPTR
JB       ACC.6, STARTIF
JB       ACC.5, RXIF
JB       ACC.4, TXIF
JB       ACC.3, STOPIF
JMP      LOOP

STARTIF:
ANL      A, #NOT 40H         ; 处理 START 事件
MOVX     @DPTR, A
SETB     ISDA
JMP      LOOP

RXIF:
ANL      A, #NOT 20H         ; 处理 RECV 事件
MOVX     @DPTR, A
MOV      DPTR, #I2CRXD
MOVX     A, @DPTR
JBC      ISDA, RXDA
JBC      ISMA, RXMA
MOV      R0, ADDR           ; 处理 RECV 事件 (RECV DATA)
MOVX     @R0, A
INC      ADDR

```

```

    JMP      LOOP
RXDA:
    JMP      LOOP                ;处理 RECV 事件 (RECV DEVICE ADDR)
RXMA:
    MOV      ADDR,A              ;处理 RECV 事件 (RECV MEMORY ADDR)
    MOV      R0,A
    MOVX     A,@R0
    MOV      DPTR,#I2CTXD
    MOVX     @DPTR,A
    JMP      LOOP
TXIF:
    ANL      A,#NOT 10H          ;处理 SEND 事件
    MOVX     @DPTR,A
    JB       ACC.1,RXNAK
    INC      ADDR
    MOV      R0,ADDR
    MOVX     A,@R0
    MOV      DPTR,#I2CTXD
    MOVX     @DPTR,A
    JMP      LOOP
RXNAK:
    MOVX     A,#0FFH
    MOV      DPTR,#I2CTXD
    MOVX     @DPTR,A
    JMP      LOOP
STOPIF:
    ANL      A,#NOT 08H          ;处理 STOP 事件
    MOVX     @DPTR,A
    SETB     ISDA
    SETB     ISMA
    JMP      LOOP
END
```

### 21.5.6 测试 I<sup>2</sup>C 从机模式代码的主机代码

#### C 语言代码

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

sbit      SDA      =    P1^4;
sbit      SCL      =    P1^5;

void Wait()
{
    while (!(I2CMSST & 0x40));
    I2CMSST &= ~0x40;
}

void Start()
{
    I2CMSCR = 0x01;                //发送 START 命令
    Wait();
}
```

```
}

void SendData(char dat)
{
    I2CTXD = dat;                //写数据到数据缓冲区
    I2CMSCR = 0x02;              //发送SEND 命令
    Wait();
}

void RecvACK()
{
    I2CMSCR = 0x03;              //发送读ACK 命令
    Wait();
}

char RecvData()
{
    I2CMSCR = 0x04;              //发送RECV 命令
    Wait();
    return I2CRXD;
}

void SendACK()
{
    I2CMSST = 0x00;              //设置ACK 信号
    I2CMSCR = 0x05;              //发送ACK 命令
    Wait();
}

void SendNAK()
{
    I2CMSST = 0x01;              //设置NAK 信号
    I2CMSCR = 0x05;              //发送ACK 命令
    Wait();
}

void Stop()
{
    I2CMSCR = 0x06;              //发送STOP 命令
    Wait();
}

void Delay()
{
    int i;

    for (i=0; i<3000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P_SW2 /= 0x80;               //使能访问XFR
```



```

P0M0 = 0x00;
P0M1 = 0x00;
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

I2CCFG = 0xe0; //使能I2C 主机模式
I2CMSST = 0x00;

Start(); //发送起始命令
SendData(0x5a); //发送设备地址(010_1101B)+写命令(0B)
RecvACK();
SendData(0x00); //发送存储地址
RecvACK();
SendData(0x12); //写测试数据1
RecvACK();
SendData(0x78); //写测试数据2
RecvACK();
Stop(); //发送停止命令

Start(); //发送起始命令
SendData(0x5a); //发送设备地址(010_1101B)+写命令(0B)
RecvACK();
SendData(0x00); //发送存储地址高字节
RecvACK();
Start(); //发送起始命令
SendData(0x5b); //发送设备地址(010_1101B)+读命令(1B)
RecvACK();
P0 = RecvData(); //读取数据1
SendACK();
P2 = RecvData(); //读取数据2
SendNAK();
Stop(); //发送停止命令

while (1);
}

```

## 汇编代码

;测试工作频率为11.0592MHz

|          |       |        |
|----------|-------|--------|
| P_SW2    | DATA  | 0BAH   |
| I2CCFG   | XDATA | 0FE80H |
| I2CMSCR  | XDATA | 0FE81H |
| I2CMSST  | XDATA | 0FE82H |
| I2CSLCR  | XDATA | 0FE83H |
| I2CSLST  | XDATA | 0FE84H |
| I2CSLADR | XDATA | 0FE85H |
| I2CTXD   | XDATA | 0FE86H |
| I2CRXD   | XDATA | 0FE87H |

|                  |             |                      |                     |
|------------------|-------------|----------------------|---------------------|
| <i>SDA</i>       | <i>BIT</i>  | <i>P1.4</i>          |                     |
| <i>SCL</i>       | <i>BIT</i>  | <i>P1.5</i>          |                     |
| <i>P1M1</i>      | <i>DATA</i> | <i>091H</i>          |                     |
| <i>P1M0</i>      | <i>DATA</i> | <i>092H</i>          |                     |
| <i>P0M1</i>      | <i>DATA</i> | <i>093H</i>          |                     |
| <i>P0M0</i>      | <i>DATA</i> | <i>094H</i>          |                     |
| <i>P2M1</i>      | <i>DATA</i> | <i>095H</i>          |                     |
| <i>P2M0</i>      | <i>DATA</i> | <i>096H</i>          |                     |
| <i>P3M1</i>      | <i>DATA</i> | <i>0B1H</i>          |                     |
| <i>P3M0</i>      | <i>DATA</i> | <i>0B2H</i>          |                     |
| <i>P4M1</i>      | <i>DATA</i> | <i>0B3H</i>          |                     |
| <i>P4M0</i>      | <i>DATA</i> | <i>0B4H</i>          |                     |
| <i>P5M1</i>      | <i>DATA</i> | <i>0C9H</i>          |                     |
| <i>P5M0</i>      | <i>DATA</i> | <i>0CAH</i>          |                     |
|                  | <i>ORG</i>  | <i>0000H</i>         |                     |
|                  | <i>LJMP</i> | <i>MAIN</i>          |                     |
|                  | <i>ORG</i>  | <i>0100H</i>         |                     |
| <i>START:</i>    |             |                      |                     |
|                  | <i>MOV</i>  | <i>A,#00000001B</i>  | <i>;发送 START 命令</i> |
|                  | <i>MOV</i>  | <i>DPTR,#I2CMSCR</i> |                     |
|                  | <i>MOVX</i> | <i>@DPTR,A</i>       |                     |
|                  | <i>JMP</i>  | <i>WAIT</i>          |                     |
| <i>SENDDATA:</i> |             |                      |                     |
|                  | <i>MOV</i>  | <i>DPTR,#I2CTXD</i>  | <i>;写数据到数据缓冲区</i>   |
|                  | <i>MOVX</i> | <i>@DPTR,A</i>       |                     |
|                  | <i>MOV</i>  | <i>A,#00000010B</i>  | <i>;发送 SEND 命令</i>  |
|                  | <i>MOV</i>  | <i>DPTR,#I2CMSCR</i> |                     |
|                  | <i>MOVX</i> | <i>@DPTR,A</i>       |                     |
|                  | <i>JMP</i>  | <i>WAIT</i>          |                     |
| <i>RECVACK:</i>  |             |                      |                     |
|                  | <i>MOV</i>  | <i>A,#00000011B</i>  | <i>;发送读 ACK 命令</i>  |
|                  | <i>MOV</i>  | <i>DPTR,#I2CMSCR</i> |                     |
|                  | <i>MOVX</i> | <i>@DPTR,A</i>       |                     |
|                  | <i>JMP</i>  | <i>WAIT</i>          |                     |
| <i>RECVDATA:</i> |             |                      |                     |
|                  | <i>MOV</i>  | <i>A,#00000100B</i>  | <i>;发送 RECV 命令</i>  |
|                  | <i>MOV</i>  | <i>DPTR,#I2CMSCR</i> |                     |
|                  | <i>MOVX</i> | <i>@DPTR,A</i>       |                     |
|                  | <i>CALL</i> | <i>WAIT</i>          |                     |
|                  | <i>MOV</i>  | <i>DPTR,#I2CRXD</i>  | <i>;从数据缓冲区读取数据</i>  |
|                  | <i>MOVX</i> | <i>A,@DPTR</i>       |                     |
|                  | <i>RET</i>  |                      |                     |
| <i>SENDACK:</i>  |             |                      |                     |
|                  | <i>MOV</i>  | <i>A,#00000000B</i>  | <i>;设置 ACK 信号</i>   |
|                  | <i>MOV</i>  | <i>DPTR,#I2CMSST</i> |                     |
|                  | <i>MOVX</i> | <i>@DPTR,A</i>       |                     |
|                  | <i>MOV</i>  | <i>A,#00000101B</i>  | <i>;发送 ACK 命令</i>   |
|                  | <i>MOV</i>  | <i>DPTR,#I2CMSCR</i> |                     |
|                  | <i>MOVX</i> | <i>@DPTR,A</i>       |                     |
|                  | <i>JMP</i>  | <i>WAIT</i>          |                     |
| <i>SENDNAK:</i>  |             |                      |                     |
|                  | <i>MOV</i>  | <i>A,#00000001B</i>  | <i>;设置 NAK 信号</i>   |
|                  | <i>MOV</i>  | <i>DPTR,#I2CMSST</i> |                     |
|                  | <i>MOVX</i> | <i>@DPTR,A</i>       |                     |
|                  | <i>MOV</i>  | <i>A,#00000101B</i>  | <i>;发送 ACK 命令</i>   |
|                  | <i>MOV</i>  | <i>DPTR,#I2CMSCR</i> |                     |

```

MOVX    @DPTR,A
JMP      WAIT

STOP:
MOV      A,#00000110B          ;发送 STOP 命令
MOV      DPTR,#I2CMSCR
MOVX     @DPTR,A
JMP      WAIT

WAIT:
MOV      DPTR,#I2CMSST        ;清中断标志
MOVX     A,@DPTR
JNB      ACC.6,WAIT
ANL      A,#NOT 40H
MOVX     @DPTR,A
RET

DELAY:
MOV      R0,#0
MOV      RI,#0

DELAY1:
NOP
NOP
NOP
NOP
DJNZ     RI,DELAY1
DJNZ     R0,DELAY1
RET

MAIN:
MOV      SP,#5FH
ORL      P_SW2,#80H          ;使能访问 XFR

MOV      P0M0,#00H
MOV      P0M1,#00H
MOV      P1M0,#00H
MOV      P1M1,#00H
MOV      P2M0,#00H
MOV      P2M1,#00H
MOV      P3M0,#00H
MOV      P3M1,#00H
MOV      P4M0,#00H
MOV      P4M1,#00H
MOV      P5M0,#00H
MOV      P5M1,#00H

MOV      A,#11100000B          ;设置 I2C 模块为主机模式
MOV      DPTR,#I2CCFG
MOVX     @DPTR,A
MOV      A,#00000000B
MOV      DPTR,#I2CMSST
MOVX     @DPTR,A

CALL     START                ;发送起始命令
MOV      A,#5AH
;
CALL     SENDDATA              ;发送设备地址(010_1101B)+写命令(0B)
CALL     RECVACK
MOV      A,#000H              ;发送存储地址
CALL     SENDDATA
CALL     RECVACK
MOV      A,#12H                ;写测试数据 1

```

---

```
CALL    SENDDATA
CALL    RECVACK
MOV     A,#78H                ;写测试数据2
CALL    SENDDATA
CALL    RECVACK
CALL    STOP                  ;发送停止命令

CALL    DELAY                 ;等待设备写数据

CALL    START                 ;发送起始命令
MOV     A,#5AH                ;发送设备地址(010_1101B)+写命令(0B)
CALL    SENDDATA
CALL    RECVACK
MOV     A,#000H               ;发送存储地址
CALL    SENDDATA
CALL    RECVACK
CALL    START                 ;发送起始命令
MOV     A,#5BH                ;发送设备地址(010_1101B)+读命令(1B)
CALL    SENDDATA
CALL    RECVACK
CALL    RECVDATA              ;读取数据1
MOV     P0,A
CALL    SENDACK
CALL    RECVDATA              ;读取数据2
MOV     P2,A
CALL    SENDNAK
CALL    STOP                  ;发送停止命令

JMP     $

END
```

---

---

## 22 16 位高级 PWM 定时器，支持正交编码器

| 产品线              | 高级 PWM |
|------------------|--------|
| STC8H1K08 系列     | ●      |
| STC8H1K28 系列     | ●      |
| STC8H3K64S4 系列   | ●      |
| STC8H3K64S2 系列   | ●      |
| STC8H8K64U 系列    | ●      |
| STC8H4K64TL 系列   | ●      |
| STC8H4K64TLCD 系列 | ●      |
| STC8H1K08T 系列    | ●      |

STC8H 系列的单片机内部集成了 8 通道 16 位高级 PWM 定时器，分成两组周期可不同的 PWM，分别命名为 PWMA 和 PWMB（之前的数据手册曾命名为 PWM1 和 PWM2，但容易与芯片管脚名称混淆，故更改为 PWMA 和 PWMB），可分别单独设置。第一组 PWM/PWMA 可配置成 4 组互补/对称/死区控制的 PWM 或捕捉外部信号，第二组 PWM/PWMB 可配置成 4 路 PWM 输出或捕捉外部信号。

第一组 PWM/PWMA 的时钟频率可以是系统时钟经过寄存器 [PWMA\\_PSCRH](#) 和 [PWMA\\_PSCRL](#) 进行分频后的时钟，分频值可以是 1~65535 之间的任意值。第二组 PWM/PWMB 的时钟频率可以是系统时钟经过寄存器 [PWMB\\_PSCRH](#) 和 [PWMB\\_PSCRL](#) 进行分频后的时钟，分频值可以是 1~65535 之间的任意值。两组 PWM 的时钟频率可分别独立设置。

第一组 PWM 定时器/PWMA 有 4 个通道（PWM1P/PWM1N、PWM2P/PWM2N、PWM3P/PWM3N、PWM4P/PWM4N），每个通道都可独立实现 PWM 输出（可设置带死区的互补对称 PWM 输出）、捕获和比较功能；第二组 PWM 定时器/PWMB 有 4 个通道（PWM5、PWM6、PWM7、PWM8），每个通道也可独立实现 PWM 输出、捕获和比较功能。两组 PWM 定时器唯一的区别是第一组可输出带死区的互补对称 PWM，而第二组只能输出单端的 PWM，其他功能完全相同。下面关于高级 PWM 定时器的介绍只以第一组为例进行说明。

当使用第一组 PWM 定时器输出 PWM 波形时，可单独使能 PWM1P/PWM2P/PWM3P/PWM4P 输出，也可单独使能 PWM1N/PWM2N/PWM3N/PWM4N 输出。例如：若单独使能了 PWM1P 输出，则 PWM1N 就不能再独立输出，除非 PWM1P 和 PWM1N 组成一组互补对称输出。PWMA 的 4 路输出是可分别独立设置的，例如：可单独使能 PWM1P 和 PWM2N 输出，也可单独使能 PWM2N 和 PWM3N 输出。若需要使用第一组 PWM 定时器进行捕获功能或者测量脉宽时，输入信号只能从每路的正端输入，即只有 PWM1P/PWM2P/PWM3P/PWM4P 才有捕获功能和测量脉宽功能。

两组高级 PWM 定时器对外部信号进行捕获时，可选择上升沿捕获或者下降沿捕获。如果需要同时捕获上升沿和下降沿，则可将输入信号同时接入到两路 PWM，使能其中一路捕获上升沿，另外一路捕获下降沿即可。**更强悍的是，将外部输入信号同时接入到两路 PWM 时，可同时捕获信号的周期值和占空比值。**

### STC 三种硬件 PWM 比较：

**兼容传统 8051 的 PCA/CCP/PWM：**可输出 PWM 波形、捕获外部输入信号以及输出高速脉冲。可对外输出 6 位/7 位/8 位/10 位的 PWM 波形，6 位 PWM 波形的频率为 PCA 模块时钟源频率/64；7 位 PWM 波形的频率为 PCA 模块时钟源频率/128；8 位 PWM 波形的频率为 PCA 模块时钟源频率/256；10 位

PWM 波形的频率为 PCA 模块时钟源频率/1024。捕获外部输入信号,可捕获上升沿、下降沿或者同时捕获上升沿和下降沿。

**STC8G 系列的 15 位增强型 PWM:** 只能对外输出 PWM 波形,无输入捕获功能。对外输出 PWM 的频率以及占空比均可任意设置。通过软件干预,可实现多路互补/对称/带死区的 PWM 波形。有外部异常检测功能以及实时触发 ADC 转换功能。

**STC8H 系列的 16 位高级 PWM 定时器:** 是目前 STC 功能最强的 PWM,可对外输出任意频率以及任意占空比的 PWM 波形。无需软件干预即可输出互补/对称/带死区的 PWM 波形。能捕获外部输入信号,可捕获上升沿、下降沿或者同时捕获上升沿和下降沿,测量外部波形时,可同时测量波形的周期值和占空比值。有正交编码功能、外部异常检测功能以及实时触发 ADC 转换功能。

下面的说明中, **PWMA** 代表第一组 PWM 定时器, **PWMB** 代表第二组 PWM 定时器

### 第 1 组高级 PWM 定时器/PWMA 内部信号说明

**TI1:** 外部时钟输入信号 1 (PWM1P 管脚信号或者 PWM1P/PWM2P/PWM3P 相异或后的信号)

**TI1F:** 经过 IC1F 数字滤波后的 TI1 信号

**TI1FP:** 经过 CC1P/CC2P 边沿检测器后的 TI1F 信号

**TI1F\_ED:** TI1F 的边沿信号

**TI1FP1:** 经过 CC1P 边沿检测器后的 TI1F 信号

**TI1FP2:** 经过 CC2P 边沿检测器后的 TI1F 信号

**IC1:** 通过 CC1S 选择的通道 1 的捕获输入信号

**OC1REF:** 输出通道 1 输出的参考波形 (中间波形)

**OC1:** 通道 1 的主输出信号 (经过 CC1P 极性处理后的 OC1REF 信号)

**OC1N:** 通道 1 的互补输出信号 (经过 CC1NP 极性处理后的 OC1REF 信号)

**TI2:** 外部时钟输入信号 2 (PWM2P 管脚信号)

**TI2F:** 经过 IC2F 数字滤波后的 TI2 信号

**TI2F\_ED:** TI2F 的边沿信号

**TI2FP:** 经过 CC1P/CC2P 边沿检测器后的 TI2F 信号

**TI2FP1:** 经过 CC1P 边沿检测器后的 TI2F 信号

**TI2FP2:** 经过 CC2P 边沿检测器后的 TI2F 信号

**IC2:** 通过 CC2S 选择的通道 2 的捕获输入信号

**OC2REF:** 输出通道 2 输出的参考波形 (中间波形)

**OC2:** 通道 2 的主输出信号 (经过 CC2P 极性处理后的 OC2REF 信号)

**OC2N:** 通道 2 的互补输出信号 (经过 CC2NP 极性处理后的 OC2REF 信号)

**TI3:** 外部时钟输入信号 3 (PWM3P 管脚信号)

**TI3F:** 经过 IC3F 数字滤波后的 TI3 信号

**TI3F\_ED:** TI3F 的边沿信号

**TI3FP:** 经过 CC3P/CC4P 边沿检测器后的 TI3F 信号

**TI3FP3:** 经过 CC3P 边沿检测器后的 TI3F 信号

**TI3FP4:** 经过 CC4P 边沿检测器后的 TI3F 信号

**IC3:** 通过 CC3S 选择的通道 3 的捕获输入信号

**OC3REF:** 输出通道 3 输出的参考波形 (中间波形)

**OC3:** 通道 3 的主输出信号 (经过 CC3P 极性处理后的 OC3REF 信号)

**OC3N:** 通道 3 的互补输出信号 (经过 CC3NP 极性处理后的 OC3REF 信号)

**TI4:** 外部时钟输入信号 4 (PWM4P 管脚信号)

**TI4F:** 经过 IC4F 数字滤波后的 TI4 信号

**TI4F\_ED:** TI4F 的边沿信号

**TI4FP:** 经过 CC3P/CC4P 边沿检测器后的 TI4F 信号

**TI4FP3:** 经过 CC3P 边沿检测器后的 TI4F 信号

**TI4FP4:** 经过 CC4P 边沿检测器后的 TI4F 信号

**IC4:** 通过 CC4S 选择的通道 4 的捕获输入信号

**OC4REF:** 输出通道 4 输出的参考波形 (中间波形)

**OC4:** 通道 4 的主输出信号 (经过 CC4P 极性处理后的 OC4REF 信号)

**OC4N:** 通道 4 的互补输出信号 (经过 CC4NP 极性处理后的 OC4REF 信号)

**ITR1:** 内部触发输入信号 1

**ITR2:** 内部触发输入信号 2

**TRC:** 固定为 TI1\_ED

**TRGI:** 经过 TS 多路选择器后的触发输入信号

**TRGO:** 经过 MMS 多路选择器后的触发输出信号

**ETR:** 外部触发输入信号 (PWMETI1 管脚信号)

**ETRP:** 经过 ETP 边沿检测器以及 ETPS 分频器后的 ETR 信号

**ETRF:** 经过 ETF 数字滤波后的 ETRP 信号

**BRK:** 刹车输入信号 (PWMFLT)

**CK\_PSC:** 预分频时钟, PWMA\_PSCR 预分频器的输入时钟

**CK\_CNT:** PWMA\_PSCR 预分频器的输出时钟, PWM 定时器的时钟

## 第 2 组高级 PWM 定时器/PWMB 内部信号说明

**TI5:** 外部时钟输入信号 5 (PWM5 管脚信号或者 PWM5/PWM6/PWM7 相异或后的信号)

**TI5F:** 经过 IC5F 数字滤波后的 TI5 信号

**TI5FP:** 经过 CC5P/CC6P 边沿检测器后的 TI5F 信号

**TI5F\_ED:** TI5F 的边沿信号

**TI5FP5:** 经过 CC5P 边沿检测器后的 TI5F 信号

**TI5FP6:** 经过 CC6P 边沿检测器后的 TI5F 信号

**IC5:** 通过 CC5S 选择的通道 5 的捕获输入信号

**OC5REF:** 输出通道 5 输出的参考波形 (中间波形)

**OC5:** 通道 5 的主输出信号 (经过 CC5P 极性处理后的 OC5REF 信号)

**TI6:** 外部时钟输入信号 6 (PWM6 管脚信号)

**TI6F:** 经过 IC6F 数字滤波后的 TI6 信号

**TI6F\_ED:** TI6F 的边沿信号

**TI6FP:** 经过 CC5P/CC6P 边沿检测器后的 TI6F 信号

**TI6FP5:** 经过 CC5P 边沿检测器后的 TI6F 信号

**TI6FP6:** 经过 CC6P 边沿检测器后的 TI6F 信号

**IC6:** 通过 CC6S 选择的通道 6 的捕获输入信号



**OC6REF**: 输出通道 6 输出的参考波形 (中间波形)

**OC6**: 通道 6 的主输出信号 (经过 CC6P 极性处理后的 OC6REF 信号)

**TI7**: 外部时钟输入信号 7 (PWM7 管脚信号)

**TI7F**: 经过 IC7F 数字滤波后的 TI7 信号

**TI7F\_ED**: TI7F 的边沿信号

**TI7FP**: 经过 CC7P/CC8P 边沿检测器后的 TI7F 信号

**TI7FP7**: 经过 CC7P 边沿检测器后的 TI7F 信号

**TI7FP8**: 经过 CC8P 边沿检测器后的 TI7F 信号

**IC7**: 通过 CC7S 选择的通道 7 的捕获输入信号

**OC7REF**: 输出通道 7 输出的参考波形 (中间波形)

**OC7**: 通道 7 的主输出信号 (经过 CC7P 极性处理后的 OC7REF 信号)

**TI8**: 外部时钟输入信号 8 (PWM8 管脚信号)

**TI8F**: 经过 IC8F 数字滤波后的 TI8 信号

**TI8F\_ED**: TI8F 的边沿信号

**TI8FP**: 经过 CC7P/CC8P 边沿检测器后的 TI8F 信号

**TI8FP7**: 经过 CC7P 边沿检测器后的 TI8F 信号

**TI8FP8**: 经过 CC8P 边沿检测器后的 TI8F 信号

**IC8**: 通过 CC8S 选择的通道 8 的捕获输入信号

**OC8REF**: 输出通道 8 输出的参考波形 (中间波形)

**OC8**: 通道 8 的主输出信号 (经过 CC8P 极性处理后的 OC8REF 信号)

## 22.1 简介

PWMA 由一个 16 位的自动装载计数器组成, 它由一个可编程的预分频器驱动。

PWMA 适用于许多不同的用途:

- 基本的定时
- 测量输入信号的脉冲宽度 (输入捕获)
- 产生输出波形 (输出比较, PWM 和单脉冲模式)
- 对应与不同事件 (捕获, 比较, 溢出, 刹车, 触发) 的中断
- 与 PWMB 或者外部信号 (外部时钟, 复位信号, 触发和使能信号) 同步

PWMA 广泛的适用于各种控制应用中, 包括那些需要中间对齐模式 PWM 的应用, 该模式支持互补输出和死区时间控制。PWMA 的时钟源可以是内部时钟, 也可以是外部的信号, 可以通过配置寄存器来进行选择。

## 22.2 主要特性

PWMA 的特性包括:

- 16 位向上、向下、向上/下自动装载计数器
- 允许在指定数目的计数器周期之后更新定时器寄存器的重复计数器
- 16 位可编程 (可以实时修改) 预分频器, 计数器时钟频率的分频系数为 1~65535 之间的任意数值
- 同步电路, 用于使用外部信号控制定时器以及定时器互联
- 多达 4 个独立通道可以配置成:



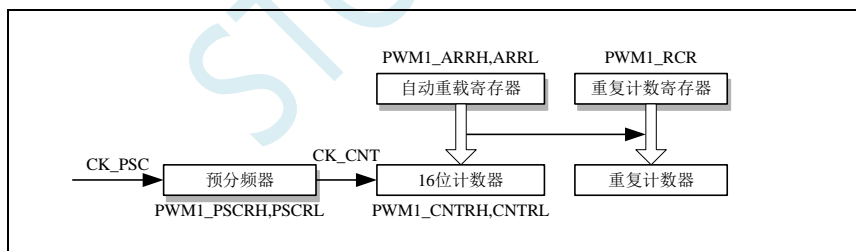
- 输入捕获
- 输出比较
- PWM 输出（边缘或中间对齐模式）
- 六步 PWM 输出
- 单脉冲模式输出
- 支持 4 个死区时间可编程的通道上互补输出
- 刹车输入信号（PWMFLT）可以将定时器输出信号置于复位状态或者一个确定状态
- 外部触发输入引脚（PWMETI）
- 产生中断的事件包括：
  - 更新：计数器向上溢出/向下溢出，计数器初始化（通过软件或者内部/外部触发）
  - 触发事件（计数器启动、停止、初始化或者由内部/外部触发计数）
  - 输入捕获，测量脉宽
  - 外部中断
  - 输出比较
  - 刹车信号输入

## 22.3 时基单元

PWMA 的时基单元包含：

- 16 位向上/向下计数器
- 16 位自动重载寄存器
- 重复计数器
- 预分频器

PWMA 时基单元



16 位计数器、预分频器、自动重载寄存器和重复计数器寄存器都可以通过软件进行读写操作。自动重载寄存器由预装载寄存器和影子寄存器组成。

可在在两种模式下写自动重载寄存器：

- 自动预装载已使能（PWMA\_CR1 寄存器的 ARPE 位为 1）。在此模式下，写入自动重载寄存器的数据将被保存在预装载寄存器中，并在下一个更新事件（UEV）时传送到影子寄存器。
- 自动预装载已禁止（PWMA\_CR1 寄存器的 ARPE 位为 0）。在此模式下，写入自动重载寄存器的数据将立即写入影子寄存器。

更新事件的产生条件：

- 计数器向上或向下溢出。
- 软件置位了 PWMA\_EGR 寄存器的 UG 位。
- 时钟/触发控制器产生了触发事件。

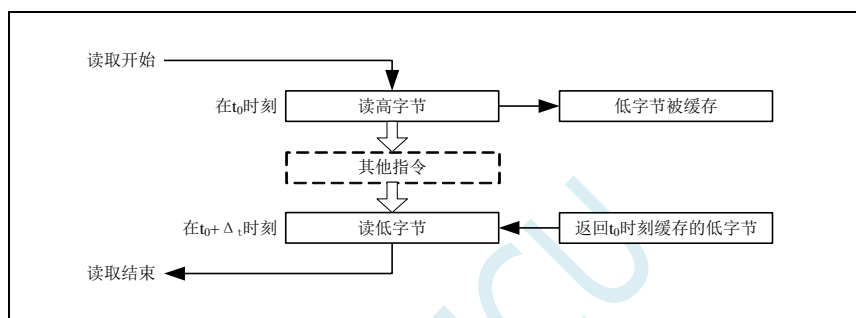
在预装载使能时 (ARPE=1)，如果发生了更新事件，预装载寄存器中的数值 (PWMA\_ARR) 将写入影子寄存器中，并且 PWMA\_PSCR 寄存器中的值将写入预分频器中。置位 PWMA\_CR1 寄存器的 UDIS 位将禁止更新事件 (UEV)。预分频器的输出 CK\_CNT 驱动计数器，而 CK\_CNT 仅在 PWMA\_CR1 寄存器的计数器使能位 (CEN) 被置位时才有效。

注意：实际的计数器在 CEN 位使能的一个时钟周期后才开始计数。

### 22.3.1 读写 16 位计数器

写计数器的操作没有缓存，在任何时候都可以写 PWMA\_CNTRH 和 PWMA\_CNTRL 寄存器，因此为避免写入了错误的数值，一般建议不要在计数器运行时写入新的数值。

读计数器的操作带有 8 位的缓存。用户必须先读定时器的高字节，在用户读了高字节后，低字节将被自动缓存，缓存的数据将会一直保持直到 16 位数据的读操作完成。



### 22.3.2 16 位 PWMA\_ARR 寄存器的写操作

预装载寄存器中的值将写入 16 位的 PWMA\_ARR 寄存器中，此操作由两条指令完成，每条指令写入 1 个字节。必须先写高字节，后写低字节。

影子寄存器在写入高字节时被锁定，并保持到低字节写完。

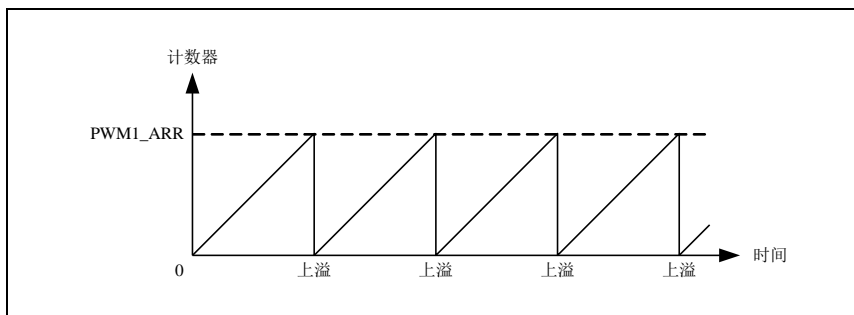
### 22.3.3 预分频器

PWMA 的预分频器基于一个由 16 位寄存器 (PWMA\_PSCR) 控制的 16 位计数器。由于这个控制寄存器带有缓冲器，因此它能够在运行时被改变。预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。预分频器的值由预装载寄存器写入，保存了当前使用值的影子寄存器在低字节写入时被载入。由于需两次单独的写操作来写 16 位寄存器，因此必须保证高字节先写入。新的预分频器的值在下次更新事件到来时被采用。对 PWMA\_PSCR 寄存器的读操作通过预装载寄存器完成。

计数器的频率计算公式:  $f_{CK\_CNT} = f_{CK\_PSC} / (PSCR[15:0] + 1)$

### 22.3.4 向上计数模式

在向上计数模式中，计数器从 0 计数到用户定义的比较值 (PWMA\_ARR 寄存器的值)，然后重新从 0 开始计数并产生一个计数器溢出事件，此时如果 PWMA\_CR1 寄存器的 UDIS 位是 0，将会产生一个更新事件 (UEV)。



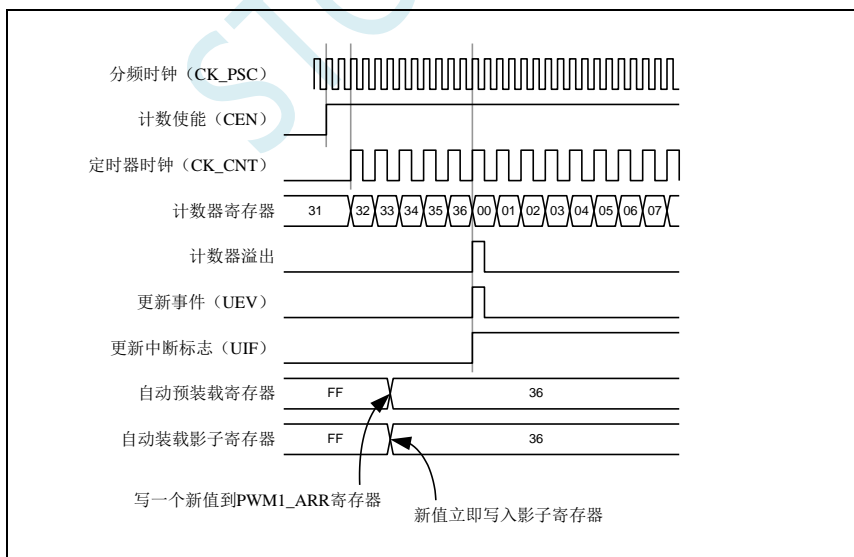
通过软件方式或者通过使用触发控制器置位 **PWMA\_EGR** 寄存器的 **UG** 位同样也可以产生一个更新事件。使用软件置位 **PWMA\_CR1** 寄存器的 **UDIS** 位，可以禁止更新事件，这样可以避免在更新预装载寄存器时更新影子寄存器。在 **UDIS** 位被清除之前，将不产生更新事件。但是在应该产生更新事件时，计数器仍会被清 0，同时预分频器的计数也被清 0（但预分频器的数值不变）。此外，如果设置了 **PWMA\_CR1** 寄存器中的 **URS** 位（选择更新请求），设置 **UG** 位将产生一个更新事件 **UEV**，但硬件不设置 **UIF** 标志（即不产生中断请求）。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件依据 **URS** 位同时设置更新标志位（**PWMA\_SR** 寄存器的 **UIF** 位）：

- 自动装载影子寄存器被重新置入预装载寄存器的值（**PWMA\_ARR**）。
- 预分频器的缓存器被置入预装载寄存器的值（**PWMA\_PSC**）。

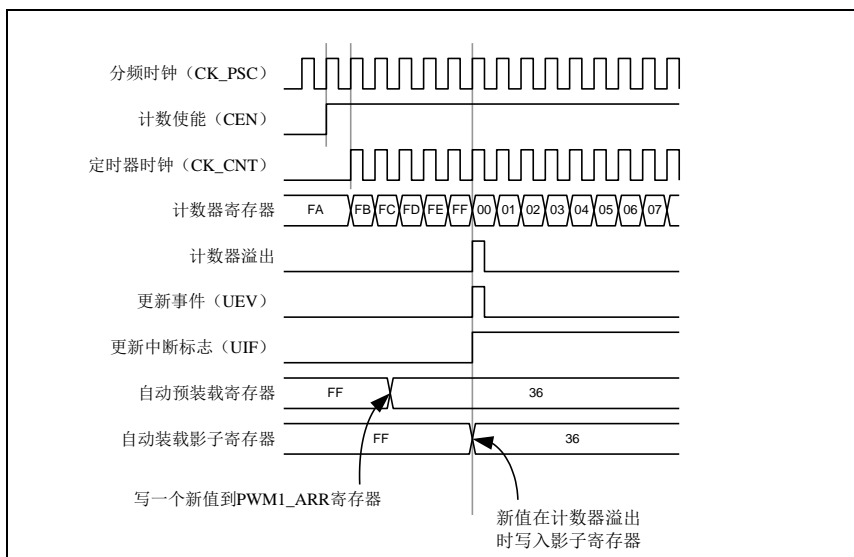
下图给出一些例子，说明当 **PWMA\_ARR=0x36** 时，计数器在不同时钟频率下的动作。图中预分频为 2，因此计数器的时钟（**CK\_CNT**）频率是预分频时钟（**CK\_PSC**）频率的一半。图中禁止了自动装载功能（**ARPE=0**），所以在计数器达到 0x36 时，计数器溢出，影子寄存器立刻被更新，同时产生一个更新事件。

当 **ARPE=0**（**ARR** 不预装载），预分频为 2 时的计数器更新：



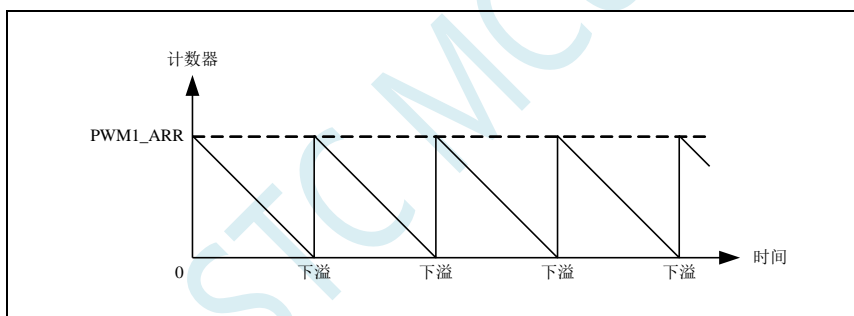
下图的预分频为 1，因此 **CK\_CNT** 的频率与 **CK\_PSC** 一致。图中使能了自动重载（**ARPE=1**），所以在计数器达到 0xFF 产生溢出。0x36 将在溢出时被写入，同时产生一个更新事件。

当 **ARPE=1**（**PWMA\_ARR** 预装载），预分频为 1 时的计数器更新：



### 22.3.5 向下计数模式

在向下模式中，计数器从自动装载的值 (PWMA\_ARR 寄存器的值) 开始向下计数到 0，然后再从自动装载的值重新开始计数，并产生一个计数器向下溢出事件。如果 PWMA\_CR1 寄存器的 UDIS 位被清除，还会产生一个更新事件 (UEV)。



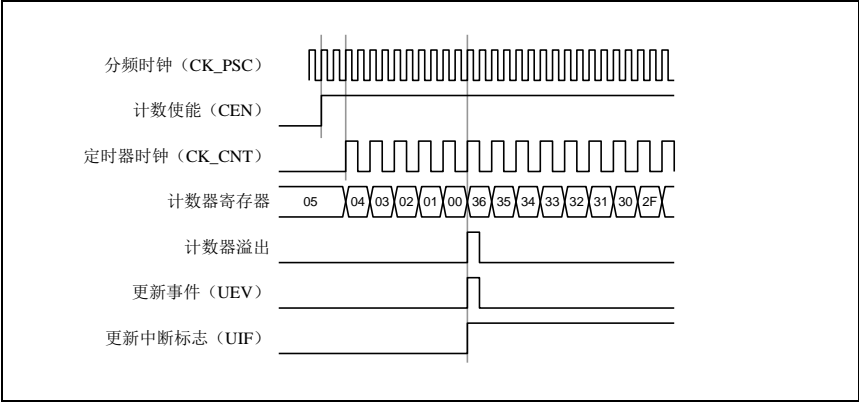
通过软件方式或者通过使用触发控制器置位 PWMA\_EGR 寄存器的 UG 位同样也可以产生一个更新事件。置位 PWMA\_CR1 寄存器的 UDIS 位可以禁止 UEV 事件。这样可以避免在更新预装载寄存器时更新影子寄存器。因此 UDIS 位清除之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，并且预分频器的计数器重新从 0 开始 (但预分频器不能被修改)。此外，如果设置了 PWMA\_CR1 寄存器中的 URS 位 (选择更新请求)，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志 (因此不产生中断)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，硬件依据 URS 位同时设置更新标志位 (PWMA\_SR 寄存器的 UIF 位)：

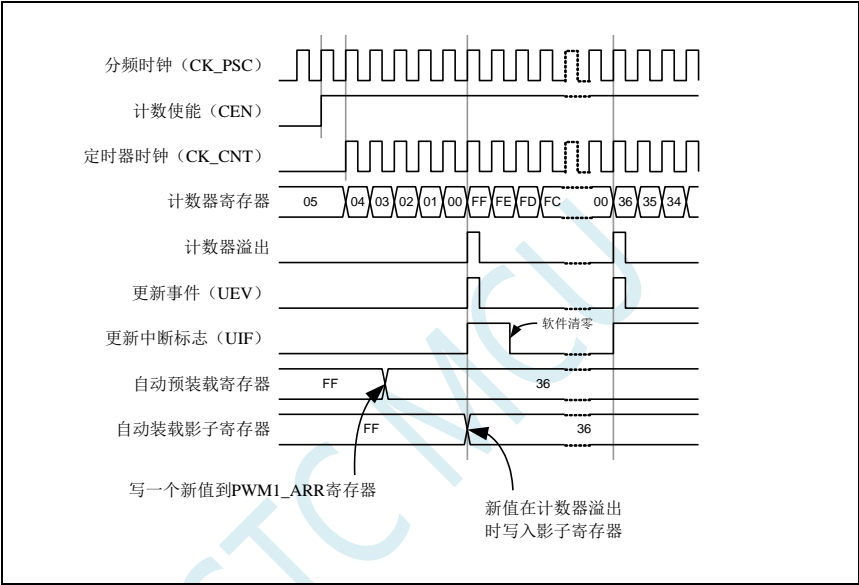
- 自动装载影子寄存器被重新置入预装载寄存器的值 (PWMA\_ARR)。
- 预分频器的缓存器被置入预装载寄存器的值 (PWMA\_PSC)。

以下是一些当 PWMA\_ARR=0x36 时，计数器在不同时钟频率下的图表。下图描述了在向下计数模式下，预装载不使能时新的数值在下个周期时被写入。

当 ARPE=0 (ARR 不预装载)，预分频为 2 时的计数器更新：

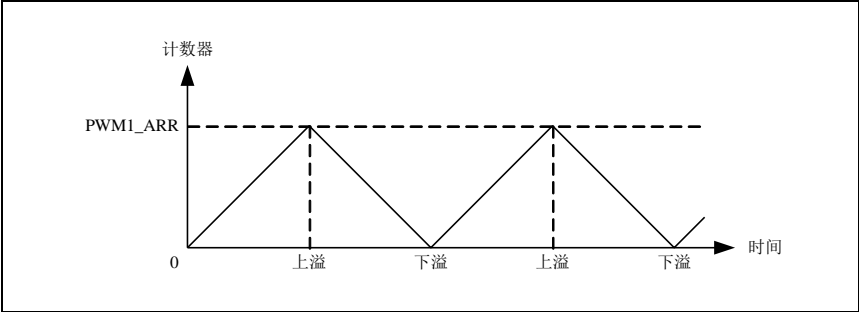


当 ARPE=1 (ARR 预装载), 预分频为 1 时的计数器更新



22.3.6 中间对齐模式（向上/向下计数）

在中央对齐模式，计数器从 0 开始计数到 PWM1\_ARR 寄存器的值，产生一个计数器上溢事件，然后从 PWM1\_ARR 寄存器的值向下计数到 0 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。在此模式下，不能写入 PWM1\_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。



如果定时器带有重复计数器，在重复了指定次数 (PWM1\_RCR 的值) 的向上和向下溢出之后会产生更新事件 (UEV)。否则每一次的向上向下溢出都会产生更新事件。通过软件方式或者通过使用触发控制器置位 PWM1\_EGR 寄存器的 UG 位同样也可以产生一个更新事件。此时，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。设置 PWM1\_CR1 寄存器中的 UDIS 位可以禁止 UEV 事件。这样可

以避免在更新预装载寄存器时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。如果定时器带有重复计数器，由于重复寄存器没有双重的缓冲，新的重复数值将立刻生效，因此在修改时需要小心。此外，如果设置了 PWMA\_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志（因此不产生中断），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

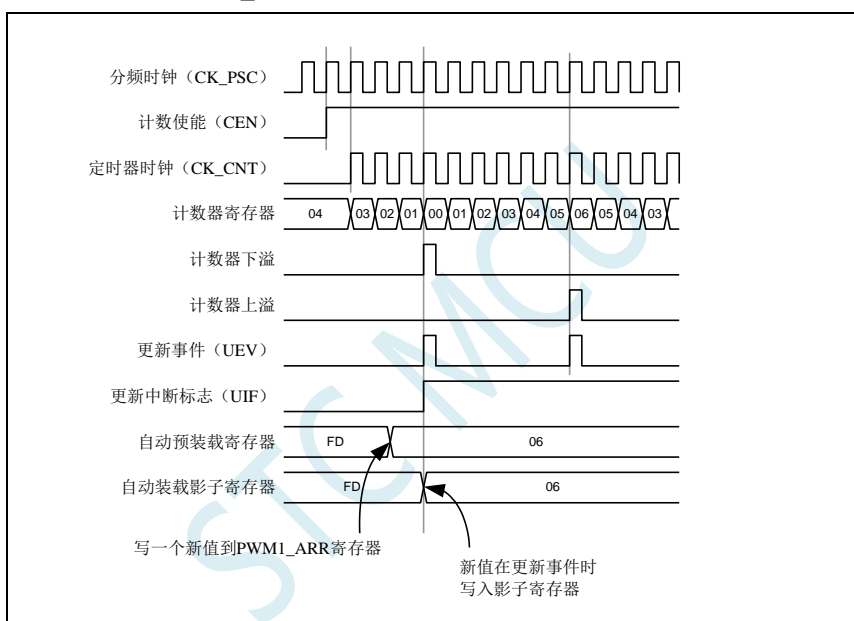
当发生更新事件时，所有的寄存器都被更新，硬件依据 URS 位更新标志位（PWMA\_SR 寄存器中的 UIF 位）：

- 预分频器的缓存器被加载为预装载的值（PWMA\_PSCR）。
- 当前的自动加载寄存器被更新为预装载值（PWMA\_ARR）。

要注意到如果因为计数器溢出而产生更新，自动重装载寄存器将在计数器重载入之前被更新，因此下一个周期才是预期的值（计数器被装载为新的值）。

以下是一些计数器在不同时钟频率下的操作的例子：

内部时钟分频因子为 1，PWMA\_ARR=0x6，ARPE=1



使用中央对齐模式的提示：

- 启动中央对齐模式时，计数器将按照原有的向上/向下的配置计数。也就是说 PWMA\_CR1 寄存器中的 DIR 位将决定计数器是向上还是向下计数。此外，软件不能同时修改 DIR 位和 CMS 位的值。
- 不推荐在中央对齐模式下，计数器正在计数时写计数器的值，这将导致不能预料的后果。具体的说：
  - 向计数器写入了比自动装载值更大的数值时（PWMA\_CNT > PWMA\_ARR），但计数器的计数方向不发生改变。例如计数器已经向上溢出，但计数器仍然向上计数。
  - 向计数器写入了 0 或者 PWMA\_ARR 的值，但更新事件不发生。
- 安全使用中央对齐模式的计数器的方法是在启动计数器之前先用软件（置位 PWMA\_EGR 寄存器的 UG 位）产生一个更新事件，并且不在计数器计数时修改计数器的值。

## 22.3.7 重复计数器

时基单元解释了计数器向上/向下溢出时更新事件（UEV）是如何产生的，然而事实上它只能在重复计数器的值达到 0 的时候产生。这个特性对产生 PWM 信号非常有用。



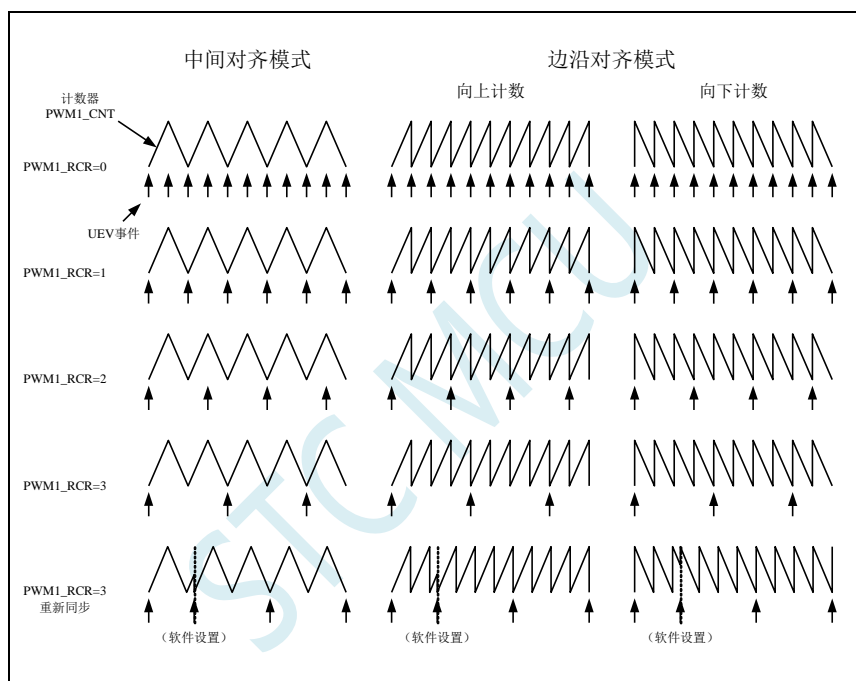
这意味着在每 N 次计数上溢或下溢时，数据从预装载寄存器传输到影子寄存器（PWMA\_ARR 自动重载入寄存器，PWMA\_PSCR 预装载寄存器，还有在比较模式下的捕获/比较寄存器 PWMA\_CCRx），N 是 PWMA\_RCR 重复计数寄存器中的值。

重复计数器在下述任一条件成立时递减：

- 向上计数模式下每次计数器向上溢出时
- 向下计数模式下每次计数器向下溢出时
- 中央对齐模式下每次上溢和每次下溢时。虽然这样限制了 PWM 的最大循环周期为 128，但它能够在每个 PWM 周期 2 次更新占空比。在中央对齐模式下，因为波形是对称的，如果每个 PWM 周期中仅刷新一次比较寄存器，则最大的分辨率为  $2 \cdot t_{CK\_PSC}$ 。

重复计数器是自动加载的，重复速率由 PWMA\_RCR 寄存器的值定义。当更新事件由软件产生或者通过硬件的时钟/触发控制器产生，则无论重复计数器的值是多少，立即发生更新事件，并且 PWMA\_RCR 寄存器中的内容被重载入到重复计数器。

不同模式下更新速率的例子，及 PWMA\_RCR 的寄存器设置



## 22.4 时钟/触发控制器

时钟/触发控制器允许用户选择计数器的时钟源，输入触发信号和输出信号，

### 22.4.1 预分频时钟（CK\_PSC）

时基单元的预分频时钟（CK\_PSC）可以由以下源提供：

- 内部时钟（ $f_{MASTER}$ ）
- 外部时钟模式 1：外部时钟输入（TIX）
- 外部时钟模式 2：外部触发输入 ETR
- 内部触发输入（ITRx）：使用一个 PWM 的 TRGO 做为另一个 PWM 的预分频时钟。

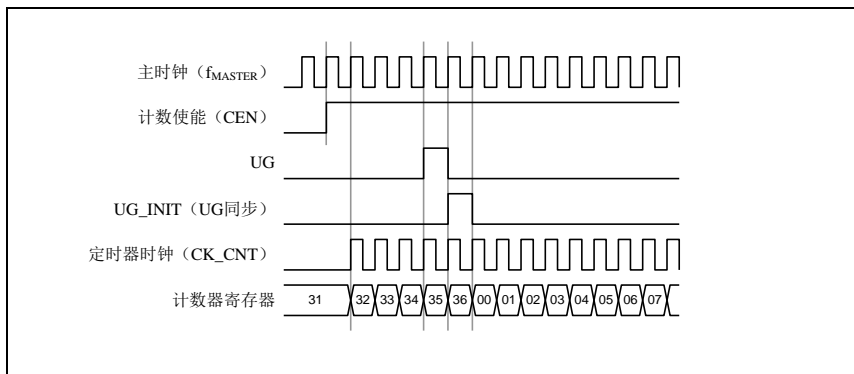
### 22.4.2 内部时钟源（ $f_{MASTER}$ ）

如果同时禁止了时钟/触发模式控制器和外部触发输入（PWMA\_SMCR 寄存器的 SMS=000，PWMA\_ETR 寄存器的 ECE=0），则 CEN、DIR 和 UG 位是实际上的控制位，并且只能被软件修改（UG

位仍被自动清除)。一旦 CEN 位被写成 1，预分频器的时钟就由内部时钟提供。

下图描述了控制电路和向上计数器在普通模式下，不带预分频器时的操作。

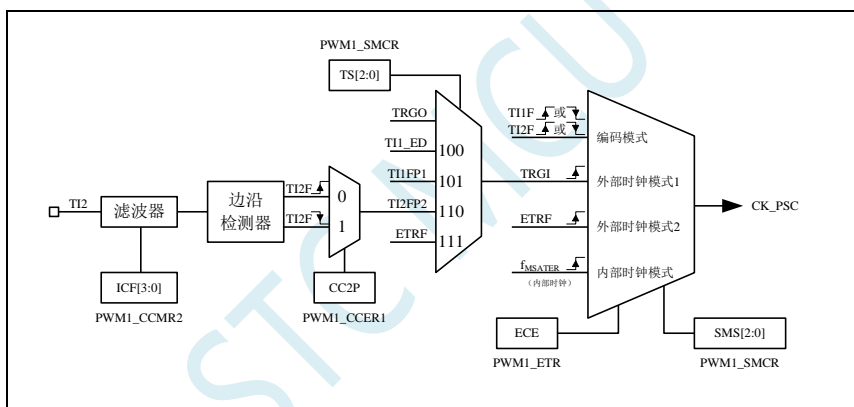
普通模式下的控制电路， $f_{MASTER}$  分频因子为 1



### 22.4.3 外部时钟源模式 1

当 PWMA\_SMCR 寄存器的 SMS=111 时，此模式被选中。然后再通过 PWMA\_SMCR 寄存器的 TS 选择 TRGI 的信号源。计数器可以在选定输入端的每个上升沿或下降沿计数。

下面的例子以 TI2 作为外部时钟



例如，要配置向上计数器在 TI2 输入端的上升沿计数，使用下列步骤：

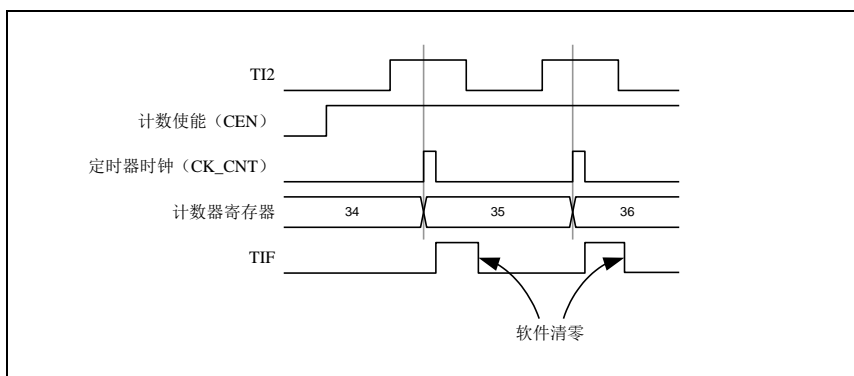
1. 配置 PWMA\_CCMR2 寄存器的 CC2S=01，使用通道 2 检测 TI2 输入的上升沿
2. 配置 PWMA\_CCMR2 寄存器的 IC2F[3:0]位，选择输入滤波器带宽
3. 配置 PWMA\_CCER1 寄存器的 CC2P=0，选定上升沿极性
4. 配置 PWMA\_SMCR 寄存器的 SMS=111，配置计数器使用外部时钟模式 1
5. 配置 PWMA\_SMCR 寄存器的 TS=110，选定 TI2 作为输入源
6. 设置 PWMA\_CR1 寄存器的 CEN=1，启动计数器

当上升沿出现在 TI2，计数器计数一次，且触发标识位（PWMA\_SR1 寄存器的 TIF 位）被置 1，如果使能了中断（在 PWMA\_IER 寄存器中配置）则会产生中断请求。

在 TI2 的上升沿和计数器实际时钟之间的延时取决于在 TI2 输入端的重新同步电路。

外部时钟模式 1 下的控制电路

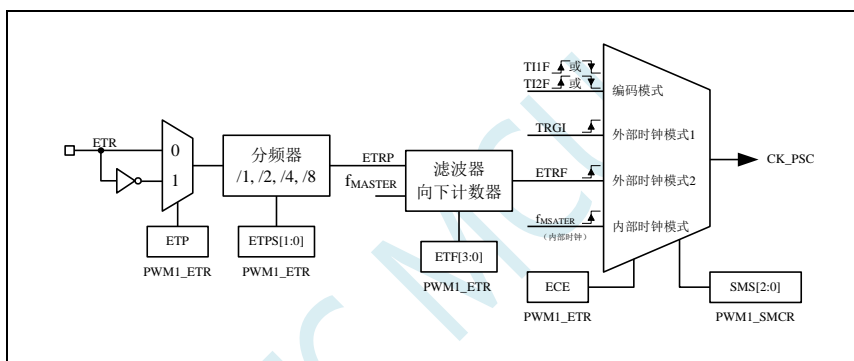




## 22.4.4 外部时钟源模式 2

计数器能够在外部触发输入 ETR 信号的每一个上升沿或下降沿计数。将 PWMA\_ETR 寄存器的 ECE 位写 1，即可选定此模式。(PWMA\_SMCR 寄存器的 SMS=111 且 PWMA\_SMCR 寄存器的 TS=111 时，也可选择此模式)

外部触发输入的总体框图：

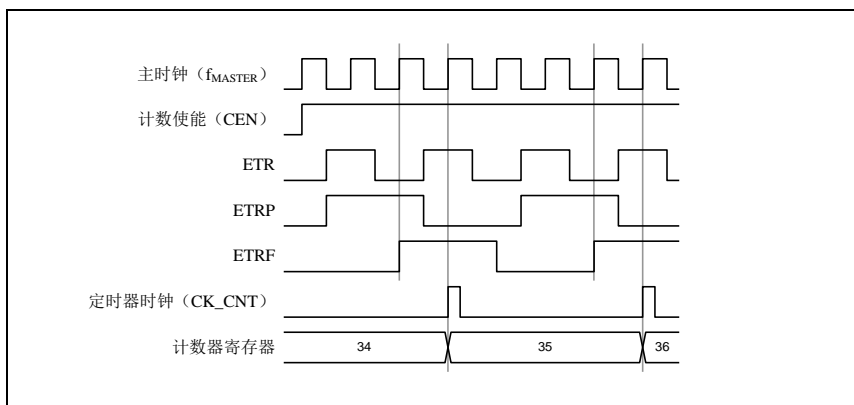


例如，要配置计数器在 ETR 信号的每 2 个上升沿时向上计数一次，需使用下列步骤：

1. 本例中不需要滤波器，配置 PWMA\_ETR 寄存器的 ETF[3:0]=0000
2. 设置预分频器，配置 PWMA\_ETR 寄存器的 ETPS[1:0]=01
3. 选择 ETR 的上升沿检测，配置 PWMA\_ETR 寄存器的 ETP=0
4. 开启外部时钟模式 2，配置 PWMA\_ETR 寄存器中的 ECE=1
5. 启动计数器，写 PWMA\_CR1 寄存器的 CEN=1

计数器在每 2 个 ETR 上升沿计数一次。

外部时钟模式 2 下的控制电路



## 22.4.5 触发同步

PWMA 的计数器使用三种模式与外部的触发信号同步:

- 标准触发模式
- 复位触发模式
- 门控触发模式

### 标准触发模式

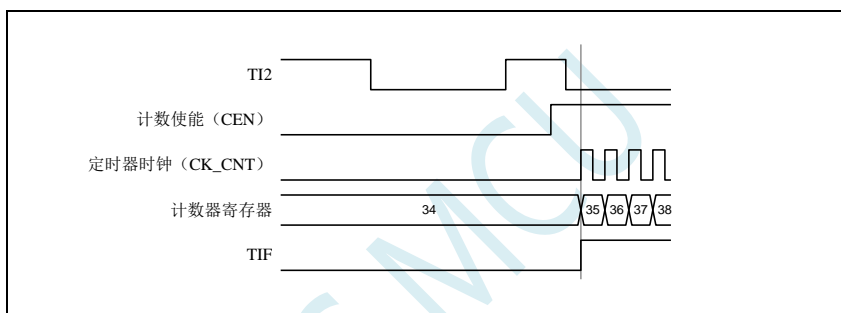
计数器的使能 (CEN) 依赖于选中的输入端上的事件。

在下面的例子中, 计数器在 TI2 输入的上升沿开始向上计数:

1. 配置 PWMA\_CCER1 寄存器的 CC2P=0, 选择 TI2 的上升沿做为触发条件。
2. 配置 PWMA\_SMCR 寄存器的 SMS=110, 选择计数器为触发模式。配置 PWMA\_SMCR 寄存器的 TS=110, 选择 TI2 作为输入源。

当 TI2 出现一个上升沿时, 计数器开始在内部时钟驱动下计数, 同时置位 TIF 标志。TI2 上升沿和计数器启动计数之间的延时取决于 TI2 输入端的重同步电路。

标准触发模式的控制电路



### 复位触发模式

在发生一个触发输入事件时, 计数器和它的预分频器能够重新被初始化。同时, 如果 PWMA\_CR1 寄存器的 URS 位为低, 还产生一个更新事件 UEV, 然后所有的预装载寄存器 (PWMA\_ARR, PWMA\_CCRx) 都会被更新。

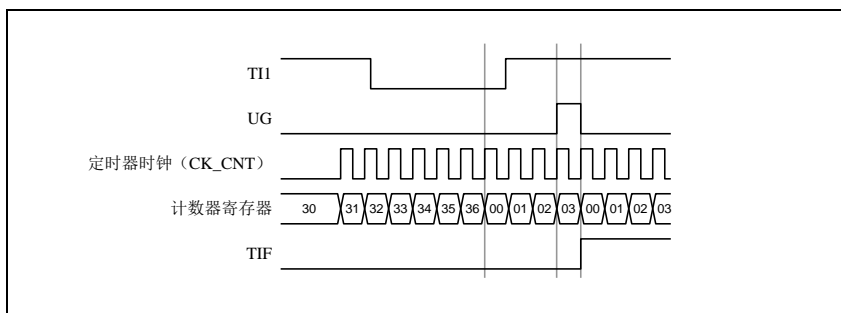
在以下的例子中, TI1 输入端的上升沿导致向上计数器被清零:

1. 配置 PWMA\_CCER1 寄存器的 CC1P=0 来选择 TI1 的极性 (只检测 TI1 的上升沿)。
2. 配置 PWMA\_SMCR 寄存器的 SMS=100, 选择定时器为复位触发模式。配置 PWMA\_SMCR 寄存器的 TS=101, 选择 TI1 作为输入源。
3. 配置 PWMA\_CR1 寄存器的 CEN=1, 启动计数器。

计数器开始依据内部时钟计数, 然后正常计数直到 TI1 出现一个上升沿。此时, 计数器被清零然后从 0 重新开始计数。同时, 触发标志 (PWMA\_SR1 寄存器的 TIF 位) 被置位, 如果使能了中断 (PWMA\_IER 寄存器的 TIE 位), 则产生一个中断请求。

下图显示当自动重装载寄存器 PWMA\_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时取决于 TI1 输入端的重同步电路。

复位触发模式下的控制电路



## 门控触发模式

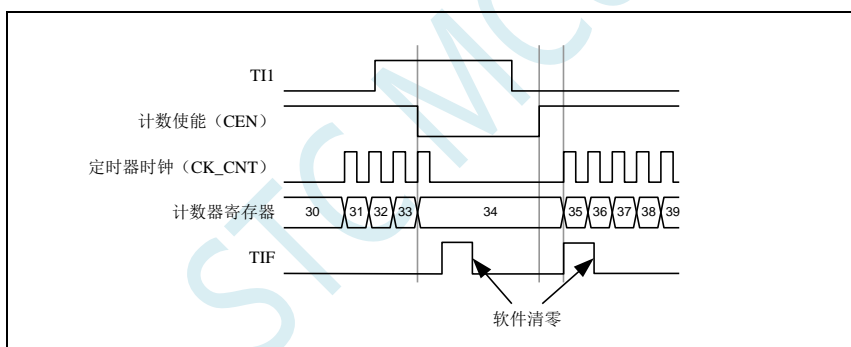
计数器由选中的输入端信号的电平使能。

在如下的例子中，计数器只在 TI1 为低时向上计数：

1. 配置 PWMA\_CCER1 寄存器的 CC1P=1 来确定 TI1 的极性（只检测 TI1 上的低电平）。
2. 配置 PWMA\_SMCR 寄存器的 SMS=101，选择定时器为门控触发模式，配置 PWMA\_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
3. 配置 PWMA\_CR1 寄存器的 CEN=1，启动计数器（在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何）。

只要 TI1 为低，计数器开始依据内部时钟计数，一旦 TI1 变高则停止计数。当计数器开始或停止时 TIF 标志位都会被置位。TI1 上升沿和计数器实际停止之间的延时取决于 TI1 输入端的重同步电路。

门控触发模式下的控制电路



## 外部时钟模式 2 联合触发模式

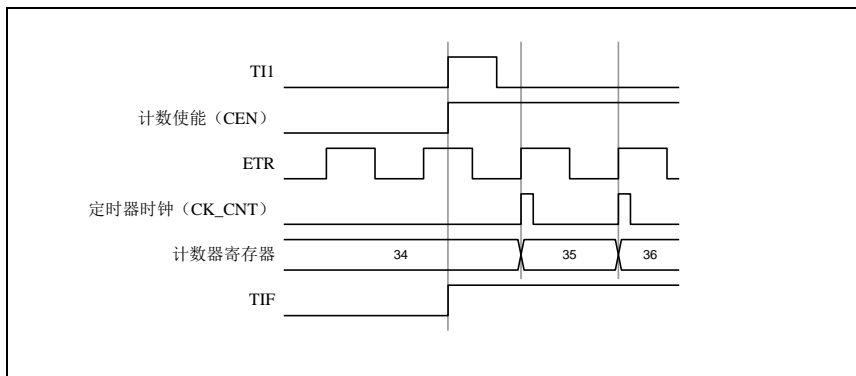
外部时钟模式 2 可以与另一个输入信号的触发模式一起使用。例如，ETR 信号被用作外部时钟的输入，另一个输入信号可用作触发输入（支持标准触发模式，复位触发模式和门控触发模式）。注意不能通过 PWMA\_SMCR 寄存器的 TS 位把 ETR 配置成 TRGI。

在下面的例子中，一旦在 TI1 上出现一个上升沿，计数器即在 ETR 的每一个上升沿向上计数一次：

1. 通过 PWMA\_ETR 寄存器配置外部触发输入电路。配置 ETPS=00 禁止预分频，配置 ETP=0 监测 ETR 信号的上升沿，配置 ECE=1 使能外部时钟模式 2。
2. 配置 PWMA\_CCER1 寄存器的 CC1P=0 来选择 TI1 的上升沿触发。
3. 配置 PWMA\_SMCR 寄存器的 SMS=110 来选择定时器为触发模式。配置 PWMA\_SMCR 寄存器的 TS=101 来选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。TI1 信号的上升沿和计数器实际时钟之间的延时取决于 TI1 输入端的重同步电路。ETR 信号的上升沿和计数器实际时钟之间的延时取决于 ETRP 输入端的重同步电路。

外部时钟模式 2+触发模式下的控制电路



## 22.4.6 与 PWMB 同步

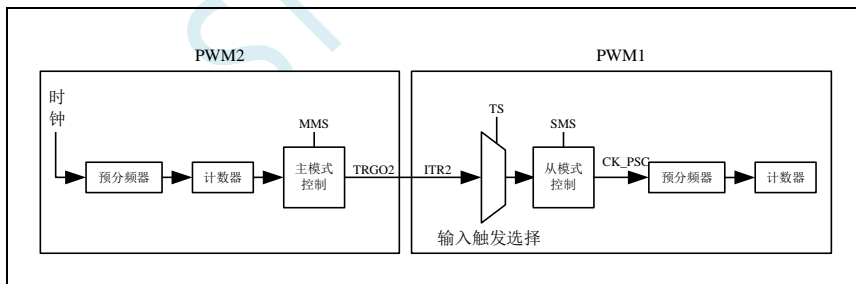
在芯片中, 定时器在内部互相联结, 用于定时器的同步或链接。当某个定时器配置成主模式时, 可以输出触发信号 (TRGO) 到那些配置为从模式的定时器来完成复位操作、启动操作、停止操作或者作为那些定时器的驱动时钟。

### 使用 PWMB 的 TRGO 作为 PWMA 的预分频时钟

例如, 用户可以配置 PWMB 作为 PWMA 的预分频时钟, 需进行如下配置:

1. 配置 PWMB 为主模式, 使得在每个更新事件 (UEV) 时输出周期性的触发信号。配置 PWMB\_CR2 寄存器的 MMS=010, 使每个更新事件时 TRGO 能输出一个上升沿。
2. PWMB 输出的 TRGO 信号链接到 PWMA。PWMA 需要配置成触发从模式, 使用 ITR2 作为输入触发信号。以上操作可以通过配置 PWMA\_SMCR 寄存器的 TS=010 实现。
3. 配置 PWMA\_SMCR 寄存器的 SMS=111 将时钟/触发控制器设置为外部时钟模式 1。此操作将使 PWMB 输出的周期性触发信号 TRGO 的上升沿驱动 PWMA 的时钟。
4. 最后, 置位 PWMB 的 CEN 位 (PWMB\_CR1 寄存器中), 使能两个 PWM。

主触发从模式的例子



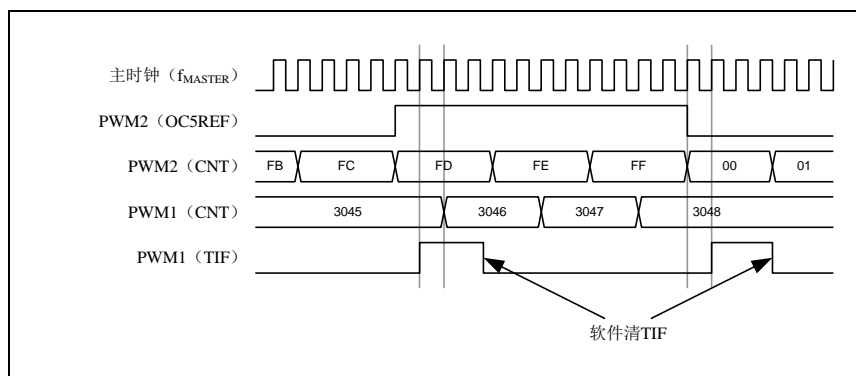
### 使用 PWMB 使能 PWMA

在本例中, 我们用 PWMB 的比较输出使能 PWMA。PWMA 仅在 PWMB 的 OC1REF 信号为高时按照自己的驱动时钟计数。两个 PWM 都使用 4 分频的  $f_{MASTER}$  为时钟 ( $f_{CK\_CNT} = f_{MASTER}/4$ )。

1. 配置 PWMB 为主模式, 将比较输出信号 (OC5REF) 作为触发信号输出。(配置 PWMB\_CR2 寄存器的 MMS=100)。
2. 配置 PWMB 的 OC5REF 信号的波形 (PWMB\_CCMR1 寄存器)。
3. 配置 PWMA 把 PWMB 的输出作为自己的触发输入信号 (配置 PWMA\_SMCR 寄存器的 TS=010)。
4. 配置 PWMA 为门控触发模式 (配置 PWMA\_SMCR 寄存器的 SMS=101)。
5. 置位 CEN 位 (PWMA\_CR1 寄存器), 使能 PWMA。
6. 置位 CEN 位 (PWMB\_CR1 寄存器), 使能 PWMB。

注意: 两个 PWM 的时钟并不同步, 但仅影响 PWMA 的使能信号。

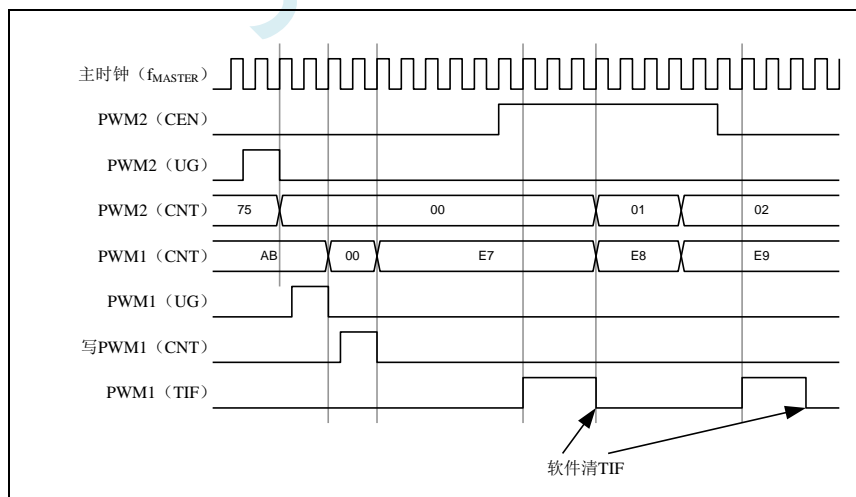
PWMB 的输出门控触发 PWMA



上图中, PWMA 的计数器和预分频器都没有在启动前初始化, 所以都是从现有值开始计数的。如果在启动 PWMB 之前复位两个定时器, 用户就可以写入期望的数值到 PWMA 的计数器, 使之从指定值开始计数。对 PWMA 的复位操作可以通过软件写 PWMA\_EGR 寄存器的 UG 位实现。

在下面这个例子中, 我们使 PWMB 和 PWMA 同步。PWMB 为主模式并从 0 启动计数。PWMA 为触发从模式, 并从 0xE7 启动计数。两个 PWM 采用相同的分频系数。当清除 PWMB\_CR1 寄存器的 CEN 位时, PWMB 被禁止, 同时 PWMA 停止计数。

1. 配置 PWMB 为主模式, 将比较输出信号 (OC5REF) 作为触发信号输出。(配置 PWMB\_CR2 寄存器的 MMS=100)。
2. 配置 PWMB 的 OC5REF 信号的波形 (PWMB\_CCMR1 寄存器)。
3. 配置 PWMA 把 PWMB 的输出作为自己的触发输入信号 (配置 PWMA\_SMCR 寄存器的 TS=010)。
4. 配置 PWMA 为门控触发模式 (配置 PWMA\_SMCR 寄存器的 SMS=101)。
5. 通过对 UG 位 (PWMB\_EGR 寄存器) 写 1, 复位 PWMB。
6. 通过对 UG 位 (PWMA\_EGR 寄存器) 写 1, 复位 PWMA。
7. 将 0xE7 写入 PWMA 的计数器中 (PWMA\_CNTRL), 初始化 PWMA。
8. 通过对 CEN 位 (PWMA\_CR1 寄存器) 写 1, 使能 PWMA。
9. 通过对 CEN 位 (PWMB\_CR1 寄存器) 写 1, 启动 PWMB。
10. 通过对 CEN 位 (PWMB\_CR1 寄存器) 写 0, 停止 PWMB。



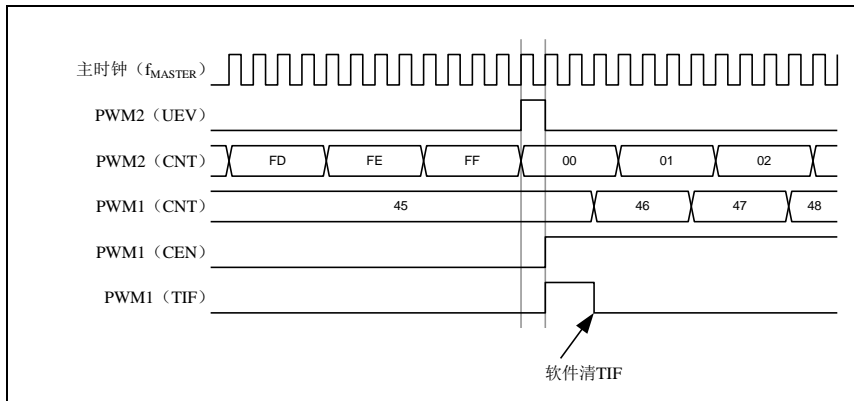
### 使用 PWMB 启动 PWMA

在本例中, 我们用 PWMB 的更新事件来启动 PWMA。

PWMA 在 PWMB 发生更新事件时按照 PWMA 自己的驱动时钟从它的现有值开始计数 (可以是非 0 值)。PWMA 在收到触发信号后自动使能 CEN 位, 并开始计数, 一直持续到用户向 PWMA\_CR1 寄存器的 CEN 位写 0。两个 PWM 都使用 4 分频的  $f_{MASTER}$  作为驱动时钟 ( $f_{CK\_CNT} = f_{MASTER}/4$ )。

1. 配置 PWMB 为主模式，输出更新信号 (UEV)。(配置 PWMB\_CR2 寄存器的 MMS=010)。
2. 配置 PWMB 的周期 (PWMB\_ARR 寄存器)。
3. 配置 PWMA 用 PWMB 的输出作为输入的触发信号 (配置 PWMA\_SMCR 寄存器的 TS=010)。
4. 配置 PWMA 为触发模式 (配置 PWMA\_SMCR 寄存器的 SMS=110)。
5. 置位 CEN 位 (PWMB\_CR1 寄存器) 启动 PWMB。

PWMB 的更新事件 (PWMB-UEV) 触发 PWMA



如同前面的例子，用户也可以在启动计数器前对它们初始化。

### 用外部信号同步的触发两个 PWM

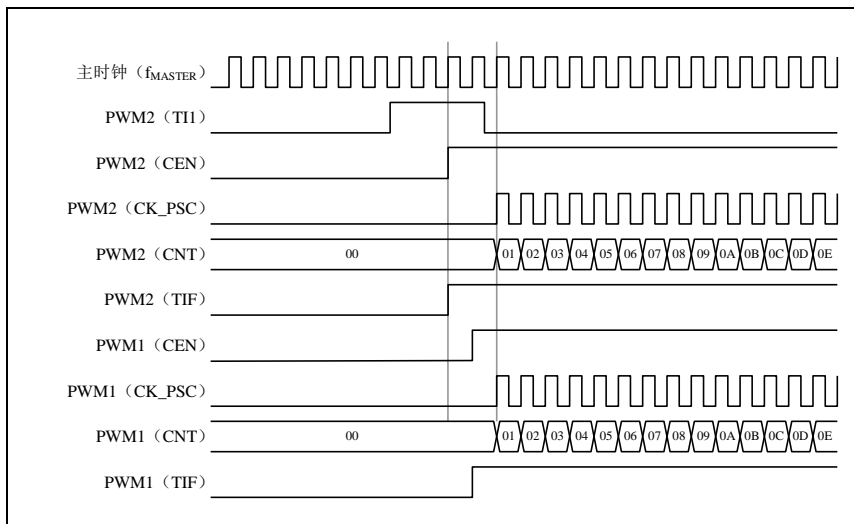
在本例中，使用 TI1 的上升沿使能 PWMB，并同时使能 PWMA。为了保持定时器的对齐，PWMB 需要配置成主/从模式（对于 TI1 信号为从模式，对于 PWMA 为主模式）。

1. 配置 PWMB 为主模式，以输出使能信号作为 PWMA 的触发 (配置 PWMB\_CR2 寄存器的 MMS=001)。
2. 配置 PWMB 为从模式，把 TI1 信号作为输入的触发信号 (配置 PWMB\_SMCR 寄存器的 TS=100)。
3. 配置 PWMB 的触发模式 (配置 PWMB\_SMCR 寄存器的 SMS=110)。
4. 配置 PWMB 为主/从模式 (配置 PWMB\_SMCR 寄存器的 MSM=1)。
5. 配置 PWMA 以 PWMB 的输出为输入触发信号 (配置 PWMA\_SMCR 寄存器的 TS=010)。
6. 配置 PWMA 的触发模式 (配置 PWMA\_SMCR 寄存器的 SMS=110)。

当 TI1 上出现上升沿时，两个定时器同步的开始计数，并且 TIF 位都被置起。

注意：在本例中，两个定时器在启动前都进行了初始化（设置 UG 位），所以它们都从 0 开始计数，但是用户也可以通过修改计数器寄存器 (PWMA\_CNT) 来插入一个偏移量，这样的话，在 PWMB 的 CK\_PSC 信号和 CNT\_EN 信号间会插入延时。

PWMB 的 TI1 信号触发 PWMB 和 PWMA



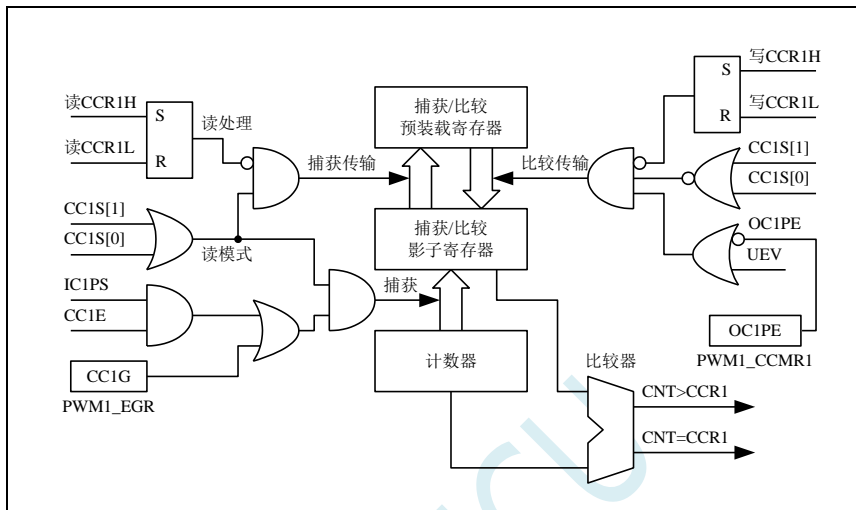


## 22.5 捕获/比较通道

PWM1P、PWM2P、PWM3P、PWM4P 可以用作输入捕获，PWM1P/PWM1N、PWM2P/PWM2N、PWM3P/PWM3N、PWM4P/PWM4N 可以输出比较，这个功能可以通过配置捕获/比较通道模式寄存器（PWMA\_CCMR*i*）的 CCiS 通道选择位来实现，此处的 *i* 代表 1~4 的通道数。

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器（包含影子寄存器）来构建的，包括捕获的输入部分（数字滤波、多路复用和预分频器）和输出部分（比较器和输出控制）。

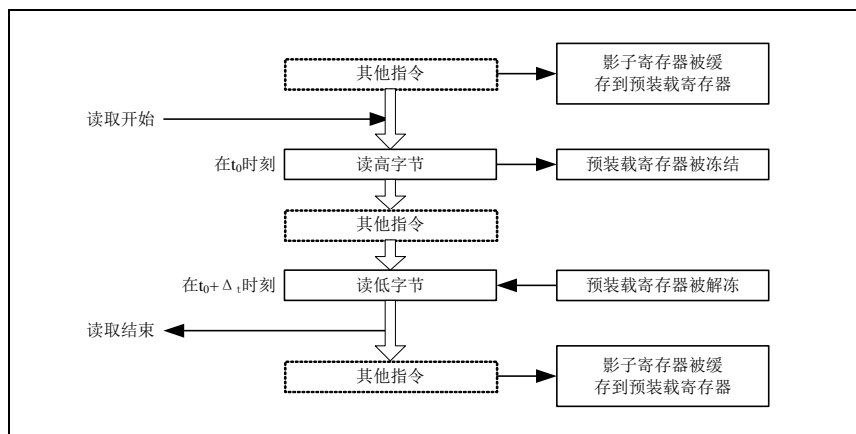
捕获/比较通道 1 的主要电路（其他通道与此类似）



捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

当通道被配置成输出模式时，可以随时访问 PWMA CCRi 寄存器。

当通道被配置成输入模式时，对 PWMA\_CCRi 寄存器的读操作类似于计数器的读操作。当捕获发生时，计数器的内容被捕获到 PWMA\_CCRi 影子寄存器，随后再复制到预装载寄存器中。在读操作进行中，预装载寄存器是被冻结的。



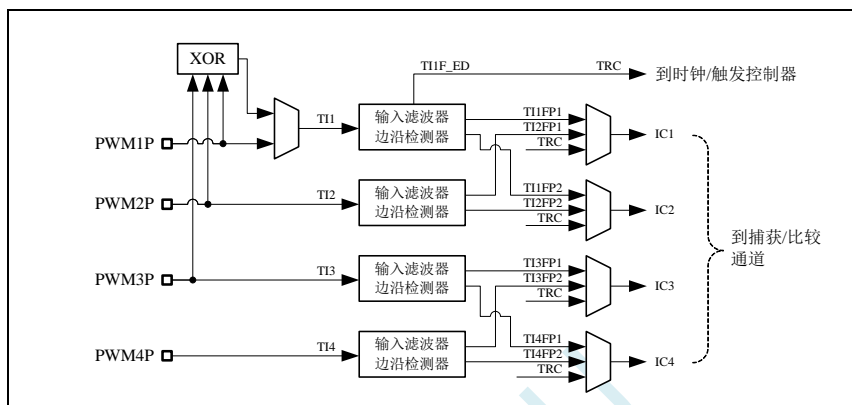
上图描述了 16 位的 CCRi 寄存器的读操作流程，被缓存的数据将保持不变直到读流程结束。在整个读流程结束后，如果仅仅读了 PWMA\_CCRiL 寄存器，返回计数器数值的低位。如果在读了低位数据以后再读高位数据，将不再返回同样的低位数据。

## 22.5.1 16 位 PWMA\_CCRi 寄存器的写流程

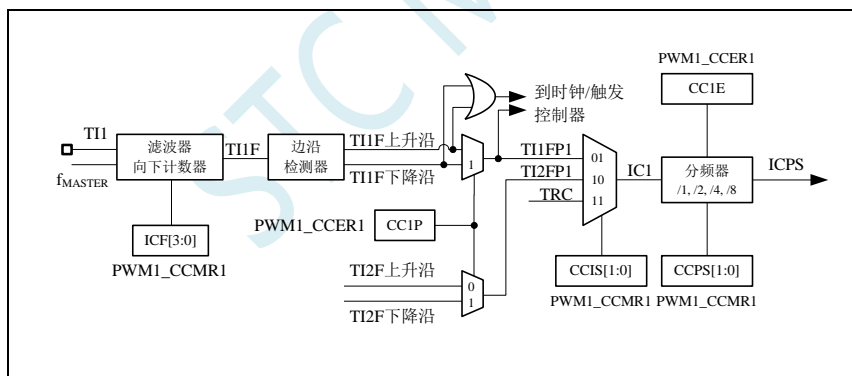
16 位 PWMA\_CCRi 寄存器的写操作通过预装载寄存器完成。必需使用两条指令来完成整个流程，一条指令对应一个字节。必需先写高位字节。在写高位字节时，影子寄存器的更新被禁止直到低位字节的写操作完成。

## 22.5.2 输入模块

输入模块的框图



如图，输入部分对相应的  $TIx$  输入信号采样，并产生一个滤波后的信号  $TIxF$ 。然后，一个带极性选择的边缘监测器产生一个信号 ( $TIxFPx$ )，它可以作为触发模式控制器的输入触发或者作为捕获控制。该信号通过预分频后进入捕获寄存器 ( $ICxPS$ )。



## 22.5.3 输入捕获模式

在输入捕获模式下，当检测到  $ICi$  信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器 ( $PWMA\_CCRx$ ) 中。当发生捕获事件时，相应的  $CCiIF$  标志 ( $PWMA\_SR$  寄存器) 被置 1。如果  $PWMA\_IER$  寄存器的  $CCiIE$  位被置位，也就是使能了中断，则将产生中断请求。如果发生捕获事件时  $CCiIF$  标志已经为高，那么重复捕获标志  $CCiOF$  ( $PWMA\_SR2$  寄存器) 被置 1。写  $CCiIF=0$  或读取存储在  $PWMA\_CCRiL$  寄存器中的捕获数据都可清除  $CCiIF$ 。写  $CCiOF=0$  可清除  $CCiOF$ 。

### PWM 输入信号上升沿时捕获

以下例子说明如何在  $TI1$  输入的上升沿时捕获计数器的值到  $PWMA\_CCR1$  寄存器中，步骤如下：

1. 选择有效输入端，设置  $PWMA\_CCMR1$  寄存器中的  $CC1S=01$ ，此时通道被配置为输入，并且  $PWMA\_CCR1$  寄存器变为只读。
2. 根据输入信号  $TI1$  的特点，可通过配置  $PWMA\_CCMR1$  寄存器中的  $IC1F$  位来设置相应的输入滤波



器的滤波时间。假设输入信号在最多 5 个时钟周期的时间内抖动, 我们须配置滤波器的带宽长于 5 个时钟周期; 因此我们可以连续采样 8 次, 以确认在 TI1 上一次真实的边沿变换, 即在 PWMA\_CCMR1 寄存器中写入 IC1F=0011, 此时, 只有连续采样到 8 个相同的 TI1 信号, 信号才为有效 (采样频率为  $f_{MASTER}$ )。

3. 选择 TI1 通道的有效转换边沿, 在 PWMA\_CCER1 寄存器中写入 CC1P=0 (上升沿)。
4. 配置输入预分频器。在本例中, 我们希望捕获发生在每一个有效的电平转换时刻, 因此预分频器被禁止 (写 PWMA\_CCMR1 寄存器的 IC1PS=00)。
5. 设置 PWMA\_CCER1 寄存器的 CC1E=1, 允许捕获计数器的值到捕获寄存器中。
6. 如果需要, 通过设置 PWMA\_IER 寄存器中的 CC1IE 位允许相关中断请求。

当发生一个输入捕获时:

- 当产生有效的电平转换时, 计数器的值被传送到 PWMA\_CCR1 寄存器。
- CC1IF 标志被设置。当发生至少 2 个连续的捕获时, 而 CC1IF 未曾被清除时, CC1OF 也被置 1。
- 如设置了 CC1IE 位, 则会产生一个中断。

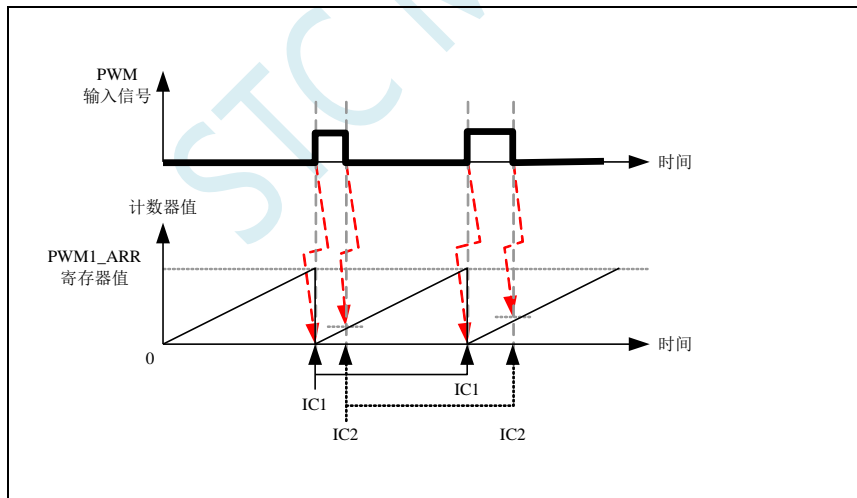
为了处理捕获溢出事件 (CC1OF 位), 建议在读出重复捕获标志之前读取数据, 这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的重复捕获信息。

注意: 设置 PWMA\_EGR 寄存器中相应的 CCiG 位, 可以通过软件产生输入捕获中断。

### PWM 输入信号测量

该模式是输入捕获模式的一个特例, 除下列区别外, 操作与输入捕获模式相同:

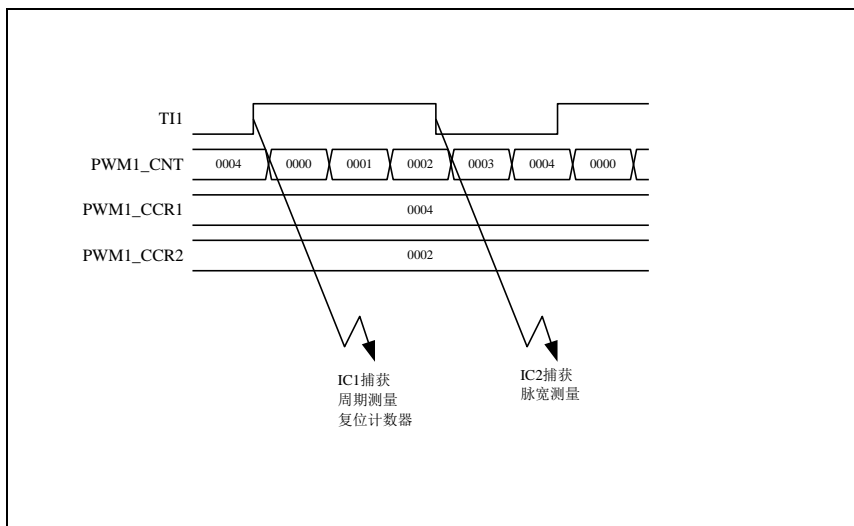
- 两个 ICi 信号被映射至同一个 TIi 输入。
- 这两个 ICi 信号的有效边沿的极性相反。
- 其中一个 TIiFP 信号被作为触发输入信号, 而触发模式控制器被配置成复位触发模式。



例如, 你可以用以下方式测量 TI1 上输入的 PWM 信号的周期 (PWMA\_CCR1 寄存器) 和占空比 (PWMA\_CCR2 寄存器)。

1. 选择 PWMA\_CCR1 的有效输入: 置 PWMA\_CCMR1 寄存器的 CC1S=01 (选中 TI1FP1)。
2. 选择 TI1FP1 的有效极性: 置 CC1P=0 (上升沿有效)。
3. 选择 PWMA\_CCR2 的有效输入: 置 PWMA\_CCMR2 寄存器的 CC2S=10 (选中 TI1FP2)。
4. 选择 TI1FP2 的有效极性 (捕获数据到 PWMA\_CCR2): 置 CC2P=1 (下降沿有效)。
5. 选择有效的触发输入信号: 置 PWMA\_SMCR 寄存器中的 TS=101 (选择 TI1FP1)。
6. 配置触发模式控制器为复位触发模式: 置 PWMA\_SMCR 中的 SMS=100。
7. 使能捕获: 置 PWMA\_CCER1 寄存器中 CC1E=1, CC2E=1。

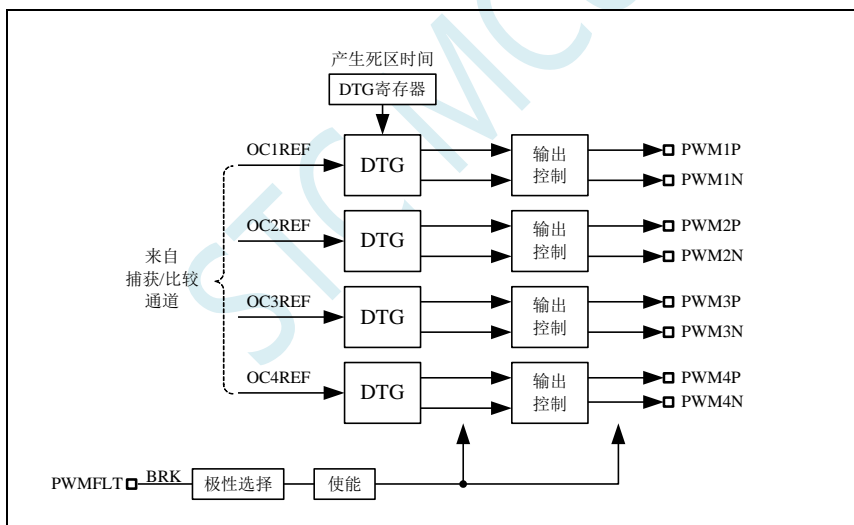
PWM 输入信号测量实例



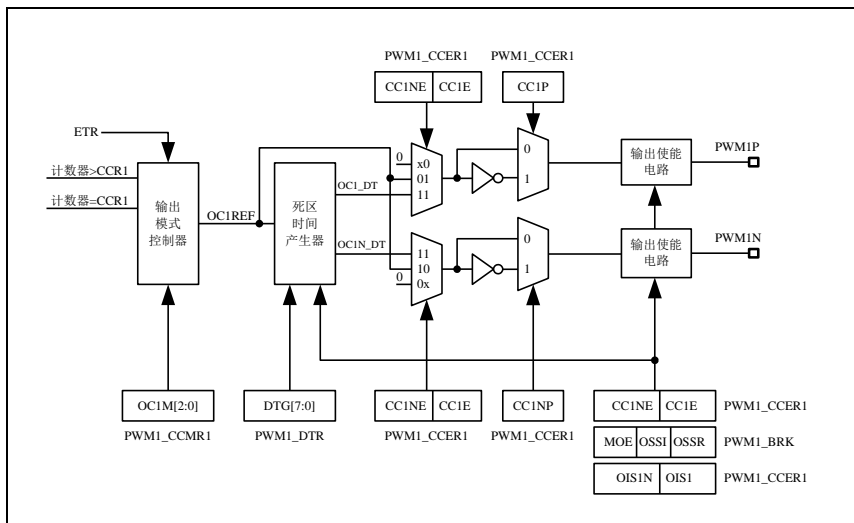
## 22.5.4 输出模块

输出模块会产生一个用来做参考的中间波形，称为 OCiREF（高有效）。刹车功能和极性的处理都在模块的最后处理。

输出模块框图



通道 1 详细的带互补输出的输出模块框图（其他通道类似）



### 22.5.5 强制输出模式

在输出模式下，输出比较信号能够直接由软件强制为高或低状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 PWMA\_CCMRi 寄存器的 OCiM=101，可强制 OCiREF 信号为高。

置 PWMA\_CCMRi 寄存器的 OCiM=100，可强制 OCiREF 信号为低。

OCi/OCiN 的输出是高还是低则取决于 CCiP/CCiNP 极性标志位。

该模式下，在 PWMA\_CCRi 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改，也仍然会产生相应的中断。

### 22.5.6 输出比较模式

此模式用来控制一个输出波形或者指示一段给定的时间已经达到。

当计数器与捕获/比较寄存器的内容相匹配时，有如下操作：

- 根据不同的输出比较模式，相应的 OCi 输出信号：
  - 保持不变 (OCiM=000)
  - 设置为有效电平 (OCiM=001)
  - 设置为无效电平 (OCiM=010)
  - 翻转 (OCiM=011)
- 设置中断状态寄存器中的标志位 (PWMA\_SR1 寄存器中的 CCiIF 位)。
- 若设置了相应的中断使能位 (PWMA\_IER 寄存器中的 CCiIE 位)，则产生一个中断。

PWMA\_CCMRi 寄存器的 OCiM 位用于选择输出比较模式，而 PWMA\_CCMRi 寄存器的 CCiP 位用于选择有效和无效的电平极性。PWMA\_CCMRi 寄存器的 OCiPE 位用于选择 PWMA\_CCRi 寄存器是否需要使用预装载寄存器。在输出比较模式下，更新事件 UEV 对 OCiREF 和 OCi 输出没有影响。时间精度为计数器的一个计数周期。输出比较模式也能用来输出一个单脉冲。

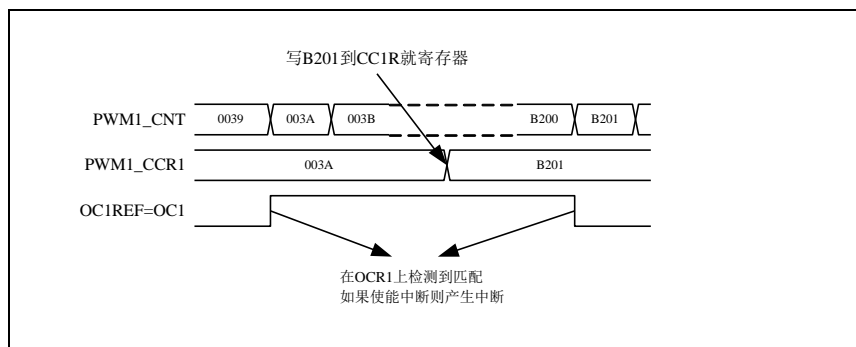
输出比较模式的配置步骤：

1. 选择计数器时钟（内部、外部或者预分频器）。
2. 将相应的数据写入 PWMA\_ARR 和 PWMA\_CCRi 寄存器中。
3. 如果要产生一个中断请求，设置 CCiIE 位。
4. 选择输出模式步骤：
  1. 设置 OCiM=011，在计数器与 CCRi 匹配时翻转 OCiM 管脚的输出
  2. 设置 OCiPE = 0，禁用预装载寄存器

3. 设置  $CCiP = 0$ ，选择高电平为有效电平
4. 设置  $CCiE = 1$ ，使能输出
5. 设置  $PWMA\_CR1$  寄存器的  $CEN$  位来启动计数器

$PWMA\_CCRi$  寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器 ( $OCiPE=0$ )，否则  $PWMA\_CCRi$  的影子寄存器只能在发生下一次更新事件时被更新。

输出比较模式，翻转  $OC1$



## 22.5.7 PWM 模式

脉冲宽度调制 (PWM) 模式可以产生一个由  $PWMA\_ARR$  寄存器确定频率，由  $PWMA\_CCRi$  寄存器确定占空比的信号。

在  $PWMA\_CCMRi$  寄存器中的  $OCiM$  位写入 110 (PWM 模式 1) 或 111 (PWM 模式 2)，能够独立地设置每个  $OCi$  输出通道产生一路 PWM。必须设置  $PWMA\_CCMRi$  寄存器的  $OCiPE$  位使能相应的预装载寄存器，也可以设置  $PWMA\_CR1$  寄存器的  $ARPE$  位使能自动重载的预装载寄存器（在向上计数模式或中央对称模式中）。

由于仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置  $PWMA\_EGR$  寄存器的  $UG$  位来初始化所有的寄存器。

$OCi$  的极性可以通过软件在  $PWMA\_CCERi$  寄存器中的  $CCiP$  位设置，它可以设置为高电平有效或低电平有效。 $OCi$  的输出使能通过  $PWMA\_CCERi$  和  $PWMA\_BKR$  寄存器中的  $CCiE$ 、 $MOE$ 、 $OISi$ 、 $OSSR$  和  $OSSI$  位的组合来控制。

在 PWM 模式（模式 1 或模式 2）下， $PWMA\_CNT$  和  $PWMA\_CCRi$  始终在进行比较，（依据计数器的计数方向）以确定是否符合  $PWMA\_CCRi \leq PWMA\_CNT$  或者  $PWMA\_CNT \leq PWMA\_CCRi$ 。

根据  $PWMA\_CR1$  寄存器中  $CMS$  位域的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的 PWM 信号。

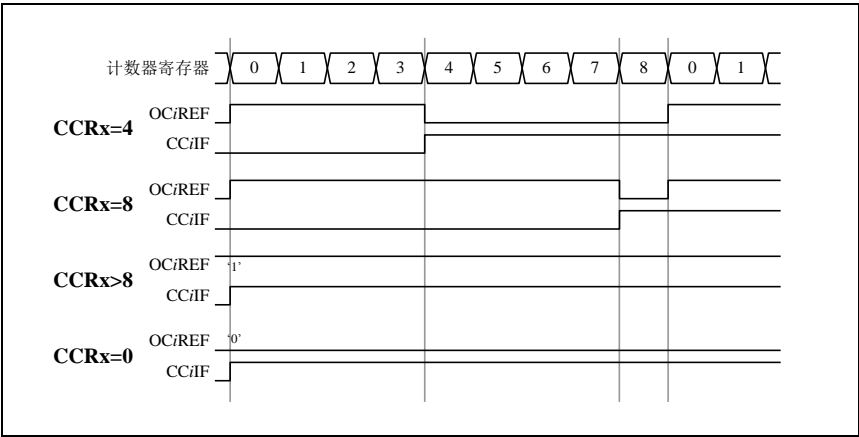
### PWM 边沿对齐模式

#### 向上计数配置

当  $PWMA\_CR1$  寄存器中的  $DIR$  位为 0 时，执行向上计数。

下面是一个 PWM 模式 1 的例子。当  $PWMA\_CNT < PWMA\_CCRi$  时，PWM 参考信号  $OCiREF$  为高，否则为低。如果  $PWMA\_CCRi$  中的比较值大于自动重载载值 ( $PWMA\_ARR$ )，则  $OCiREF$  保持为高。如果比较值为 0，则  $OCiREF$  保持为低。

边沿对齐，PWM 模式 1 的波形 ( $ARR=8$ )



向下计数的配置

当 PWMA\_CR1 寄存器的 DIR 位为 1 时，执行向下计数。

在 PWM 模式 1 时，当 PWMA\_CNT>PWMA\_CCRi 时参考信号 OCiREF 为低，否则为高。如果 PWMA\_CCRi 中的比较值大于 PWMA\_ARR 中的自动重装载值，则 OCiREF 保持为高。该模式下不能产生占空比为 0% 的 PWM 波形。

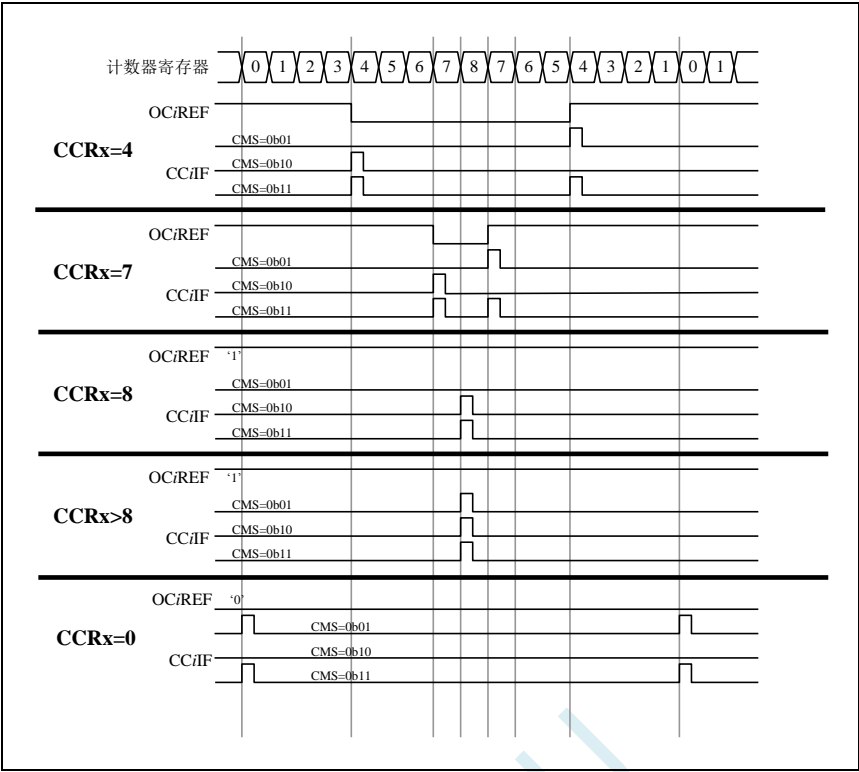
PWM 中央对齐模式

当 PWMA\_CR1 寄存器中的 CMS 位不为'00'时为中央对齐模式（所有其他的配置对 OCiREF/OCi 信号都有相同的作用）。

根据不同的 CMS 位的设置，比较标志可以在计数器向上计数，向下计数，或向上和向下计数时被置 1。PWMA\_CR1 寄存器中的计数方向位（DIR）由硬件更新，不要用软件修改它。

下面给出了一些中央对齐的 PWM 波形的例子：

- PWMA\_ARR=8
- PWM 模式 1
- 标志位在以下三种情况下被置位：
  - 只有在计数器向下计数时（CMS=01）
  - 只有在计数器向上计数时（CMS=10）
  - 在计数器向上和向下计数时（CMS=11）
- 中央对齐的 PWM 波形（ARR=8）



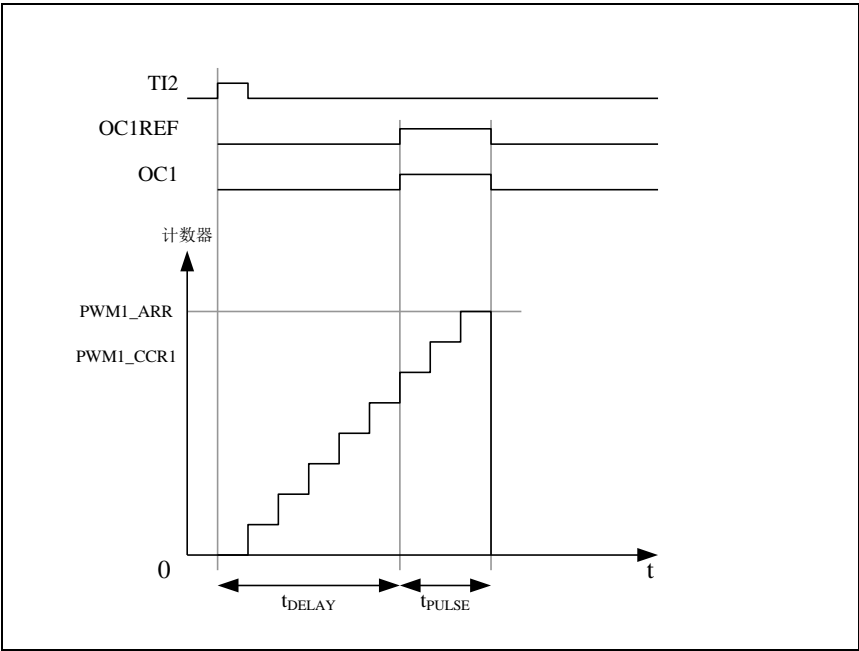
单脉冲模式

单脉冲模式（OPM）是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可控的脉冲。

可以通过时钟/触发控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 PWMA\_CR1 寄存器的 OPM 位将选择单脉冲模式，此时计数器自动地在下一个更新事件 UEV 时停止。仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前（当定时器正在等待触发），必须如下配置：

- 向上计数方式：计数器  $CNT < CCRi \leq ARR$ ，
- 向下计数方式：计数器  $CNT > CCRi$ 。

单脉冲模式图例



例如, 在从 TI2 输入脚上检测到一个上升沿之后延迟  $t_{\text{DELAY}}$ , 在 OC1 上产生一个  $t_{\text{PULSE}}$  宽度的正脉冲: (假定 IC2 作为触发 1 通道的触发源)

- 置 PWMA\_CCMR2 寄存器的 CC2S=01, 把 IC2 映射到 TI2。
- 置 PWMA\_CCER1 寄存器的 CC2P=0, 使 IC2 能够检测上升沿。
- 置 PWMA\_SMCR 寄存器的 TS=110, 使 IC2 作为时钟/触发控制器的触发源 (TRGI)。
- 置 PWMA\_SMCR 寄存器的 SMS=110 (触发模式), IC2 被用来启动计数器。OPM 的波形由写入比较寄存器的数值决定 (要考虑时钟频率和计数器预分频器)。
- $t_{\text{DELAY}}$  由 PWMA\_CCR1 寄存器中的值定义。
- $t_{\text{PULSE}}$  由自动装载值和比较值之间的差值定义 (PWMA\_ARR - PWMA\_CCR1)。
- 假定当发生比较匹配时要产生从 0 到 1 的波形, 当计数器达到预装载值时要产生一个从 1 到 0 的波形, 首先要置 PWMA\_CCMR1 寄存器的 OCiM=111, 进入 PWM 模式 2, 根据需要有选择的设置 PWMA\_CCMR1 寄存器的 OCiPE=1, 置位 PWMA\_CR1 寄存器中的 ARPE, 使能预装载寄存器, 然后在 PWMA\_CCR1 寄存器中填写比较值, 在 PWMA\_ARR 寄存器中填写自动装载值, 设置 UG 位来产生一个更新事件, 然后等待在 TI2 上的一个外部触发事件。

在这个例子中, PWMA\_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲, 所以设置 PWMA\_CR1 寄存器中的 OPM=1, 在下一个更新事件 (当计数器从自动装载值翻转到 0) 时停止计数。

### OCx 快速使能 (特殊情况)

在单脉冲模式下, 对 TIi 输入脚的边沿检测会设置 CEN 位以启动计数器, 然后计数器和比较值间的比较操作产生了单脉冲的输出。但是这些操作需要一定的时钟周期, 因此它限制了可得到的最小延时  $t_{\text{DELAY}}$ 。

如果要以最小延时输出波形, 可以设置 PWMA\_CCMRi 寄存器中的 OCiFE 位, 此时强制 OCiREF (和 OCx) 直接响应激励而不再依赖比较的结果, 输出的波形与比较匹配时的波形一样。OCiFE 只在通道配置为 PWMA 和 PWMB 模式时起作用。

### 互补输出和死区插入

PWMA 能够输出两路互补信号, 并且能够管理输出的瞬时关断和接通, 这段时间通常被称为死区, 用户应该根据连接的输出器件和它们的特性 (电平转换的延时、电源开关的延时等) 来调整死区时间。

配置 PWMA\_CCERi 寄存器中的 CCiP 和 CCiNP 位, 可以为每一个输出独立地选择极性 (主输出 OCi 或互补输出 OCiN)。互补信号 OCi 和 OCiN 通过下列控制位的组合进行控制: PWMA\_CCERi 寄存器的 CCiE 和 CCiNE 位, PWMA\_BKR 寄存器中的 MOE、OISi、OISiN、OSSI 和 OSSR 位。特别的是, 在转换到 IDLE 状态时 (MOE 下降到 0) 死区控制被激活。

同时设置 CCiE 和 CCiNE 位将插入死区, 如果存在刹车电路, 则还要设置 MOE 位。每一个通道都有一个 8 位的死区发生器。

如果 OCi 和 OCiN 为高有效:

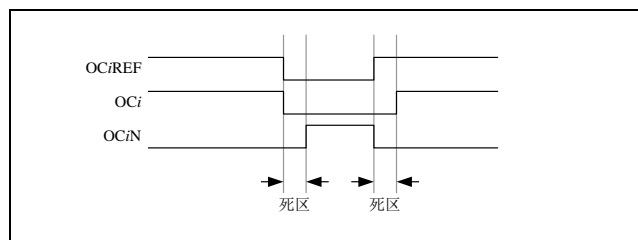
- OCi 输出信号与 OCiREF 相同, 只是它的上升沿相对于 OCiREF 的上升沿有一个延迟。
- OCiN 输出信号与 OCiREF 相反, 只是它的上升沿相对于 OCiREF 的下降沿有一个延迟。

如果延迟大于当前有效的输出宽度 (OCi 或者 OCiN), 则不会产生相应的脉冲。

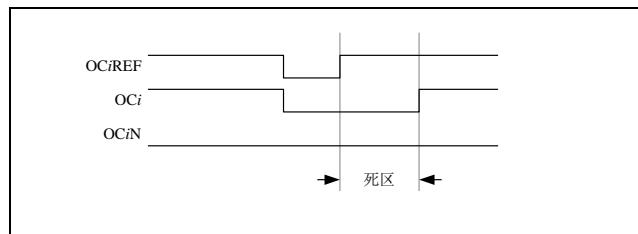
下列几张图显示了死区发生器的输出信号和当前参考信号 OCiREF 之间的关系。(假设 CCiP=0、CCiNP=0、MOE=1、CCiE=1 并且 CCiNE=1)

带死区插入的互补输出

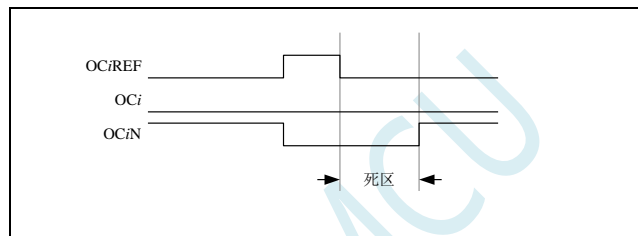




死区波形延迟大于负脉冲



死区波形延迟大于正脉冲



每一个通道的死区延时都是相同的，是由 PWMA\_DTR 寄存器中的 DTG 位编程配置。

### 重定向 OCiREF 到 OCi 或 OCiN

在输出模式下(强制输出、输出比较或 PWM 输出),通过配置 PWMA\_CCERi 寄存器的 CCiE 和 CCiNE 位, OCiREF 可以被重定向到 OCi 或者 OCiN 的输出。

这个功能可以在互补输出处于无效电平时,在某个输出上送出一个特殊的波形(例如 PWM 或者静态有效电平)。另一个作用是,让两个输出同时处于无效电平,或同时处于有效电平(此时仍然是带死区的互补输出)。

注:当只使能 OCiN (CCiE=0, CCiNE=1) 时,它不会反相,而当 OCiREF 变高时立即有效。例如,如果 CCiNP=0,则 OCiN=OCiREF。另一方面,当 OCi 和 OCiN 都被使能时(CCiE=CCiNE=1),当 OCiREF 为高时 OCi 有效;而 OCiN 相反,当 OCiREF 低时 OCiN 变为有效。

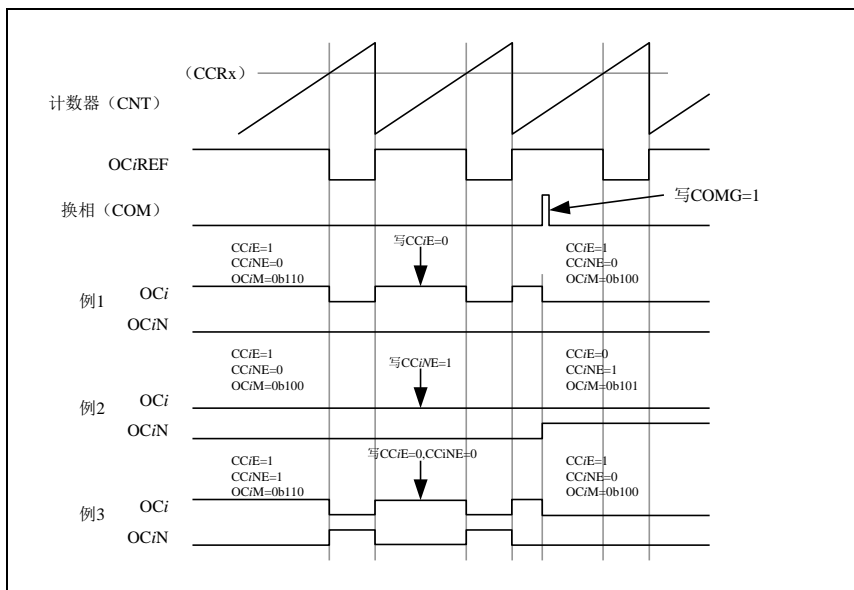
### 针对马达控制的六步 PWM 输出

当在一个通道上需要互补输出时,预装载位有 OCiM、CCiE 和 CCiNE。在发生 COM 换相事件时,这些预装载位被传送到影子寄存器位。这样你就可以预先设置好下一步骤配置,并在同一个时刻同时修改所有通道的配置。COM 可以通过设置 PWMA\_EGR 寄存器的 COMG 位由软件产生,或在 TRGI 上升沿由硬件产生。

下图显示当发生 COM 事件时,三种不同配置下 OCx 和 OCxN 输出。

产生六步 PWM, 使用 COM 的例子 (OSSR=1)





## 22.5.8 使用刹车功能 (PWMFLT)

刹车功能常用于马达控制中。当使用刹车功能时,依据相应的控制位(PWMA\_BKR 寄存器中的 MOE、OSSI 和 OSSR 位), 输出使能信号和无效电平都会被修改。

系统复位后, 刹车电路被禁止, MOE 位为低。设置 PWMA\_BKR 寄存器中的 BKE 位可以使能刹车功能。刹车输入信号的极性可以通过配置同一个寄存器中的 BKP 位选择。BKE 和 BKP 可以被同时修改。

MOE 下降沿相对于时钟模块可以是异步的, 因此在实际信号(作用在输出端)和同步控制位(在 PWMA\_BKR 寄存器中)之间设置了一个再同步电路。这个再同步电路会在异步信号和同步信号之间产生延迟。特别的, 如果当它为低时写 MOE=1, 则读出它之前必须先插入一个延时(空指令)才能读到正确的值。这是因为写入的是异步信号而读的是同步信号。

当发生刹车时(在刹车输入端出现选定的电平), 有下述动作:

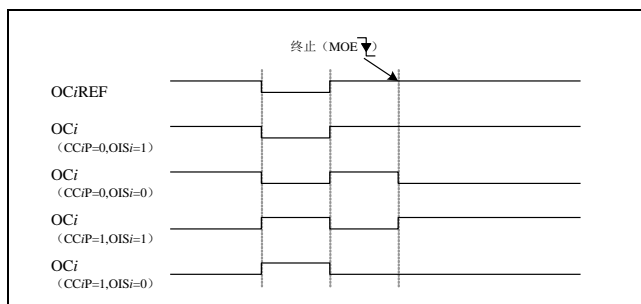
- MOE 位被异步地清除, 将输出置于无效状态、空闲状态或者复位状态(由 OSSI 位选择)。这个特性在 MCU 的振荡器关闭时依然有效。
- 一旦 MOE=0, 每一个输出通道输出由 PWMA\_OISR 寄存器的 OISi 位设定的电平。如果 OSSI=0, 则定时器不再控制输出使能信号, 否则输出使能信号始终为高。
- 当使用互补输出时:
  - 输出首先被置于复位状态即无效的状态(取决于极性)。这是异步操作, 即使定时器没有时钟时, 此功能也有效。
  - 如果定时器的时钟依然存在, 死区生成器将会重新生效, 在死区之后根据 OISi 和 OISiN 位指示的电平驱动输出端口。即使在这种情况下, OCi 和 OCiN 也不能被同时驱动到有效的电平。
- 如果设置了 PWMA\_IER 寄存器的 BIE 位, 当刹车状态标志(PWMA\_SR1 寄存器中的 BIF 位)为 1 时, 则产生一个中断。
- 如果设置了 PWMA\_BKR 寄存器中的 AOE 位, 在下一个更新事件 UEV 时 MOE 位被自动置位。例如这可以用来进行波形控制, 否则, MOE 始终保持低直到被再次置 1。这个特性可以被用在安全方面, 你可以把刹车输入连到电源驱动的报警输出、热敏传感器或者其他安全器件上。

注: 刹车输入为电平有效。所以, 当刹车输入有效时, 不能同时(自动地或者通过软件)设置 MOE。同时, 状态标志 BIF 不能被清除。

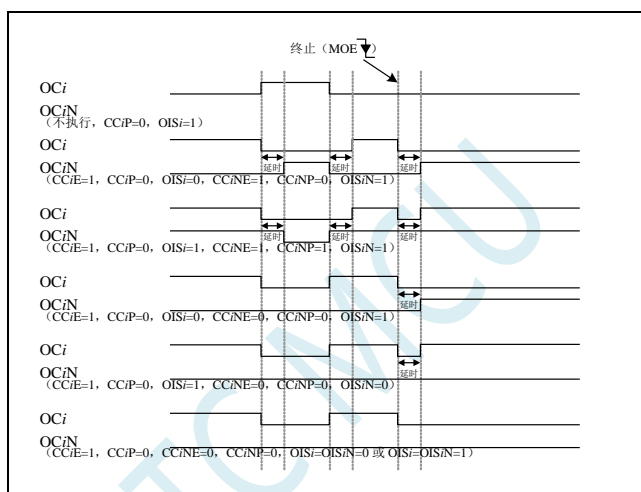
刹车由 BRK 输入产生, 它的有效极性是可编程的, 且由 PWMA\_BKR 寄存器的 BKE 位开启或禁止。除了刹车输入和输出管理, 刹车电路中还实现了写保护以保证应用程序的安全。它允许用户冻结几个配

置参数 (OCi 极性和被禁止时的状态, OCiM 配置, 刹车使能和极性)。用户可以通过 PWMA\_BKR 寄存器的 LOCK 位, 从三种级别的保护中选择一种。在 MCU 复位后 LOCK 位域只能被修改一次。

刹车响应的输出 (不带互补输出的通道)



带互补输出的刹车响应的输出 (PWMA 互补输出)



## 22.5.9 在外部事件发生时清除 OCiREF 信号

对于一个给定的通道, 在 ETRF 输入端 (设置 PWMA\_CCMRi 寄存器中对应的 OCiCE 位为 '1') 的高电平能够把 OCiREF 信号拉低, OCiREF 信号将保持为低直到发生下一次的更新事件 UEV。该功能只能用于输出比较模式和 PWM 模式, 而不能用于强制模式。

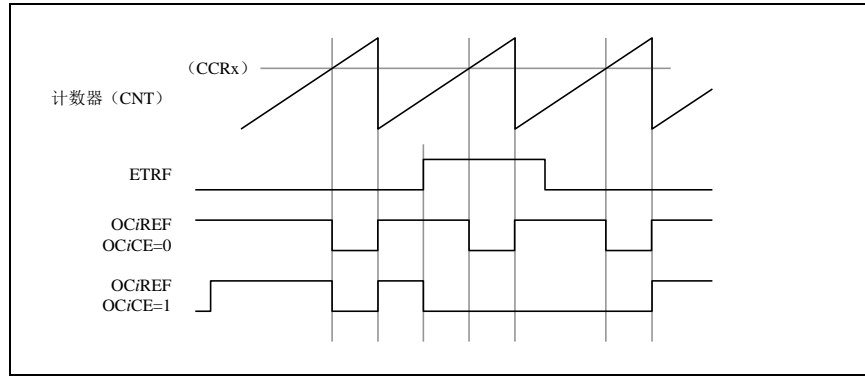
例如, OCiREF 信号可以联到一个比较器的输出, 用于控制电流。这时, ETR 必须配置如下:

1. 外部触发预分频器必须处于关闭: PWMA\_ETR 寄存器中的 ETPS[1:0]=00。
2. 必须禁止外部时钟模式 2: PWMA\_ETR 寄存器中的 ECE=0。
3. 外部触发极性 (ETP) 和外部触发滤波器 (ETF) 可以根据需要配置。

下图显示了当 ETRF 输入变为高时, 对应不同 OCiCE 的值, OCiREF 信号的动作。

在这个例子中, 定时器 PWMA 被置于 PWM 模式。

ETR 清除 PWMA 的 OCiREF



## 22.5.10 编码器接口模式

编码器接口模式一般用于马达控制。

选择编码器接口模式的方法是：

- 如果计数器只在 TI2 的边沿计数，则置 PWMA\_SMCR 寄存器中的 SMS=001；
- 如果只在 TI1 边沿计数，则置 SMS=010；
- 如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=011。

通过设置 PWMA\_CCER1 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以对输入滤波器编程。

两个输入 TI1 和 TI2 被用来作为增量编码器的接口。假定计数器已经启动（PWMA\_CR1 寄存器中的 CEN=1），则计数器在每次 TI1FP1 或 TI2FP2 上产生有效跳变时计数。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号。如果没有滤波和极性变换，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 PWMA\_CR1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数，在任一输入端（TI1 或者 TI2）的跳变都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 PWMA\_ARR 寄存器的自动装载值之间连续计数（根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数）。所以在开始计数之前必须配置 PWMA\_ARR。在这种模式下捕获器、比较器、预分频器、重复计数器、触发输出特性等仍工作如常。编码器模式和外部时钟模式 2 不兼容，因此不能同时操作。

编码器接口模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指示着编码器的位置，计数方向与相连的传感器旋转的方向对应。

下表列出了所有可能的组合（假设 TI1 和 TI2 不同时变换）。

计数方向与编码器信号的关系

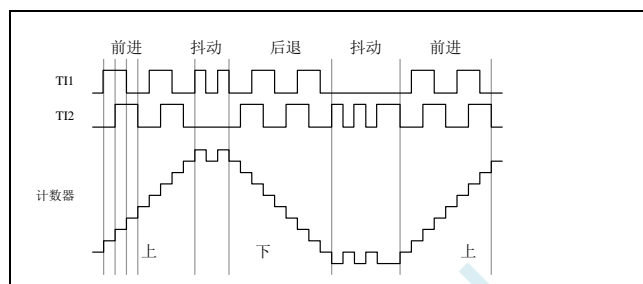
| 有效边沿            | 相对信号的电平<br>(TI1FP1 对应 TI2, TI2FP2 对应 TI1) | TI1FP1 信号 |      | TI2FP2 信号 |      |
|-----------------|---|-----------|------|-----------|------|
|                 |   | 上升        | 下降   | 上升        | 下降   |
| 仅在 TI1 计数       | 高   | 向下计数      | 向上计数 | 不计数       | 不计数  |
|                 | 低   | 向上计数      | 向下计数 | 不计数       | 不计数  |
| 仅在 TI2 计数       | 高   | 不计数       | 不计数  | 向上计数      | 向下计数 |
|                 | 低   | 不计数       | 不计数  | 向下计数      | 向上计数 |
| 在 TI1 和 TI2 上计数 | 高   | 向下计数      | 向上计数 | 向上计数      | 向下计数 |
|                 | 低   | 向上计数      | 向下计数 | 向下计数      | 向上计数 |

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般使用比较器将编码器的差分输出转换成数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下面是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

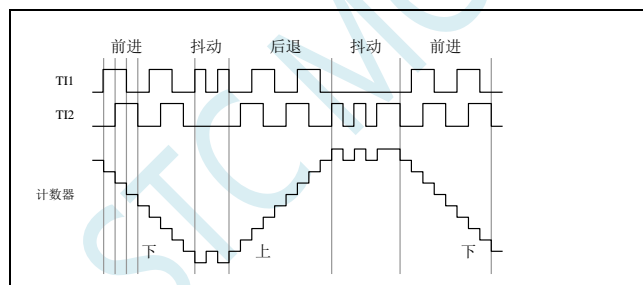
- CC1S=01 (PWMA\_CCMR1 寄存器, IC1FP1 映射到 TI1)
- CC2S=01 (PWMA\_CCMR2 寄存器, IC2FP2 映射到 TI2)
- CC1P=0 (PWMA\_CCER1 寄存器, IC1 不反相, IC1=TI1)
- CC2P=0 (PWMA\_CCER1 寄存器, IC2 不反相, IC2=TI2)
- SMS=011 (PWMA\_SMCR 寄存器, 所有的输入均在上升沿和下降沿有效)。
- CEN=1 (PWMA\_CR1 寄存器, 计数器使能)

编码器模式下的计数器操作实例



下图为当 IC1 极性反相时计数器的操作实例 (CC1P=1, 其他配置与上例相同)

IC1 反相的编码器接口模式实例



当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用另外一个配置在捕获模式下的定时器测量两个编码器事件的间隔，可以获得动态的信息（速度、加速度、减速度）。指示机械零点的编码器输出可被用做此目的。根据两个事件间的间隔，可以按照一定的时间间隔读出计数器。如果可能的话，你可以把计数器的值锁存到第三个输入捕获寄存器（捕获信号必须是周期的并且可以由另一个定时器产生）。

## 22.6 中断

PWMA/PWMB 各有 8 个中断请求源：

- 刹车中断
- 触发中断
- COM 事件中断
- 输入捕捉/输出比较 4 中断
- 输入捕捉/输出比较 3 中断
- 输入捕捉/输出比较 2 中断
- 输入捕捉/输出比较 1 中断
- 更新事件中断（如：计数器上溢，下溢及初始化）

为了使用中断特性，对每个被使用的中断通道，设置 PWM\_IER/PWMB\_IER 寄存器中相应的中断使能位：即 BIE，TIE，COMIE，CCiIE，UIE 位。通过设置 PWMA\_EGR/PWMB\_EGR 寄存器中的相应位，也可以用软件产生上述各个中断源。

STC MCU

## 22.7 PWMA/PWMB 寄存器描述

### 22.7.1 高级 PWM 功能脚切换

| 符号         | 地址    | B7        | B6 | B5        | B4 | B3        | B2     | B1          | B0 |
|------------|-------|-----------|----|-----------|----|-----------|--------|-------------|----|
| PWMA_PS    | FEB2H | C4PS[1:0] |    | C3PS[1:0] |    | C2PS[1:0] |        | C1PS[1:0]   |    |
| PWMB_PS    | FEB6H | C8PS[1:0] |    | C7PS[1:0] |    | C6PS[1:0] |        | C5PS[1:0]   |    |
| PWMA_ETRPS | FEB0H |           |    |           |    |           | BRKAPS | ETRAPS[1:0] |    |
| PWMB_ETRPS | FEB4H |           |    |           |    |           | BRKBPS | ETRBPS[1:0] |    |

C1PS[1:0]: 高级 PWM 通道 1 输出脚选择位

| C1PS[1:0] | PWM1P | PWM1N |
|-----------|-------|-------|
| 00        | P1.0  | P1.1  |
| 01        | P2.0  | P2.1  |
| 10        | P6.0  | P6.1  |
| 11        | -     | -     |

C2PS[1:0]: 高级 PWM 通道 2 输出脚选择位

| C2PS[1:0] | PWM2P                                   | PWM2N |
|-----------|---|-------|
| 00        | <a href="#">P1.2/P5.4<sup>[1]</sup></a> | P1.3  |
| 01        | P2.2                                    | P2.3  |
| 10        | P6.2                                    | P6.3  |
| 11        | -                                       | -     |

注<sup>[1]</sup>: 对于部分没有 P1.2 口的单片机型号, 此功能在 P5.4 口上

C3PS[1:0]: 高级 PWM 通道 3 输出脚选择位

| C3PS[1:0] | PWM3P | PWM3N |
|-----------|-------|-------|
| 00        | P1.4  | P1.5  |
| 01        | P2.4  | P2.5  |
| 10        | P6.4  | P6.5  |
| 11        | -     | -     |

C4PS[1:0]: 高级 PWM 通道 4 输出脚选择位

| C4PS[1:0] | PWM4P | PWM4N |
|-----------|-------|-------|
| 00        | P1.6  | P1.7  |
| 01        | P2.6  | P2.7  |
| 10        | P6.6  | P6.7  |
| 11        | P3.4  | P3.3  |

C5PS[1:0]: 高级 PWM 通道 5 输出脚选择位

| C5PS[1:0] | PWM5 |
|-----------|------|
| 00        | P2.0 |
| 01        | P1.7 |
| 10        | P0.0 |
| 11        | P7.4 |

C6PS[1:0]: 高级 PWM 通道 6 输出脚选择位

| C6PS[1:0] | PWM6 |
|-----------|------|
| 00        | P2.1 |

|    |      |
|----|------|
| 01 | P5.4 |
| 10 | P0.1 |
| 11 | P7.5 |

C7PS[1:0]: 高级 PWM 通道 7 输出脚选择位

| C7PS[1:0] | PWM7 |
|-----------|------|
| 00        | P2.2 |
| 01        | P3.3 |
| 10        | P0.2 |
| 11        | P7.6 |

C8PS[1:0]: 高级 PWM 通道 8 输出脚选择位

| C8PS[1:0] | PWM8 |
|-----------|------|
| 00        | P2.3 |
| 01        | P3.4 |
| 10        | P0.3 |
| 11        | P7.7 |

ETRAPs[1:0]: 高级 PWMA 的外部触发脚 ERI 选择位

| ETRAPs [1:0] | PWMETI |
|--------------|--------|
| 00           | P3.2   |
| 01           | P4.1   |
| 10           | P7.3   |
| 11           | -      |

ETRBPS[1:0]: 高级 PWMB 的外部触发脚 ERIB 选择位

| ETRBPS [1:0] | PWMETI2 |
|--------------|---------|
| 00           | P3.2    |
| 01           | P0.6    |
| 10           | -       |
| 11           | -       |

BRKAPS: 高级 PWMA 的刹车脚 PWMFLT 选择位

| BRKAPS | PWMFLT |
|--------|--------|
| 0      | P3.5   |
| 1      | 比较器的输出 |

BRKBPS: 高级 PWMB 的刹车脚 PWMFLT2 选择位

| BRKBPS | PWMFLT2 |
|--------|---------|
| 0      | P3.5    |
| 1      | 比较器的输出  |

## 22.7.2 输出使能寄存器 (PWM<sub>x</sub>\_ENO)

| 符号       | 地址    | B7    | B6    | B5    | B4    | B3    | B2    | B1    | B0    |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| PWMA_ENO | FEB1H | ENO4N | ENO4P | ENO3N | ENO3P | ENO2N | ENO2P | ENO1N | ENO1P |
| PWMB_ENO | FEB5H | -     | ENO8P | -     | ENO7P | -     | ENO6P | -     | ENO5P |

ENO8P: PWM8 输出控制位

0: 禁止 PWM8 输出

1: 使能 PWM8 输出  
 ENO7P: PWM7 输出控制位  
 0: 禁止 PWM7 输出  
 1: 使能 PWM7 输出  
 ENO6P: PWM6 输出控制位  
 0: 禁止 PWM6 输出  
 1: 使能 PWM6 输出  
 ENO5P: PWM5 输出控制位  
 0: 禁止 PWM5 输出  
 1: 使能 PWM5 输出  
 ENO4N: PWM4N 输出控制位  
 0: 禁止 PWM4N 输出  
 1: 使能 PWM4N 输出  
 ENO4P: PWM4P 输出控制位  
 0: 禁止 PWM4P 输出  
 1: 使能 PWM4P 输出  
 ENO3N: PWM3N 输出控制位  
 0: 禁止 PWM3N 输出  
 1: 使能 PWM3N 输出  
 ENO3P: PWM3P 输出控制位  
 0: 禁止 PWM3P 输出  
 1: 使能 PWM3P 输出  
 ENO2N: PWM2N 输出控制位  
 0: 禁止 PWM2N 输出  
 1: 使能 PWM2N 输出  
 ENO2P: PWM2P 输出控制位  
 0: 禁止 PWM2P 输出  
 1: 使能 PWM2P 输出  
 ENO1N: PWM1N 输出控制位  
 0: 禁止 PWM1N 输出  
 1: 使能 PWM1N 输出  
 ENO1P: PWM1P 输出控制位  
 0: 禁止 PWM1P 输出  
 1: 使能 PWM1P 输出

### 22.7.3 输出附加使能寄存器 (PWMx\_IOAUX)

| 符号         | 地址    | B7    | B6    | B5    | B4    | B3    | B2    | B1    | B0    |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| PWMA_IOAUX | FEB3H | AUX4N | AUX4P | AUX3N | AUX3P | AUX2N | AUX2P | AUX1N | AUX1P |
| PWMB_IOAUX | FEB7H | -     | AUX8P | -     | AUX7P | -     | AUX6P | -     | AUX5P |

AUX8P: PWM8 输出附加控制位  
 0: PWM8 的输出直接由 ENO8P 控制  
 1: PWM8 的输出由 ENO8P 和 PWMB\_BKR 共同控制  
 AUX7P: PWM7 输出附加控制位  
 0: PWM7 的输出直接由 ENO7P 控制



1: PWM7 的输出由 ENO7P 和 PWMB\_BKR 共同控制  
AUX6P: PWM6 输出附加控制位  
0: PWM6 的输出直接由 ENO6P 控制  
1: PWM6 的输出由 ENO6P 和 PWMB\_BKR 共同控制  
AUX5P: PWM5 输出附加控制位  
0: PWM5 的输出直接由 ENO5P 控制  
1: PWM5 的输出由 ENO5P 和 PWMB\_BKR 共同控制  
AUX4N: PWM4N 输出附加控制位  
0: PWM4N 的输出直接由 ENO4N 控制  
1: PWM4N 的输出由 ENO4N 和 PWMA\_BKR 共同控制  
AUX4P: PWM4P 输出附加控制位  
0: PWM4P 的输出直接由 ENO4P 控制  
1: PWM4P 的输出由 ENO4P 和 PWMA\_BKR 共同控制  
AUX3N: PWM3N 输出附加控制位  
0: PWM3N 的输出直接由 ENO3N 控制  
1: PWM3N 的输出由 ENO3N 和 PWMA\_BKR 共同控制  
AUX3P: PWM3P 输出附加控制位  
0: PWM3P 的输出直接由 ENO3P 控制  
1: PWM3P 的输出由 ENO3P 和 PWMA\_BKR 共同控制  
AUX2N: PWM2N 输出附加控制位  
0: PWM2N 的输出直接由 ENO2N 控制  
1: PWM2N 的输出由 ENO2N 和 PWMA\_BKR 共同控制  
AUX2P: PWM2P 输出附加控制位  
0: PWM2P 的输出直接由 ENO2P 控制  
1: PWM2P 的输出由 ENO2P 和 PWMA\_BKR 共同控制  
AUX1N: PWM1N 输出附加控制位  
0: PWM1N 的输出直接由 ENO1N 控制  
1: PWM1N 的输出由 ENO1N 和 PWMA\_BKR 共同控制  
AUX1P: PWM1P 输出附加控制位  
0: PWM1P 的输出直接由 ENO1P 控制  
1: PWM1P 的输出由 ENO1P 和 PWMA\_BKR 共同控制

## 22.7.4 控制寄存器 1 (PWM<sub>x</sub>\_CR1)

| 符号       | 地址    | B7    | B6        | B5 | B4   | B3   | B2   | B1    | B0   |
|----------|-------|-------|-----------|----|------|------|------|-------|------|
| PWMA_CR1 | FEC0H | ARPEA | CMSA[1:0] |    | DIRA | OPMA | URSA | UDISA | CENA |
| PWMB_CR1 | FEE0H | ARPEB | CMSB[1:0] |    | DIRB | OPMB | URSB | UDISB | CENB |

ARPE<sub>n</sub>: 自动预装载允许位 (n=A,B)

0: PWM<sub>n</sub>\_ARR 寄存器没有缓冲, 它可以被直接写入

1: PWM<sub>n</sub>\_ARR 寄存器由预装载缓冲器缓冲

CMS<sub>n</sub>[1:0]: 选择对齐模式 (n= A,B)

| CMS <sub>n</sub> [1:0] | 对齐模式    | 说明                     |
|------------------------|---------|------------------------|
| 00                     | 边沿对齐模式  | 计数器依据方向位 (DIR) 向上或向下计数 |
| 01                     | 中央对齐模式1 | 计数器交替地向上和向下计数。         |

|    |         |  |
|----|---------|--|
|    |         | 配置为输出的通道的输出比较中断标志位，只在计数器向下计数时被置1。                      |
| 10 | 中央对齐模式2 | 计数器交替地向上和向下计数。<br>配置为输出的通道的输出比较中断标志位，只在计数器向上计数时被置1。    |
| 11 | 中央对齐模式3 | 计数器交替地向上和向下计数。<br>配置为输出的通道的输出比较中断标志位，在计数器向上和向下计数时均被置1。 |

注 1: 在计数器开启时 (CEN=1)，不允许从边沿对齐模式转换到中央对齐模式。

注 2: 在中央对齐模式下，编码器模式 (SMS=001, 010, 011) 必须被禁止。

DIRn: 计数器的计数方向 (n= A,B)

0: 计数器向上计数;

1: 计数器向下计数。

注: 当计数器配置为中央对齐模式或编码器模式时，该位为只读。

OPMn: 单脉冲模式 (n= A,B)

0: 在发生更新事件时，计数器不停止;

1: 在发生下一次更新事件时，清除 CEN 位，计数器停止。

URSn: 更新请求源 (n= A,B)

0: 如果 UDIS 允许产生更新事件，则下述任一事件产生一个更新中断:

- 寄存器被更新 (计数器上溢/下溢)
- 软件设置 UG 位
- 时钟/触发控制器产生的更新

1: 如果 UDIS 允许产生更新事件，则只有当下列事件发生时才产生更新中断，并 UIF 置 1:

- 寄存器被更新 (计数器上溢/下溢)

UDISn: 禁止更新 (n= A,B)

0: 一旦下列事件发生，产生更新 (UEV) 事件:

- 计数器溢出/下溢
- 产生软件更新事件
- 时钟/触发模式控制器产生的硬件复位 被缓存的寄存器被装入它们的预装载值。

1: 不产生更新事件，影子寄存器 (ARR、PSC、CCR<sub>x</sub>) 保持它们的值。如果设置了 UG 位或时钟/触发控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。

CENn: 允许计数器 (n= A,B)

0: 禁止计数器;

1: 使能计数器。

注: 在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。然而触发模式可以自动地通过硬件设置 CEN 位。

## 22.7.5 控制寄存器 2 (PWM<sub>x</sub>\_CR2)，及实时触发 ADC

| 符号       | 地址    | B7   | B6        | B5 | B4 | B3 | B2    | B1 | B0    |
|----------|-------|------|-----------|----|----|----|-------|----|-------|
| PWMA_CR2 | FEC1H | TI1S | MMSA[2:0] |    |    | -  | COMSA | -  | CCPCA |
| PWMB_CR2 | FEE1H | TI5S | MMSB[2:0] |    |    | -  | COMSB | -  | CCPCB |

TI1S: 第一组 PWM/PWMA 的 TI1 选择

0: PWM1P 输入管脚连到 TI1 (数字滤波器的输入);

1: PWM1P、PWM2P 和 PWM3P 管脚经异或后连到第一组 PWM 的 TI1。

TI5S: 第二组 PWM/PWMB 的 TI5 选择

0: PWM5 输入管脚连到 TI5 (数字滤波器的输入);

1: PWM5、PWM6 和 PWM7 管脚经异或后连到第二组 PWM/PWMB 的 TI5。

MMSA[2:0]: 主模式选择

| MMSA[2:0] | 主模式  | 说明   |
|-----------|------|--|
| 000       | 复位   | PWMA_EGR寄存器的UG位被用于作为触发输出 (TRGO)。如果触发输入 (时钟/触发控制器配置为复位模式) 产生复位, 则TRGO上的信号相对实际的复位会有一个延迟  |
| 001       | 使能   | 计数器使能信号被用于作为触发输出 (TRGO)。其用于启动ADC, 以便控制在一段时间内使能ADC。计数器使能信号是通过CEN控制位和门控模式下的触发输入信号的逻辑或产生。除非选择了主/从模式, 当计数器使能信号受控于触发输入时, TRGO上会有一个延迟。<br><b>注: 当需要使用PWM触发ADC转换时, 需要先设置ADC_CONTR寄存器中的ADC_POWER、ADC_CHS以及ADC_EPWMT, 当PWM产生TRGO内部信号时, 系统会自动设置ADC_START来启动AD转换。详细使用请参考范例程序“<a href="#">使用PWM的CEN启动PWMA定时器, 实时触发ADC</a>”</b> |
| 010       | 更新   | 更新事件被选为触发输出 (TRGO)   |
| 011       | 比较脉冲 | 一旦发生一次捕获或一次比较成功, 当CC1IF标志被置1时, 触发输出送出一个正脉冲 (TRGO)  |
| 100       | 比较   | OC1REF信号被用于作为触发输出 (TRGO)   |
| 101       | 比较   | OC2REF信号被用于作为触发输出 (TRGO)   |
| 110       | 比较   | OC3REF信号被用于作为触发输出 (TRGO)   |
| 111       | 比较   | OC4REF信号被用于作为触发输出 (TRGO)   |

MMSB[2:0]: 主模式选择

| MMSB[2:0] | 主模式  | 说明  |
|-----------|------|---|
| 000       | 复位   | PWMB_EGR寄存器的UG位被用于作为触发输出 (TRGO)。如果触发输入 (时钟/触发控制器配置为复位模式) 产生复位, 则TRGO上的信号相对实际的复位会有一个延迟   |
| 001       | 使能   | 计数器使能信号被用于作为触发输出 (TRGO)。其用于启动多个PWM, 以便控制在一段时间内使能从PWM。计数器使能信号是通过CEN控制位和门控模式下的触发输入信号的逻辑或产生。除非选择了主/从模式, 当计数器使能信号受控于触发输入时, TRGO上会有一个延迟。 |
| 010       | 更新   | 更新事件被选为触发输出 (TRGO)  |
| 011       | 比较脉冲 | 一旦发生一次捕获或一次比较成功, 当CC5IF标志被置1  |

|     |    |                          |
|-----|----|--------------------------|
|     |    | 时, 触发输出送出一个正脉冲 (TRGO)    |
| 100 | 比较 | OC5REF信号被用于作为触发输出 (TRGO) |
| 101 | 比较 | OC6REF信号被用于作为触发输出 (TRGO) |
| 110 | 比较 | OC7REF信号被用于作为触发输出 (TRGO) |
| 111 | 比较 | OC8REF信号被用于作为触发输出 (TRGO) |

**注: 只有第一组 PWM (PWMA) 的 TRGO 可用于触发启动 ADC**

**注: 只有第二组 PWM (PWMB) 的 TRGO 可用于第一组 PWM (PWMA) 的 ITR2**

COMSn: 捕获/比较控制位的更新控制选择 (n=A,B)

0: 当 CCPCn=1 时, 只有在 COMG 位置 1 的时候这些控制位才被更新

1: 当 CCPCn=1 时, 只有在 COMG 位置 1 或 TRGI 发生上升沿的时候这些控制位才被更新

CCPCn: 捕获/比较预装载控制位 (n= A,B)

0: CCIE, CCINE, CCiP, CCiNP 和 OCIM 位不是预装载的

1: CCIE, CCINE, CCiP, CCiNP 和 OCIM 位是预装载的; 设置该位后, 它们只在设置了 COMG 位后被更新。

注: 该位只对具有互补输出的通道起作用。

## 22.7.6 从模式控制寄存器(PWMx\_SMCR)

| 符号        | 地址    | B7   | B6       | B5 | B4 | B3 | B2        | B1 | B0 |
|-----------|-------|------|----------|----|----|----|-----------|----|----|
| PWMA_SMCR | FEC2H | MSMA | TSA[2:0] |    |    | -  | SMSA[2:0] |    |    |
| PWMB_SMCR | FEE2H | MSMB | TSB[2:0] |    |    | -  | SMSB[2:0] |    |    |

MSMn: 主/从模式 (n= A,B)

0: 无作用

1: 触发输入 (TRGI) 上的事件被延迟了, 以允许 PWMn 与它的从 PWM 间的完美同步 (通过 TRGO)

TSA[2:0]: 触发源选择

| TSA[2:0] | 触发源                 |
|----------|---------------------|
| 000      | -                   |
| 001      | -                   |
| 010      | 内部触发 ITR2           |
| 011      | -                   |
| 100      | TI1的边沿检测器 (TI1F_ED) |
| 101      | 滤波后的定时器输入1 (TI1FP1) |
| 110      | 滤波后的定时器输入2 (TI2FP2) |
| 111      | 外部触发输入 (ETRF)       |

TSB[2:0]: 触发源选择

| TSB[2:0] | 触发源                 |
|----------|---------------------|
| 000      | -                   |
| 001      | -                   |
| 010      | -                   |
| 011      | -                   |
| 100      | TI5的边沿检测器 (TI5F_ED) |

|     |                     |
|-----|---------------------|
| 101 | 滤波后的定时器输入1 (TI5FP5) |
| 110 | 滤波后的定时器输入2 (TI5FP6) |
| 111 | 外部触发输入 (ETRF)       |

注: 这些位只能在 SMS=000 时被改变, 以避免在改变时产生错误的边沿检测。

#### SMSA[2:0]: 时钟/触发/从模式选择

| SMSA[2:0] | 功能      | 说明   |
|-----------|---------|--|
| 000       | 内部时钟模式  | 如果CEN=1, 则预分频器直接由内部时钟驱动  |
| 001       | 编码器模式1  | 根据TI1FP1的电平, 计数器在TI2FP2的边沿向上/下计数   |
| 010       | 编码器模式2  | 根据TI2FP2的电平, 计数器在TI1FP1的边沿向上/下计数   |
| 011       | 编码器模式3  | 根据另一个输入的电平, 计数器在TI1FP1和TI2FP2的边沿向上/下计数   |
| 100       | 复位模式    | 在选中的触发输入 (TRGI) 的上升沿时重新初始化计数器, 并且产生一个更新 寄存器的信号   |
| 101       | 门控模式    | 当触发输入 (TRGI) 为高时, 计数器的时钟开启。一旦触发输入变为低, 则计数器停止 (但不复位)。计数器的启动和停止都是受控的   |
| 110       | 触发模式    | 计数器在触发输入TRGI的上升沿启动 (但不复位), 只有计数器的启动是受控 的   |
| 111       | 外部时钟模式1 | 选中的触发输入 (TRGI) 的上升沿驱动计数器。<br>注: 如果TI1F_ED被选为触发输入 (TS=100) 时, 不要使用门控模式。这是因为TI1F_ED在每次TI1F变化时只是输出一个脉冲, 然而门控模式是要检查触发输入的电平 |

#### SMSB[2:0]: 时钟/触发/从模式选择

| SMSB[2:0] | 功能      | 说明   |
|-----------|---------|--|
| 000       | 内部时钟模式  | 如果CEN=1, 则预分频器直接由内部时钟驱动  |
| 001       | 编码器模式1  | 根据TI5FP5的电平, 计数器在TI6FP6的边沿向上/下计数                                   |
| 010       | 编码器模式2  | 根据TI6FP6的电平, 计数器在TI5FP5的边沿向上/下计数                                   |
| 011       | 编码器模式3  | 根据另一个输入的电平, 计数器在TI5FP5和TI6FP6的边沿向上/下计数                             |
| 100       | 复位模式    | 在选中的触发输入 (TRGI) 的上升沿时重新初始化计数器, 并且产生一个更新 寄存器的信号                     |
| 101       | 门控模式    | 当触发输入 (TRGI) 为高时, 计数器的时钟开启。一旦触发输入变为低, 则计数器停止 (但不复位)。计数器的启动和停止都是受控的 |
| 110       | 触发模式    | 计数器在触发输入TRGI的上升沿启动 (但不复位), 只有计数器的启动是受控 的                           |
| 111       | 外部时钟模式1 | 选中的触发输入 (TRGI) 的上升沿驱动计数器。<br>注: 如果TI5F_ED被选为触发输入 (TS=100) 时, 不要使用  |

门控模式。这是因为TI5F\_ED在每次TI5F变化时只是输出一个脉冲，然而门控模式是要检查触发输入的电平

## 22.7.7 外部触发寄存器(PWMx\_ETR)

| 符号       | 地址    | B7   | B6   | B5         | B4 | B3        | B2 | B1 | B0 |
|----------|-------|------|------|------------|----|-----------|----|----|----|
| PWMA_ETR | FEC3H | ETP1 | ECEA | ETPSA[1:0] |    | ETFA[3:0] |    |    |    |
| PWMB_ETR | FEE3H | ETP2 | ECEB | ETPSB[1:0] |    | ETFB[3:0] |    |    |    |

ETPn: 外部触发 ETR 的极性 (n=A,B)

0: 高电平或上升沿有效

1: 低电平或下降沿有效

ECEn: 外部时钟使能 (n=A,B)

0: 禁止外部时钟模式 2

1: 使能外部时钟模式 2，计数器的时钟为 ETRF 的有效沿。

注 1: ECE 置 1 的效果与选择把 TRGI 连接到 ETRF 的外部时钟模式 1 相同 (PWMn\_SMCR 寄存器中，SMS=111，TS=111)。

注 2: 外部时钟模式 2 可与下列模式同时使用：触发标准模式；触发复位模式；触发门控模式。但是，此时 TRGI 决不能与 ETRF 相连 (PWMn\_SMCR 寄存器中，TS 不能为 111)。

注 3: 外部时钟模式 1 与外部时钟模式 2 同时使能，外部时钟输入为 ETRF。

ETPSn: 外部触发预分频器外部触发信号 EPRP 的频率最大不能超过 fMASTER/4。可用预分频器来降低 ETRP 的频率，当 EPRP 的频率很高时，它非常有用：(n=A,B)

00: 预分频器关闭

01: EPRP 的频率/2

02: EPRP 的频率/4

03: EPRP 的频率/8

ETFn[3:0]: 外部触发滤波器选择，该位域定义了 ETRP 的采样频率及数字滤波器长度。(n=A,B)

| ETFn[3:0] | 时钟数 | ETF[3:0] | 时钟数 |
|-----------|-----|----------|-----|
| 0000      | 1   | 1000     | 48  |
| 0001      | 2   | 1001     | 64  |
| 0010      | 4   | 1010     | 80  |
| 0011      | 8   | 1011     | 96  |
| 0100      | 12  | 1100     | 128 |
| 0101      | 16  | 1101     | 160 |
| 0110      | 24  | 1110     | 192 |
| 0111      | 32  | 1111     | 256 |

## 22.7.8 中断使能寄存器(PWMx\_IER)

| 符号       | 地址    | B7   | B6   | B5     | B4    | B3    | B2    | B1    | B0   |
|----------|-------|------|------|--------|-------|-------|-------|-------|------|
| PWMA_IER | FEC4H | BIEA | TIEA | COMIEA | CC4IE | CC3IE | CC2IE | CC1IE | UIEA |
| PWMB_IER | FEE4H | BIEB | TIEB | COMIEB | CC8IE | CC7IE | CC6IE | CC5IE | UIEB |

BIEn: 允许刹车中断 (n=A,B)

0: 禁止刹车中断;



1: 允许刹车中断。

TIE: 触发中断使能 (n= A,B)

0: 禁止触发中断;

1: 使能触发中断。

COMIE: 允许 COM 中断 (n= A,B)

0: 禁止 COM 中断;

1: 允许 COM 中断。

CCnIE: 允许捕获/比较 n 中断 (n=1,2,3,4,5,6,7,8)

0: 禁止捕获/比较 n 中断;

1: 允许捕获/比较 n 中断。

UIEn: 允许更新中断 (n= A,B)

0: 禁止更新中断;

1: 允许更新中断。

## 22.7.9 状态寄存器 1(PWMx\_SR1)

| 符号       | 地址    | B7   | B6   | B5     | B4    | B3    | B2    | B1    | B0   |
|----------|-------|------|------|--------|-------|-------|-------|-------|------|
| PWMA_SR1 | FEC5H | BIFA | TIFA | COMIFA | CC4IF | CC3IF | CC2IF | CC1IF | UIFA |
| PWMB_SR1 | FEE5H | BIFB | TIFB | COMIFB | CC8IF | CC7IF | CC6IF | CC5IF | UIFB |

BIFn: 刹车中断标记。一旦刹车输入有效, 由硬件对该位置 1。如果刹车输入无效, 则该位可由软件清 0。(n= A,B)

0: 无刹车事件产生

1: 刹车输入上检测到有效电平

TIFn: 触发器中断标记。当发生触发事件时由硬件对该位置 1。由软件清 0。(n= A,B)

0: 无触发器事件产生

1: 触发中断等待响应

COMIFn: COM 中断标记。一旦产生 COM 事件该位由硬件置 1。由软件清 0。(n= A,B)

0: 无 COM 事件产生

1: COM 中断等待响应

CC8IF: 捕获/比较8中断标记, 参考CC1IF描述

CC7IF: 捕获/比较7中断标记, 参考CC1IF描述

CC6IF: 捕获/比较6中断标记, 参考CC1IF描述

CC5IF: 捕获/比较5中断标记, 参考CC1IF描述

CC4IF: 捕获/比较4中断标记, 参考CC1IF描述

CC3IF: 捕获/比较3中断标记, 参考CC1IF描述

CC2IF: 捕获/比较2中断标记, 参考CC1IF描述

CC1IF: 捕获/比较1中断标记。

**如果通道CC1配置为输出模式:**

当计数器值与比较值匹配时该位由硬件置1, 但在中心对称模式下除外。它由软件清0。

0: 无匹配发生;

1: PWMA\_CNT 的值与 PWMA\_CCR1 的值匹配。

注: 在中心对称模式下, 当计数器值为 0 时, 向上计数, 当计数器值为 ARR 时, 向下计数 (它从 0 向上计数到 ARR-1, 再由 ARR 向下计数到 1)。因此, 对所有的 SMS 位值, 这两个值都不置标记。但是, 如果 CCR1>ARR, 则当 CNT 达到 ARR 值时, CC1IF 置 1。

**如果通道CC1配置为输入模式:**

当捕获事件发生时该位由硬件置1，它由软件清0或通过读PWMA\_CCR1L清0。

0: 无输入捕获产生

1: 计数器值已被捕获至 PWMA\_CCR1

UIFn: 更新中断标记 当产生更新事件时该位由硬件置 1。它由软件清 0。(n= A,B)

0: 无更新事件产生

1: 更新事件等待响应。当寄存器被更新时该位由硬件置 1

– 若 PWMn\_CR1 寄存器的 UDIS=0，当计数器上溢或下溢时

– 若 PWMn\_CR1 寄存器的 UDIS=0、URS=0，当设置 PWMn\_EGR 寄存器的 UG 位软件对计数器 CNT 重新初始化时

– 若 PWMn\_CR1 寄存器的 UDIS=0、URS=0，当计数器 CNT 被触发事件重新初始化时

## 22.7.10 状态寄存器 2(PWMx\_SR2)

| 符号       | 地址    | B7 | B6 | B5 | B4    | B3    | B2    | B1    | B0 |
|----------|-------|----|----|----|-------|-------|-------|-------|----|
| PWMA_SR2 | FEC6H | -  | -  | -  | CC4OF | CC3OF | CC2OF | CC1OF | -  |
| PWMB_SR2 | FEE6H | -  | -  | -  | CC8OF | CC7OF | CC6OF | CC5OF | -  |

CC8OF: 捕获/比较8重复捕获标记。参见CC1OF描述。

CC7OF: 捕获/比较7重复捕获标记。参见CC1OF描述。

CC6OF: 捕获/比较6重复捕获标记。参见CC1OF描述。

CC5OF: 捕获/比较5重复捕获标记。参见CC1OF描述。

CC4OF: 捕获/比较4重复捕获标记。参见CC1OF描述。

CC3OF: 捕获/比较3重复捕获标记。参见CC1OF描述。

CC2OF: 捕获/比较2重复捕获标记。参见CC1OF描述。

CC1OF: 捕获/比较1重复捕获标记。仅当相应的通道被配置为输入捕获时，该标记可由硬件置1。写0可清除该位。

0: 无重复捕获产生；

1: 计数器的值被捕获到 PWMA\_CCR1 寄存器时，CC1IF 的状态已经为 1。

## 22.7.11 事件产生寄存器 (PWMx\_EGR)

| 符号       | 地址    | B7  | B6  | B5    | B4   | B3   | B2   | B1   | B0  |
|----------|-------|-----|-----|-------|------|------|------|------|-----|
| PWMA_EGR | FEC7H | BGA | TGA | COMGA | CC4G | CC3G | CC2G | CC1G | UGA |
| PWMB_EGR | FEE7H | BGB | TGB | COMGB | CC8G | CC7G | CC6G | CC5G | UGB |

BGn: 产生刹车事件。该位由软件置 1，用于产生一个刹车事件，由硬件自动清 0 (n= A,B)

0: 无动作

1: 产生一个刹车事件。此时 MOE=0、BIF=1，若开启对应的中断 (BIE=1)，则产生相应的中断

TGn: 产生触发事件。该位由软件置 1，用于产生一个触发事件，由硬件自动清 0 (n= A,B)

0: 无动作

1: TIF=1，若开启对应的中断 (TIE=1)，则产生相应的中断

COMGn: 捕获/比较事件，产生控制更新。该位由软件置 1，由硬件自动清 0 (n= A,B)

0: 无动作

1: CCPC=1，允许更新 CCIE、CCINE、CCiP，CCiNP，OCIM 位。

注：该位只对拥有互补输出的通道有效

CC8G: 产生捕获/比较 8 事件。参考 CC1G 描述

CC7G: 产生捕获/比较 7 事件。参考 CC1G 描述



CC6G: 产生捕获/比较 6 事件。参考 CC1G 描述

CC5G: 产生捕获/比较 5 事件。参考 CC1G 描述

CC4G: 产生捕获/比较 4 事件。参考 CC1G 描述

CC3G: 产生捕获/比较 3 事件。参考 CC1G 描述

CC2G: 产生捕获/比较 2 事件。参考 CC1G 描述

CC1G: 产生捕获/比较 1 事件。产生捕获/比较 1 事件。该位由软件置 1，用于产生一个捕获/比较事件，由硬件自动清 0。

0: 无动作;

1: 在通道 CC1 上产生一个捕获/比较事件。

若通道 CC1 配置为输出: 设置 CC1IF=1, 若开启对应的中断, 则产生相应的中断。

若通道 CC1 配置为输入: 当前的计数器值被捕获至 PWMA\_CCR1 寄存器, 设置 CC1IF=1, 若开启对应的中断, 则产生相应的中断。若 CC1IF 已经为 1, 则设置 CC1OF=1。

UGn: 产生更新事件 该位由软件置 1, 由硬件自动清 0。(n= A,B)

0: 无动作;

1: 重新初始化计数器, 并产生一个更新事件。

注意预分频器的计数器也被清 0 (但是预分频系数不变)。若在中心对称模式下或 DIR=0 (向上计数) 则计数器被清 0; 若 DIR=1 (向下计数) 则计数器取 PWMn\_ARR 的值。

## 22.7.12 捕获/比较模式寄存器 1 (PWMx\_CCMR1)

通道可用于捕获输入模式或比较输出模式, 通道的方向由相应的 CCnS 位定义。该寄存器其它位的作用在输入和输出模式下不同。OCxx 描述了通道在输出模式下的功能, ICxx 描述了通道在输入模式下的功能。因此必须注意, 同一个位在输出模式和输入模式下的功能是不同的。

通道配置为比较输出模式

| 符号         | 地址    | B7    | B6        | B5 | B4 | B3    | B2    | B1        | B0 |
|------------|-------|-------|-----------|----|----|-------|-------|-----------|----|
| PWMA_CCMR1 | FEC8H | OC1CE | OC1M[2:0] |    |    | OC1PE | OC1FE | CC1S[1:0] |    |
| PWMB_CCMR1 | FEB8H | OC5CE | OC5M[2:0] |    |    | OC5PE | OC5FE | CC5S[1:0] |    |

OCnCE: 输出比较 n 清零使能。该位用于使能使用 PWMETI 引脚上的外部事件来清通道 n 的输出信号 (OCnREF) (n=1,5)

0: OCnREF 不受 ETRF 输入的影响;

1: 一旦检测到 ETRF 输入高电平, OCnREF=0。

OCnM[2:0]: 输出比较 n 模式。该 3 位定义了输出参考信号 OCnREF 的动作, 而 OCnREF 决定了 OCn 的值。OCnREF 是高电平有效, 而 OCn 的有效电平取决于 CCnP 位。(n=1,5)

| OCnM[2:0] | 模式               | 说明                                |
|-----------|------------------|-----------------------------------|
| 000       | 冻结               | PWMn_CCR1与PWMn_CNT间的比较对OCnREF不起作用 |
| 001       | 匹配时设置通道n的输出为有效电平 | 当PWMn_CCR1=PWMn_CNT时, OCnREF输出高   |
| 010       | 匹配时设置通道n的输出为无效电平 | 当PWMn_CCR1=PWMn_CNT时, OCnREF输出低   |
| 011       | 翻转               | 当PWMn_CCR1=PWMn_CNT时, 翻转OCnREF    |
| 100       | 强制为无效电平          | 强制OCnREF为低                        |
| 101       | 强制为有效电平          | 强制OCnREF为高                        |

|     |        |  |
|-----|--------|--|
| 110 | PWM模式1 | 在向上计数时, 当PWMn_CNT<PWMn_CCR1时<br>OCnREF输出高, 否则OCnREF输出低<br>在向下计数时, 当PWMn_CNT>PWMn_CCR1时<br>OCnREF输出低, 否则OCnREF输出高 |
| 111 | PWM模式2 | 在向上计数时, 当PWMn_CNT<PWMn_CCR1时<br>OCnREF输出低, 否则OCnREF输出高<br>在向下计数时, 当PWMn_CNT>PWMn_CCR1时<br>OCnREF输出高, 否则OCnREF输出低 |

注 1: 一旦 LOCK 级别设为 3 (PWMn\_BKR 寄存器中的 LOCK 位) 并且 CCnS=00 (该通道配置成输出) 则该位不能被修改。

注 2: 在 PWM 模式 1 或 PWM 模式 2 中, 只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时, OCnREF 电平才改变。

注 3: 在有互补输出的通道上, 这些位是预装载的。如果 PWMn\_CR2 寄存器的 CCPC=1, OCM 位只有在 COM 事件发生时, 才从预装载位取新值。

OCnPE: 输出比较 n 预装载使能 (n=1,5)

0: 禁止 PWMn\_CCR1 寄存器的预装载功能, 可随时写入 PWMn\_CCR1 寄存器, 并且新写入的数值立即起作用。

1: 开启 PWMn\_CCR1 寄存器的预装载功能, 读写操作仅对预装载寄存器操作, PWMn\_CCR1 的预装载值在更新事件到来时被加载至当前寄存器中。

注 1: 一旦 LOCK 级别设为 3 (PWMn\_BKR 寄存器中的 LOCK 位) 并且 CCnS=00 (该通道配置成输出) 则该位不能被修改。

注 2: 为了操作正确, 在 PWM 模式下必须使能预装载功能。但在单脉冲模式下 (PWMn\_CR1 寄存器的 OPM=1), 它不是必须的。

OCnFE: 输出比较 n 快速使能。该位用于加快 CC 输出对触发输入事件的响应。(n=1,5)

0: 根据计数器与 CCRn 的值, CCn 正常操作, 即使触发器是打开的。当触发器的输入有一个有效沿时, 激活 CCn 输出的最小延时为 5 个时钟周期。

1: 输入到触发器的有效沿的作用就象发生了一次比较匹配。因此, OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。OCFE 只在通道被配置成 PWMA 或 PWMB 模式时起作用。

CC1S[1:0]: 捕获/比较 1 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC1S[1:0] | 方向 | 输入脚   |
|-----------|----|---|
| 00        | 输出 |   |
| 01        | 输入 | IC1映射在TI1FP1上                                       |
| 10        | 输入 | IC1映射在TI2FP1上                                       |
| 11        | 输入 | IC1映射在TRC上。此模式仅工作在内部触发器输入被选中时 (由PWMA_SMCR寄存器的TS位选择) |

CC5S[1:0]: 捕获/比较 5 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC5S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC5映射在TI5FP5上 |
| 10        | 输入 | IC5映射在TI6FP5上 |

|    |    |  |
|----|----|--|
| 11 | 输入 | IC5映射在TRC上。此模式仅工作在内部触发器输入被选中时（由PWM5_SMCR寄存器的TS位选择） |
|----|----|--|

注：CC1S 仅在通道关闭时（PWMA\_CCER1 寄存器的 CC1E=0）才是可写的。

注：CC5S 仅在通道关闭时（PWM5\_CCER1 寄存器的 CC5E=0）才是可写的。

STC MCU

通道配置为捕获输入模式

| 符号         | 地址    | B7        | B6 | B5 | B4 | B3          | B2 | B1        | B0 |
|------------|-------|-----------|----|----|----|-------------|----|-----------|----|
| PWMA_CCMR1 | FEC8H | IC1F[3:0] |    |    |    | IC1PSC[1:0] |    | CC1S[1:0] |    |
| PWMB_CCMR1 | FEE8H | IC5F[3:0] |    |    |    | IC5PSC[1:0] |    | CC5S[1:0] |    |

ICnF[3:0]: 输入捕获 n 滤波器选择, 该位域定义了 TI<sub>n</sub> 的采样频率及数字滤波器长度。(n=1,5)

| ICnF[3:0] | 时钟数 | ICnF[3:0] | 时钟数 |
|-----------|-----|-----------|-----|
| 0000      | 1   | 1000      | 48  |
| 0001      | 2   | 1001      | 64  |
| 0010      | 4   | 1010      | 80  |
| 0011      | 8   | 1011      | 96  |
| 0100      | 12  | 1100      | 128 |
| 0101      | 16  | 1101      | 160 |
| 0110      | 24  | 1110      | 192 |
| 0111      | 32  | 1111      | 256 |

注: 即使对于带互补输出的通道, 该位域也是非预装载的, 并且不会考虑 CCPC (PWM<sub>n</sub>\_CR2 寄存器) 的值

ICnPSC[1:0]: 输入/捕获 n 预分频器。这两位定义了 CC<sub>n</sub> 输入 (IC<sub>1</sub>) 的预分频系数。(n=1,5)

00: 无预分频器, 捕获输入口上检测到的每一个边沿都触发一次捕获

01: 每 2 个事件触发一次捕获

10: 每 4 个事件触发一次捕获

11: 每 8 个事件触发一次捕获

CC1S[1:0]: 捕获/比较 1 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC1S[1:0] | 方向 | 输入脚   |
|-----------|----|---|
| 00        | 输出 |   |
| 01        | 输入 | IC1映射在TI1FP1上                                       |
| 10        | 输入 | IC1映射在TI2FP1上                                       |
| 11        | 输入 | IC1映射在TRC上。此模式仅工作在内部触发器输入被选中时 (由PWMA_SMCR寄存器的TS位选择) |

CC5S[1:0]: 捕获/比较 5 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC5S[1:0] | 方向 | 输入脚   |
|-----------|----|---|
| 00        | 输出 |   |
| 01        | 输入 | IC5映射在TI5FP5上                                       |
| 10        | 输入 | IC5映射在TI6FP5上                                       |
| 11        | 输入 | IC5映射在TRC上。此模式仅工作在内部触发器输入被选中时 (由PWM5_SMCR寄存器的TS位选择) |

注: CC1S 仅在通道关闭时 (PWMA\_CCER1 寄存器的 CC1E=0) 才是可写的。

注: CC5S 仅在通道关闭时 (PWM5\_CCER1 寄存器的 CC5E=0) 才是可写的。

## 22.7.13 捕获/比较模式寄存器 2 (PWM<sub>x</sub>\_CCMR2)

通道配置为比较输出模式

| 符号         | 地址    | B7    | B6        | B5 | B4 | B3    | B2    | B1        | B0 |
|------------|-------|-------|-----------|----|----|-------|-------|-----------|----|
| PWMA_CCMR2 | FEC9H | OC2CE | OC2M[2:0] |    |    | OC2PE | OC2FE | CC2S[1:0] |    |
| PWMB_CCMR2 | FEE9H | OC6CE | OC6M[2:0] |    |    | OC6PE | OC6FE | CC6S[1:0] |    |

OCnCE: 输出比较 n 清零使能。该位用于使能使用 PWMETI 引脚上的外部事件来清通道 n 的输出信号 (OCnREF) (n=2,6)

0: OCnREF 不受 ETRF 输入的影响;

1: 一旦检测到 ETRF 输入高电平, OCnREF=0。

OCnM[2:0]: 输出比较 2 模式, 参考 OC1M。(n=2,6)

OCnPE: 输出比较 2 预装载使能, 参考 OP1PE。(n=2,6)

CC2S[1:0]: 捕获/比较 2 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC2S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC2映射在TI2FP2上 |
| 10        | 输入 | IC2映射在TI1FP2上 |
| 11        | 输入 | IC2映射在TRC上。   |

CC6S[1:0]: 捕获/比较 6 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC6S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC6映射在TI6FP6上 |
| 10        | 输入 | IC6映射在TI5FP6上 |
| 11        | 输入 | IC6映射在TRC上。   |

通道配置为捕获输入模式

| 符号         | 地址    | B7        | B6 | B5 | B4 | B3          | B2 | B1        | B0 |
|------------|-------|-----------|----|----|----|-------------|----|-----------|----|
| PWMA_CCMR2 | FEC9H | IC2F[3:0] |    |    |    | IC2PSC[1:0] |    | CC2S[1:0] |    |
| PWMB_CCMR2 | FEE9H | IC6F[3:0] |    |    |    | IC6PSC[1:0] |    | CC6S[1:0] |    |

ICnF[3:0]: 输入捕获 n 滤波器选择, 参考 IC1F。(n=2,6)

ICnPSC[1:0]: 输入/捕获 n 预分频器, 参考 IC1PSC。(n=2,6)

CC2S[1:0]: 捕获/比较 2 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC2S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC2映射在TI2FP2上 |
| 10        | 输入 | IC2映射在TI1FP2上 |
| 11        | 输入 | IC2映射在TRC上。   |

CC6S[1:0]: 捕获/比较 6 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC6S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC6映射在TI6FP6上 |

|    |    |               |
|----|----|---------------|
| 10 | 输入 | IC6映射在TI5FP6上 |
| 11 | 输入 | IC6映射在TRC上。   |

## 22.7.14 捕获/比较模式寄存器 3 (PWMx\_CCMR3)

通道配置为比较输出模式

| 符号         | 地址    | B7    | B6        | B5 | B4 | B3    | B2    | B1        | B0 |
|------------|-------|-------|-----------|----|----|-------|-------|-----------|----|
| PWMA_CCMR3 | FECAH | OC3CE | OC3M[2:0] |    |    | OC3PE | OC3FE | CC3S[1:0] |    |
| PWMB_CCMR3 | FEEAH | OC7CE | OC7M[2:0] |    |    | OC7PE | OC7FE | CC7S[1:0] |    |

OCnCE: 输出比较 n 清零使能。该位用于使能使用 PWMETI 引脚上的外部事件来清通道 n 的输出信号 (OCnREF) (n=3,7)

0: OCnREF 不受 ETRF 输入的影响;

1: 一旦检测到 ETRF 输入高电平, OCnREF=0。

OCnM[2:0]: 输出比较 3 模式, 参考 OC1M。 (n=3,7)

OCnPE: 输出比较 3 预装载使能, 参考 OP1PE。 (n=3,7)

CC3S[1:0]: 捕获/比较 3 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC3S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC3映射在TI3FP3上 |
| 10        | 输入 | IC3映射在TI4FP3上 |
| 11        | 输入 | IC3映射在TRC上。   |

CC7S[1:0]: 捕获/比较 7 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC7S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC7映射在TI7FP7上 |
| 10        | 输入 | IC7映射在TI8FP7上 |
| 11        | 输入 | IC7映射在TRC上。   |

通道配置为捕获输入模式

| 符号         | 地址    | B7        | B6 | B5 | B4 | B3          | B2 | B1        | B0 |
|------------|-------|-----------|----|----|----|-------------|----|-----------|----|
| PWMA_CCMR3 | FECAH | IC3F[3:0] |    |    |    | IC3PSC[1:0] |    | CC3S[1:0] |    |
| PWMB_CCMR3 | FEEAH | IC7F[3:0] |    |    |    | IC7PSC[1:0] |    | CC7S[1:0] |    |

ICnF[3:0]: 输入捕获 n 滤波器选择, 参考 IC1F。 (n=3,7)

ICnPSC[1:0]: 输入/捕获 n 预分频器, 参考 IC1PSC。 (n=3,7)

CC3S[1:0]: 捕获/比较 3 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC3S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC3映射在TI3FP3上 |
| 10        | 输入 | IC3映射在TI4FP3上 |
| 11        | 输入 | IC3映射在TRC上。   |

CC7S[1:0]: 捕获/比较 7 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC7S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC7映射在TI7FP7上 |
| 10        | 输入 | IC7映射在TI8FP7上 |
| 11        | 输入 | IC7映射在TRC上。   |

## 22.7.15 捕获/比较模式寄存器 4 (PWMx\_CCMR4)

通道配置为比较输出模式

| 符号         | 地址    | B7    | B6        | B5 | B4 | B3    | B2    | B1        | B0 |
|------------|-------|-------|-----------|----|----|-------|-------|-----------|----|
| PWMA_CCMR4 | FECBH | OC4CE | OC4M[2:0] |    |    | OC4PE | OC4FE | CC4S[1:0] |    |
| PWMB_CCMR4 | FEEBH | OC8CE | OC8M[2:0] |    |    | OC8PE | OC8FE | CC8S[1:0] |    |

OCnCE: 输出比较 n 清零使能。该位用于使能使用 PWMETI 引脚上的外部事件来清通道 n 的输出信号 (OCnREF) (n=4,8)

0: OCnREF 不受 ETRF 输入的影响;

1: 一旦检测到 ETRF 输入高电平, OCnREF=0。

OCnM[2:0]: 输出比较 n 模式, 参考 OC1M。 (n=4,8)

OCnPE: 输出比较 n 预装载使能, 参考 OP1PE。 (n=4,8)

CC4S[1:0]: 捕获/比较 4 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC4S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC4映射在TI4FP4上 |
| 10        | 输入 | IC4映射在TI3FP4上 |
| 11        | 输入 | IC4映射在TRC上。   |

CC8S[1:0]: 捕获/比较 8 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC8S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC8映射在TI8FP8上 |
| 10        | 输入 | IC8映射在TI7FP8上 |
| 11        | 输入 | IC8映射在TRC上。   |

通道配置为捕获输入模式

| 符号         | 地址    | B7        | B6 | B5 | B4 | B3          | B2 | B1        | B0 |
|------------|-------|-----------|----|----|----|-------------|----|-----------|----|
| PWMA_CCMR4 | FECBH | IC4F[3:0] |    |    |    | IC4PSC[1:0] |    | CC4S[1:0] |    |
| PWMB_CCMR4 | FEEBH | IC8F[3:0] |    |    |    | IC8PSC[1:0] |    | CC8S[1:0] |    |

ICnF[3:0]: 输入捕获 n 滤波器选择, 参考 IC1F。 (n=4,8)

ICnPSC[1:0]: 输入/捕获 n 预分频器, 参考 IC1PSC。 (n=4,8)

CC4S[1:0]: 捕获/比较 4 选择。这两位定义通道的方向 (输入/输出), 及输入脚的选择

| CC4S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC4映射在TI4FP4上 |



|    |    |               |
|----|----|---------------|
| 10 | 输入 | IC4映射在TI3FP4上 |
| 11 | 输入 | IC4映射在TRC上。   |

CC8S[1:0]: 捕获/比较 8 选择。这两位定义通道的方向（输入/输出），及输入脚的选择

| CC8S[1:0] | 方向 | 输入脚           |
|-----------|----|---------------|
| 00        | 输出 |               |
| 01        | 输入 | IC8映射在TI8FP8上 |
| 10        | 输入 | IC8映射在TI7FP8上 |
| 11        | 输入 | IC8映射在TRC上。   |

## 22.7.16 捕获/比较使能寄存器 1 (PWMx\_CCER1)

| 符号         | 地址    | B7    | B6    | B5   | B4   | B3    | B2    | B1   | B0   |
|------------|-------|-------|-------|------|------|-------|-------|------|------|
| PWMA_CCER1 | FECCH | CC2NP | CC2NE | CC2P | CC2E | CC1NP | CC1NE | CC1P | CC1E |
| PWMB_CCER1 | FEECH | -     | -     | CC6P | CC6E | -     | -     | CC5P | CC5E |

CC6P: OC6 输入捕获/比较输出极性。参考 CC1P

CC6E: OC6 输入捕获/比较输出使能。参考 CC1E

CC5P: OC5 输入捕获/比较输出极性。参考 CC1P

CC5E: OC5 输入捕获/比较输出使能。参考 CC1E

CC2NP: OC2N 比较输出极性。参考 CC1NP

CC2NE: OC2N 比较输出使能。参考 CC1NE

CC2P: OC2 输入捕获/比较输出极性。参考 CC1P

CC2E: OC2 输入捕获/比较输出使能。参考 CC1E

CC1NP: OC1N 比较输出极性

0: 高电平有效;

1: 低电平有效。

注 1: 一旦 LOCK 级别 (PWMA\_BKR 寄存器中的 LOCK 位) 设为 3 或 2 且 CC1S=00 (通道配置为输出), 则该位不能被修改。

注 2: 对于有互补输出的通道, 该位是预装载的。如果 CCPC=1 (PWMA\_CR2 寄存器), 只有在 COM 事件发生时, CC1NP 位才从预装载位中取新值。

CC1NE: OC1N 比较输出使能

0: 关闭比较输出。

1: 开启比较输出, 其输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值。

注: 对于有互补输出的通道, 该位是预装载的。如果 CCPC=1 (PWMA\_CR2 寄存器), 只有在 COM 事件发生时, CC1NE 位才从预装载位中取新值。

CC1P: OC1 输入捕获/比较输出极性

CC1 通道配置为输出:

0: 高电平有效

1: 低电平有效

CC1 通道配置为输入或者捕获:

0: 捕获发生在 TI1F 或 TI2F 的上升沿;

1: 捕获发生在 TI1F 或 TI2F 的下降沿。

注 1: 一旦 LOCK 级别 (PWMA\_BKR 寄存器中的 LOCK 位) 设为 3 或 2, 则该位不能被修改。

注 2: 对于有互补输出的通道, 该位是预装载的。如果 CCPC=1 (PWMA\_CR2 寄存器), 只有在 COM 事件发生时, CC1P 位才从预装载位中取新值。



CC1E: OC1 输入捕获/比较输出使能

0: 关闭输入捕获/比较输出;

1: 开启输入捕获/比较输出。

注: 对于有互补输出的通道, 该位是预装载的。如果 CCPC=1 (PWMA\_CR2 寄存器), 只有在 COM 事件发生时, CC1E 位才从预装载位中取新值。

STC MCU

带刹车功能的互补输出通道 OC<sub>i</sub> 和 OC<sub>iN</sub> 的控制位

| 控制位 |      |      |      |       | 输出状态  |   |
|-----|------|------|------|-------|---|---|
| MOE | OSSI | OSSR | CCiE | CCiNE | OC <sub>i</sub> 输出状态  | OC <sub>iN</sub> 输出状态                           |
| 1   | X    | 0    | 0    | 0     | 输出禁止  | 输出禁止  |
|     |      | 0    | 0    | 1     | 输出禁止  | 带极性的 OC <sub>i</sub> REF                        |
|     |      | 0    | 1    | 0     | 带极性的 OC <sub>i</sub> REF  | 输出禁止  |
|     |      | 0    | 1    | 1     | 带极性和死区的 OC <sub>i</sub> REF   | 带极性和死区的反向 OC <sub>i</sub> REF                   |
|     |      | 1    | 0    | 0     | 输出禁止  | 输出禁止  |
|     |      | 1    | 0    | 1     | 关闭状态<br>(输出使能且为无效电平)<br>OC <sub>i</sub> =CCiP   | 带极性的 OC <sub>i</sub> REF                        |
|     |      | 1    | 1    | 0     | 带极性的 OC <sub>i</sub> REF  | 关闭状态<br>(输出使能且为无效电平)<br>OC <sub>iN</sub> =CCiNP |
|     |      | 1    | 1    | 1     | 带极性和死区的 OC <sub>i</sub> REF   | 带极性和死区的反向 OC <sub>i</sub> REF                   |
| 0   | 0    | X    | X    | X     | 输出禁止  |   |
|     | 1    |      |      |       | 关闭状态 (输出使能且为无效电平) 异步地: OC <sub>i</sub> =CCiP, OC <sub>iN</sub> =CCiNP;<br>然后, 若时钟存在: 经过一个死区时间后 OC <sub>i</sub> =OISi, OC <sub>iN</sub> =OISiN, 假设<br>OISi 与 OISiN 并不都对应 OC <sub>i</sub> 和 OC <sub>iN</sub> 的有效电平。 |   |

注: 管脚连接到互补的 OC<sub>i</sub> 和 OC<sub>iN</sub> 通道的外部 I/O 管脚的状态, 取决于 OC<sub>i</sub> 和 OC<sub>iN</sub> 通道状态和 GPIO 寄存器。

## 22.7.17 捕获/比较使能寄存器 2 (PWM<sub>x</sub>\_CCER2)

| 符号         | 地址     | B7    | B6    | B5   | B4   | B3    | B2    | B1   | B0   |
|------------|--------|-------|-------|------|------|-------|-------|------|------|
| PWMA_CCER2 | FECDDH | CC4NP | CC4NE | CC4P | CC4E | CC3NP | CC3NE | CC3P | CC3E |
| PWMB_CCER2 | FEEDH  | -     | -     | CC8P | CC8E | -     | -     | CC7P | CC7E |

CC8P: OC8 输入捕获/比较输出极性。参考 CC1P

CC8E: OC8 输入捕获/比较输出使能。参考 CC1E

CC7P: OC7 输入捕获/比较输出极性。参考 CC1P

CC7E: OC7 输入捕获/比较输出使能。参考 CC1E

CC4NP: OC4N 比较输出极性。参考 CC1NP

CC4NE: OC4N 比较输出使能。参考 CC1NE

CC4P: OC4 输入捕获/比较输出极性。参考 CC1P

CC4E: OC4 输入捕获/比较输出使能。参考 CC1E

CC3NP: OC3N 比较输出极性。参考 CC1NP

CC3NE: OC3N 比较输出使能。参考 CC1NE

CC3P: OC3 输入捕获/比较输出极性。参考 CC1P

CC3E: OC3 输入捕获/比较输出使能。参考 CC1E

## 22.7.18 计数器高 8 位 (PWM<sub>x</sub>\_CNTRH)

| 符号         | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|------------|----|----|----|----|----|----|----|
| PWMA_CNTRH | FECEH | CNT1[15:8] |    |    |    |    |    |    |    |
| PWMB_CNTRH | FEFEH | CNT2[15:8] |    |    |    |    |    |    |    |

CNT<sub>n</sub>[15:8]: 计数器的高 8 位值 (n= A,B)

## 22.7.19 计数器低 8 位 (PWM<sub>x</sub>\_CNTRL)

| 符号         | 地址    | B7        | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|-----------|----|----|----|----|----|----|----|
| PWMA_CNTRL | FECFH | CNT1[7:0] |    |    |    |    |    |    |    |
| PWMB_CNTRL | FEFFH | CNT2[7:0] |    |    |    |    |    |    |    |

CNT<sub>n</sub>[7:0]: 计数器的低 8 位值 (n= A,B)

## 22.7.20 预分频器高 8 位 (PWM<sub>x</sub>\_PSCRH), 输出频率计算公式

| 符号         | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|------------|----|----|----|----|----|----|----|
| PWMA_PSCRH | FED0H | PSC1[15:8] |    |    |    |    |    |    |    |
| PWMB_PSCRH | FEF0H | PSC2[15:8] |    |    |    |    |    |    |    |

PSC<sub>n</sub>[15:8]: 预分频器的高 8 位值。(n= A,B)

预分频器用于对 CK\_PSC 进行分频。计数器的时钟频率(fCK\_CNT)等于 fCK\_PSC/(PSCR[15:0]+1)。

PSCR 包含了当更新事件产生时装入当前预分频器寄存器的值 (更新事件包括计数器被 TIM\_EGR 的 UG 位清 0 或被工作在复位模式的从控制器清 0)。这意味着为了使新的值起作用, 必须产生一个更新事件。

## PWM 输出频率计算公式

PWMA 和 PWMB 两组 PWM 的输出频率计算公式相同, 且每组可设置不同的频率。

| 对齐模式 | PWM输出频率计算公式   |
|------|---|
| 边沿对齐 | $\text{PWM输出频率} = \frac{\text{系统工作频率SYSclk}}{(\text{PWMx\_PSCR} + 1) \times (\text{PWMx\_ARR} + 1)}$    |
| 中间对齐 | $\text{PWM输出频率} = \frac{\text{系统工作频率SYSclk}}{(\text{PWMx\_PSCR} + 1) \times \text{PWMx\_ARR} \times 2}$ |

## 22.7.21 预分频器低 8 位 (PWM<sub>x</sub>\_PSCRL)

| 符号         | 地址    | B7        | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|-----------|----|----|----|----|----|----|----|
| PWMA_PSCRL | FED1H | PSC1[7:0] |    |    |    |    |    |    |    |
| PWMB_PSCRL | FEF1H | PSC2[7:0] |    |    |    |    |    |    |    |

PSC<sub>n</sub>[7:0]: 预分频器的低 8 位值。(n= A,B)

## 22.7.22 自动重装载寄存器高 8 位 (PWM<sub>x</sub>\_ARRH)

| 符号        | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----------|-------|------------|----|----|----|----|----|----|----|
| PWMA_ARRH | FED2H | ARR1[15:8] |    |    |    |    |    |    |    |
| PWMB_ARRH | FEF2H | ARR2[15:8] |    |    |    |    |    |    |    |

ARR<sub>n</sub>[15:8]: 自动重装载高 8 位值 (n= A,B)

ARR 包含了将要装载入实际的自动重装载寄存器的值。当自动重装载的值为 0 时, 计数器不工作。

## 22.7.23 自动重装载寄存器低 8 位 (PWM<sub>x</sub>\_ARRL)

| 符号        | 地址    | B7        | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----------|-------|-----------|----|----|----|----|----|----|----|
| PWMA_ARRL | FED3H | ARR1[7:0] |    |    |    |    |    |    |    |
| PWMB_ARRL | FEF3H | ARR2[7:0] |    |    |    |    |    |    |    |

ARR<sub>n</sub>[7:0]: 自动重装载低 8 位值 (n= A,B)

## 22.7.24 重复计数器寄存器 (PWM<sub>x</sub>\_RCR)

| 符号       | 地址    | B7        | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----------|-------|-----------|----|----|----|----|----|----|----|
| PWMA_RCR | FED4H | REP1[7:0] |    |    |    |    |    |    |    |
| PWMB_RCR | FEF4H | REP2[7:0] |    |    |    |    |    |    |    |

REP<sub>n</sub>[7:0]: 重复计数器值 (n= A,B)

开启了预装载功能后, 这些位允许用户设置比较寄存器的更新速率 (即周期性地从预装载寄存器 传输到当前寄存器); 如果允许产生更新中断, 则会同时影响产生更新中断的速率。每次向下计数器 REP\_CNT 达到 0, 会产生一个更新事件并且计数器 REP\_CNT 重新从 REP 值开始计数。由于 REP\_CNT 只有在周期更新事件 U\_RC 发生时才重载 REP 值, 因此对 PWM<sub>n</sub>\_RCR 寄存器写入的新值只在下次周期更新事件发生时才起作用。这意味着在 PWM 模式中, (REP+1) 对应着:

- 一 在边沿对齐模式下, PWM 周期的数目;
- 一 在中心对称模式下, PWM 半周期的数目;

## 22.7.25 捕获/比较寄存器 1/5 高 8 位 (PWM<sub>x</sub>\_CCR1H)

| 符号         | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|------------|----|----|----|----|----|----|----|
| PWMA_CCR1H | FED5H | CCR1[15:8] |    |    |    |    |    |    |    |
| PWMB_CCR5H | FEF5H | CCR5[15:8] |    |    |    |    |    |    |    |

CCR<sub>n</sub>[15:8]: 捕获/比较 n 的高 8 位值 (n=1,5)

若 CC<sub>n</sub> 通道配置为输出: CCR<sub>n</sub> 包含了装入当前比较值 (预装载值)。如果在 PWM<sub>n</sub>\_CCMR1 寄存器 (OC<sub>n</sub>PE 位) 中未选择预装载功能, 写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时, 此预装载值才传输至当前捕获/比较 n 寄存器中。当前比较值同计数器 PWM<sub>n</sub>\_CNT 的值相比较, 并在 OC<sub>n</sub> 端口上产生输出信号。

若 CC<sub>n</sub> 通道配置为输入: CCR<sub>n</sub> 包含了上一次输入捕获事件发生时的计数器值 (此时该寄存器为只读)。

## 22.7.26 捕获/比较寄存器 1/5 低 8 位 (PWM<sub>x</sub>\_CCR1L)

| 符号 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----|----|----|----|----|----|----|----|----|----|
|----|----|----|----|----|----|----|----|----|----|

|            |       |           |
|------------|-------|-----------|
| PWMA_CCR1L | FED6H | CCR1[7:0] |
| PWMB_CCR5L | FEF6H | CCR5[7:0] |

CCRN[7:0]: 捕获/比较 n 的低 8 位值 (n=1,5)

### 22.7.27 捕获/比较寄存器 2/6 高 8 位 (PWMx\_CCR2H)

| 符号         | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|------------|----|----|----|----|----|----|----|
| PWMA_CCR2H | FED7H | CCR2[15:8] |    |    |    |    |    |    |    |
| PWMB_CCR6H | FEF7H | CCR6[15:8] |    |    |    |    |    |    |    |

CCRN[15:8]: 捕获/比较 n 的高 8 位值 (n=2,6)

### 22.7.28 捕获/比较寄存器 2/6 低 8 位 (PWMx\_CCR2L)

| 符号         | 地址    | B7        | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|-----------|----|----|----|----|----|----|----|
| PWMA_CCR2L | FED8H | CCR2[7:0] |    |    |    |    |    |    |    |
| PWMB_CCR6L | FEF8H | CCR6[7:0] |    |    |    |    |    |    |    |

CCRN[7:0]: 捕获/比较 n 的低 8 位值 (n=2,6)

### 22.7.29 捕获/比较寄存器 3/7 高 8 位 (PWMx\_CCR3H)

| 符号         | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|------------|----|----|----|----|----|----|----|
| PWMA_CCR3H | FED9H | CCR3[15:8] |    |    |    |    |    |    |    |
| PWMB_CCR7H | FEF9H | CCR7[15:8] |    |    |    |    |    |    |    |

CCRN[15:8]: 捕获/比较 n 的高 8 位值 (n=3,7)

### 22.7.30 捕获/比较寄存器 3/7 低 8 位 (PWMx\_CCR3L)

| 符号         | 地址     | B7        | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|--------|-----------|----|----|----|----|----|----|----|
| PWMA_CCR3L | FEDA H | CCR3[7:0] |    |    |    |    |    |    |    |
| PWMB_CCR7L | FEFA H | CCR7[7:0] |    |    |    |    |    |    |    |

CCRN[7:0]: 捕获/比较 n 的低 8 位值 (n=3,7)

### 22.7.31 捕获/比较寄存器 4/8 高 8 位 (PWMx\_CCR4H)

| 符号         | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|------------|----|----|----|----|----|----|----|
| PWMA_CCR4H | FEDBH | CCR4[15:8] |    |    |    |    |    |    |    |
| PWMB_CCR8H | FEFBH | CCR8[15:8] |    |    |    |    |    |    |    |

CCRN[15:8]: 捕获/比较 n 的高 8 位值 (n=4,8)

### 22.7.32 捕获/比较寄存器 4/8 低 8 位 (PWMx\_CCR4L)

| 符号         | 地址    | B7        | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|-----------|----|----|----|----|----|----|----|
| PWMA_CCR4L | FEDCH | CCR4[7:0] |    |    |    |    |    |    |    |
| PWMB_CCR8L | FEFCH | CCR8[7:0] |    |    |    |    |    |    |    |

CCRN[7:0]: 捕获/比较 n 的低 8 位值 (n=4,8)

### 22.7.33 刹车寄存器 (PWMx\_BKR)

| 符号       | 地址    | B7   | B6   | B5   | B4   | B3    | B2    | B1         | B0 |
|----------|-------|------|------|------|------|-------|-------|------------|----|
| PWMA_BKR | FEDDH | MOEA | AOEA | BKPA | BKEA | OSSRA | OSSIA | LOCKA[1:0] |    |
| PWMB_BKR | FEFDH | MOEB | AOEB | BKPB | BKEB | OSSRB | OSSIB | LOCKB[1:0] |    |

**MOEn:** 主输出使能。一旦刹车输入有效, 该位被硬件异步清 0。根据 AOE 位的设置值, 该位可以由软件置 1 或被自动置 1。它仅对配置为输出的通道有效。(n= A,B)

0: 禁止 OC 和 OCN 输出或强制为空闲状态

1: 如果设置了相应的使能位 (PWMn\_CCERX 寄存器的 CCIE 位), 则使能 OC 和 OCN 输出。

**AOEn:** 自动输出使能 (n= A,B)

0: MOE 只能被软件置 1;

1: MOE 能被软件置 1 或在下一个更新事件被自动置 1 (如果刹车输入无效)。

注: 一旦 LOCK 级别 (PWMn\_BKR 寄存器中的 LOCK 位) 设为 1, 则该位不能被修改

**BKPn:** 刹车输入极性 (n= A,B)

0: 刹车输入低电平有效

1: 刹车输入高电平有效

注: 一旦 LOCK 级别 (PWMn\_BKR 寄存器中的 LOCK 位) 设为 1, 则该位不能被修改

**BKE n:** 刹车功能使能 (n= A,B)

0: 禁止刹车输入 (BRK)

1: 开启刹车输入 (BRK)

注: 一旦 LOCK 级别 (PWMn\_BKR 寄存器中的 LOCK 位) 设为 1, 则该位不能被修改。

**OSSRn:** 运行模式下“关闭状态”选择。该位在 MOE=1 且通道设为输出时有效 (n= A,B)

0: 当 PWM 不工作时, 禁止 OC/OCN 输出 (OC/OCN 使能输出信号=0);

1: 当 PWM 不工作时, 一旦 CCiE=1 或 CCiNE=1, 首先开启 OC/OCN 并输出无效电平, 然后置 OC/OCN 使能输出信号=1。

注: 一旦 LOCK 级别 (PWMn\_BKR 寄存器中的 LOCK 位) 设为 2, 则该位不能被修改。

**OSSI n:** 空闲模式下“关闭状态”选择。该位在 MOE=0 且通道设为输出时有效。(n= A,B)

0: 当 PWM 不工作时, 禁止 OC/OCN 输出 (OC/OCN 使能输出信号=0);

1: 当 PWM 不工作时, 一旦 CCiE=1 或 CCiNE=1, OC/OCN 首先输出其空闲电平, 然后 OC/OCN 使能输出信号=1。

注: 一旦 LOCK 级别 (PWMn\_BKR 寄存器中的 LOCK 位) 设为 2, 则该位不能被修改。

**LOCKn[1:0]:** 锁定设置。该位为防止软件错误而提供的写保护措施 (n= A,B)

| LOCKn[1:0] | 保护级别  | 保护内容  |
|------------|-------|---|
| 00         | 无保护   | 寄存器无写保护   |
| 01         | 锁定级别1 | 不能写入PWMn_BKR寄存器的BKE、BKP、AOE位和PWMn_OISR寄存器的OISI位 |
| 10         | 锁定级别2 | 不能写入锁定级别1中的各位,<br>也不能写入CC极性位以及OSSR/OSSI位        |
| 11         | 锁定级别3 | 不能写入锁定级别2中的各位,<br>也不能写入CC控制位                    |

注: 由于 BKE、BKP、AOE、OSSR、OSSI 位可被锁定 (依赖于 LOCK 位), 因此在第一次写 PWMn\_BKR 寄存器时必须对它们进行设置。

## 22.7.34 死区寄存器 (PWMx\_DTR)

| 符号       | 地址    | B7        | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----------|-------|-----------|----|----|----|----|----|----|----|
| PWMA_DTR | FEDEH | DTGA[7:0] |    |    |    |    |    |    |    |
| PWMB_DTR | FEFEH | DTGB[7:0] |    |    |    |    |    |    |    |

DTGn[7:0]: 死区发生器设置。(n= A,B)

这些位定义了插入互补输出之间的死区持续时间。(t<sub>CK\_PSC</sub> 为 PWMn 的时钟脉冲)

| DTGn[7:5] | 死区时间  |
|-----------|---|
| 000       | DTGn[7:0] * t <sub>CK_PSC</sub>             |
| 001       |   |
| 010       |   |
| 011       |   |
| 100       | (64 + DTGn[6:0]) * 2 * t <sub>CK_PSC</sub>  |
| 101       |   |
| 110       | (32 + DTGn[5:0]) * 8 * t <sub>CK_PSC</sub>  |
| 111       | (32 + DTGn[4:0]) * 16 * t <sub>CK_PSC</sub> |

注: 一旦 LOCK 级别 (PWMx\_BKR 寄存器中的 LOCK 位) 设为 1、2 或 3 时, 则该位不能被修改。

## 22.7.35 输出空闲状态寄存器 (PWMx\_OISR)

| 符号        | 地址    | B7    | B6   | B5    | B4   | B3    | B2   | B1    | B0   |
|-----------|-------|-------|------|-------|------|-------|------|-------|------|
| PWMA_OISR | FEDFH | OIS4N | OIS4 | OIS3N | OIS3 | OIS2N | OIS2 | OIS1N | OIS1 |
| PWMB_OISR | FEFFH | -     | OIS8 | -     | OIS7 | -     | OIS6 | -     | OIS5 |

OIS8: 空闲状态时 OC8 输出电平

OIS7: 空闲状态时 OC7 输出电平

OIS6: 空闲状态时 OC6 输出电平

OIS5: 空闲状态时 OC5 输出电平

OIS4N: 空闲状态时 OC4N 输出电平

OIS4: 空闲状态时 OC4 输出电平

OIS3N: 空闲状态时 OC3N 输出电平

OIS3: 空闲状态时 OC3 输出电平

OIS2N: 空闲状态时 OC2N 输出电平

OIS2: 空闲状态时 OC2 输出电平

OIS1N: 空闲状态时 OC1N 输出电平

0: 当 MOE=0 时, 则在一个死区时间后, OC1N=0;

1: 当 MOE=0 时, 则在一个死区时间后, OC1N=1。

注: 一旦 LOCK 级别 (PWMx\_BKR 寄存器中的 LOCK 位) 设为 1、2 或 3 时, 则该位不能被修改。

OIS1: 空闲状态时 OC1 输出电平

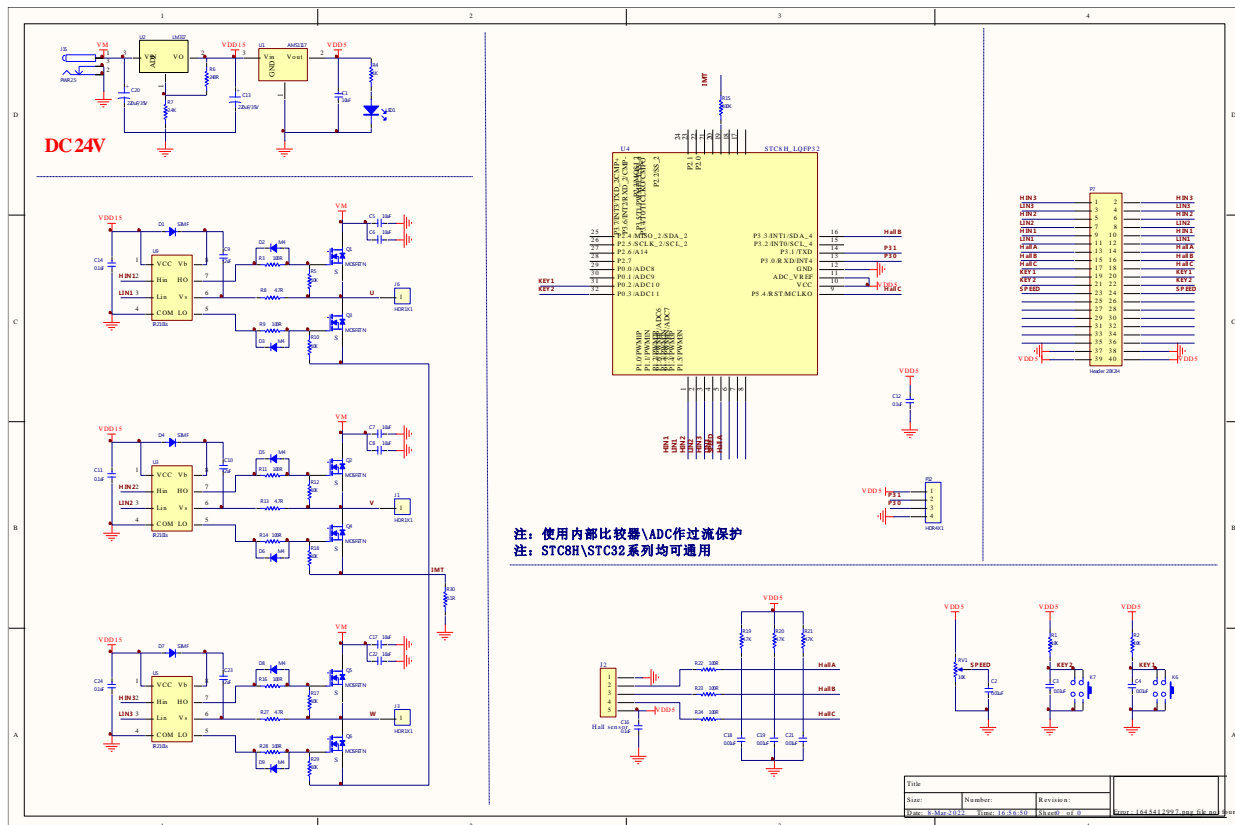
0: 当 MOE=0 时, 如果 OC1N 使能, 则在一个死区后, OC1=0;

1: 当 MOE=0 时, 如果 OC1N 使能, 则在一个死区后, OC1=1。

注: 一旦 LOCK 级别 (PWMx\_BKR 寄存器中的 LOCK 位) 设为 1、2 或 3 时, 则该位不能被修改。

## 22.8 范例程序

### 22.8.1 BLDC 无刷直流电机驱动(带 HALL)



#### C 语言代码

//测试工作频率为11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
typedef unsigned char u8;
typedef unsigned int u16;
```

```
#define TRUE 1
#define FALSE 0
```

```
#define RV09_CH 6
```

```
#define TIM1_Period ((u16)0x0180)
#define TIM1_STPulse ((u16)342)
```

```
#define START 0x1A
#define RUN 0x1B
#define STOP 0x1C
#define IDLE 0x1D
```

```
#define TIM1_OCMODE_MASK ((u8)0x70)
#define TIM1_OCCE_ENABLE ((u8)0x80)
```



```

#define TIM1_OCCE_DISABLE ((u8)0x00)
#define TIM1_OCMODE_TIMING ((u8)0x00)
#define TIM1_OCMODE_ACTIVE ((u8)0x10)
#define TIM1_OCMODE_INACTIVE ((u8)0x20)
#define TIM1_OCMODE_TOGGLE ((u8)0x30)
#define TIM1_FORCE_INACTIVE ((u8)0x40)
#define TIM1_FORCE_ACTIVE ((u8)0x50)
#define TIM1_OCMODE_PWMA ((u8)0x60)
#define TIM1_OCMODE_PWMB ((u8)0x70)
#define CC1_POLARITY_HIGH ((u8)0x02)
#define CC1N_POLARITY_HIGH ((u8)0x08)
#define CC2_POLARITY_HIGH ((u8)0x20)
#define CC2N_POLARITY_HIGH ((u8)0x80)
#define CC1_POLARITY_LOW ((u8)~0x02)
#define CC1N_POLARITY_LOW ((u8)~0x08)
#define CC2_POLARITY_LOW ((u8)~0x20)
#define CC2N_POLARITY_LOW ((u8)~0x80)
#define CC1_OCENABLE ((u8)0x01)
#define CC1N_OCENABLE ((u8)0x04)
#define CC2_OCENABLE ((u8)0x10)
#define CC2N_OCENABLE ((u8)0x40)
#define CC1_OCDISABLE ((u8)~0x01)
#define CC1N_OCDISABLE ((u8)~0x04)
#define CC2_OCDISABLE ((u8)~0x10)
#define CC2N_OCDISABLE ((u8)~0x40)
#define CC3_POLARITY_HIGH ((u8)0x02)
#define CC3N_POLARITY_HIGH ((u8)0x08)
#define CC4_POLARITY_HIGH ((u8)0x20)
#define CC4N_POLARITY_HIGH ((u8)0x80)
#define CC3_POLARITY_LOW ((u8)~0x02)
#define CC3N_POLARITY_LOW ((u8)~0x08)
#define CC4_POLARITY_LOW ((u8)~0x20)
#define CC4N_POLARITY_LOW ((u8)~0x80)
#define CC3_OCENABLE ((u8)0x01)
#define CC3N_OCENABLE ((u8)0x04)
#define CC4_OCENABLE ((u8)0x10)
#define CC4N_OCENABLE ((u8)0x40)
#define CC3_OCDISABLE ((u8)~0x01)
#define CC3N_OCDISABLE ((u8)~0x04)
#define CC4_OCDISABLE ((u8)~0x10)
#define CC4N_OCDISABLE ((u8)~0x40)

```

```
void LED_OUT(u8 X);
```

```
//LED 单字节串行移位函数
```

```

unsigned char code LED_0F[] =
{
    0xC0,0xF9,0xA4,0xB0,
    0x99,0x92,0x82,0xF8,
    0x80,0x90,0x8C,0xBF,
    0xC6,0xA1,0x86,0xFF,
    0xbf
};

```

```

#define DIO          P23
#define RCLK         P24
#define SCLK         P25

```

```

//串行数据输入
//时钟脉冲信号——上升沿有效
//打入信号——上升沿有效

```

```

void DelayXus(unsigned char delayTime);
void DelayXms(unsigned char delayTime);

```

```

unsigned int ADC_Convert(u8 ch);
void PWM_Init(void);
void SPEED_ADJ();
unsigned char RD_HALL();
void MOTOR_START();
void MOTOR_STOP();
unsigned char KEY_detect();
void LED4_Display (unsigned int dat,unsigned char num);

```

```

unsigned char Display_num=1;
unsigned int Display_dat=0;
unsigned int Motor_speed;
unsigned char Motor_sta = IDLE;
unsigned char BRK_occur=0;
unsigned int TIM2_CAP1_y=0;
unsigned int CAP1_avg=0;
unsigned char CAP1_cnt=0;
unsigned long CAP1_sum=0;

```

```
void main(void)
```

```
{
```

```
    P_SW2 /= 0x80;
```

```
    P1 = 0x00;
```

```
    P0M1 = 0x0C;
```

```
    P0M0 = 0x01;
```

```
    P1M1 = 0xc0;
```

```
    P1M0 = 0x3F;
```

```
    P2M1 = 0x00;
```

```
    P2M0 = 0x38;
```

```
    P3M1 = 0x28;
```

```
    P3M0 = 0x00;
```

```
    ET0=1;
```

```
    TR0=1;
```

```
    ADCCFG = 0x0f;
```

```
    ADC_CONTR = 0x80;
```

```
    PWMA_ENO = 0x3F;
```

```
//PWMA 输出使能
```

```
    PWMB_ENO = 0x00;
```

```
//PWMB 输出使能
```

```
    PWMA_PS = 0x00;
```

```
//PWMA pin 选择
```

```
    PWMB_PS = 0xd5;
```

```
//PWMB pin 选择
```

```
/******
```

```
输出比较模式 PWMx_duty = [CCRx/(ARR + 1)]*100
```

```
*****
```

```
/******PWMB 接 hall 传感器*****
```

```
//////////时基单元//////////
```

```
    PWMB_PSCRL = 15;
```

```
    PWMB_ARRH = 0xff;
```

```
//自动重载寄存器, 计数器 overflow 点
```

```
    PWMB_ARRL = 0xff;
```

```
    PWMB_CCR4H = 0x00;
```

```
    PWMB_CCR4L = 0x05;
```

```
//////////通道配置//////////
```

```
    PWMB_CCMR1 = 0x43;
```

```
//通道模式配置
```

```
    PWMB_CCMR2 = 0x41;
```

```
    PWMB_CCMR3 = 0x41;
```

```
PWMB_CCMR4 = 0x70;
PWMB_CCER1 = 0x11;
PWMB_CCER2 = 0x11;
```

```
////////// 模式配置//////////
```

```
PWMB_CR2 = 0xf0;
PWMB_CR1 = 0x81;
PWMB_SMCR = 0x44;
```

```
////////// 使能& 中断配置//////////
```

```
PWMB_BKR = 0x80;           //主输出使能
PWMB_IER = 0x02;          //使能中断
```

```
/******PWMA 控制马达换相******/
```

```
////////// 时基单元//////////
```

```
PWMA_PSCRH = 0x00;         //预分频寄存器
PWMA_PSCRL = 0x00;
PWMA_ARRH = (u8)(TIM1_Period >> 8);
PWMA_ARRL = (u8)(TIM1_Period);
```

```
////////// 通道配置//////////
```

```
PWMA_CCMR1 = 0x70;         //通道模式配置
PWMA_CCMR2 = 0x70;
PWMA_CCMR3 = 0x70;
PWMA_CCER1 = 0x11;         //配置通道输出使能和极性
PWMA_CCER2 = 0x01;         //配置通道输出使能和极性
PWMA_OISR = 0xAA;          //配置MOE=0 时各通道输出电平
```

```
////////// 模式配置//////////
```

```
PWMA_CR1 = 0xA0;
PWMA_CR2 = 0x24;
PWMA_SMCR = 0x20;
```

```
////////// 使能& 中断配置//////////
```

```
PWMA_BKR = 0x1c;
PWMA_CR1 |= 0x01;          //使能计数器
```

```
EA = 1;
```

```
while (1)
```

```
{
```

```
    P22=~P22;
```

```
    Display_dat = Motor_speed;           //Motor_speed
```

```
    switch(Motor_sta)
```

```
    {
```

```
        case START:
```

```
            MOTOR_START();
```

```
            Motor_sta = RUN;
```

```
            break;
```

```
        case RUN:
```

```
            SPEED_ADJ();
```

```
            if((KEY_detect() == 2)||(BRK_occur == TRUE))
```

```
                Motor_sta = STOP;
```

```
            break;
```

```
        case STOP:
```

```
            MOTOR_STOP();
```

```
            Motor_sta = IDLE;
```

```
            break;
```

```

        case IDLE:
            if(KEY_detect()==1)
                Motor_sta = START;
            BRK_occur = FALSE;
            Motor_speed = 0;
            CAPI_avg = 0;
            CAPI_cnt = 0;
            CAPI_sum = 0;
            break;
        }
    }
}

void TIM0_ISR() interrupt 1
{
    TH0=0xf0;
    if(Display_num>8)
        Display_num=1;
    LED4_Display(Display_dat,Display_num);
    Display_num=(Display_num<<1);
}

void PWMA_ISR() interrupt 26
{
    if((PWMA_SR1 & 0x20))
    {
        switch(RD_HALL())
        {
            case 3:
                PWMA_CCMR3 &= ~TIM1_OCMODE_MASK;
                PWMA_CCMR3 /= TIM1_FORCE_INACTIVE;
                PWMA_CCMR1 &= ~TIM1_OCMODE_MASK;
                PWMA_CCMR1 /= TIM1_OCMODE_PWMB;
                break;
            case 2:
                PWMA_CCER1 &= CC2N_POLARITY_LOW;
                PWMA_CCER2 /= CC3N_POLARITY_HIGH;
                break;
            case 6:
                PWMA_CCMR1 &= ~TIM1_OCMODE_MASK;
                PWMA_CCMR1 /= TIM1_FORCE_INACTIVE;
                PWMA_CCMR2 &= ~TIM1_OCMODE_MASK;
                PWMA_CCMR2 /= TIM1_OCMODE_PWMB;
                break;
            case 4:
                PWMA_CCER1 /= CC1N_POLARITY_HIGH;
                PWMA_CCER2 &= CC3N_POLARITY_LOW;
                break;
            case 5:
                PWMA_CCMR2 &= ~TIM1_OCMODE_MASK;
                PWMA_CCMR2 /= TIM1_FORCE_INACTIVE;
                PWMA_CCMR3 &= ~TIM1_OCMODE_MASK;
                PWMA_CCMR3 /= TIM1_OCMODE_PWMB;
                break;
            case 1:
                PWMA_CCER1 &= CC1N_POLARITY_LOW;
                PWMA_CCER1 /= CC2N_POLARITY_HIGH;
                break;
        }
    }
}

```

```

    CAP1_sum += TIM2_CAP1_v;
    CAP1_cnt++;
    if(CAP1_cnt==128)
    {
        CAP1_cnt=0;
        CAP1_avg = (CAP1_sum>>7);
        CAP1_sum = 0;
        Motor_speed = 5000000/CAP1_avg;
    }

    PWMA_SRI &=~0x20;                //清零
}
if((PWMA_SRI & 0x80))                //BRK
{
    BRK_occur = TRUE;
    PWMA_SRI &=~0x80;                //清零
}
}

void PWMB_ISR() interrupt 27
{
    if((PWMB_SRI & 0x02))
    {
        TIM2_CAP1_v = PWMB_CCR1H;
        TIM2_CAP1_v = (TIM2_CAP1_v<<8) + PWMB_CCR1L;
        PWMB_SRI &=~0x02;
    }
}

void DelayXus(unsigned char delayTime)
{
    int i = 0;
    while( delayTime--)
    {
        for( i = 0 ; i < 1 ; i++);
    }
}

void DelayXms( unsigned char delayTime )
{
    int i = 0;
    while( delayTime--)
    {
        for( i = 0 ; i < 2 ; i++)
        {
            DelayXus(100);
        }
    }
}

unsigned int ADC_Convert(u8 ch)
{
    u16 res=0;

    ADC_CONTR &= ~0x0f;
    ADC_CONTR |= ch;
    ADC_CONTR |= 0x40;
    DelayXus(1);
}

```

```

    while (!(ADC_CONTR & 0x20));
    ADC_CONTR &= ~0x20;

    res = ADC_RES;
    res = (res<<2)+(ADC_RESL>>6);
    return res;
}

void SPEED_ADJ()
{
    u16 ADC_result;

    ADC_result = (ADC_Convert(RV09_CH)/3);
    PWMA_CCR1H = (u8)(ADC_result >> 8);           //计数器比较值
    PWMA_CCR1L = (u8)(ADC_result);
    PWMA_CCR2H = (u8)(ADC_result >> 8);
    PWMA_CCR2L = (u8)(ADC_result);
    PWMA_CCR3H = (u8)(ADC_result >> 8);
    PWMA_CCR3L = (u8)(ADC_result);
}

unsigned char RD_HALL()
{
    unsigned char Hall_sta = 0;

    (P17)? (Hall_sta|=0x01) : (Hall_sta&=~0x01);
    (P54)? (Hall_sta|=0x02) : (Hall_sta&=~0x02);
    (P33)? (Hall_sta|=0x04) : (Hall_sta&=~0x04);

    return Hall_sta;
}

void MOTOR_START()
{
    u16 temp;
    u16 ADC_result;

    PWMA_CCR1H = (u8)(TIM1_STPulse >> 8);           //计数器比较值
    PWMA_CCR1L = (u8)(TIM1_STPulse);
    PWMA_CCR2H = (u8)(TIM1_STPulse >> 8);
    PWMA_CCR2L = (u8)(TIM1_STPulse);
    PWMA_CCR3H = (u8)(TIM1_STPulse >> 8);
    PWMA_CCR3L = (u8)(TIM1_STPulse);
    PWMA_BKR /= 0x80;                               //主输出使能相当于总开关
    PWMA_IER /= 0xA0;                               //使能中断

    switch(RD_HALL())
    {
    case 1:
        PWMA_CCER1 &= CC1N_POLARITY_LOW;
        PWMA_CCER1 /= CC2N_POLARITY_HIGH;
        PWMA_CCER2 &= CC3N_POLARITY_LOW;
        PWMA_CCMR3 &= ~TIM1_OCMODE_MASK;
        PWMA_CCMR3 /= TIM1_FORCE_INACTIVE;
        PWMA_CCMR2 &= ~TIM1_OCMODE_MASK;
        PWMA_CCMR2 /= TIM1_FORCE_INACTIVE;
        PWMA_CCMR1 &= ~TIM1_OCMODE_MASK;
        PWMA_CCMR1 /= TIM1_OCMODE_PWMB;
        break;
    }
}

```

case 3:

```
PWMA_CCMR3 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR3 /= TIM1_FORCE_INACTIVE;
PWMA_CCMR2 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR2 /= TIM1_FORCE_INACTIVE;
PWMA_CCMR1 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR1 /= TIM1_OCMODE_PWMB;
PWMA_CCER1 &= CC1N_POLARITY_LOW;
PWMA_CCER1 &= CC2N_POLARITY_LOW;
PWMA_CCER2 /= CC3N_POLARITY_HIGH;
break;
```

case 2:

```
PWMA_CCER1 &= CC1N_POLARITY_LOW;
PWMA_CCER1 &= CC2N_POLARITY_LOW;
PWMA_CCER2 /= CC3N_POLARITY_HIGH;
PWMA_CCMR1 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR1 /= TIM1_FORCE_INACTIVE;
PWMA_CCMR2 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR2 /= TIM1_OCMODE_PWMB;
PWMA_CCMR3 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR3 /= TIM1_FORCE_INACTIVE;
break;
```

case 6:

```
PWMA_CCMR1 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR1 /= TIM1_FORCE_INACTIVE;
PWMA_CCMR2 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR2 /= TIM1_OCMODE_PWMB;
PWMA_CCMR3 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR3 /= TIM1_FORCE_INACTIVE;
PWMA_CCER1 /= CC1N_POLARITY_HIGH;
PWMA_CCER1 &= CC2N_POLARITY_LOW;
PWMA_CCER2 &= CC3N_POLARITY_LOW;
break;
```

case 4:

```
PWMA_CCER1 /= CC1N_POLARITY_HIGH;
PWMA_CCER1 &= CC2N_POLARITY_LOW;
PWMA_CCER2 &= CC3N_POLARITY_LOW;
PWMA_CCMR1 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR1 /= TIM1_FORCE_INACTIVE;
PWMA_CCMR2 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR2 /= TIM1_FORCE_INACTIVE;
PWMA_CCMR3 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR3 /= TIM1_OCMODE_PWMB;
break;
```

case 5:

```
PWMA_CCMR1 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR1 /= TIM1_FORCE_INACTIVE;
PWMA_CCMR2 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR2 /= TIM1_FORCE_INACTIVE;
PWMA_CCMR3 &= ~TIM1_OCMODE_MASK;
PWMA_CCMR3 /= TIM1_OCMODE_PWMB;
PWMA_CCER1 &= CC1N_POLARITY_LOW;
PWMA_CCER1 /= CC2N_POLARITY_HIGH;
PWMA_CCER2 &= CC3N_POLARITY_LOW;
break;
```

}

```
ADC_result = (ADC_Convert(RV09_CH)/3);
```

```
for(temp = TIM1_STPulse; temp > ADC_result; temp--)
```

```

    {
        PWMA_CCR1H = (u8)(temp >> 8);           //计数器比较值
        PWMA_CCR1L = (u8)(temp);
        PWMA_CCR2H = (u8)(temp >> 8);
        PWMA_CCR2L = (u8)(temp);
        PWMA_CCR3H = (u8)(temp >> 8);
        PWMA_CCR3L = (u8)(temp);
        DelayXms(10);
    }
}

void MOTOR_STOP()
{
    PWMA_BKR &= ~0x80;
    PWMA_IER &= ~0xA0;
}

void LED4_Display (u16 dat,u8 num)
{
    switch(num)
    {
        case 0x01:
            LED_OUT(LED_0F[(dat/1)%10]);
            LED_OUT(0x01);
            RCLK = 0;
            RCLK = 1;
            break;
        case 0x02:
            LED_OUT(LED_0F[(dat/10)%10]);
            LED_OUT(0x02);
            RCLK = 0;
            RCLK = 1;
            break;
        case 0x04:
            LED_OUT(LED_0F[(dat/100)%10]);
            LED_OUT(0x04);
            RCLK = 0;
            RCLK = 1;
            break;
        case 0x08:
            LED_OUT(LED_0F[(dat/1000)%10]);
            LED_OUT(0x08);
            RCLK = 0;
            RCLK = 1;
            break;
    }
}

void LED_OUT(u8 X)
{
    u8 i;

    for(i=8;i>=1;i--)
    {
        if (X&0x80) DIO=1;
        else DIO=0;
        X<<=1;
        SCLK = 0;
        SCLK = 1;
    }
}

```

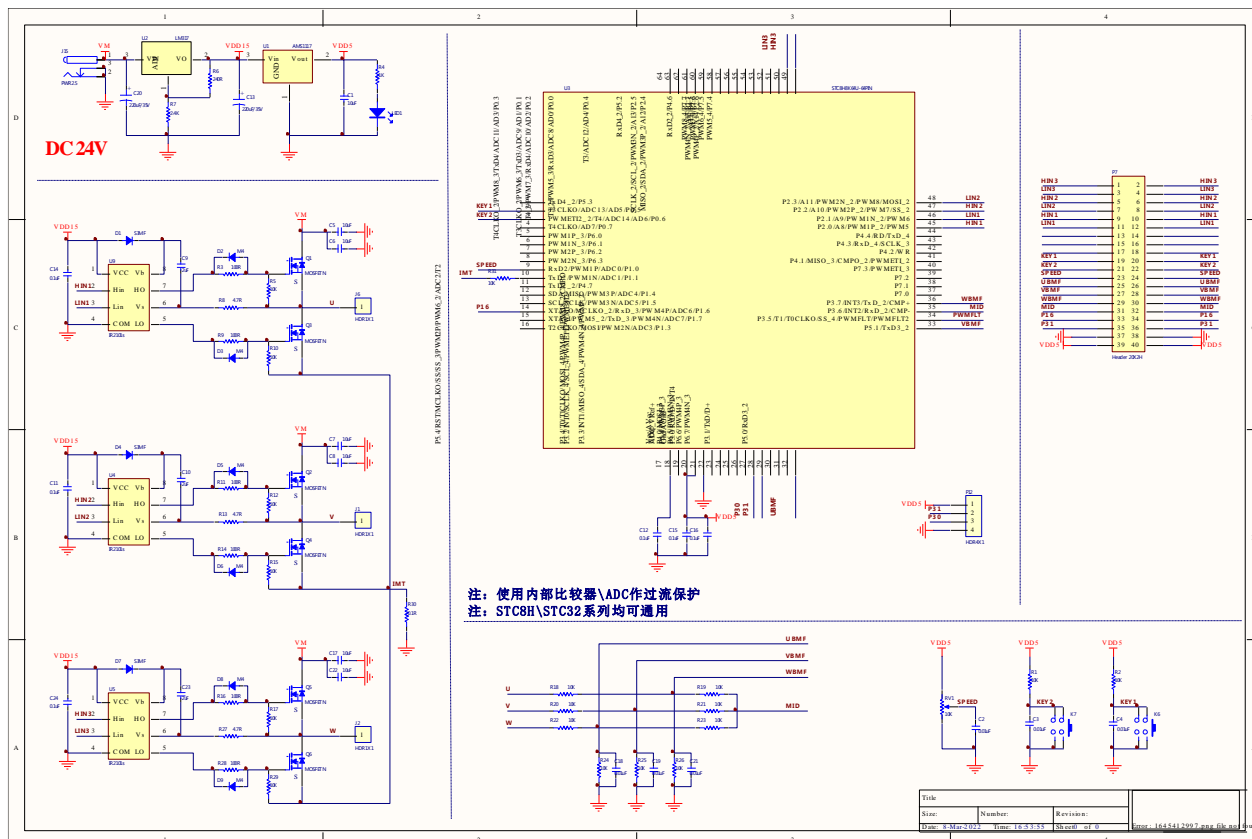


```
    }  
}  
  
unsigned char KEY_detect()  
{  
    if(!P02)  
    {  
        DelayXms(10);  
        if(!P02)  
        {  
            return 1;  
        }  
        else return 0;  
    }  
    else if(!P03)  
    {  
        DelayXms(10);  
        if(!P03)  
        {  
            return 2;  
        }  
        else return 0;  
    }  
    else return 0;  
}
```

---

## 22.8.2 BLDC 无刷直流电机驱动(无 HALL), 一个比较器当 3 个比较器分时复用

——可实现无霍尔 100000 转高速运转



## C 语言代码

//测试工作频率为11.0592MHz

//本例程实现如下功能: 侦测反电动势, 驱动无霍尔无刷直流马达运转

//目前使用MCU 内置单比较器可实现**无霍尔100000 转高速运转**, 还有提升空间

/\*include \*\*\*\*\*/

#include "STC8H.h"

/\*typedef \*\*\*\*\*/

typedef unsigned char u8;

typedef unsigned int u16;

/\*define \*\*\*\*\*/

#define TIM1\_Period ((u16)200)

#define START 0x1A

#define RUN 0x1B

#define STOP 0x1C

#define IDLE 0x1D

/\*Variable \*\*\*\*\*/

u8 Hall\_sta = 0;

u16 Aim\_Pulse, Start\_Pulse;

u8 Motor\_sta = IDLE;

u8 Hall\_a=0, Hall\_b=0, Hall\_c=0;

u8 CMP\_ch=0;

u8 BMF\_en=0;

u8 BMF=0;

/\*function \*\*\*\*\*/

void DelayXus(u8 delayTime);

void DelayXms(u8 delayTime);

void MOTOR\_START(void);

```
void MOTOR_STOP(void);
u8 RD_BMF(void);
u8 KEY_detect(void);
```

```

/***** main *****/
void main(void)
{
    P_SW2 /= 0x80;                //使能XFR 访问

    P0M1 = 0x0F;
    P0M0 = 0x00;
    P1M1 = 0xc1;
    P1M0 = 0x30;
    P2M1 = 0x00;
    P2M0 = 0xff;
    P3M1 = 0xE0;
    P3M0 = 0x13;
    P5M1 = 0x03;
    P5M0 = 0x00;
    RSTCFG = 0x40;                //P54 用作IO

    ADCCFG = 0x0f;
    ADC_CONTR = 0x80;

    CMPCR2 = 0x1f;
    CMPEXCFG = 0x00;
    CMPCR1 = 0xB0;

    PWM1_ENO = 0x15;              //IO 输出PWM
    PWM1_PS = 0x55;

    PWMA_PSCRH = 0x00;            //预分频寄存器
    PWMA_PSCRL = 0x02;
    PWMA_DTR = 0x02;              // 死区时间配置
    PWMA_ARRH = (u8)(TIM1_Period >> 8);
    PWMA_ARRL = (u8)(TIM1_Period);
    PWMA_CCMR1 = 0x60;            // 通道模式配置
    PWMA_CCMR2 = 0x60;
    PWMA_CCMR3 = 0x60;
    PWMA_CCER1 = 0x11;            // 配置通道输出使能和极性
    PWMA_CCER2 = 0x01;            // 配置通道输出使能和极性
    PWMA_OISR = 0x00;            // 配置 MOE=0 时各通道输出电平

    PWMA_CR1 = 0xA0;
    PWMA_CR2 = 0x24;
    PWMA_SMCR = 0x20;
    PWMA_BKR = 0x0c;
    PWMA_CR1 /= 0x01;            // 使能计数器

    EA = 1;
    Aim_Pulse = 100;

    while (1)
    {
        switch(Motor_sta)
        {
            case START:
                MOTOR_START();
        }
    }
}

```

```

        Motor_sta = RUN;
        break;
    case RUN:
        if(KEY_detect()==1)
            Motor_sta = STOP;
        break;
    case STOP:
        MOTOR_STOP();
        Motor_sta = IDLE;
        break;
    case IDLE:
        if(KEY_detect()==1)
            Motor_sta = START;
        break;
    default: break;
}
}
}

void CMP_Isr() interrupt 21
{
    CMPCR1 &= ~0x30; //关比较器中断

    if(CMPCR1 & 0x01)
    {
        if((CMPEXCFG&0X03) == 0) {Hall_a=1;}
        if((CMPEXCFG&0X03) == 1) {Hall_b=1;}
        if((CMPEXCFG&0X03) == 2) {Hall_c=1;}
    }
    else
    {
        if((CMPEXCFG&0X03) == 0) {Hall_a=0;}
        if((CMPEXCFG&0X03) == 1) {Hall_b=0;}
        if((CMPEXCFG&0X03) == 2) {Hall_c=0;}
    }
    BMF = RD_BMF();
    if(BMF_en)
    {
        switch(BMF)
        {
            case 3:
                PWMA_CCR3H = 0;
                PWMA_CCR3L = 0;
                PWMA_CCR1H = (u8)(Aim_Pulse >> 8);
                PWMA_CCR1L = (u8)(Aim_Pulse);
                PWMA_CCR2H = 0;
                PWMA_CCR2L = 0;
                P21=0;P23=1;P25=0;
                break;

            case 2:
                PWMA_CCR3H = 0;
                PWMA_CCR3L = 0;
                PWMA_CCR1H = (u8)(Aim_Pulse >> 8);
                PWMA_CCR1L = (u8)(Aim_Pulse);
                PWMA_CCR2H = 0;
                PWMA_CCR2L = 0;
                P21=0;P23=0;P25=1;
                break;

            case 6:

```

```

        PWMA_CCR1H = 0;
        PWMA_CCR1L = 0;
        PWMA_CCR2H = (u8)(Aim_Pulse >> 8);
        PWMA_CCR2L = (u8)(Aim_Pulse);
        PWMA_CCR3H = 0;
        PWMA_CCR3L = 0;
        P21=0;P23=0;P25=1;
        break;
    case 4:
        PWMA_CCR1H = 0;
        PWMA_CCR1L = 0;
        PWMA_CCR2H = (u8)(Aim_Pulse >> 8);
        PWMA_CCR2L = (u8)(Aim_Pulse);
        PWMA_CCR3H = 0;
        PWMA_CCR3L = 0;
        P21=1;P23=0;P25=0;
        break;
    case 5:
        PWMA_CCR1H = 0;
        PWMA_CCR1L = 0;
        PWMA_CCR2H = 0;
        PWMA_CCR2L = 0;
        PWMA_CCR3H = (u8)(Aim_Pulse >> 8);
        PWMA_CCR3L = (u8)(Aim_Pulse);
        P21=1;P23=0;P25=0;
        break;
    case 1:
        PWMA_CCR1H = 0;
        PWMA_CCR1L = 0;
        PWMA_CCR2H = 0;
        PWMA_CCR2L = 0;
        PWMA_CCR3H = (u8)(Aim_Pulse >> 8);
        PWMA_CCR3L = (u8)(Aim_Pulse);
        P21=0;P23=1;P25=0;
        break;
    default : break;
}

CMP_ch++;
if(CMP_ch==3) CMP_ch=0;
CMPEXCFG &= ~0x03;
CMPEXCFG /= CMP_ch;

CMPCR1 &= ~0x40; //清比较器中断标志
CMPCR1 /= 0x30; //开比较器中断
}

u8 RD_BMF()
{
    Hall_sta = 0;
    (Hall_b)? (Hall_sta|=0x01):(Hall_sta&=~0x01);
    (Hall_a)? (Hall_sta|=0x02):(Hall_sta&=~0x02);
    (Hall_c)? (Hall_sta|=0x04):(Hall_sta&=~0x04);
    return Hall_sta;
}

void MOTOR_START()
{

```

```

u16 i,timer=120;
u8 HA = 3;
u8 Start_cnt = 0;

Start_Pulse = 0;
PWMA_BKR /= 0x80; // 主输出使能

while(1)
{
    for(i=0;i<timer; i++)
    {
        DelayXus(78); //等待
    }
    timer-= timer/25;

    switch(HA)
    {
        case 3:
            PWMA_CCR1H = (u8)(Start_Pulse >> 8);
            PWMA_CCR1L = (u8)(Start_Pulse);
            PWMA_CCR2H = 0;
            PWMA_CCR2L = 0;
            PWMA_CCR3H = 0;
            PWMA_CCR3L = 0;
            P21=0;P23=1;P25=0;
            HA = 2;
            break;

        case 2:
            PWMA_CCR1H = (u8)(Start_Pulse >> 8);
            PWMA_CCR1L = (u8)(Start_Pulse);
            PWMA_CCR2H = 0;
            PWMA_CCR2L = 0;
            PWMA_CCR3H = 0;
            PWMA_CCR3L = 0;
            P21=0;P23=0;P25=1;
            HA = 6;
            break;

        case 6:
            PWMA_CCR1H = 0;
            PWMA_CCR1L = 0;
            PWMA_CCR2H = (u8)(Start_Pulse >> 8);
            PWMA_CCR2L = (u8)(Start_Pulse);
            PWMA_CCR3H = 0;
            PWMA_CCR3L = 0;
            P21=0;P23=0;P25=1;
            HA = 4;
            break;

        case 4:
            PWMA_CCR1H = 0;
            PWMA_CCR1L = 0;
            PWMA_CCR2H = (u8)(Start_Pulse >> 8);
            PWMA_CCR2L = (u8)(Start_Pulse);
            PWMA_CCR3H = 0;
            PWMA_CCR3L = 0;
            P21=1;P23=0;P25=0;
            HA = 5;
            break;

        case 5:
            PWMA_CCR1H = 0;

```

```

        PWMA_CCR1L = 0;
        PWMA_CCR2H = 0;
        PWMA_CCR2L = 0;
        PWMA_CCR3H = (u8)(Start_Pulse >> 8);
        PWMA_CCR3L = (u8)(Start_Pulse);
        P21=1;P23=0;P25=0;
        HA = 1;
        break;
    case 1:
        PWMA_CCR1H = 0;
        PWMA_CCR1L = 0;
        PWMA_CCR2H = 0;
        PWMA_CCR2L = 0;
        PWMA_CCR3H = (u8)(Start_Pulse >> 8);
        PWMA_CCR3L = (u8)(Start_Pulse);
        P21=0;P23=1;P25=0;
        HA = 3;
        break;
    default : break;
}
if(++Start_cnt>80)
{
    BMF_en=1;
    break;
}
else
{
    if(Start_Pulse<Aim_Pulse) Start_Pulse += 4;
    else Start_Pulse = Aim_Pulse;
}
}
}

void MOTOR_STOP()
{
    BMF_en = 0;
    PWMA_BKR &= ~0x80; //关比较器中断
    P21=0;P23=0;P25=0;
}

u8 KEY_detect()
{
    if(!P05)
    {
        DelayXms(30);
        if(!P05)
        {
            while(!P05);
            return 1;
        }
    }
    return 0;
}

void DelayXus(u8 delayTime)
{
    int i = 0;
    while( delayTime-- )
    {

```

```

        for( i = 0 ; i < 1 ; i++);
    }
}
void DelayXms( u8  delayTime )
{
    int  i = 0;
    while( delayTime-- )
    {
        for( i = 0 ; i < 2 ; i++ )
        {
            DelayXus(100);
        }
    }
}

```

## 22.8.3 使用高级 PWM 实现编码器

### C 语言代码

//测试工作频率为 11.0592MHz

```
#define  MAIN_Fosc  11059200L//定义主时钟
```

```
#include  <reg51.h>
```

/\*\*\*\*\*\* 功能说明\*\*\*\*\*

PWMA 模块工作于编码器模式. PWMA 模块只能接一个编码器.  
串口 1(RXD-->P3.0 TXD-->P3.1) 返回读数结果, 串口设置 115200,8,n,1;  
编码器 A 相输入: PWM1P (P1.0)  
编码器 B 相输入: PWM2P (P1.2)

编码器模式: 模式 1: 每个脉冲两个边沿加减 2.  
模式 2: 每个脉冲两个边沿加减 2.  
模式 3: 每个脉冲两个边沿加减 4.

\*\*\*\*\*/

```

sfr      P1M1  =    0x91;
sfr      P1M0  =    0x92;
sfr      AUXR  =    0x8E;
sfr      P_SW1 =    0xA2;
sfr      P_SW2 =    0xBA;

```

```

#define  PWMA_CNTR    (*(unsigned int volatile xdata *)0xfece)
#define  PWMA_PSCR    (*(unsigned int volatile xdata *)0xfed0)
#define  PWMA_ARR     (*(unsigned int volatile xdata *)0xfed2)
#define  PWMA_CCMR1   (*(unsigned char volatile xdata *)0xfec8)
#define  PWMA_CCMR2   (*(unsigned char volatile xdata *)0xfec9)
#define  PWMA_CR1     (*(unsigned char volatile xdata *)0xfec0)
#define  PWMA_SMCR     (*(unsigned char volatile xdata *)0xfec2)
#define  PWMA_ENO     (*(unsigned char volatile xdata *)0xfeb1)
#define  PWMA_PS      (*(unsigned char volatile xdata *)0xfeb2)
#define  PWMA_CCER1   (*(unsigned char volatile xdata *)0xfec3)
#define  PWMA_IER     (*(unsigned char volatile xdata *)0xfec4)
#define  PWMA_SRI     (*(unsigned char volatile xdata *)0xfec5)

```



```

unsigned int    pulse;                //编码器脉冲
bit            B_Change;             //编码器计数改变

bit            B_TX1_Busy;           // 发送忙标志

void            PWMA_config(void);
void            UART1_config(unsigned long brt);    // brt: 通信波特率
void            UART1_TxByte(unsigned char dat);

void main(void)
{
    unsigned int j;

    P_SW2 /= 0x80;                    //使能XFR 访问

    P1M1 = 0x00;
    P1M0 = 0x00;

    UART1_config(115200UL);           // brt: 通信波特率

    EA = 1;

    PWMA_config();
    pulse = 10;

    while (1)
    {
        if(B_Change)
        {
            B_Change = 0;
            j = pulse;
            UART1_TxByte(j/10000+'0');    //转成十进制文本并发送
            UART1_TxByte((j%10000)/1000+'0');
            UART1_TxByte((j%1000)/100+'0');
            UART1_TxByte((j%100)/10+'0');
            UART1_TxByte(j%10+'0');
            UART1_TxByte(0x0d);
            UART1_TxByte(0x0a);
        }
    }
}

//=====
// 函数: void PWMA_config(void)
// 描述: PPWM 配置函数。
// 参数: noe.
// 返回: none.
// 版本: V1.0, 2021-5-10
// 备注:
//=====
void PWMA_config(void)
{
    PWMA_PSCR = 0;                    //预分频寄存器,分频Fck_cnt = Fck_psc/(PSCR[15:0]+1),
                                     //边沿对齐PWM 频率 =SYSclk/((PSCR+1)*(ARR+1)),
                                     //中央对齐PWM 频率 =SYSclk/((PSCR+1)*(ARR+1)*2).
    PWMA_ARR = 0xffff;               //自动重装载寄存器,控制PWM 周期
}

```

```

    PWMA_CNTR = 0;
    PWMA_ENO = 0;

    PWMA_CCMR1 = 0x01+(10<<4);

    PWMA_CCMR2 = 0x01+(10<<4);

    PWMA_SMCR = 2;

    PWMA_CCER1 = 0x55;
    PWMA_PS = 0;
    PWMA_IER = 0x02;
    PWMA_CR1 = 0x01;

}

//=====
// 函数: void PWMA_ISR(void) interrupt PWMA_VECTOR
// 描述: PWMA 中断处理程序
// 参数: None
// 返回: none.
// 版本: V1.0, 2021-6-1
//=====
void PWMA_ISR(void) interrupt 26
{
    if(PWMA_SRI & 0x02)
    {
        pulse = PWMA_CNTR;
        B_Change = 1;
    }
    PWMA_SRI = 0;
}

//=====
// 函数: void          UART1_config(u32 brt)
// 描述: UART1 初始化函数。
// 参数: brt:          通信波特率
// 返回: none.
// 版本: VER1.0
// 日期: 2018-4-2
// 备注:
//=====
void UART1_config(unsigned long brt)
{
    Brt = 65536UL - (MAIN_Fosc / 4) / brt;
    AUXR &= ~0x01;
    AUXR |= (1<<6);
    TMOD &= 0x0f;
    TH1 = (unsigned char)(brt >> 8);
    TL1 = (unsigned char)brt;
    TR1 = 1;
    P_SW1 &= ~0xc0;
    SCON = (SCON & 0x3f) / (1<<6);
    ES = 1;
    REN = 1;
}

```

//清零编码器计数器值  
//IO 禁止输出 PWM  
//通道1 模式配置, 配置成输入通道  
//0~15 对应输入滤波时钟数:  
//1 2 4 8 12 16 24 32 48 64 80 96 128 160 192 256  
//通道2 模式配置, 配置成输入通道  
//0~15 对应输入滤波时钟数:  
//1 2 4 8 12 16 24 32 48 64 80 96 128 160 192 256  
//编码器模式: 模式1 或模式2: 每个脉冲两个边沿加减2.  
//模式3: 每个脉冲四个边沿加减4.  
//配置通道输入使能和极性, 允许输入, 下降沿  
//IO 选择 P1.0 P1.2  
//使能中断  
//使能计数器, 允许自动重载寄存器缓冲,  
//边沿对齐模式, 向上计数,  
//bit7=1: 写自动重载寄存器缓冲(本周期不会被打扰),  
// =0: 直接写自动重载寄存器本(周期可能会乱掉)

//编码器中断  
//读取当前编码器计数值  
//标志已有捕捉值

// brt: 通信波特率  
//S1 BRT Use Timer1;  
//Timer1 set as 1T mode  
//Timer1 16bits AutoReload;  
// 运行 Timer1  
//串口1 切换到 P3.0 P3.1  
// 8 位数据, 1 位起始位, 1 位停止位, 无校验  
//允许中断  
//允许接收

```
}

//=====
// 函数: void UART1_TxByte(u8 dat)
// 描述: 串口1 查询发送一个字节函数
// 参数: dat: 要发送的字节数据
// 返回: none.
// 版本: VER1.0
// 日期: 2018-4-2
// 备注:
//=====
void UART1_TxByte(unsigned char dat)
{
    B_TX1_Busy = 1;                //标志发送忙
    SBUF = dat;                    //发一个字节
    while(B_TX1_Busy);             //等待发送完成
}

//=====
// 函数: void UART1_int (void) interrupt UART1_VECTOR
// 描述: 串口1 中断函数
// 参数: none.
// 返回: none.
// 版本: VER1.0
// 日期: 2018-4-2
// 备注:
//=====
void UART1_int (void) interrupt 4
{
    if(RI)
        RI = 0;

    if(TI)
    {
        TI = 0;
        B_TX1_Busy = 0;
    }
}
```

## 22.8.4 正交编码器模式

### C 语言代码

//测试工作频率为11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
unsigned char cnt_H, cnt_L;
```

```
void main(void)
```

```
{
```

```
    P_SW2 |= 0x80;
```

//使能XFR 访问

```
    PIM1 = 0x0f;
```

```
    PIM0 = 0x00;
```

```
    PWMA_ENO = 0x00;
```

//配置成TRGI 的pin 需关掉ENO 对应bit 并配成input

```

    PWMA_PS = 0x00;                                //00:PWM at P1

    PWMA_PSCRH = 0x00;                               //预分频寄存器
    PWMA_PSCRL = 0x00;

    PWMA_CCMR1 = 0x21;                               //通道模式配置为输入, 接编码器, 滤波器4 时钟
    PWMA_CCMR2 = 0x21;                               //通道模式配置为输入, 接编码器, 滤波器4 时钟

    PWMA_SMCR = 0x03;                               //编码器模式3

    PWMA_CCER1 = 0x55;                               //配置通道使能和极性
    PWMA_CCER2 = 0x55;                               //配置通道使能和极性

    PWMA_IER = 0x02;                                 //使能中断

    PWMA_CR1 /= 0x01;                                //使能计数器

    EA = 1;

    while (1);
}

/***** PWM 中断读编码器计数值 *****/
void PWMA_ISR() interrupt 26
{
    if (PWMA_SRI & 0X02)
    {
        P03 = ~P03;
        cnt_H = PWMA_CCR1H;
        cnt_L = PWMA_CCR1L;
        PWMA_SRI &= ~0X02;
    }
}

```

## 22.8.5 单脉冲模式（触发控制脉冲输出）

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

void main(void)
{
    P_SW2 /= 0x80;                                //使能XFR 访问

    P0M1 = 0x00;
    P0M0 = 0xFF;
    P1M1 = 0x0c;
    P1M0 = 0xF3;

    PWMA_ENO = 0xF3;                               //IO 输出PWM
    PWMA_PS = 0x00;                                //00:PWM at P1

    /***** PWMx_duty = [CCRx/(ARR + 1)]*100 *****/
    PWMx_duty = [CCRx/(ARR + 1)]*100
    *****/

```

```

//配置成TRGI 的 pin 需关掉 ENO 对应 bit 并配成 input
PWMA_PSCRH = 0x00;           //预分频寄存器
PWMA_PSCRL = 0x00;
PWMA_DTR = 0x00;           //死区时间配置

PWMA_CCMR1 = 0x68;          //通道模式配置
PWMA_CCMR2 = 0x01;          //配置成输入通道
PWMA_CCMR3 = 0x68;
PWMA_CCMR4 = 0x68;

PWMA_SMCR = 0x66;

PWMA_ARRH = 0x08;          //自动重载寄存器, 计数器 overflow 点
PWMA_ARRL = 0x00;

PWMA_CCR1H = 0x04;         //计数器比较值
PWMA_CCR1L = 0x00;
PWMA_CCR2H = 0x02;
PWMA_CCR2L = 0x00;
PWMA_CCR3H = 0x01;
PWMA_CCR3L = 0x00;
PWMA_CCR4H = 0x01;
PWMA_CCR4L = 0x00;

PWMA_CCER1 = 0x55;         //配置通道输出使能和极性
PWMA_CCER2 = 0x55;         //配置通道输出使能和极性

PWMA_BKR = 0x80;          //主输出使能 相当于总开关
PWMA_IER = 0x02;          //使能中断
PWMA_CR1 = 0x08;          //单脉冲模式
PWMA_CR1 |= 0x01;         //使能计数器

EA = 1;
while (1);
}

void PWMA_ISR() interrupt 26
{
    if (PWMA_SRI & 0X02)
    {
        P03 = ~P03;
        PWMA_SRI &= ~0X02;
    }
}

```

## 22.8.6 门控模式（输入电平使能计数器）

### C 语言代码

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"
#include "intrins.h"

void main(void)
{
    P_SW2 |= 0x80;          //使能 XFR 访问

    P0M1 = 0x00;

```

```

P0M0 = 0xFF;
P1M1 = 0x00;
P1M0 = 0xFF;
P3M1 = 0x04;
P3M0 = 0x00;

PWMA_ENO = 0xFF;           //IO 输出PWM
PWMA_PS = 0x00;           //00:PWM at P1

/*****
PWMx_duty = [CCRx/(ARR + 1)]*100
*****/
//配置成TRGI 的pin 需关掉ENO 对应bit 并配成input
PWMA_PSCRH = 0x00;         //预分频寄存器
PWMA_PSCRL = 0x00;
PWMA_DTR = 0x00;          //死区时间配置

PWMA_CCMR1 = 0x68;        //通道模式配置
PWMA_CCMR2 = 0x68;        //配置成输入通道
PWMA_CCMR3 = 0x68;
PWMA_CCMR4 = 0x68;

PWMA_SMCR = 0x75;         //门控触发模式ETRF 输入

PWMA_ARRH = 0x08;         //自动重载寄存器, 计数器overflow 点
PWMA_ARRL = 0x00;

PWMA_CCR1H = 0x04;        //计数器比较值
PWMA_CCR1L = 0x00;        //
PWMA_CCR2H = 0x02;        //
PWMA_CCR2L = 0x00;        //
PWMA_CCR3H = 0x01;        //
PWMA_CCR3L = 0x00;        //
PWMA_CCR4H = 0x01;        //
PWMA_CCR4L = 0x00;        //

PWMA_CCER1 = 0x55;        //配置通道输出使能和极性
PWMA_CCER2 = 0x55;        //配置通道输出使能和极性

PWMA_BKR = 0x80;          //主输出使能 相当于总开关
PWMA_IER = 0x02;          //使能中断

PWMA_CR1 /= 0x01;         //使能计数器

EA = 1;
while (1);
}

void PWMA_ISR() interrupt 26
{
    if(PWMA_SRI & 0X02)
    {
        P03 = ~P03;
        PWMA_SRI &= ~0X02;
    }
}

```

## 22.8.7 外部时钟模式

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void main(void)
```

```
{
```

```
    P_SW2 |= 0x80; //使能 XFR 访问
```

```
    P0M1 = 0x00;
```

```
    P0M0 = 0xFF;
```

```
    P1M1 = 0x00;
```

```
    P1M0 = 0xFF;
```

```
    P3M1 = 0x04;
```

```
    P3M0 = 0x00;
```

```
    PWMA_ENO = 0xFF;
```

```
//IO 输出 PWM
```

```
    PWMA_PS = 0x00;
```

```
//00: PWM at P1
```

```
/******
```

```
PWMx_duty = [CCRx/(ARR + 1)]*100
```

```
*****/
```

```
//配置成 TRGI 的 pin 需关掉 ENO 对应 bit 并配成 input
```

```
    PWMA_PSCRH = 0x00;
```

```
//预分频寄存器
```

```
    PWMA_PSCRL = 0x00;
```

```
    PWMA_DTR = 0x00;
```

```
//死区时间配置
```

```
    PWMA_CCMR1 = 0x68;
```

```
//通道模式配置
```

```
    PWMA_CCMR2 = 0x68;
```

```
//配置成输入通道
```

```
    PWMA_CCMR3 = 0x68;
```

```
    PWMA_CCMR4 = 0x68;
```

```
    PWMA_SMCR = 0x77;
```

```
//ETRF 输入
```

```
    PWMA_ARRH = 0x08;
```

```
//自动重载寄存器, 计数器 overflow 点
```

```
    PWMA_ARRL = 0x00;
```

```
    PWMA_CCR1H = 0x04;
```

```
//计数器比较值
```

```
    PWMA_CCR1L = 0x00;
```

```
    PWMA_CCR2H = 0x02;
```

```
    PWMA_CCR2L = 0x00;
```

```
    PWMA_CCR3H = 0x01;
```

```
    PWMA_CCR3L = 0x00;
```

```
    PWMA_CCR4H = 0x01;
```

```
    PWMA_CCR4L = 0x00;
```

```
    PWMA_CCER1 = 0x55;
```

```
//配置通道输出使能和极性
```

```
    PWMA_CCER2 = 0x55;
```

```
//配置通道输出使能和极性
```

```
    PWMA_BKR = 0x80;
```

```
//主输出使能 相当于总开关
```

```
    PWMA_IER = 0x02;
```

```
//使能中断
```

```
    PWMA_CR1 |= 0x01;
```

```
//使能计数器
```

```
    EA = 1;
```

```
    while (1);
```

```
}  
  
void PWMA_ISR() interrupt 26  
{  
    if(PWMA_SRI & 0X02)  
    {  
        P03 = ~P03;  
        PWMA_SRI &=~0X02;  
    }  
}
```

---

STC MCU



## 22.8.8 输入捕获模式测量脉冲周期（捕获上升沿到上升沿或者下降沿到下降沿）

原理：使用高级 PWM 内部的某一通道的捕获模块 CCx，捕获外部的端口的上升沿或者下降沿，两个上升沿之间或者两个下降沿之间的时间即为脉冲的周期，也就是说，两次捕获计数值的差值即为周期值。

范例：使用 PWMA 的第一组捕获模块 CC1 捕获功能，捕获 PWM1P（P1.0）管脚上的上升沿，在中断中对前后两次的捕获值相减得到周期

注意：只有 PWM1P、PWM2P、PWM3P、PWM4P、PWM5、PWM6、PWM7、PWM8 这些管脚以及相应切换管脚才有捕获功能

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
int      cap;
int      cap_new;
int      cap_old;
```

```
void main(void)
```

```
{
```

```
    P_SW2 |= 0x80;
```

//使能 XFR 访问

```
    P0M1 = 0x00;
```

```
    P0M0 = 0xFF;
```

```
    P1M1 = 0x0c;
```

```
    P1M0 = 0xF3;
```

```
    PWMA_ENO = 0xF3;
```

//IO 输出 PWM

```
    PWMA_PS = 0x00;
```

//00: PWM at P1

/\*配置成 TRGI 的 pin 需关掉 ENO 对应 bit 并配成 input\*/

```
    PWMA_PSCRH = 0x00;
```

//预分频寄存器

```
    PWMA_PSCRL = 0x00;
```

```
    PWMA_DTR = 0x00;
```

//死区时间配置

```
    PWMA_CCMR1 = 0x01;
```

//通道模式配置

```
    PWMA_CCMR2 = 0x68;
```

```
    PWMA_CCMR3 = 0x68;
```

```
    PWMA_CCMR4 = 0x68;
```

```
    PWMA_SMCR = 0x66;
```

```
    PWMA_CCER1 = 0x55;
```

//配置通道输出使能和极性

```
    PWMA_CCER2 = 0x55;
```

//配置通道输出使能和极性

```
    PWMA_IER = 0x04;
```

//使能中断

```
    PWMA_CR1 |= 0x01;
```

//使能计数器

```
    EA = 1;
```

```
    while (1);  
}  
  
void PWMA_ISR() interrupt 26  
{  
    if(PWMA_SRI & 0X02)  
    {  
        P03 = ~P03;  
        PWMA_SRI &=~0X02;  
    }  
    if(PWMA_SRI & 0X04)  
    {  
        cap_old = cap_new;  
        cap_new = PWMA_CCRI;           //读取CCRI  
        cap = cap_new - cap_old;  
        PWMA_SRI &= ~0x02;  
    }  
}
```

---

STC MCU

## 22.8.9 输入捕获模式测量脉冲高电平宽度（捕获上升沿到下降沿）

原理：使用高级 PWM 内部的两通道的捕获模块 CCx 和 CCx+1 同时捕获外部的同一个管脚，CCx 捕获此管脚的上升沿，CCx+1 捕获此管脚的下降沿，然利用 CCx+1 的捕获值减去 CCx 的捕获值，其差值即为脉冲高电平的宽度。

范例：使用 PWMA 的第一组捕获模块 CC1 和第二组捕获模块 CC2，同时捕获 PWM1P 管脚（P1.0），其中 CC1 捕获 PWM1P 的上升沿，CC2 捕获 PWM1P 的下降沿，在中断中使用 CC2 的捕获值减去 CC1 的捕获值，其差值即为脉冲高电平的宽度。

注意：1、使用的是芯片内部的两路捕获模块同时捕获外部的同一个管脚，所以不需要将外部的多个管脚相连接。

2、只有 CC1+CC2、CC3+CC4、CC5+CC6、CC7+CC8 这 4 种组合才能完成上面的功能。CC1+CC2 组合可以同时捕获 PWM1P 管脚，也可以同时捕获 PWM2P 管脚；CC3+CC4 组合可以同时捕获 PWM3P 管脚，也可以同时捕获 PWM4P 管脚；CC5+CC6 组合可以同时捕获 PWM5 管脚，也可以同时捕获 PWM6 管脚；CC7+CC8 组合可以同时捕获 PWM7 管脚，也可以同时捕获 PWM8 管脚

### C 语言代码

```
//测试工作频率为 11.0592MHz
#include "stc8h.h"
#include "intrins.h"

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P1M0 = 0x00;
    P1M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    PWMA_CCER1 = 0x00; // (CC1 捕获 T1I 上升沿, CC2 捕获 T1I 下降沿)
    PWMA_CCMR1 = 0x01; // CC1 为输入模式, 且映射到 T1IFP1 上
    PWMA_CCMR2 = 0x02; // CC2 为输入模式, 且映射到 T1IFP2 上
    PWMA_CCER1 = 0x11; // 使能 CC1/CC2 上的捕获功能
    PWMA_CCER1 /= 0x00; // 设置捕获极性为 CC1 的上升沿
    PWMA_CCER1 = 0x20; // 设置捕获极性为 CC2 的下降沿
    PWMA_CR1 = 0x01;

    PWMA_IER = 0x04; // 使能 CC2 捕获中断
    EA = 1;

    while (1);
}

void PWMA_ISR() interrupt 26
{
    unsigned int cnt;

    if (PWMA_SRI & 0x04)
    {
        PWMA_SRI &= ~0x04;
    }
}
```

```
        cnt = PWMA_CCR2 - PWMA_CCRI;           //差值即为高电平宽度
    }
}
```

---

STC MCU

## 22.8.10 输入捕获模式测量脉冲低电平宽度（捕获下降沿到上升沿）

原理：使用高级 PWM 内部的两通道的捕获模块 CCx 和 CCx+1 同时捕获外部的同一个管脚，CCx 捕获此管脚的下降沿，CCx+1 捕获此管脚的上升沿，然利用 CCx+1 的捕获值减去 CCx 的捕获值，其差值即为脉冲低电平的宽度。

范例：使用 PWMA 的第一组捕获模块 CC1 和第二组捕获模块 CC2，同时捕获 PWM1P 管脚（P1.0），其中 CC1 捕获 PWM1P 的下降沿，CC2 捕获 PWM1P 的上升沿，在中断中使用 CC2 的捕获值减去 CC1 的捕获值，其差值即为脉冲低电平的宽度。

注意：1、使用的是芯片内部的两路捕获模块同时捕获外部的同一个管脚，所以不需要将外部的多个管脚相连接。

2、只有 CC1+CC2、CC3+CC4、CC5+CC6、CC7+CC8 这 4 种组合才能完成上面的功能。CC1+CC2 组合可以同时捕获 PWM1P 管脚，也可以同时捕获 PWM2P 管脚；CC3+CC4 组合可以同时捕获 PWM3P 管脚，也可以同时捕获 PWM4P 管脚；CC5+CC6 组合可以同时捕获 PWM5 管脚，也可以同时捕获 PWM6 管脚；CC7+CC8 组合可以同时捕获 PWM7 管脚，也可以同时捕获 PWM8 管脚

### C 语言代码

```
//测试工作频率为 11.0592MHz
#include "stc8h.h"
#include "intrins.h"

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P1M0 = 0x00;
    P1M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    PWMA_CCER1 = 0x00; // (CC1 捕获 T1I 上升沿, CC2 捕获 T1I 下降沿)
    PWMA_CCMR1 = 0x01; // CC1 为输入模式, 且映射到 TIIFP1 上
    PWMA_CCMR2 = 0x02; // CC2 为输入模式, 且映射到 TIIFP2 上
    PWMA_CCER1 = 0x11; // 使能 CC1/CC2 上的捕获功能
    PWMA_CCER1 /= 0x00; // 设置捕获极性为 CC1 的上升沿
    PWMA_CCER1 = 0x20; // 设置捕获极性为 CC2 的下降沿
    PWMA_CR1 = 0x01;

    PWMA_IER = 0x02; // 使能 CC1 捕获中断
    EA = 1;

    while (1);
}

void PWMA_ISR() interrupt 26
{
    unsigned int cnt;

    if (PWMA_SRI & 0x02)
    {
        PWMA_SRI &= ~0x02;
    }
}
```

```
        cnt = PWMA_CCRI - PWMA_CCR2;           //差值即为低电平宽度
    }
}
```

---

STC MCU

## 22.8.11 输入捕获模式同时测量脉冲周期和占空比

原理: 使用高级 PWM 内部的两通道的捕获模块 CCx 和 CCx+1 同时捕获外部的同一个管脚, CCx 捕获此管脚的上升沿, CCx+1 捕获此管脚的下降沿, 同时使能此管脚的上升沿信号为复位触发信号, CCx 的捕获值即为周期, CCx+1 的捕获值即为占空比。

范例: 使用 PWMA 的第一组捕获模块 CC1 和第二组捕获模块 CC2, 同时捕获 PWM1P 管脚 (P1.0), 其中 CC1 捕获 PWM1P 的上升沿, CC2 捕获 PWM1P 的下降沿, 并设置 PWM1P 的上升沿信号为复位触发信号, CC1 的捕获值即为周期, CC2 的捕获值即为占空比。

注意: 1、使用的是芯片内部的两路捕获模块同时捕获外部的同一个管脚, 所以不需要将外部的多个管脚相连接。

2、只有 CC1+CC2、CC5+CC6 这两种组合才能完成上面的功能。CC1+CC2 组合可以同时捕获 PWM1P 管脚, 也可以同时捕获 PWM2P 管脚; CC5+CC6 组合可以同时捕获 PWM5 管脚, 也可以同时捕获 PWM6 管脚

3、由于设置了复位触发信号, 所以捕获值即为周期值和占空比值, 无需再减前一次的捕获值。

### C 语言代码

```
//测试工作频率为 11.0592MHz
```

```
#include "stc8h.h"
```

```
#include "intrins.h"
```

```
void main()
```

```
{
```

```
    P_SW2 /= 0x80;
```

```
//使能访问 XFR
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
    PWMA_CCER1 = 0x00;
```

```
    PWMA_CCMR1 = 0x01;
```

```
    PWMA_CCMR2 = 0x02;
```

```
    PWMA_CCER1 = 0x11;
```

```
    PWMA_CCER1 /= 0x00;
```

```
    PWMA_CCER1 /= 0x20;
```

```
    PWMA_SMCR = 0x54;
```

```
    PWMA_CR1 = 0x01;
```

```
//(CC1 捕获 T11 上升沿, CC2 捕获 T11 下降沿)
```

```
//CC1 捕获周期宽度, CC2 捕获高电平宽度
```

```
//CC1 为输入模式, 且映射到 TI1FP1 上
```

```
//CC2 为输入模式, 且映射到 TI1FP2 上
```

```
//使能 CC1/CC2 上的捕获功能
```

```
//设置捕获极性为 CC1 的上升沿
```

```
//设置捕获极性为 CC2 的下降沿
```

```
//TS=TI1FP1, SMS=TI1 上升沿复位模式
```

```
    PWMA_IER = 0x06;
```

```
    EA = 1;
```

```
//使能 CC1/CC2 捕获中断
```

```
    while (1);
```

```
}
```

```
void PWMA_ISR() interrupt 26
```

```
{
```

```
    unsigned int cnt;
```

```
    if (PWMA_SRI & 0x02)
```

```
    {
```

```
        PWMA_SRI &= ~0x02;
```

```
        cnt = PWMA_CCR1;                //CC1 捕获周期宽度
    }
    if (PWMA_SRI & 0x04)
    {
        PWMA_SRI &= ~0x04;

        cnt = PWMA_CCR2;                //CC2 捕获占空比 (高电平宽度)
    }
}
```

---

STC MCU



## 22.8.12 同时捕获 4 路输入信号的周期和占空比

原理: 使用高级 PWM 内部的两通道的捕获模块 CCx 和 CCx+1 同时捕获外部的同一个管脚, CCx 捕获此管脚的上升沿, CCx+1 捕获此管脚的下降沿, CCx 的两次捕获值的差值即为周期, CCx+1 的捕获值与 CCx 的前一次捕获值的差值即为占空比。

范例: 使用 PWMA 的第一组捕获模块 CC1 和第二组捕获模块 CC2, 同时捕获 PWM1P 管脚 (P1.0), 其中 CC1 捕获 PWM1P 的上升沿, CC2 捕获 PWM1P 的下降沿, CC1 的捕获值减去前一次捕获值即为周期, CC2 的捕获值减去 CC1 的前一次捕获值即为占空比。PWMB 的 CC5 和 CC6 同时捕获 PWM5 (P2.0)、PWMB 的 CC7 和 CC8 同时捕获 PWM7 (P2.2)、PWMA 的 CC3 和 CC4 同时捕获 PWM3P (P1.4)。另外使用定时器 0 在 P1.0 上产生波形、定时器 1 在 P1.4 上产生波形、定时器 3 在 P2.0 上产生波形、定时器 4 在 P2.2 上产生波形。捕获值通过串口送到 PC。

注意: 1、使用的是芯片内部的两路捕获模块同时捕获外部的同一个管脚, 所以不需要将外部的多个管脚相连接。

2、由于没有设置复位触发信号, 所以周期值和占空比值均需要作相应的减法运算才能得到。

---

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
#include "stdio.h"
```

```
#define FOSC 12000000UL
#define BRT (65536 - FOSC / 115200 / 4)
#define T10K (65536 - FOSC / 10000)
#define T11K (65536 - FOSC / 11000)
#define T12K (65536 - FOSC / 12000)
#define T13K (65536 - FOSC / 13000)
```

```
unsigned int ccr1;
unsigned int ccr3;
unsigned int ccr5;
unsigned int ccr7;
```

```
unsigned int cycle1;
unsigned int duty1;
unsigned int cycle2;
unsigned int duty2;
unsigned int cycle3;
unsigned int duty3;
unsigned int cycle4;
unsigned int duty4;
```

```
bit f1, f2, f3, f4;
```

```
void main()
{
```

```
    P_SW2 /= 0x80;
```

```
    //使能访问 XFR
```

```
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
```

```

P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

AUXR /= 0x80;
AUXR /= 0x40;
TMOD = 0x00;
TL0 = T10K;
TH0 = T10K >> 8;
TL1 = T11K;
TH1 = T11K >> 8;
TR0 = 1;
TR1 = 1;
ET0 = 1;
ET1 = 1;

T3L = T12K;
T3H = T12K >> 8;
T4L = T13K;
T4H = T13K >> 8;
T4T3M = 0xaa;
IE2 /= 0x20;
IE2 /= 0x40;

SCON = 0x52;
T2L = BRT;
T2H = BRT >> 8;
AUXR /= 0x15;

printf("PWM Test .\n");

PWMA_CCER1 = 0x00;
PWMA_CCMR1 = 0x01;
PWMA_CCMR2 = 0x02;
PWMA_CCER1 = 0x11;
PWMA_CCER1 /= 0x00;
PWMA_CCER1 /= 0x20;

PWMA_CCER2 = 0x00;
PWMA_CCMR3 = 0x01;
PWMA_CCMR4 = 0x02;
PWMA_CCER2 = 0x11;
PWMA_CCER2 /= 0x00;
PWMA_CCER2 /= 0x20;
PWMA_CR1 = 0x01;

PWMA_IER = 0x1e;

PWMB_CCER1 = 0x00;
PWMB_CCMR1 = 0x01;
PWMB_CCMR2 = 0x02;
PWMB_CCER1 = 0x11;
PWMB_CCER1 /= 0x00;
PWMB_CCER1 /= 0x20;

PWMB_CCER2 = 0x00;
PWMB_CCMR3 = 0x01;

```

//定时器0 使用1T 模式  
//定时器1 使用1T 模式  
//定时器0/1 使用16 位自动重载模式  
//定时器0 周期10K  
//定时器1 周期11K  
//定时器0 开始计数  
//定时器1 开始计数  
//使能定时器0 中断  
//使能定时器1 中断  
//定时器3 周期12K  
//定时器4 周期13K  
//定时器3/4 使用1T 模式  
//使能定时器3 中断  
//使能定时器4 中断  
//CC1 为输入模式,且映射到/TI1FP1 上  
//CC2 为输入模式,且映射到/TI1FP2 上  
//使能CC1 上的捕获功能,使能CC2 上的捕获功能  
//设置捕获极性为CC1 的上升沿  
//设置捕获极性为CC2 的下降沿  
//CC3 为输入模式,且映射到/TI3FP3 上  
//CC4 为输入模式,且映射到/TI3FP4 上  
//使能CC3 上的捕获功能,使能CC4 上的捕获功能  
//设置捕获极性为CC3 的上升沿  
//设置捕获极性为CC4 的下降沿  
//使能CC1/CC2/CC3/CC4 捕获中断  
//CC5 为输入模式,且映射到/TI5FP5 上  
//CC6 为输入模式,且映射到/TI5FP6 上  
//使能CC5 上的捕获功能,使能CC6 上的捕获功能  
//设置捕获极性为CC5 的上升沿  
//设置捕获极性为CC6 的下降沿  
//CC7 为输入模式,且映射到/TI7FP8 上

```

PWMB_CCMR4 = 0x02;           //CC8 为输入模式,且映射到TI7FP8 上
PWMB_CCER2 = 0x11;           //使能CC7 上的捕获功能,使能CC8 上的捕获功能
PWMB_CCER2 |= 0x00;           //设置捕获极性为CC7 的上升沿
PWMB_CCER2 |= 0x20;           //设置捕获极性为CC8 的下降沿
PWMB_CR1 = 0x01;

PWMB_IER = 0x1e;              //使能CC5/CC6/CC7/CC8 捕获中断

EA = 1;

while (1)
{
    if (f1)
    {
        f1 = 0;
        printf("cycle1 = %04x duty1 = %04x\n", cycle1, duty1);
    }
    if (f2)
    {
        f2 = 0;
        printf("cycle2 = %04x duty2 = %04x\n", cycle2, duty2);
    }
    if (f3)
    {
        f3 = 0;
        printf("cycle3 = %04x duty3 = %04x\n", cycle3, duty3);
    }
    if (f4)
    {
        f4 = 0;
        printf("cycle4 = %04x duty4 = %04x\n", cycle4, duty4);
    }
}

void TMR0_ISR() interrupt TMR0_VECTOR           //产生CCI 波形到P1.0 口
{
    static unsigned int cnt = 0;

    cnt++;
    if (cnt == 10)
    {
        P10 = 0;
    }
    else if (cnt == 30)
    {
        P10 = 1;
        cnt = 0;
    }
}

void TMRI_ISR() interrupt TMRI_VECTOR           //产生CC3 波形到P1.4 口
{
    static unsigned int cnt = 0;

    cnt++;
    if (cnt == 10)
    {
        P14 = 0;
    }
}

```

```

    }
    else if (cnt == 30)
    {
        P14 = 1;
        cnt = 0;
    }
}

```

**void TMR3\_ISR() interrupt TMR3\_VECTOR** //产生 CC5 波形到 P2.0 口

```

{
    static unsigned int cnt = 0;

    cnt++;
    if (cnt == 10)
    {
        P20 = 0;
    }
    else if (cnt == 30)
    {
        P20 = 1;
        cnt = 0;
    }
}

```

**void TMR4\_ISR() interrupt TMR4\_VECTOR** //产生 CC7 波形到 P2.2 口

```

{
    static unsigned int cnt = 0;

    cnt++;
    if (cnt == 10)
    {
        P22 = 0;
    }
    else if (cnt == 30)
    {
        P22 = 1;
        cnt = 0;
    }
}

```

**void PWMA\_ISR() interrupt PWMA\_VECTOR**

```

{
    unsigned int ccr;

    if (PWMA_SRI & 0x02) //CC1 捕获中断
    {
        PWMA_SRI &= ~0x02;

        ccr = (PWMA_CCR1H << 8) + PWMA_CCR1L; //读取捕获值
        cycle1 = ccr - ccr1; //计算周期
        ccr1 = ccr; //保存当前捕获值
        f1 = 1; //波形 1 的周期和占空比捕获完成，触发串口发送
    }
    if (PWMA_SRI & 0x04) //CC2 捕获中断
    {
        PWMA_SRI &= ~0x04;

        ccr = (PWMA_CCR2H << 8) + PWMA_CCR2L; //读取捕获值
        duty1 = ccr - ccr1; //计算占空比
    }
}

```

```

}

if (PWMA_SRI & 0x08)                                     //CC3 捕获中断
{
    PWMA_SRI &= ~0x08;

    ccr = (PWMA_CCR3H << 8) + PWMA_CCR3L;               //读取捕获值
    cycle2 = ccr - ccr3;                                 //计算周期
    ccr3 = ccr;                                          //保存当前捕获值
    f2 = 1;                                              //波形2 的周期和占空比捕获完成, 触发串口发送
}
if (PWMA_SRI & 0x10)                                     //CC4 捕获中断
{
    PWMA_SRI &= ~0x10;

    ccr = (PWMA_CCR4H << 8) + PWMA_CCR4L;               //读取捕获值
    duty2 = ccr - ccr3;                                 //计算占空比
}
}

void PWMB_ISR() interrupt PWMB_VECTOR
{
    unsigned int ccr;

    if (PWMB_SRI & 0x02)                                   //CC5 捕获中断
    {
        PWMB_SRI &= ~0x02;

        ccr = (PWMB_CCR1H << 8) + PWMB_CCR1L;           //读取捕获值
        cycle3 = ccr - ccr5;                             //计算周期
        ccr5 = ccr;                                       //保存当前捕获值
        f3 = 1;                                           //波形3 的周期和占空比捕获完成, 触发串口发送
    }
    if (PWMB_SRI & 0x04)                                   //CC6 捕获中断
    {
        PWMB_SRI &= ~0x04;

        ccr = (PWMB_CCR2H << 8) + PWMB_CCR2L;           //读取捕获值
        duty3 = ccr - ccr5;                               //计算占空比
    }

    if (PWMB_SRI & 0x08)                                   //CC7 捕获中断
    {
        PWMB_SRI &= ~0x08;

        ccr = (PWMB_CCR3H << 8) + PWMB_CCR3L;           //读取捕获值
        cycle4 = ccr - ccr7;                             //计算周期
        ccr7 = ccr;                                       //保存当前捕获值
        f4 = 1;                                           //波形4 的周期和占空比捕获完成, 触发串口发送
    }
    if (PWMB_SRI & 0x10)                                   //CC8 捕获中断
    {
        PWMB_SRI &= ~0x10;

        ccr = (PWMB_CCR4H << 8) + PWMB_CCR4L;           //读取捕获值
        duty4 = ccr - ccr7;                               //计算占空比
    }
}

```

STC MCU

## 22.8.13 输出占空比为 100% 和 0% 的 PWM 波形的方方法（以 PWM1P 为例）

### 22.8.13.1 方法 1：设置 PWMx\_ENO 禁止输出 PWM

#### C 语言代码

```
//测试工作频率为 11.0592MHz
#include "stc8h.h"
#include "intrins.h"

void main()
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P1M0 = 0x00;
    P1M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    PWMA_ENO &= ~0x01;           //禁止 PWM1P 端口输出
                                //此时 PWM1P 端口为 GPIO，可通过
                                //直接操作 I/O 口寄存器强制输出高电平或者低电平

    while (1);
}
```

### 22.8.13.2 方法 2：设置 PWMx\_CCMRn 寄存器强制输出有效/无效电平

#### C 语言代码

```
//测试工作频率为 11.0592MHz
#include "stc8h.h"
#include "intrins.h"

void main()
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P1M0 = 0x00;
    P1M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    PWMA_CCER1 = 0x00;
    PWMA_CCMR1 &= ~0x03;         //CCI 为输出模式
    PWMA_CCMR1 /= 0x40;         //CCI 强制输出无效电平（占空比 0%）
}
```

```
// PWMA_CCMR1 /= 0x50;
PWMA_CCER1 /= 0x01;
PWMA_ENO = 0x01;
PWMA_BKR = 0x80;
PWMA_CR1 = 0x01;

while (1);
}
```

//CC1 强制输出有效电平 (占空比 100%)  
//使能CC1 输出, 且设置高电平为有效电平  
//使能PWM1P 端口输出  
//使能主输出  
//开始计时

## 22.8.14带死区控制的 PWM 互补输出

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void main(void)
```

```
{
    P_SW2 /= 0x80; //使能XFR 访问

    P0M1 = 0x00;
    P0M0 = 0xFF;
    P1M1 = 0x00;
    P1M0 = 0xFF;

    PWMA_ENO = 0xFF; //IO 输出PWM
    PWMA_PS = 0x00; //00:PWM at P1

    /***/
    PWMx_duty = [CCRx/(ARR + 1)]*100
    /***/

    PWMA_PSCRH = 0x00; //预分频寄存器
    PWMA_PSCRL = 0x00;
    PWMA_DTR = 0x00; //死区时间配置

    PWMA_CCMR1 = 0x68; //通道模式配置
    PWMA_CCMR2 = 0x68;
    PWMA_CCMR3 = 0x68;
    PWMA_CCMR4 = 0x68;

    PWMA_ARRH = 0x08; //自动重载寄存器, 计数器overflow 点
    PWMA_ARRL = 0x00;

    PWMA_CCR1H = 0x04; //计数器比较值
    PWMA_CCR1L = 0x00;
    PWMA_CCR2H = 0x02;
    PWMA_CCR2L = 0x00;
    PWMA_CCR3H = 0x01;
    PWMA_CCR3L = 0x00;
    PWMA_CCR4H = 0x01;
    PWMA_CCR4L = 0x00;

    PWMA_CCER1 = 0x55; //配置通道输出使能和极性
    PWMA_CCER2 = 0x55; //配置通道输出使能和极性

    PWMA_BKR = 0x80; //主输出使能 相当于总开关
}
```



```
PWMA_IER = 0x02;           //使能中断
PWMA_CR1 = 0x01;           //使能计数器

EA = 1;
while (1);
}

void PWMA_ISR() interrupt 26
{
    if(PWMA_SRI & 0X02)
    {
        P03 = ~P03;
        PWMA_SRI &=~0X02;
    }
}
```

---

## 22.8.15 PWM 端口做外部中断（下降沿中断或者上升沿中断）

### C 语言代码

---

```
//测试工作频率为 11.0592MHz

#include "stc8h.h"
#include "intrins.h"

void main(void)
{
    P_SW2 /= 0x80;           //使能 XFR 访问

    P1M1 = 0x00;
    P1M0 = 0x00;
    P3M1 = 0x00;
    P3M0 = 0x00;

    PWMA_CCER1 = 0x00;
    PWMA_CCMR1 = 0x01;
    PWMA_CCER1 = 0x01;
    PWMA_CCER1 = 0x00;
    // PWMA_CCER1 /= 0x02;
    PWMA_CR1 = 0x01;
    PWMA_IER = 0x02;
    EA = 1;

    while (1);
}

void PWMA_ISR() interrupt 26
{
    if(PWMA_SRI & 0X02)
    {
        P37 = ~P37;
        PWMA_SRI &=~0X02;
    }
}
```

---

## 22.8.16 输出任意周期和任意占空比的波形

### C 语言代码

---

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void main()
```

```
{
```

```
    P_SW2 /= 0x80;
```

```
    //使能访问 XFR
```

```
    P0M0 = 0x00;
```

```
    P0M1 = 0x00;
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P2M0 = 0x00;
```

```
    P2M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P4M0 = 0x00;
```

```
    P4M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
    PWMA_CCER1 = 0x00;
```

```
    //写 CCMRx 前必须先清零 CCERx 关闭通道
```

```
    PWMA_CCMR1 = 0x60;
```

```
    //设置 CCI 为 PWMA 输出模式
```

```
    PWMA_CCER1 = 0x01;
```

```
    //使能 CCI 通道
```

```
    PWMA_CCR1 = 100;
```

```
    //设置占空比时间
```

```
    PWMA_ARR = 500;
```

```
    //设置周期时间
```

```
    PWMA_ENO = 0x01;
```

```
    //使能 PWMIP 端口输出
```

```
    PWMA_BKR = 0x80;
```

```
    //使能主输出
```

```
    PWMA_CR1 = 0x01;
```

```
    //开始计时
```

```
    while (1);
```

```
}
```

---

## 22.8.17 使用 PWM 的 CEN 启动 PWMA 定时器，实时触发 ADC

### C 语言代码

---

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void delay()
```

```
{
```

```
    int i;
```

```
    for (i=0; i<100; i++);
```

```
}
```

```
void main()
```

```
{
```

```
P_SW2 /= 0x80; //使能访问 XFR

P1M0 = 0x00;
P1M1 = 0x01;
P3M0 = 0x00;
P3M1 = 0x00;

ADC_CONTR = ADC_POWER | ADC_EPWMT | 0; //选择 P1.0 为 ADC 输入通道
delay(); //等待 ADC 电源稳定
EADC = 1;

PWMA_CR2 = 0x10; //CEN 信号为 TRGO, 可用于触发 ADC
PWMA_ARR = 5000;
PWMA_IER = 0x01;
PWMA_CR1 = 0x01; //设置 CEN 启动 PWMA 定时器, 实时触发 ADC
EA = 1;

while (1);
}

void ADC_ISR() interrupt 5
{
    ADC_CONTR &= ~ADC_FLAG;
}

void PWMA_ISR() interrupt 26
{
    if(PWMA_SRI & 0x01)
    {
        PWMA_SRI &= ~0x01;
    }
}
```

## 22.8.18 PWM 周期重复触发 ADC

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void delay()
{
    int i;
    for (i=0; i<100; i++);
}
```

```
void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P1M0 = 0x00;
    P1M1 = 0x01;
    P3M0 = 0x00;
    P3M1 = 0x00;
```

```
    ADC_CONTR = ADC_POWER | ADC_EPWMT | 0; //选择 P1.0 为 ADC 输入通道
```

```

delay();
EADC = 1;

PWMA_CR2 = 0x20;
PWMA_ARR = 5000;
PWMA_IER = 0x01;
PWMA_CR1 = 0x01;
EA = 1;

while (1);
}

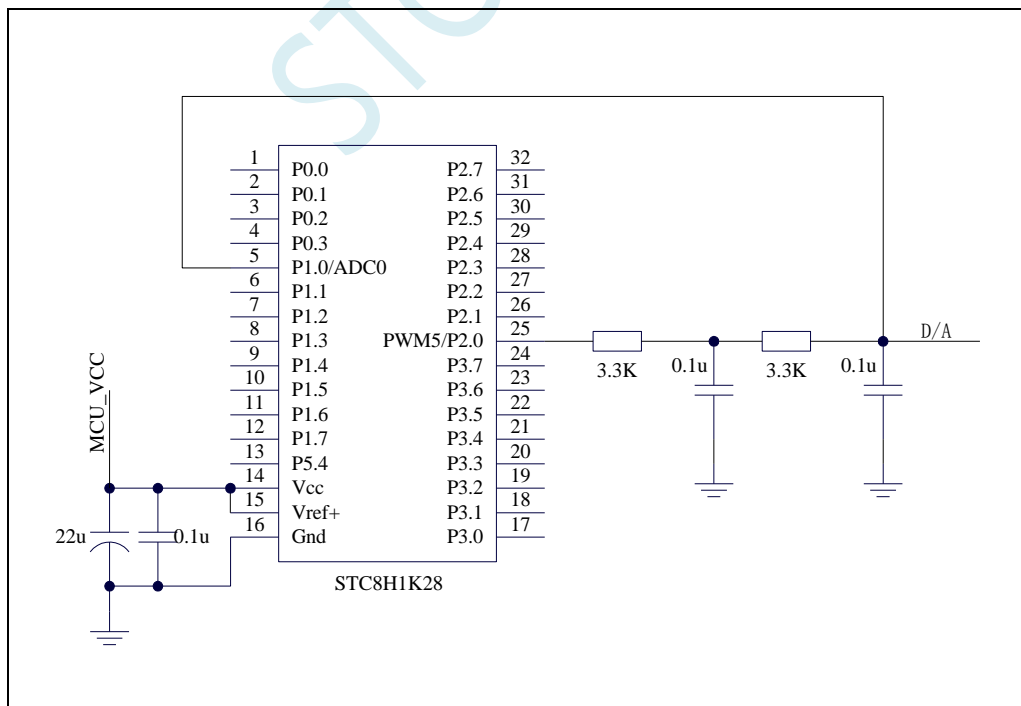
void ADC_ISR() interrupt 5
{
    ADC_CONTR &= ~ADC_FLAG;
}

void PWMA_ISR() interrupt 26
{
    if(PWMA_SRI & 0x01)
    {
        PWMA_SRI &= ~0x01;
    }
}

```

## 22.8.19 利用 PWM 实现 16 位 DAC 的参考线路图

STC8H 系列单片机的高级 PWM 定时器可输出 16 位的 PWM 波形，再经过两级低通滤波即可产生 16 位的 DAC 信号，通过调节 PWM 波形的高电平占空比即可实现 DAC 信号的改变。应用线路图如下图所示，输出的 DAC 信号可输入到 MCU 的 ADC 进行反馈测量。



## 22.8.20 利用 PWM 实现互补 SPWM

高级 PWM 定时器 PWM1P/PWM1N, PWM2P/PWM2N, PWM3P/PWM3N, PWM4P/PWM4N 每个通道

都可独立实现 PWM 输出, 或者两两互补对称输出。演示使用 PWM1P, PWM1N 产生互补的 SPWM。主时钟选择 24MHZ, PWM 时钟选择 1T, PWM 周期 2400, 死区 12 个时钟(0.5us), 正弦波表用 200 点, 输出正弦波频率 =  $24000000 / 2400 / 200 = 50 \text{ HZ}$ 。

本程序仅仅是一个 SPWM 的演示程序, 用户可以通过上面的计算方法修改 PWM 周期和正弦波的点数和幅度。本程序输出频率固定, 如果需要变频, 请用户自己设计变频方案。

## C 语言代码

//测试工作频率为24MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define MAIN_Fosc 24000000L //定义主时钟
```

```
typedef unsigned char u8;
typedef unsigned int u16;
typedef unsigned long u32;
```

/\*\*\*\*\*\* 用户定义宏 \*\*\*\*\*/

```
#define PWMA_ENO (*(unsigned char volatile xdata *) 0xFEB1)
#define PWMA_PS (*(unsigned char volatile xdata *) 0xFEB2)
#define PWMB_ENO (*(unsigned char volatile xdata *) 0xFEB5)
#define PWMB_PS (*(unsigned char volatile xdata *) 0xFEB6)

#define PWMA_CR1 (*(unsigned char volatile xdata *) 0xFEC0)
#define PWMA_CR2 (*(unsigned char volatile xdata *) 0xFEC1)
#define PWMA_SMCR (*(unsigned char volatile xdata *) 0xFEC2)
#define PWMA_ETR (*(unsigned char volatile xdata *) 0xFEC3)
#define PWMA_IER (*(unsigned char volatile xdata *) 0xFEC4)
#define PWMA_SR1 (*(unsigned char volatile xdata *) 0xFEC5)
#define PWMA_SR2 (*(unsigned char volatile xdata *) 0xFEC6)
#define PWMA_EGR (*(unsigned char volatile xdata *) 0xFEC7)
#define PWMA_CCMR1 (*(unsigned char volatile xdata *) 0xFEC8)
#define PWMA_CCMR2 (*(unsigned char volatile xdata *) 0xFEC9)
#define PWMA_CCMR3 (*(unsigned char volatile xdata *) 0xFECA)
#define PWMA_CCMR4 (*(unsigned char volatile xdata *) 0xFECB)
#define PWMA_CCER1 (*(unsigned char volatile xdata *) 0xFECC)
#define PWMA_CCER2 (*(unsigned char volatile xdata *) 0xFECD)
#define PWMA_CNTRH (*(unsigned char volatile xdata *) 0xFECE)
#define PWMA_CNTRL (*(unsigned char volatile xdata *) 0xFECF)
#define PWMA_PSCRH (*(unsigned char volatile xdata *) 0xFED0)
#define PWMA_PSCRL (*(unsigned char volatile xdata *) 0xFED1)
#define PWMA_ARRH (*(unsigned char volatile xdata *) 0xFED2)
#define PWMA_ARRL (*(unsigned char volatile xdata *) 0xFED3)
#define PWMA_RCR (*(unsigned char volatile xdata *) 0xFED4)
#define PWMA_CCR1H (*(unsigned char volatile xdata *) 0xFED5)
#define PWMA_CCR1L (*(unsigned char volatile xdata *) 0xFED6)
#define PWMA_CCR2H (*(unsigned char volatile xdata *) 0xFED7)
#define PWMA_CCR2L (*(unsigned char volatile xdata *) 0xFED8)
#define PWMA_CCR3H (*(unsigned char volatile xdata *) 0xFED9)
#define PWMA_CCR3L (*(unsigned char volatile xdata *) 0xFEDA)
#define PWMA_CCR4H (*(unsigned char volatile xdata *) 0xFEDB)
#define PWMA_CCR4L (*(unsigned char volatile xdata *) 0xFEDC)
#define PWMA_BKR (*(unsigned char volatile xdata *) 0xFEDD)
#define PWMA_DTR (*(unsigned char volatile xdata *) 0xFEDE)
#define PWMA_OISR (*(unsigned char volatile xdata *) 0xFEDF)
```

/\*\*\*\*\*  
 \*\*\*\*\*/

```
#define PWMA_1 0x00 //P:P1.0 N:P1.1
#define PWMA_2 0x01 //P:P2.0 N:P2.1
#define PWMA_3 0x02 //P:P6.0 N:P6.1
```

```
#define PWMB_1 0x00 //P:P1.2 N:P1.3
#define PWMB_2 0x04 //P:P2.2 N:P2.3
#define PWMB_3 0x08 //P:P6.2 N:P6.3
```

```
#define PWM3_1      0x00      //P:P1.4  N:P1.5
#define PWM3_2      0x10      //P:P2.4  N:P2.5
#define PWM3_3      0x20      //P:P6.4  N:P6.5
```

```
#define PWM4_1      0x00      //P:P1.6  N:P1.7
#define PWM4_2      0x40      //P:P2.6  N:P2.7
#define PWM4_3      0x80      //P:P6.6  N:P6.7
#define PWM4_4      0xC0      //P:P3.4  N:P3.3
```

```
#define ENO1P 0x01
#define ENO1N 0x02
#define ENO2P 0x04
#define ENO2N 0x08
#define ENO3P 0x10
#define ENO3N 0x20
#define ENO4P 0x40
#define ENO4N 0x80
```

```

/***** 本地变量声明 *****/

```

```

unsigned int code T_SinTable[]={
    1220, 1256, 1292, 1328, 1364, 1400, 1435, 1471,
    1506, 1541, 1575, 1610, 1643, 1677, 1710, 1742,
    1774, 1805, 1836, 1866, 1896, 1925, 1953, 1981,
    2007, 2033, 2058, 2083, 2106, 2129, 2150, 2171,
    2191, 2210, 2228, 2245, 2261, 2275, 2289, 2302,
    2314, 2324, 2334, 2342, 2350, 2356, 2361, 2365,
    2368, 2369, 2370, 2369, 2368, 2365, 2361, 2356,
    2350, 2342, 2334, 2324, 2314, 2302, 2289, 2275,
    2261, 2245, 2228, 2210, 2191, 2171, 2150, 2129,
    2106, 2083, 2058, 2033, 2007, 1981, 1953, 1925,
    1896, 1866, 1836, 1805, 1774, 1742, 1710, 1677,
    1643, 1610, 1575, 1541, 1506, 1471, 1435, 1400,
    1364, 1328, 1292, 1256, 1220, 1184, 1148, 1112,
    1076, 1040, 1005, 969, 934, 899, 865, 830,
    797, 763, 730, 698, 666, 635, 604, 574,
    544, 515, 487, 459, 433, 407, 382, 357,
    334, 311, 290, 269, 249, 230, 212, 195,
    179, 165, 151, 138, 126, 116, 106, 98,
    90, 84, 79, 75, 72, 71, 70, 71,
    72, 75, 79, 84, 90, 98, 106, 116,
    126, 138, 151, 165, 179, 195, 212, 230,
    249, 269, 290, 311, 334, 357, 382, 407,
    433, 459, 487, 515, 544, 574, 604, 635,
    666, 698, 730, 763, 797, 830, 865, 899,
    934, 969, 1005, 1040, 1076, 1112, 1148, 1184,
};

```

*u16 PWMA\_Duty;*

*u8 PWM\_Index;*

*//SPWM 查表索引*

*/\*\*\*\*\*\* 主函数 \*\*\*\*\*\*/*

*void main(void)*

*{*

*P\_SW2 /= 0x80;*

*P0M1 = 0; P0M0 = 0; //设置为双向口*

*P1M1 = 0; P1M0 = 0; //设置为双向口*

*P2M1 = 0; P2M0 = 0; //设置为双向口*

*P3M1 = 0; P3M0 = 0; //设置为双向口*

*P4M1 = 0; P4M0 = 0; //设置为双向口*

*P5M1 = 0; P5M0 = 0; //设置为双向口*

*P6M1 = 0; P6M0 = 0; //设置为双向口*

*P7M1 = 0; P7M0 = 0; //设置为双向口*

*PWMA\_Duty = 1220;*

*PWMA\_CCER1 = 0x00;*

*//写 CCMRx 前必须先清零 CCxE 关闭通道*

*PWMA\_CCER2 = 0x00;*

*PWMA\_CCMR1 = 0x60;*

*//通道模式配置*

*// PWMA\_CCMR2 = 0x60;*

*// PWMA\_CCMR3 = 0x60;*

*// PWMA\_CCMR4 = 0x60;*

*PWMA\_CCER1 = 0x05;*

*//配置通道输出使能和极性*

*// PWMA\_CCER2 = 0x55;*

*PWMA\_ARRH = 0x09;*

*//设置周期时间*

*PWMA\_ARRL = 0x60;*

*PWMA\_CCR1H = (u8)(PWMA\_Duty >> 8);*

*//设置占空比时间*

*PWMA\_CCR1L = (u8)(PWMA\_Duty);*

*PWMA\_DTR = 0x0C;*

*//设置死区时间*

*PWMA\_ENO = 0x00;*

*PWMA\_ENO /= ENO1P;*

*//使能输出*

*PWMA\_ENO /= ENO1N;*

*//使能输出*

*// PWMA\_ENO /= ENO2P;*

*//使能输出*

*// PWMA\_ENO /= ENO2N;*

*//使能输出*

*// PWMA\_ENO /= ENO3P;*

*//使能输出*

*// PWMA\_ENO /= ENO3N;*

*//使能输出*

*// PWMA\_ENO /= ENO4P;*

*//使能输出*

*// PWMA\_ENO /= ENO4N;*

*//使能输出*

*PWMA\_PS = 0x00;*

*//高级 PWM 通道输出脚选择位*

*PWMA\_PS /= PWMA\_3;*

*//选择 PWMA\_3 通道*

*// PWMA\_PS /= PWMB\_3;*

*//选择 PWMB\_3 通道*

*// PWMA\_PS /= PWM3\_3;*

*//选择 PWM3\_3 通道*

*// PWMA\_PS /= PWM4\_3;*

*//选择 PWM4\_3 通道*

*PWMA\_BKR = 0x80;*

*//使能主输出*

*PWMA\_IER = 0x01;*

*//使能中断*

*PWMA\_CR1 /= 0x01;*

*//开始计时*

*EA = 1;*

*//打开总中断*

```

    while (1)
    {
    }
}

/***** 中断函数 *****/
void PWMA_ISR() interrupt 26
{
    if (PWMA_SRI & 0x01)
    {
        PWMA_SRI &= ~0x01;
        PWMA_Duty = T_SinTable[PWM_Index];
        if (++PWM_Index >= 200)
            PWM_Index = 0;

        PWMA_CCR1H = (u8)(PWMA_Duty >> 8);    //设置占空比时间
        PWMA_CCR1L = (u8)(PWMA_Duty);
    }
    PWMA_SRI = 0;
}

```

## 22.8.21 高级 PWM 输出-频率可调-脉冲计数（软件方式）

### C 语言代码

//测试工作频率为24MHz

/\*\*\*\*\* 功能说明 \*\*\*\*\*/

本例程基于 STC8H8K64U 为主控芯片的实验箱 8 进行编写测试, STC8H 系列芯片可通用参考。  
 高级 PWM 定时器实现高速 PWM 脉冲输出。  
 周期/占空比可调, 通过比较/捕获中断进行脉冲个数计数。  
 通过 P6 口演示输出, 每隔 10ms 输出一次 PWM, 计数 10 个脉冲后停止输出。  
 定时器每 1ms 调整 PWM 周期。  
 下载时, 选择时钟 24MHZ (用户可自行修改频率)。

\*\*\*\*\*/

```

#include "stc8h.h"
#include "intrins.h"

```

```

#define MAIN_Fosc      24000000L

```

```

typedef unsigned char    u8;
typedef unsigned int     u16;
typedef unsigned long    u32;

```

/\*\*\*\*\* 用户定义宏 \*\*\*\*\*/

```

#define Timer0_Reload    (65536UL-(MAIN_Fosc / 1000))    //Timer0 中断频率,1000 次/秒

#define PWM1_1           0x00                            //P:P1.0  N:P1.1
#define PWM1_2           0x01                            //P:P2.0  N:P2.1
#define PWM1_3           0x02                            //P:P6.0  N:P6.1

#define PWM2_1           0x00                            //P:P1.2  N:P1.3
#define PWM2_2           0x04                            //P:P2.2  N:P2.3
#define PWM2_3           0x08                            //P:P6.2  N:P6.3

```



```

#define PWM3_1      0x00          //P:P1.4 N:P1.5
#define PWM3_2      0x10          //P:P2.4 N:P2.5
#define PWM3_3      0x20          //P:P6.4 N:P6.5

#define PWM4_1      0x00          //P:P1.6 N:P1.7
#define PWM4_2      0x40          //P:P2.6 N:P2.7
#define PWM4_3      0x80          //P:P6.6 N:P6.7
#define PWM4_4      0xC0          //P:P3.4 N:P3.3

#define ENO1P       0x01
#define ENO1N       0x02
#define ENO2P       0x04
#define ENO2N       0x08
#define ENO3P       0x10
#define ENO3N       0x20
#define ENO4P       0x40
#define ENO4N       0x80

```

/\*\*\*\*\*\* 本地变量声明 \*\*\*\*\*\*/

```

bit B_1ms;          //1ms 标志
bit PWM1_Flag;

```

```

u16 Period;
u8 Counter;
u8 msSecond;

```

```

void UpdatePwm(void);
void TxPulse(void);

```

/\*\*\*\*\*\* 主函数 \*\*\*\*\*\*/

```

void main(void)
{
    P_SW2 /= 0x80;          //使能 XFR 访问

    P0M1 = 0x00;    P0M0 = 0x00;    //设置为双向口
    P1M1 = 0x00;    P1M0 = 0x00;    //设置为双向口
    P2M1 = 0x00;    P2M0 = 0x00;    //设置为双向口
    P3M1 = 0x00;    P3M0 = 0x00;    //设置为双向口
    P4M1 = 0x00;    P4M0 = 0x00;    //设置为双向口
    P5M1 = 0x00;    P5M0 = 0x00;    //设置为双向口
    P6M1 = 0x00;    P6M0 = 0x00;    //设置为双向口
    P7M1 = 0x00;    P7M0 = 0x00;    //设置为双向口

    PWM1_Flag = 0;
    Counter = 0;
    Period = 0x1000;

    //Timer0 初始化
    AUXR = 0x80;          //Timer0 set as 1T,16 bits timer auto-reload,
    TH0 = (u8)(Timer0_Reload / 256);
    TL0 = (u8)(Timer0_Reload % 256);
    ET0 = 1;              //Timer0 interrupt enable
    TR0 = 1;              //Timer0 run

    PWMA_ENO = 0x00;
    PWMA_ENO /= ENO1P;    //使能输出

    PWMA_PS = 0x00;      //高级 PWM 通道输出脚选择位
}

```

```

    PWMA_PS /= PWM1_3;                                //选择 PWM1_3 通道

    UpdatePwm();
    PWMA_BKR = 0x80;                                  //使能主输出
    PWMA_CR1 /= 0x01;                                  //开始计时

    P40 = 0;                                           //给 LED 供电
    EA = 1;                                           //打开总中断

    while (1)
    {
        if(B_1ms)
        {
            B_1ms = 0;
            msSecond++;
            if(msSecond >= 10)
            {
                msSecond = 0;
                TxPulse();                            //10ms 启动一次 PWM 输出
            }
        }
    }
}

/***** 发送脉冲函数 *****/
void TxPulse(void)
{
    PWMA_CCER1 = 0x00;                                //写 CCMRx 前必须先清零 CCxE 关闭通道
    PWMA_CCMR1 = 0x60;                                //设置 PWM1 模式1 输出
    PWMA_CCER1 = 0x01;                                //使能 CC1E 通道, 高电平有效
    PWMA_SRI = 0;                                       //清标志位
    PWMA_CNTR = 0;                                       //清计数器
    PWMA_IER = 0x02;                                    //使能捕获/比较 1 中断
}

/***** Timer0 1ms 中断函数 *****/
void timer0(void) interrupt 1
{
    B_1ms = 1;
    if(PWM1_Flag)
    {
        Period++;                                     //周期递增
        if(Period >= 0x1000) PWM1_Flag = 0;
    }
    else
    {
        Period--;                                     //周期递减
        if(Period <= 0x0100) PWM1_Flag = 1;
    }
    UpdatePwm();                                       //设置周期、占空比
}

/***** PWM 中断函数 *****/
void PWMA_ISR() interrupt 26
{
    if(PWMA_SRI & 0X02)
    {
        PWMA_SRI &= ~0X02;                            //清标志位
    }
}

```

```

Counter++;
if(Counter >= 10)
{
    Counter = 0;
    PWMA_CCER1 = 0x00;
    PWMA_CCMR1 = 0x40;
    PWMA_CCER1 = 0x01;
    PWMA_IER = 0x00;
}
}

//=====
// 函数: UpdatePwm(void)
// 描述: 更新PWM 周期占空比.
// 参数: none.
// 返回: none.
// 版本: V1.0, 2012-11-22
//=====
void UpdatePwm(void)
{
    PWMA_ARR = Period;
    PWMA_CCR1 = (Period >> 1);
}

```

## 22.8.22 高级 PWM 输出-频率可调-脉冲计数（硬件方式）

### C 语言代码

//测试工作频率为24MHz

/\*\*\*\*\*\* 功能说明 \*\*\*\*\*/

本例程基于STC8H8K64U 为主控芯片的实验箱9 进行编写测试, STC8H 系列芯片可通用参考.

高级PWM 定时器实现高速PWM 脉冲输出.

周期/占空比可调, 通过比较捕获中断进行脉冲个数计数.

通过P6 口演示输出, 每隔10ms 输出一次PWM, 计数10 个脉冲后停止输出.

使用单脉冲模式配合重复计数寄存器, 纯硬件控制脉冲个数.

定时器每1ms 调整PWM 周期.

下载时, 选择时钟 24MHZ (用户可自行修改频率).

\*\*\*\*\*/

```
#include "stc8h.h"
```

```
#include "intrins.h"
```

```
#define MAIN_Fosc 24000000L
```

//定义主时钟

```
typedef unsigned char u8;
```

```
typedef unsigned int u16;
```

```
typedef unsigned long u32;
```

/\*\*\*\*\*\* 用户定义宏 \*\*\*\*\*/

```
#define Timer0_Reload (65536UL-(MAIN_Fosc / 1000)) //Timer 0 中断频率, 1000 次/秒
```

\*\*\*\*\*/

```
#define PWM1_1 0x00
```

//P:P1.0 N:P1.1

```
#define PWM1_2 0x01
```

//P:P2.0 N:P2.1

```

#define PWM1_3      0x02                                //P:P6.0  N:P6.1

#define PWM2_1      0x00                                //P:P1.2/P5.4  N:P1.3
#define PWM2_2      0x04                                //P:P2.2  N:P2.3
#define PWM2_3      0x08                                //P:P6.2  N:P6.3

#define PWM3_1      0x00                                //P:P1.4  N:P1.5
#define PWM3_2      0x10                                //P:P2.4  N:P2.5
#define PWM3_3      0x20                                //P:P6.4  N:P6.5

#define PWM4_1      0x00                                //P:P1.6  N:P1.7
#define PWM4_2      0x40                                //P:P2.6  N:P2.7
#define PWM4_3      0x80                                //P:P6.6  N:P6.7
#define PWM4_4      0xC0                                //P:P3.4  N:P3.3

#define ENO1P      0x01
#define ENO1N      0x02
#define ENO2P      0x04
#define ENO2N      0x08
#define ENO3P      0x10
#define ENO3N      0x20
#define ENO4P      0x40
#define ENO4N      0x80

/***** 本地变量声明 *****/
bit B_1ms; //1ms 标志
bit PWM1_Flag;

u16 Period;
u8 Counter;
u8 msSecond;

void UpdatePwm(void);
void TxPulse(u8 rep);

/***** 主函数 *****/
void main(void)
{
    P_SW2 /= 0x80; //扩展寄存器(XFR)访问使能

    P0M1 = 0x00; P0M0 = 0x00; //设置为准双向口
    P1M1 = 0x00; P1M0 = 0x00; //设置为准双向口
    P2M1 = 0x00; P2M0 = 0x00; //设置为准双向口
    P3M1 = 0x00; P3M0 = 0x00; //设置为准双向口
    P4M1 = 0x00; P4M0 = 0x00; //设置为准双向口
    P5M1 = 0x00; P5M0 = 0x00; //设置为准双向口
    P6M1 = 0x00; P6M0 = 0x00; //设置为准双向口
    P7M1 = 0x00; P7M0 = 0x00; //设置为准双向口

    PWM1_Flag = 0;
    Counter = 0;
    Period = 0x1000;

    AUXR = 0x80; //Timer0 初始化
    TH0 = (u8)(Timer0_Reload / 256); //Timer0 set as 1T, 16 bits timer auto-reload,
    TL0 = (u8)(Timer0_Reload % 256);
    ET0 = 1; //Timer0 interrupt enable
    TR0 = 1; //Tiner0 run

```

```

PWMA_ENO = 0x00;
PWMA_ENO /= ENOIP;                                     //使能输出

PWMA_CCER1 = 0x00;                                     //写 CCMRx 前必须先清零 CCxE  关闭通道
PWMA_CCMR1 = 0x68;                                     //设置 PWM1 模式1 输出
//PWMA_CCER1 = 0x01;                                   //使能    CC1E 通道, 高电平有效
PWMA_CCER1 = 0x03;                                     //使能 CC1E 通道, 低电平有效

PWMA_PS = 0x00;                                        //高级 PWM 通道输出脚选择位
PWMA_PS /= PWM1_3;                                    //选择 PWM1_3 通道

UpdatePwm();
PWMA_BKR = 0x80;                                       //使能主输出
//PWMA_CR1 /= 0x89;                                    //使能ARR 预装载, 单脉冲模式, 开始计时

P40 = 0;                                                //给 LED 供电
EA = 1;                                                 //打开总中断

while (1)
{
    if(B_1ms)
    {
        B_1ms = 0;
        msSecond++;
        if(msSecond >= 10)                             //10ms 启动一次 PWM 输出
        {
            msSecond = 0;
            TxPulse(10);                                //输出 10 个脉冲
        }
    }
}

/***** 发送脉冲函数 *****/
void TxPulse(u8 rep)
{
    if(rep == 0) return;
    rep -= 1;

    PWMA_RCR = rep;                                     //重复计数寄存器, 计数 rep 个脉冲后产生更新事件
    PWMA_CR1 /= 0x89;                                   //使能ARR 预装载, 单脉冲模式, 开始计时
}

/***** Timer0 1ms 中断函数 *****/
void timer0(void) interrupt 1
{
    B_1ms = 1;
    if(PWM1_Flag)
    {
        Period++;                                     //周期递增
        if(Period >= 0x1000) PWM1_Flag = 0;
    }
    else
    {
        Period--;                                     //周期递减
        if(Period <= 0x0100) PWM1_Flag = 1;
    }
    UpdatePwm();                                       //设置周期、占空比
}

```

```
}
```

```
//=====
// 函数: UpdatePwm(void)
// 描述: 更新PWM 周期占空比
// 参数: none.
// 返回: none.
// 版本: V1.0, 2012-11-22
//=====
void UpdatePwm(void)
{
    PWMA_ARR = Period;
    PWMA_CCR1 = (Period >> 1);           //设置占空比时间: Period/2
}
```

STC MCU

## 22.8.23 产生 3 路相位差 120 度的互补 PWM 波形（网友提供）

//测试工作频率为24MHz

/\*\*\*\*\*\*

主要功能: P2.0-P2.5 输出互补的三路相位差 120 度的 PWM

第 1 路 P2.0/P2.1 为 PWM 输出模式, 第 2 路 P2.2/P2.3 和第 3 路 P2.4/P2.5 为比较输出模式

程序下载进目标芯片, 输出 50hz 的 SPWM, 占空比 25%

\*\*\*\*\*/

#include "stc8h.h"

#define FOSC 2400000UL

#define PWM\_PSC (240-1)

//定义 PWM 时钟预分频系数

#define PWM\_PERIOD 2000

//定义 PWM 周期值

//(频率=FOSC/(PWM\_PSC+1)/PWM\_PERIOD=50Hz)

#define PWM\_DUTY 500

//定义 PWM 的占空比值

//(占空比=PWM\_DUTY/PWM\_PERIOD\*100%=25%)

void SYS\_Init();

void PWM\_Init();

void main()

{

SYS\_Init();

PWM\_Init();

EA = 1;

//打开总中断

while (1);

}

void SYS\_Init()

{

P\_SW2 /= 0x80;

//扩展寄存器(XFR)访问使能

P0M1 = 0x00; P0M0 = 0x00;

P1M1 = 0x00; P1M0 = 0x00;

P2M1 = 0x00; P2M0 = 0x00;

P3M1 = 0x00; P3M0 = 0x00;

P4M1 = 0x00; P4M0 = 0x00;

P5M1 = 0x00; P5M0 = 0x00;

P6M1 = 0x00; P6M0 = 0x00;

P7M1 = 0x00; P7M0 = 0x00;

}

void PWM\_Init()

{

PWMA\_PSCR = PWM\_PSC;

//配置预分频系数

PWMA\_CCER1 = 0x00;

//写 CCMRx 前必须先清零 CCxE 关闭通道

PWMA\_CCER2 = 0x00;

PWMA\_CCMR1 = 0x60;

//通道模式配置 PWM 模式 1

PWMA\_CCMR2 = 0x30;

//通道模式配置输出比较模式

```

    PWMA_CCMR3 = 0x30;                //通道模式配置输出比较模式

    PWMA_CCER1 = 0x55;                //配置通道 1,2,3 输出使能和极性
    PWMA_CCER2 = 0x05;

    PWMA_ARR = PWM_PERIOD;            //设置周期时间

    PWMA_ENO = 0x3f;                  //使能PWM 输出
    PWMA_PS = 0x15;                    //高级PWM 通道输出脚选择P2.0-P2.5

    PWMA_CCR1 = PWM_DUTY;              //设置占空比时间
    PWMA_CCR2 = PWM_PERIOD/3;          //设置OC2 起始翻转位
    PWMA_CCR3 = PWM_PERIOD/3*2;        //设置OC3 起始翻转位

    PWMA_IER = 0x0d;                  //使能OC2/OC3 比较中断,更新中断

    PWMA_BKR = 0x80;                  //使能主输出
    PWMA_CR1 |= 0x01;                 //开始计时
}

void PWMA_ISR() interrupt 26
{
    if (PWMA_SRI & 0x01)
    {
        PWMA_CCR2 = PWM_PERIOD/3;      //设置占空比时间
        PWMA_CCR3 = PWM_PERIOD/3*2;    //设置占空比时间
        PWMA_SRI &= ~0x01;
    }
    else if (PWMA_SRI & 0x04)
    {
        PWMA_CCR2 = (PWM_PERIOD/3+PWM_DUTY); //设置OC2 结束翻转位
        PWMA_SRI &= ~0x04;
    }
    else if (PWMA_SRI & 0x08)
    {
        PWMA_CCR3 = (PWM_PERIOD/3*2+PWM_DUTY); //设置OC3 结束翻转位
        PWMA_SRI &= ~0x08;
    }
    else
    {
        PWMA_SRI = 0;
    }
}

```

## 22.8.24 高级 PWM 时钟输出应用（系统时钟 2 分频输出）

//测试工作频率为24MHz

```
#include "stc8h.h"
```

```

#define FOSC          24000000UL
#define PWM_PERIOD    (2-1)                //定义PWM 周期值
                                           //(频率=FOSC/(PWM_PERIOD+1)=12MHz)
#define PWM_DUTY      (1)                  //定义PWM 的占空比值
                                           //(占空比=PWM_DUTY/PWM_PERIOD*100%=50%)

```



```
void SYS_Init();
void PWM_Init();

void main()
{
    SYS_Init();
    PWM_Init();

    while (1);
}

void SYS_Init()
{
    P_SW2 /= 0x80; //扩展寄存器(XFR)访问使能

    P0M1 = 0x00; P0M0 = 0x00;
    P1M1 = 0x00; P1M0 = 0x00;
    P2M1 = 0x00; P2M0 = 0x00;
    P3M1 = 0x00; P3M0 = 0x00;
    P4M1 = 0x00; P4M0 = 0x00;
    P5M1 = 0x00; P5M0 = 0x00;
    P6M1 = 0x00; P6M0 = 0x00;
    P7M1 = 0x00; P7M0 = 0x00;
}

void PWM_Init()
{
    PWMA_CCER1 = 0x00; //写 CCMRx 前必须先清零 CCxE 关闭通道
    PWMA_CCMR1 = 0x60; //通道模式配置 PWM 模式 1
    PWMA_CCER1 = 0x01; //配置通道输出使能和极性
    PWMA_ARR = PWM_PERIOD; //设置周期时间
    PWMA_CCR1 = PWM_DUTY; //设置占空比时间
    PWMA_ENO = 0x01; //使能 PWM 输出
    PWMA_BKR = 0x80; //使能主输出
    PWMA_CR1 = 0x01; //开始计时
}
```

## 23 高速高级 PWM (HSPWM)

| 产品线                    | 高速高级 PWM |
|------------------------|----------|
| STC8H1K08 系列           |          |
| STC8H1K28 系列           |          |
| STC8H3K64S4 系列         |          |
| STC8H3K64S2 系列         |          |
| STC8H8K64U 系列          |          |
| STC8H4K64TL 系列         |          |
| STC8H4K64TLCD 系列 A 系列  |          |
| STC8H4K64TLCD 系列 A+ 系列 |          |
| STC8H1K08T 系列          | ●        |

STC8H 的部分系列单片机为高级 PWMA 和高级 PWMB 提供了高速模式(HSPWMA 和 HSPWMB)。高速高级 PWM 是以高级 PWMA 和高级 PWMB 为基础,增加了高速模式。

当系统运行在较低工作频率时,高速高级 PWM 可工作在高达 144M~192M 的频率下。从而可以达到降低内核功耗,提升外设性能的目的

### 23.1 相关寄存器

| 符号         | 描述            | 地址    | 位地址与符号    |           |    |    |    |       |         |           | 复位值       |
|------------|---------------|-------|-----------|-----------|----|----|----|-------|---------|-----------|-----------|
|            |               |       | B7        | B6        | B5 | B4 | B3 | B2    | B1      | B0        |           |
| HSPWMA_CFG | 高速 PWMA 配置寄存器 | FBF0H | -         | -         | -  | -  | -  | INTEN | ASYNCEN | 1         | xxxx,0001 |
| HSPWMA_ADR | 高速 PWMA 地址寄存器 | FBF1H | RW/BUSY   | ADDR[6:0] |    |    |    |       |         |           | 0000,0000 |
| HSPWMA_DAT | 高速 PWMA 数据寄存器 | FBF2H | DATA[7:0] |           |    |    |    |       |         | 0000,0000 |           |
| HSPWMB_CFG | 高速 PWMB 配置寄存器 | FBF4H | -         | -         | -  | -  | -  | INTEN | ASYNCEN | 1         | xxxx,0001 |
| HSPWMB_ADR | 高速 PWMB 地址寄存器 | FBF5H | RW/BUSY   | ADDR[6:0] |    |    |    |       |         |           | 0000,0000 |
| HSPWMB_DAT | 高速 PWMB 数据寄存器 | FBF6H | DATA[7:0] |           |    |    |    |       |         | 0000,0000 |           |

#### 23.1.1 HSPWM 配置寄存器 (HSPWMn\_CFG)

| 符号         | 地址    | B7 | B6 | B5 | B4 | B3 | B2    | B1      | B0 |
|------------|-------|----|----|----|----|----|-------|---------|----|
| HSPWMA_CFG | FBF0H | -  | -  | -  | -  | -  | INTEN | ASYNCEN | 1  |
| HSPWMB_CFG | FBF4H | -  | -  | -  | -  | -  | INTEN | ASYNCEN | 1  |

ASYNCEN: 异步控制模式使能位

0: 关闭异步控制。

1: 使能异步控制模式。

注: 当关闭异步控制时, 高级 PWMA/高级 PWMB 为传统模式, 此时高级 PWM 会自动选择系统工作频率, PWM 工作频率与系统工作频率相同; 若需要时 PWM 工作在高速模式, 则需要使能异

步控制模式，此时 PWM 时钟可选择主时钟（MCLK）或者 PLL 输出时钟

INTEN：异步模式中断使能位

0：关闭异步模式下的 PWM 中断。

1：使能异步模式下的 PWM 中断。

注：异步模式下，若需要响应高级 PWMA/高级 PWMB 的中断，则必须使能 INTEN 位

### 23.1.2 HSPWM 地址寄存器（HSPWMn\_ADR）

| 符号         | 地址    | B7      | B6        | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|---------|-----------|----|----|----|----|----|----|
| HSPWMA_ADR | FBF1H | RW/BUSY | ADDR[6:0] |    |    |    |    |    |    |
| HSPWMB_ADR | FBF5H | RW/BUSY | ADDR[6:0] |    |    |    |    |    |    |

ADDR[6:0]：高级 PWMA/PWMB 的特殊功能寄存器地址低 7 位

0：关闭异步控制。

1：使能异步控制模式。

RW/BUSY：读写控制位、状态位

写 0：异步方式写 PWMA/PWMB 的特殊功能寄存器。

写 1：异步方式读 PWMA/PWMB 的特殊功能寄存器。

读 0：异步读写已经完成

读 1：异步读写正在进行，处于忙状态

### 23.1.3 HSPWM 数据寄存器（HSPWMn\_DAT）

| 符号         | 地址    | B7        | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|-----------|----|----|----|----|----|----|----|
| HSPWMA_DAT | FBF2H | DATA[7:0] |    |    |    |    |    |    |    |
| HSPWMB_DAT | FBF6H | DATA[7:0] |    |    |    |    |    |    |    |

DATA[7:0]：高级 PWMA/PWMB 的特殊功能寄存器数据

写：写数据到高级 PWMA/PWMB 的特殊功能寄存器。

读：从高级 PWMA/PWMB 的特殊功能寄存器读取数据。

异步读取 PWMA 的特殊功能寄存器步骤：（PWMB 类似）

- 1、读取 HSPWMA\_ADR，等待 BUSY 为 0，确定前一个异步读写已完成
- 2、将 PWMA 的特殊功能寄存器的低 7 位写入 HSPWMA\_ADR，同时置“1” HSPWMA\_ADR.7
- 3、读取 HSPWMA\_ADR，等待 BUSY 为 0
- 4、读取 HSPWMA\_DAT

异步写 PWMA 的特殊功能寄存器步骤：（PWMB 类似）

- 1、读取 HSPWMA\_ADR，等待 BUSY 为 0，确定前一个异步读写已完成
- 2、将需要写入 PWMA 的特殊功能寄存器的数据写入 HSPWMA\_DAT
- 3、将 PWMA 的特殊功能寄存器的低 7 位写入 HSPWMA\_ADR，同时清“0” HSPWMA\_ADR.7
- 4、读取 HSPWMA\_ADR，等待 BUSY 为 0。（可跳过此步继续执行其他代码，以提高系统效率）

**特别注意：**特殊功能寄存器 PWMA\_PS 和 PWMB\_PS 属于端口控制寄存器，不属于 PWMA 和 PWMB 寄存器组，所以无论是否启动 PWM 的异步控制模式，PWMA\_PS 和 PWMB\_PS 寄存器都只能使用普通同步模式进行读写

## 23.2 范例程序

### 23.2.1 使能高级 PWM 的高速模式（异步模式）

---

//测试工作频率为24MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC 24000000UL
```

```
#define HSCK_MCLK 0
```

```
#define HSCK_PLL 1
```

```
#define HSCK_SEL HSCK_PLL
```

```
#define PLL_96M 0
```

```
#define PLL_144M 1
```

```
#define PLL_SEL PLL_144M
```

```
#define CKMS 0x80
```

```
#define HSIOCK 0x40
```

```
#define MCK2SEL_MSK 0x0c
```

```
#define MCK2SEL_SEL1 0x00
```

```
#define MCK2SEL_PLLD2 0x04
```

```
#define MCK2SEL_PLLD4 0x08
```

```
#define MCKSEL_MSK 0x03
```

```
#define MCKSEL_HIRC 0x00
```

```
#define MCKSEL_XOSC 0x01
```

```
#define MCKSEL_X32K 0x02
```

```
#define MCKSEL_IRC32K 0x03
```

```
#define ENCKM 0x80
```

```
#define PCKI_MSK 0x60
```

```
#define PCKI_D1 0x00
```

```
#define PCKI_D2 0x20
```

```
#define PCKI_D3 0x40
```

```
#define PCKI_D4 0x60
```

```
void delay()
```

```
{
```

```
    int i;
```

```
    for (i=0; i<100; i++);
```

```
}
```

```
char ReadPWMA(char addr)
```

```
{
```

```
    char dat;
```

```
    while (HSPWMA_ADR & 0x80);
```

```
    HSPWMA_ADR = addr / 0x80;
```

```
    while (HSPWMA_ADR & 0x80);
```

```
    dat = HSPWMA_DAT;
```

```
//等待前一个异步读写完成
```

```
//设置间接访问地址,只需要设置原XFR地址的低7位
```

```
//HSPWMA_ADR寄存器的最高位写1,表示读数据
```

```
//等待当前异步读取完成
```

```
//读取异步数据
```

```

    return dat;
}

void WritePWMA(char addr, char dat)
{
    while (HSPWMA_ADR & 0x80);           //等待前一个异步读写完成
    HSPWMA_DAT = dat;                     //准备需要写入的数据
    HSPWMA_ADR = addr & 0x7f;             //设置间接访问地址,只需要设置原XFR地址的低7位
                                           //HSPWMA_ADR寄存器的最高位写0,表示写数据
}

void main()
{
    P_SW2 /= 0x80;                        //使能访问XFR

    //选择PLL输出时钟
    #if (PLL_SEL == PLL_96M)
        CLKSEL &= ~CKMS;                  //选择PLL的96M作为PLL的输出时钟
    #elif (PLL_SEL == PLL_144M)
        CLKSEL /= CKMS;                   //选择PLL的144M作为PLL的输出时钟
    #else
        CLKSEL &= ~CKMS;                  //默认选择PLL的96M作为PLL的输出时钟
    #endif

    //选择PLL输入时钟分频,保证输入时钟为12M
    PLLCR &= ~PCKI_MSK;
    #if (FOSC == 12000000UL)
        PLLCR /= PCKI_D1;                 //PLL输入时钟1分频
    #elif (FOSC == 24000000UL)
        PLLCR /= PCKI_D2;                 //PLL输入时钟2分频
    #elif (FOSC == 36000000UL)
        PLLCR /= PCKI_D3;                 //PLL输入时钟3分频
    #elif (FOSC == 48000000UL)
        PLLCR /= PCKI_D4;                 //PLL输入时钟4分频
    #else
        PLLCR /= PCKI_D1;                 //默认PLL输入时钟1分频
    #endif

    //启动PLL
    PLLCR /= ENCKM;                       //使能PLL倍频

    delay();                              //等待PLL锁频

    //选择HSPWM/HSSPI时钟
    #if (HCK_SEL == HCK_MCLK)
        CLKSEL &= ~HSIOCK;                 //HSPWM/HSSPI选择主时钟为时钟源
    #elif (HCK_SEL == HCK_PLL)
        CLKSEL /= HSIOCK;                  //HSPWM/HSSPI选择PLL输出时钟为时钟源
    #else
        CLKSEL &= ~HSIOCK;                 //默认HSPWM/HSSPI选择主时钟为时钟源
    #endif

    HCLKDIV = 0;                          //HSPWM/HSSPI时钟源不分频

    HSPWMA_CFG = 0x03;                    //使能PWMA相关寄存器异步访问功能

    //通过异步方式设置PWMA的相关寄存器
    WritePWMA((char*)&PWMA_CCER1, 0x00);

```

```
WritePWMA((char)&PWMA_CCMR1, 0x00); //CC1 为输出模式
WritePWMA((char)&PWMA_CCMR1, 0x60); //OC1REF 输出 PWM1(CNT<CCR 时输出有效电平1)
WritePWMA((char)&PWMA_CCER1, 0x05); //使能CC1/CC1N 上的输出功能
WritePWMA((char)&PWMA_ENO, 0x03); //使能PWM 信号输出到端口
WritePWMA((char)&PWMA_BKR, 0x80); //使能主输出
WritePWMA((char)&PWMA_CCR1H, 200 >> 8); //设置输出PWM 的占空比
WritePWMA((char)&PWMA_CCR1L, 200);
WritePWMA((char)&PWMA_ARRH, 1000 >> 8); //设置输出PWM 的周期
WritePWMA((char)&PWMA_ARRL, 1000);
WritePWMA((char)&PWMA_DTR, 10); //设置互补对称输出PWM 的死区
WritePWMA((char)&PWMA_CR1, 0x01); //开始PWM 计数

P2M0 = 0;
P2M1 = 0;
P3M0 = 0;
P3M1 = 0;

P2 = ReadPWMA((char)&PWMA_ARRH); //异步方式读取寄存器
P3 = ReadPWMA((char)&PWMA_ARRL);

while (1);
}
```

## 24 USB 通用串行总线

| 产品线              | USB |
|------------------|-----|
| STC8H1K08 系列     |     |
| STC8H1K28 系列     |     |
| STC8H3K64S4 系列   |     |
| STC8H3K64S2 系列   |     |
| STC8H8K64U 系列    | ●   |
| STC8H4K64TL 系列   |     |
| STC8H4K64TLCD 系列 |     |
| STC8H1K08T 系列    |     |

STC8H 部分系列单片机内部集成 USB2.0/USB1.1 兼容全速 USB，6 个双向端点，支持 4 种端点传输模式（控制传输、中断传输、批量传输和同步传输），每个端点拥有 64 字节的缓冲区。

USB 模块共有 1280 字节的 FIFO，结构如下：

|                    |           |           |           |            |                    |
|--------------------|-----------|-----------|-----------|------------|--------------------|
| Control            | Endp 0-IN | 64 Bytes  | 64 Bytes  | Endp 0-OUT | Control            |
| INT<br>BULK<br>ISO | Endp 1-IN | 64 Bytes  | 64 Bytes  | Endp 1-OUT | INT<br>BULK<br>ISO |
| INT<br>BULK<br>ISO | Endp 2-IN | 64 Bytes  | 64 Bytes  | Endp 2-OUT | INT<br>BULK<br>ISO |
| INT<br>BULK<br>ISO | Endp 3-IN | 64 Bytes  | 64 Bytes  | Endp 3-OUT | INT<br>BULK<br>ISO |
| INT<br>BULK<br>ISO | Endp 4-IN | 128 Bytes | 256 Bytes | Endp 4-OUT | INT<br>BULK<br>ISO |
| INT<br>BULK<br>ISO | Endp 5-IN | 128 Bytes | 256 Bytes | Endp 5-OUT | INT<br>BULK<br>ISO |

### 24.1 USB 相关的寄存器

| 符号     | 描述          | 地址  | 位地址与符号 |           |    |     |         |         |             | 复位值       |
|--------|-------------|-----|--------|-----------|----|-----|---------|---------|-------------|-----------|
|        |             |     | B7     | B6        | B5 | B4  | B3      | B2      | B1          |           |
| USBCLK | USB 时钟控制寄存器 | DCH | ENCKM  | PCKI[1:0] |    | CRE | TST_USB | TST_PHY | PHYTST[1:0] | 0010,0000 |
| USBDAT | USB 数据寄存器   | ECH |        |           |    |     |         |         |             | 0000,0000 |

|        |           |     |       |        |           |      |      |       |    |    |           |
|--------|-----------|-----|-------|--------|-----------|------|------|-------|----|----|-----------|
| USBCON | USB 控制寄存器 | F4H | ENUSB | USBRST | PS2M      | PUEN | PDEN | DFREC | DP | DM | 0000,0000 |
| USBADR | USB 地址寄存器 | FCH | BUSY  | AUTORD | UADR[5:0] |      |      |       |    |    | 0000,0000 |

### 24.1.1 USB 控制寄存器 (USBCON)

| 符号     | 地址  | B7    | B6     | B5   | B4   | B3   | B2    | B1 | B0 |
|--------|-----|-------|--------|------|------|------|-------|----|----|
| USBCON | F4H | ENUSB | USBRST | PS2M | PUEN | PDEN | DFREC | DP | DM |

ENUSB: USB 功能与 USB 时钟控制位

0: 关闭 USB 功能与 USB 时钟

1: 使能 USB 功能与 USB 时钟

ENRST: USB 复位设置控制位

0: 关闭 USB 复位设置

1: 使能 USB 复位

PS2M: PS2 mode 功能控制位

0: 关闭 PS2 mode 功能

1: 使能 PS2 mode 功能

PUEN: DP/DM 端口上 1.5K 上拉电阻控制位

0: 禁止上拉电阻

1: 使能上拉电阻

PDEN: DP/DM 端口上 500K 下拉电阻控制位

0: 禁止下拉电阻

1: 使能下拉电阻

DFREC: 差分接收状态位 (只读)

0: 当前 DP/DM 的差分状态为 “0”

1: 当前 DP/DM 的差分状态为 “1”

DP: D+ 端口状态 (PS2 为 0 时只读, PS2 为 1 时可读写)

0: 当前 D+ 为逻辑 0 电平

1: 当前 D+ 为逻辑 1 电平

DM: D- 端口状态 (PS2 为 0 时只读, PS2 为 1 时可读写)

0: 当前 D- 为逻辑 0 电平

1: 当前 D- 为逻辑 1 电平

### 24.1.2 USB 时钟控制寄存器 (USBCLK)

| 符号     | 地址  | B7    | B6        | B5 | B4  | B3      | B2      | B1          | B0 |
|--------|-----|-------|-----------|----|-----|---------|---------|-------------|----|
| USBCLK | DCH | ENCKM | PCKI[1:0] |    | CRE | TST_USB | TST_PHY | PHYTST[1:0] |    |

ENCKM: PLL 倍频控制

0: 禁止 PLL 倍频

1: 使能 PLL 倍频

PCKI[1:0]: PLL 时钟选择

| PCKI[1:0] | PLL 时钟源 |
|-----------|---------|
| 00        | 12M     |
| 01        | 24M     |
| 10        | 36M     |
| 11        | 48M     |

CRE: 时钟追频控制位



0: 禁止时钟追频

1: 使能时钟追频

TST\_USB: USB 测试模式

0: 禁止 USB 测试模式

1: 使能 USB 测试模式

TST\_PHY: PHY 测试模式

0: 禁止 PHY 测试模式

1: 使能 PHY 测试模式

PHYTST[1:0]: USB PHY 测试

| PHYTST[1:0] | 方式             | DP | DM |
|-------------|----------------|----|----|
| 00          | 方式 0: 正常       | x  | x  |
| 01          | 方式 1: 强制 “1”   | 1  | 0  |
| 10          | 方式 2: 强制 “0”   | 0  | 1  |
| 11          | 方式 3: 强制单端 “0” | 0  | 0  |

### 24.1.3 USB 间址地址寄存器 (USBADR)

| 符号     | 地址  | B7   | B6     | B5        | B4 | B3 | B2 | B1 | B0 |
|--------|-----|------|--------|-----------|----|----|----|----|----|
| USBADR | FCH | BUSY | AUTORD | UADR[5:0] |    |    |    |    |    |

BUSY: USB 寄存器读忙标志位

写 0: 无意义

写 1: 启动 USB 间接寄存器的读操作, 地址由 USBADR 设定

读 0: USBDAT 寄存器中的数据有效

读 1: USBDAT 寄存器中的数据无效, USB 正在读取间接寄存器

AUTORD: USB 寄存器自动读标志, 用于 USB 的 FIFO 的块读取

写 0: 每次读取间接 USB 寄存器都必须先写 BUSY 标志位

写 1: 当软件读取 USBDAT 时, 下一个 USB 间接寄存器的读取将自动启动 (USBADR 不变)

UADR[5:0]: USB 间接寄存器的地址

### 24.1.4 USB 间址数据寄存器 (USBDAT)

| 符号     | 地址  | B7        | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-----|-----------|----|----|----|----|----|----|----|
| USBDAT | ECH | UDAT[7:0] |    |    |    |    |    |    |    |

UDAT[7:0]: 用于间接读写 USB 寄存器

## 24.2 USB 控制器寄存器 (SIE)

|     | 0/8     | 1/9            | 2/A     | 3/B      | 4/C      | 5/D     | 6/E                 | 7/F       |
|-----|---------|----------------|---------|----------|----------|---------|---------------------|-----------|
| 30H | UTRKCTL | UTRKSTS        |         |          |          |         |                     |           |
| 28H |         |                |         |          |          |         |                     |           |
| 20H | FIFO0   | FIFO1          | FIFO2   | FIFO3    | FIFO4    | FIFO5   |                     |           |
| 18H |         |                |         |          |          |         |                     |           |
| 10H | INMAXP  | CSR0<br>INCSR1 | INCSR2  | OUTMAXP  | OUTCSR1  | OUTCSR2 | COUNT0<br>OUTCOUNT1 | OUTCOUNT2 |
| 08H |         | INTROUT1E      |         | INTRUSBE | FRAME1   | FRAME2  | INDEX               |           |
| 00H | FADDR   | POWER          | INTRIN1 | -        | INTROUT1 | -       | INTRUSB             | INTRIN1E  |

| 符号        | 描述                | 地址  | 位地址与符号       |              |           |          |          |              |          |           | 复位值       |  |
|-----------|-------------------|-----|--------------|--------------|-----------|----------|----------|--------------|----------|-----------|-----------|--|
|           |                   |     | B7           | B6           | B5        | B4       | B3       | B2           | B1       | B0        |           |  |
| FADDR     | USB 功能地址寄存器       | 00H | UPDATE       | UADDR[6:0]   |           |          |          |              |          |           | 0000,0000 |  |
| POWER     | USB 电源管理寄存器       | 01H | ISOUD        | -            | -         | -        | USBRST   | USBRSU       | USBSUS   | ENSUS     | 0xxx,0000 |  |
| INTRIN1   | USB 端点 IN 中断标志位   | 02H | -            | -            | EP5INIF   | EP4INIF  | EP3INIF  | EP2INIF      | EP1INIF  | EP0IF     | xx00,0000 |  |
| INTROUT1  | USB 端点 OUT 中断标志位  | 04H | -            | -            | EP5OUTIF  | EP4OUTIF | EP3OUTIF | EP2OUTIF     | EP1OUTIF | -         | xx00,000x |  |
| INTRUSB   | USB 电源中断标志位       | 06H | -            | -            | -         | -        | SOFIF    | RSTIF        | RSUIF    | SUSIF     | xxxx,0000 |  |
| INTRIN1E  | USB 端点 IN 中断允许位   | 07H | -            | -            | EP5INIE   | EP4INIE  | EP3INIE  | EP2INIE      | EP1INIE  | EP0IE     | xx11,1111 |  |
| INTROUT1E | USB 端点 OUT 中断允许位  | 09H | -            | -            | EP5OUTIE  | EP4OUTIE | EP3OUTIE | EP2OUTIE     | EP1OUTIE | -         | xx11,111x |  |
| INTRUSBE  | USB 电源中断允许位       | 0BH | -            | -            | -         | -        | SOFIE    | RSTIE        | RSUIE    | SUSIE     | xxxx,0110 |  |
| FRAME1    | USB 数据帧号低字节       | 0CH | FRAME[7:0]   |              |           |          |          |              |          | 0000,0000 |           |  |
| FRAME2    | USB 数据帧号高字节       | 0DH | -            | -            | -         | -        | -        | FRAME[10:8]  |          |           | xxxx,x000 |  |
| INDEX     | USB 端点索引寄存器       | 0EH | -            | -            | -         | -        | -        | INDEX[2:0]   |          |           | xxxx,x000 |  |
| INMAXP    | IN 端点的最大数据包大小     | 10H | INMAXP[7:0]  |              |           |          |          |              |          | 0000,0000 |           |  |
| CSR0      | 端点 0 控制状态寄存器      | 11H | SSUEND       | SOPRDY       | SDSTL     | SUEND    | DATEND   | STSTL        | IPRDY    | OPRDY     | 0000,0000 |  |
| INCSR1    | IN 端点控制状态寄存器 1    | 11H | -            | CLRDT        | STSTL     | SDSTL    | FLUSH    | UNDRUN       | FIFONE   | IPRDY     | x000,0000 |  |
| INCSR2    | IN 端点控制状态寄存器 2    | 12H | AUTOSET      | ISO          | MODE      | ENDMA    | FCDT     | -            | -        | -         | 0010,0xxx |  |
| OUTMAXP   | OUT 端点的最大数据包大小    | 13H | OUTMAXP[7:0] |              |           |          |          |              |          | 0000,0000 |           |  |
| OUTCSR1   | OUT 端点控制状态寄存器 1   | 14H | CLRDT        | STSTL        | SDSTL     | FLUSH    | DATERR   | OVRRUN       | FIFOFUL  | OPRDY     | 0000,0000 |  |
| OUTCSR2   | OUT 端点控制状态寄存器 2   | 15H | AUTOCLR      | ISO          | ENDMA     | DMAMD    | -        | -            | -        | -         | 0000,xxxx |  |
| COUNT0    | 端点 0 的 OUT 长度     | 16H | -            | OUTCNT0[6:0] |           |          |          |              |          |           | x000,0000 |  |
| OUTCOUNT1 | USB 端点 OUT 长度低字节  | 16H | OUTCNT[7:0]  |              |           |          |          |              |          | 0000,0000 |           |  |
| OUTCOUNT2 | USB 端点 OUT 长度高字节  | 17H | -            | -            | -         | -        | -        | OUTCNT[10:8] |          |           | xxxx,x000 |  |
| FIFO0     | 端点 0 的 FIFO 访问寄存器 | 20H | FIFO0[7:0]   |              |           |          |          |              |          | 0000,0000 |           |  |
| FIFO1     | 端点 1 的 FIFO 访问寄存器 | 21H | FIFO1[7:0]   |              |           |          |          |              |          | 0000,0000 |           |  |
| FIFO2     | 端点 2 的 FIFO 访问寄存器 | 22H | FIFO2[7:0]   |              |           |          |          |              |          | 0000,0000 |           |  |
| FIFO3     | 端点 3 的 FIFO 访问寄存器 | 23H | FIFO3[7:0]   |              |           |          |          |              |          | 0000,0000 |           |  |
| FIFO4     | 端点 4 的 FIFO 访问寄存器 | 24H | FIFO4[7:0]   |              |           |          |          |              |          | 0000,0000 |           |  |
| FIFO5     | 端点 5 的 FIFO 访问寄存器 | 25H | FIFO5[7:0]   |              |           |          |          |              |          | 0000,0000 |           |  |
| UTRKCTL   | USB 跟踪控制寄存器       | 30H | FTM1         | FTM0         | INTV[1:0] |          | ENST5    | RES[2:0]     |          |           | 1011,1011 |  |
| UTRKSTS   | USB 跟踪状态寄存器       | 31H | INTVCNT[3:0] |              |           |          | STS[1:0] |              | TST_UTRK | UTRK_RDY  | 1111,00x0 |  |

## 24.2.1 USB 功能地址寄存器 (FADDR)

| 符号    | 地址  | B7     | B6         | B5 | B4 | B3 | B2 | B1 | B0 |
|-------|-----|--------|------------|----|----|----|----|----|----|
| FADDR | 00H | UPDATE | UADDR[6:0] |    |    |    |    |    |    |

UPDATE: 更新 USB 功能地址

0: 最后的 UADDR 地址已生效

1: 最后的 UADDR 地址还未生效

UADDR[6:0]: 保存 USB 的 7 位功能地址

## 24.2.2 USB 电源控制寄存器 (POWER)

| 符号    | 地址  | B7    | B6 | B5 | B4 | B3     | B2     | B1     | B0    |
|-------|-----|-------|----|----|----|--------|--------|--------|-------|
| POWER | 01H | ISOUD | -  | -  | -  | USBRST | USBRSU | USBSUS | ENSUS |

ISOUD (ISO Update): ISO 更新

0: 当软件向 IPRDY 写“1”时, USB 在收到下一个 IN 令牌后发送数据包

1: 当软件向 IPRDY 写“1”时, USB 在收到 SOF 令牌后发送数据包, 如果 SOF 令牌之前收到 IN 令牌, 则 USB 发送长度为 0 的数据包

USBRST (USB Reset): USB 复位控制位

向此位写“1”, 可强制产生异步 USB 复位。读取此位可以获得当前总线上的复位状态信息

0: 总线上没有检测到复位信号

1: 总线上检测到了复位信号

USBRSU (USB Resume): USB 恢复控制位

以软件方式在总线上强制产生恢复信号, 以便将 USB 设备从挂起方式进行远程唤醒。当 USB 处于挂起模式 (USBSUS=1) 时, 向此位写“1”, 将强制在 USB 总线上产生恢复信号, 软件应在 10-15ms 后向此位写“0”, 以结束恢复信号。软件向 USBRSU 写入“0”后将产生 USB 恢复中断, 此时硬件会自动将 USBSUS 清“0”

USBSUS (USB Suspend): USB 挂起控制位

当 USB 进入挂起方式时, 此位被硬件置“1”。当以软件方式在总线上强制产生恢复信号后或者在总线上检测到恢复信号时且在读取了 INTRUSB 寄存器后, 硬件自动将此位清“0”。

ENSUS (Enable Suspend Detection): 使能 USB 挂起方式检测

0: 禁止挂起检测, USB 将忽略总线上的挂起信号

1: 使能挂起检测, 当检测到总线上的挂起信号, USB 将进入挂起方式

## 24.2.3 USB 端点 IN 中断标志位 (INTRIN1)

| 符号      | 地址  | B7 | B6 | B5      | B4      | B3      | B2      | B1      | B0    |
|---------|-----|----|----|---------|---------|---------|---------|---------|-------|
| INTRIN1 | 02H | -  | -  | EP5INIF | EP4INIF | EP3INIF | EP2INIF | EP1INIF | EP0IF |

EP5INIF: 端点 5 的 IN 中断标志位

0: 端点 5 的 IN 中断无效

1: 端点 5 的 IN 中断有效

EP4INIF: 端点 4 的 IN 中断标志位

0: 端点 4 的 IN 中断无效

1: 端点 4 的 IN 中断有效

EP3INIF: 端点 3 的 IN 中断标志位

0: 端点 3 的 IN 中断无效

1: 端点 3 的 IN 中断有效

EP2INIF: 端点 2 的 IN 中断标志位

0: 端点 2 的 IN 中断无效

1: 端点 2 的 IN 中断有效

EP1INIF: 端点 1 的 IN 中断标志位

0: 端点 1 的 IN 中断无效

1: 端点 1 的 IN 中断有效

EP0IF: 端点 0 的 IN/OUT 中断标志位

0: 端点 0 的 IN/OUT 中断无效

1: 端点 0 的 IN/OUT 中断有效

在软件读取 INTRIN1 寄存器后, 硬件将自动清除 INTRIN1 中的所有的中断标志

## 24.2.4 USB 端点 OUT 中断标志位 (INTROUT1)

| 符号       | 地址  | B7 | B6 | B5       | B4       | B3       | B2       | B1       | B0 |
|----------|-----|----|----|----------|----------|----------|----------|----------|----|
| INTROUT1 | 04H | -  | -  | EP5OUTIF | EP4OUTIF | EP3OUTIF | EP2OUTIF | EP1OUTIF | -  |

EP5OUTIF: 端点 5 的 OUT 中断标志位

0: 端点 5 的 OUT 中断无效

1: 端点 5 的 OUT 中断有效

EP4OUTIF: 端点 4 的 OUT 中断标志位

0: 端点 4 的 OUT 中断无效

1: 端点 4 的 OUT 中断有效

EP3OUTIF: 端点 3 的 OUT 中断标志位

0: 端点 3 的 OUT 中断无效

1: 端点 3 的 OUT 中断有效

EP2OUTIF: 端点 2 的 OUT 中断标志位

0: 端点 2 的 OUT 中断无效

1: 端点 2 的 OUT 中断有效

EP1OUTIF: 端点 1 的 OUT 中断标志位

0: 端点 1 的 OUT 中断无效

1: 端点 1 的 OUT 中断有效

在软件读取 INTROUT1 寄存器后, 硬件将自动清除 INTROUT1 中的所有的中断标志

## 24.2.5 USB 电源中断标志 (INTRUSB)

| 符号      | 地址  | B7 | B6 | B5 | B4 | B3    | B2    | B1    | B0    |
|---------|-----|----|----|----|----|-------|-------|-------|-------|
| INTRUSB | 06H | -  | -  | -  | -  | SOFIF | RSTIF | RSUIF | SUSIF |

SOFIF: USB 帧起始信号中断标志

0: USB 帧起始信号中断无效

1: USB 帧起始信号中断有效

RSTIF: USB 复位信号中断标志

0: USB 复位信号中断无效

1: USB 复位信号中断有效

RSUIF: USB 恢复信号中断标志

0: USB 恢复信号中断无效

1: USB 恢复信号中断有效

SUSIF: USB 挂起信号中断标志

0: USB 挂起信号中断无效

1: USB 挂起信号中断有效

在软件读取 INTRUSB 寄存器后，硬件将自动清除 INTRUSB 中的所有的中断标志

## 24.2.6 USB 端点 IN 中断允许寄存器 (INTRIN1E)

| 符号       | 地址  | B7 | B6 | B5      | B4      | B3      | B2      | B1      | B0    |
|----------|-----|----|----|---------|---------|---------|---------|---------|-------|
| INTRIN1E | 07H | -  | -  | EP5INIE | EP4INIE | EP3INIE | EP2INIE | EP1INIE | EP0IE |

EP5INIE: 端点 5 的 IN 中断控制位

0: 禁止端点 5 的 IN 中断

1: 允许端点 5 的 IN 中断

EP4INIE: 端点 4 的 IN 中断控制位

0: 禁止端点 4 的 IN 中断

1: 允许端点 4 的 IN 中断

EP3INIE: 端点 3 的 IN 中断控制位

0: 禁止端点 3 的 IN 中断

1: 允许端点 3 的 IN 中断

EP2INIE: 端点 2 的 IN 中断控制位

0: 禁止端点 2 的 IN 中断

1: 允许端点 2 的 IN 中断

EP1INIE: 端点 1 的 IN 中断控制位

0: 禁止端点 1 的 IN 中断

1: 允许端点 1 的 IN 中断

EP0IE: 端点 0 的 IN/OUT 中断控制位

0: 禁止端点 0 的 IN/OUT 中断

1: 允许端点 0 的 IN/OUT 中断

## 24.2.7 USB 端点 OUT 中断允许寄存器 (INTROUT1E)

| 符号        | 地址  | B7 | B6 | B5       | B4       | B3       | B2       | B1       | B0 |
|-----------|-----|----|----|----------|----------|----------|----------|----------|----|
| INTROUT1E | 09H | -  | -  | EP5OUTIE | EP4OUTIE | EP3OUTIE | EP2OUTIE | EP1OUTIE | -  |

EP5OUTIE: 端点 5 的 OUT 中断控制位

0: 禁止端点 5 的 OUT 中断

1: 允许端点 5 的 OUT 中断

EP4OUTIE: 端点 4 的 OUT 中断控制位

0: 禁止端点 4 的 OUT 中断

1: 允许端点 4 的 OUT 中断

EP3OUTIE: 端点 3 的 OUT 中断控制位

0: 禁止端点 3 的 OUT 中断

1: 允许端点 3 的 OUT 中断

EP2OUTIE: 端点 2 的 OUT 中断控制位

0: 禁止端点 2 的 OUT 中断

1: 允许端点 2 的 OUT 中断

EP1OUTIE: 端点 1 的 OUT 中断控制位

0: 禁止端点 1 的 OUT 中断

1: 允许端点 1 的 OUT 中断

## 24.2.8 USB 电源中断允许寄存器 (INTRUSBE)

| 符号 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----|----|----|----|----|----|----|----|----|----|
|----|----|----|----|----|----|----|----|----|----|

|          |     |   |   |   |   |       |       |       |       |
|----------|-----|---|---|---|---|-------|-------|-------|-------|
| INTRUSBE | 0BH | - | - | - | - | SOFIE | RSTIE | RSUIE | SUSIE |
|----------|-----|---|---|---|---|-------|-------|-------|-------|

SOFIE: USB 帧起始信号中断控制位

0: 禁止 USB 帧起始信号中断

1: 允许 USB 帧起始信号中断

RSTIE: USB 复位信号中断控制位

0: 禁止 USB 复位信号中断

1: 允许 USB 复位信号中断

RSUIE: USB 恢复信号中断控制位

0: 禁止 USB 恢复信号中断

1: 允许 USB 恢复信号中断

SUSIE: USB 挂起信号中断控制位

0: 禁止 USB 挂起信号中断

1: 允许 USB 挂起信号中断

## 24.2.9 USB 数据帧号寄存器 (FRAME<sub>n</sub>)

| 符号     | 地址  | B7         | B6 | B5 | B4 | B3 | B2          | B1 | B0 |
|--------|-----|------------|----|----|----|----|-------------|----|----|
| FRAME1 | 0CH | FRAME[7:0] |    |    |    |    |             |    |    |
| FRAME2 | 0DH | -          | -  | -  | -  | -  | FRAME[10:8] |    |    |

FRAME[10:0]: 用于保存最后接收到的数据帧的 11 位帧号

## 24.2.10 USB 端点索引寄存器 (INDEX)

| 符号    | 地址  | B7 | B6 | B5 | B4 | B3 | B2         | B1 | B0 |
|-------|-----|----|----|----|----|----|------------|----|----|
| INDEX | 0EH | -  | -  | -  | -  | -  | INDEX[2:0] |    |    |

INDEX[2:0]: 选择 USB 端点

| INDEX[2:0] | 目标端点 |
|------------|------|
| 000        | 端点 0 |
| 001        | 端点 1 |
| 010        | 端点 2 |
| 011        | 端点 3 |
| 100        | 端点 4 |
| 101        | 端点 5 |

## 24.2.11 IN 端点的最大数据包大小 (INMAXP)

| 符号     | 地址  | B7          | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-----|-------------|----|----|----|----|----|----|----|
| INMAXP | 10H | INMAXP[7:0] |    |    |    |    |    |    |    |

INMAXP[7:0]: 设置 USB 的 IN 端点最大数据包的大小 (**注意: 数据包是以 8 字节为单位。例如若需要将端点的数据包设置为 64 字节, 则需要将此寄存器设置为 8**)

当需要获取/设置此信息时, 首先必须使用 INDEX 来选择目标端点 0~5

## 24.2.12 USB 端点 0 控制状态寄存器 (CSR0)

| 符号   | 地址  | B7     | B6     | B5    | B4    | B3     | B2    | B1    | B0    |
|------|-----|--------|--------|-------|-------|--------|-------|-------|-------|
| CSR0 | 11H | SSUEND | SOPRDY | SDSTL | SUEND | DATEND | STSTL | IPRDY | OPRDY |

SSUEND (Serviced Setup End)

SETUP 结束事件处理完成标志。当处理完成 SETUP 结束事件 (SUEND) 后, 软件需要设置 SSUEND 标志位, 硬件检测到 SSUEND 被写入 “1” 时自动将 SUEND 位清 “0”。

**SOPRDY (Serviced OPRDY)**

OPRDY 事件处理完成标志。当处理完成从端点 0 接收到的数据包后, 软件需要设置 SOPRDY 标志位, 硬件检测到 SOPRDY 被写入“1”时自动将 OPRDY 位清“0”。

**SDSTL (Send Stall)**

当接收到错误的条件或者不支持的请求时, 可以在此位写“1”来结束当前的数据传输。当 STALL 信号被发送后, 硬件自动将此位清“0”。

**SUEND (Setup End)**

SETUP 安装包结束标志。当一次控制传输在软件向 DATAEND 位写“1”之前结束时, 硬件将此只读位置“1”。当软件向 SSUEND 写“1”后, 硬件将该位清“0”。

**DATEND (Data End)**

数据结束。软件应在下列情况下将此位写“1”:

- 1、当发送最后一个数据包后, 固件向 IPRDY 写“1”时;
  - 2、当发送一个零长度数据包后, 固件向 IPRDY 写“1”时;
  - 3、当接收完最后一个数据包后, 固件向 SOPRDY 写“1”时;
- 该位将被硬件自动清“0”

**STSTL (Sent Stall)**

STALL 信号发送完成标志。发送完成 STALL 信号后, 硬件将该位置“1”。该位必须用软件清“0”。

**IPRDY (IN Packet Ready)**

IN 数据包准备完成标志。软件应在将一个要发送的数据包装入到端点 0 的 FIFO 后, 将该位置“1”。在发生下列条件之一时, 硬件将该位清“0”:

- 1、数据包已发送时;
- 2、数据包被一个 SETUP 包覆盖时;
- 3、数据包被一个 OUT 包覆盖时;

**OPRDY (OUT Packet Ready)**

OUT 数据包准备完成标志。当收到一个 OUT 数据包时, 硬件将该只读位置“1”, 并产生中断。该位只在软件向 SOPRDY 位写“1”时才被清“0”。

当需要获取/设置此信息时, 首先必须使用 INDEX 来选择目标端点 0

## 24.2.13 IN 端点控制状态寄存器 1 (INCSR1)

| 符号     | 地址  | B7 | B6    | B5    | B4    | B3    | B2     | B1     | B0    |
|--------|-----|----|-------|-------|-------|-------|--------|--------|-------|
| INCSR1 | 11H | -  | CLRDT | STSTL | SDSTL | FLUSH | UNDRUN | FIFONE | IPRDY |

**CLRDT (Clear Data Toggle):** 复位 IN 数据切换位。

当 IN 端点由于被重新配置或者被 STALL 而需要将数据切换位复位到“0”时, 软件需要向此数据位写“1”。

**STSTL (Sent Stall):** STALL 信号发送完成标志。

当 STALL 信号被发送完成后, 硬件会将该位置“1”(此时 FIFO 被清空, IPRDY 位被清“0”)。该标志必须用软件清“0”。

**SDSTL (Send Stall):** STALL 信号发送请求位。

软件应向该位写“1”以产生 STALL 信号作为对一个 IN 令牌的应答。软件应向该位写“0”以结束 STALL 信号。该位对 ISO 方式没有影响。

**FLUSH (FIFO Flush):** 清除 IN 端点的 FIFO 的下一个数据包。

向该位写“1”将从 IN 端点 FIFO 中清除待发送的下一个数据包。FIFO 指针被复位, IPRDY 位被清“0”。如果 FIFO 中包含多个数据包, 软件必须对每个数据包向 FLUSH 写“1”。当 FIFO 清空完成后, 硬件将 FLUSH 位清“0”。

**UNDRUN (Data Underrun):** 数据不足。



该位的功能取决于 IN 端点的方式:

ISO 方式: 在 IPRDY 为“0”且收到一个 IN 令牌后发送了一个零长度数据包时, 该位被置“1”。

中断/批量方式: 当使用 NAK 作为对一个 IN 令牌的应答时, 该位被置“1”。

该位必须用软件清“0”。

FIFONE (FIFO Not Empty): IN 端点的 FIFO 非空标志

0: IN 端点的 FIFO 为空

1: IN 端点的 FIFO 包含有一个或者多个数据包

IPRDY (IN Packet Ready): IN 数据包准备完成标志。

软件应在将一个要发送的数据包装入到端点的 FIFO 后, 将该位置“1”。在发生下列条件之一时, 硬件将该位清“0”:

- 1、数据包已发送时;
- 2、自动设置被使能 (AUTOSSET = ‘1’) 且端点 IN 的 FIFO 数据包达到 INMAXP 所设置的值;
- 3、如果端点处于同步方式且 ISOUD 为“1”, 在收到下一个 SOF 之前 IPRDY 的读出值总是为 0。

需要获取/设置此信息时, 首先必须使用 INDEX 来选择目标端点 1~5

## 24.2.14 IN 端点控制状态寄存器 2 (INCSR2)

| 符号     | 地址  | B7       | B6  | B5   | B4    | B3   | B2 | B1 | B0 |
|--------|-----|----------|-----|------|-------|------|----|----|----|
| INCSR2 | 12H | AUTOSSET | ISO | MODE | ENDMA | FCDT | -  | -  | -  |

AUTOSSET: 自动设置 IPRDY 标志控制位。

0: 禁止自动设置 IPRDY 标志

1: 使能自动设置 IPRDY (必须向 IN FIFO 中装载的数据达到 INMAXP 所设置的值, 否则 IPRDY 标志必须手动设置)

ISO: 同步传输使能。

0: 端点被配置为批量/中断传输方式

1: 端点被配置为同步传输方式

MODE: 端点方向选择位。

0: 选择端点方向为 OUT

1: 选择端点方向为 IN

ENDMA: IN 端点的 DMA 控制

0: 禁止 IN 端点的 DMA 请求

1: 使能 IN 端点的 DMA 请求

FCDT: 强制 DATA0/DATA1 数据切换设置。

0: 端点数据只在发送完一个数据包后且收到 ACK 时切换。

1: 端点数据在每发送完一个数据包后被强制切换, 不管是否收到 ACK。

需要获取/设置此信息时, 首先必须使用 INDEX 来选择目标端点 1~5

## 24.2.15 OUT 端点的最大数据包大小 (OUTMAXP)

| 符号      | 地址  | B7           | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------|-----|--------------|----|----|----|----|----|----|----|
| OUTMAXP | 13H | OUTMAXP[7:0] |    |    |    |    |    |    |    |

OUTMAXP[7:0]: 设置 USB 的 OUT 端点最大数据包的大小 (**注意: 数据包是以 8 字节为单位。例如若需要将端点的数据包设置为 64 字节, 则需要将此寄存器设置为 8**)

需要获取/设置此信息时, 首先必须使用 INDEX 来选择目标端点 1~5

## 24.2.16 OUT 端点控制状态寄存器 1 (OUTCSR1)

| 符号 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----|----|----|----|----|----|----|----|----|----|
|----|----|----|----|----|----|----|----|----|----|



|         |     |       |       |       |       |        |        |         |       |
|---------|-----|-------|-------|-------|-------|--------|--------|---------|-------|
| OUTCSR1 | 14H | CLRDT | STSTL | SDSTL | FLUSH | DATERR | OVRRUN | FIFOFUL | OPRDY |
|---------|-----|-------|-------|-------|-------|--------|--------|---------|-------|

CLRDT (Clear Data Toggle): 复位 OUT 数据切换位。

当 OUT 端点由于被重新配置或者被 STALL 而需要将数据切换位复位到“0”时, 软件需要向此数据位写“1”。

STSTL (Sent Stall): STALL 信号发送完成标志。

当 STALL 信号被发送完成后, 硬件会将该位置“1”。该标志必须用软件清“0”。

SDSTL (Send Stall): STALL 信号发送请求位。

软件应向该位写“1”以产生 STALL 信号作为对一个 OUT 令牌的应答。软件应向该位写“0”以结束 STALL 信号。该位对 ISO 方式没有影响。

FLUSH (FIFO Flush): 清除 OUT 端点的 FIFO 的下一个数据包。

向该位写“1”将从 OUT 端点 FIFO 中清除下一个数据包。FIFO 指针被复位, OPRDY 位被清“0”。

如果 FIFO 中包含多个数据包, 软件必须对每个数据包向 FLUSH 写“1”。当 FIFO 清空完成后, 硬件将 FLUSH 位清“0”。

DATAERR (Data Error): 数据错误。

在 ISO 方式, 如果接收到的数据包有 CRC 或位填充错误, 该位被硬件置“1”。当软件清除 OPRDY 时, 该位被清“0”。该位只在 ISO 方式有效。

OVRRUN (Data Overrun): 数据溢出。

当一个输入数据包不能被装入到 OUT 端点 FIFO 时, 该位被硬件置“1”。该位只在 ISO 方式有效。该位必须用软件清“0”。

0: 无数据溢出

1: 自该标志最后一次被清除以来, 因 FIFO 已满导致数据包丢失

FIFOFUL (FIFO Full): OUT 端点的 FIFO 数据满标志。

0: OUT 端点的 FIFO 未滿

1: OUT 端点的 FIFO 已滿

OPRDY (OUT Packet Ready): OUT 数据包接收完成标志。

当有数据包可用时硬件将该位置“1”。软件应在将每个数据包从 OUT 端点 FIFO 卸载后将该位清“0”。

需要获取/设置此信息时, 首先必须使用 INDEX 来选择目标端点 1~5

## 24.2.17 OUT 端点控制状态寄存器 2 (OUTCSR2)

| 符号      | 地址  | B7      | B6  | B5               | B4               | B3 | B2 | B1 | B0 |
|---------|-----|---------|-----|------------------|------------------|----|----|----|----|
| OUTCSR2 | 15H | AUTOCLR | ISO | <del>ENDMA</del> | <del>DMAMD</del> | -  | -  | -  | -  |

AUTOCLR: 自动清除 OPRDY 标志控制位。

0: 禁止自动清除 OPRDY 标志

1: 使能自动清除 OPRDY (必须从 OUT FIFO 中下载的数据达到 OUTMAXP 所设置的值, 否则 OPRDY 标志必须手动清除)

ISO: 同步传输使能。

0: 端点被配置为批量/中断传输方式

1: 端点被配置为同步传输方式

~~ENDMA: OUT 端点的 DMA 控制~~

~~0: 禁止 OUT 端点的 DMA 请求~~

~~1: 使能 OUT 端点的 DMA 请求~~

~~DMAMD: 设置 OUT 端点的 DMA 模式~~

需要获取/设置此信息时, 首先必须使用 INDEX 来选择目标端点 1~5

## 24.2.18 USB 端点 0 的 OUT 长度 (COUNT0)

| 符号     | 地址  | B7 | B6           | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-----|----|--------------|----|----|----|----|----|----|
| COUNT0 | 16H | -  | OUTCNT0[6:0] |    |    |    |    |    |    |

OUTCNT0[6:0]: 端点 0 的 OUT 字节长度

COUNT0 专用于保存端点 0 最后接收到的 OUT 数据包的数据长度 (由于端点 0 数据包最长只能为 64 字节, 所以只需要 7 位)。此长度值只在端点 0 的 OPRDY 位为“1”时才有效。

需要获取此长度信息时, 首先必须使用 INDEX 来选择目标端点 0

## 24.2.19 USB 端点的 OUT 长度 (OUTCOUNTn)

| 符号        | 地址  | B7          | B6 | B5 | B4 | B3 | B2           | B1 | B0 |
|-----------|-----|-------------|----|----|----|----|--------------|----|----|
| OUTCOUNT1 | 16H | OUTCNT[7:0] |    |    |    |    |              |    |    |
| OUTCOUNT2 | 17H | -           | -  | -  | -  | -  | OUTCNT[10:8] |    |    |

OUTCNT[10:0]: 端点的 OUT 字节长度

OUTCOUNT1 和 OUTCOUNT2 联合组成一个 11 位的数, 保存最后 OUT 数据包的数据长度, 适用于端点 1~5。此长度值只在端点 1~5 的 OPRDY 位为“1”时才有效。

需要获取此长度信息时, 首先必须使用 INDEX 来选择目标端点 1~5

## 24.2.20 USB 端点的 FIFO 数据访问寄存器 (FIFO<sub>n</sub>)

| 符号    | 地址  | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------|-----|------------|----|----|----|----|----|----|----|
| FIFO0 | 20H | FIFO0[7:0] |    |    |    |    |    |    |    |
| FIFO1 | 21H | FIFO1[7:0] |    |    |    |    |    |    |    |
| FIFO2 | 22H | FIFO2[7:0] |    |    |    |    |    |    |    |
| FIFO3 | 23H | FIFO3[7:0] |    |    |    |    |    |    |    |
| FIFO4 | 24H | FIFO4[7:0] |    |    |    |    |    |    |    |
| FIFO5 | 25H | FIFO5[7:0] |    |    |    |    |    |    |    |

FIFO<sub>n</sub>[7:0]: USB 各个端点的 IN/OUT 数据间接访问寄存器

## 24.3 USB 产品开发注意事项

每个 USB 产品都必须有自己唯一的 VID&PID 组合, 才能被电脑正确识别。若两个不同的 USB 产品对应的 VID&PID 组合相同, 则可能出现电脑对 USB 产品的识别出现异常, 从而无法正常使用 USB 产品。为避免出现这种情况, VID 和 PID 均需通过正规途径进行统一规划和分配。

目前 STC 已通过 USB-IF 组织取得了 STC 的专用 USB 设备的 VID 编号 13503 (十六进制: 34BF)。客户使用 STC 的 USB 芯片开发自己的 USB 产品时, 若您已通过其它途径获取了您自己的 VID, 则相应的 PID 可自行规划。若您的 USB 产品需要使用 STC 的官方 VID, 则产品的 PID 务必请您向 STC 申请。

## 24.4 范例程序

### 24.4.1 HID 人机接口设备范例

#### C 语言代码

---

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
typedef unsigned char    BYTE;
typedef unsigned int     WORD;
typedef unsigned long    DWORD;
```

```
#define FADDR            0
#define POWER            1
#define INTRIN1          2
#define EP5INIF          0x20
#define EP4INIF          0x10
#define EP3INIF          0x08
#define EP2INIF          0x04
#define EP1INIF          0x02
#define EP0IF            0x01
#define INTROUT1         4
#define EP5OUTIF         0x20
#define EP4OUTIF         0x10
#define EP3OUTIF         0x08
#define EP2OUTIF         0x04
#define EP1OUTIF         0x02
#define INTRUSB          6
#define SOFIF            0x08
#define RSTIF            0x04
#define RSUIF            0x02
#define SUSIF            0x01
#define INTRINIE         7
#define EP5INIE          0x20
#define EP4INIE          0x10
#define EP3INIE          0x08
#define EP2INIE          0x04
#define EP1INIE          0x02
#define EP0IE            0x01
#define INTROUTIE        9
#define EP5OUTIE         0x20
#define EP4OUTIE         0x10
#define EP3OUTIE         0x08
#define EP2OUTIE         0x04
#define EP1OUTIE         0x02
#define INTRUSBE         11
#define SOFIE            0x08
#define RSTIE            0x04
#define RSUIE            0x02
#define SUSIE            0x01
#define FRAME1           12
#define FRAME2           13
#define INDEX            14
#define INMAXP           16
#define CSR0              17
```

```

#define SSUEND          0x80
#define SOPRDY          0x40
#define SDSTL           0x20
#define SUEND           0x10
#define DATEND          0x08
#define STSTL           0x04
#define IPRDY           0x02
#define OPRDY           0x01
#define INCSR1          17
#define INCLRDT         0x40
#define INSTSTL         0x20
#define INSDSTL         0x10
#define INFLUSH         0x08
#define INUNDRUN        0x04
#define INFIFONE        0x02
#define INIPRDY         0x01
#define INCSR2          18
#define INAUTOSET       0x80
#define INISO           0x40
#define INMODEIN        0x20
#define INMODEOUT       0x00
#define INENDMA         0x10
#define INFCDT          0x08
#define OUTMAXP         19
#define OUTCSR1         20
#define OUTCLRDT        0x80
#define OUTSTSTL        0x40
#define OUTSDSTL        0x20
#define OUTFLUSH        0x10
#define OUTDATERR       0x08
#define OUTOVRUN        0x04
#define OUTFIFOFUL      0x02
#define OUTOPRDY        0x01
#define OUTCSR2         21
#define OUTAUTOCLR      0x80
#define OUTISO          0x40
#define OUTENDMA        0x20
#define OUTDMAMD        0x10
#define COUNT0          22
#define OUTCOUNT1     22
#define OUTCOUNT2     23
#define FIFO0           32
#define FIFO1           33
#define FIFO2           34
#define FIFO3           35
#define FIFO4           36
#define FIFO5           37
#define UTRKCTL         48
#define UTRKSTS         49

#define EPIDLE          0
#define EPSTATUS        1
#define EPDATAIN        2
#define EPDATAOUT       3
#define EPSTALL         -1

#define GET_STATUS      0x00
#define CLEAR_FEATURE   0x01
#define SET_FEATURE     0x03

```

```
#define SET_ADDRESS          0x05
#define GET_DESCRIPTOR       0x06
#define SET_DESCRIPTOR       0x07
#define GET_CONFIG           0x08
#define SET_CONFIG           0x09
#define GET_INTERFACE        0x0A
#define SET_INTERFACE        0x0B
#define SYNCH_FRAME          0x0C
```

```
#define GET_REPORT           0x01
#define GET_IDLE             0x02
#define GET_PROTOCOL         0x03
#define SET_REPORT           0x09
#define SET_IDLE             0x0A
#define SET_PROTOCOL         0x0B
```

```
#define DESC_DEVICE          0x01
#define DESC_CONFIG          0x02
#define DESC_STRING          0x03
#define DESC_HIDREPORT       0x22
```

```
#define STANDARD_REQUEST     0x00
#define CLASS_REQUEST        0x20
#define VENDOR_REQUEST       0x40
#define REQUEST_MASK         0x60
```

```
typedef struct
```

```
{
    BYTE    bmRequestType;
    BYTE    bRequest;
    BYTE    wValueL;
    BYTE    wValueH;
    BYTE    wIndexL;
    BYTE    wIndexH;
    BYTE    wLengthL;
    BYTE    wLengthH;
}SETUP;
```

```
typedef struct
```

```
{
    BYTE    bStage;
    WORD    wResidue;
    BYTE    *pData;
}EP0STAGE;
```

```
void UsbInit();
BYTE ReadReg(BYTE addr);
void WriteReg(BYTE addr, BYTE dat);
BYTE ReadFifo(BYTE fifo, BYTE *pdat);
void WriteFifo(BYTE fifo, BYTE *pdat, BYTE cnt);
```

```
char code DEVIDEDESC[18];
char code CONFIGDESC[41];
char code HIDREPORTDESC[27];
char code LANGIDDESC[4];
char code MANUFACTDESC[8];
char code PRODUCTDESC[30];
```

```
SETUP Setup;
```

```
EP0STAGE Ep0Stage;
BYTE xdata HidFreature[64];
BYTE xdata HidInput[64];
BYTE xdata HidOutput[64];

void main()
{
    P_SW2 /= 0x80;                                     //使能访问XFR

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UsbInit();

    IE2 = 0x80;
    EA = 1;

    while (1);
}

BYTE ReadReg(BYTE addr)
{
    BYTE dat;

    while (USBADR & 0x80);
    USBADR = addr / 0x80;
    while (USBADR & 0x80);
    dat = USBDAT;

    return dat;
}

void WriteReg(BYTE addr, BYTE dat)
{
    while (USBADR & 0x80);
    USBADR = addr & 0x7f;
    USBDAT = dat;
}

BYTE ReadFifo(BYTE fifo, BYTE *pdat)
{
    BYTE cnt;
    BYTE ret;

    ret = cnt = ReadReg(COUNT0);
    while (cnt--)
    {
        *pdat++ = ReadReg(fifo);
    }
}
```

```
    return ret;
}

void WriteFifo(BYTE fifo, BYTE *pdat, BYTE cnt)
{
    while (cnt--)
    {
        WriteReg(fifo, *pdat++);
    }
}

void UsbInit()
{
    P3M0 = 0x00;
    P3M1 = 0x03;

    IRC48MCR = 0x80;
    while (!(IRC48MCR & 0x01));
    USBCLK = 0x00;
    USBCON = 0x90;

    WriteReg(FADDR, 0x00);
    WriteReg(POWER, 0x08);
    WriteReg(INTRINIE, 0x3f);
    WriteReg(INTROUTIE, 0x3f);
    WriteReg(INTRUSB, 0x00);
    WriteReg(POWER, 0x01);

    Ep0Stage.bStage = EPIDLE;
}

void usb_isr() interrupt 25
{
    BYTE intrusb;
    BYTE intrin;
    BYTE introut;
    BYTE csr;
    BYTE cnt;
    WORD len;

    intrusb = ReadReg(INTRUSB);
    intrin = ReadReg(INTRINI);
    introut = ReadReg(INTROUTI);

    if (intrusb & RSTIF)
    {
        WriteReg(INDEX, 1);
        WriteReg(INCSRI, INCLRDT);
        WriteReg(INDEX, 1);
        WriteReg(OUTCSRI, OUTCLRDT);
        Ep0Stage.bStage = EPIDLE;
    }

    if (intrin & EP0IF)
    {
        WriteReg(INDEX, 0);
        csr = ReadReg(CSR0);
        if (csr & STSTL)
```

```

{
    WriteReg(CSR0, csr & ~STSTL);
    Ep0Stage.bStage = EPIDLE;
}
if (csr & SUEND)
{
    WriteReg(CSR0, csr / SSUEND);
}

switch (Ep0Stage.bStage)
{
case EPIDLE:
    if (csr & OPRDY)
    {
        Ep0Stage.bStage = EPSTATUS;
        ReadFifo(FIFO0, (BYTE *)&Setup);
        ((BYTE *)&Ep0Stage.wResidue)[0] = Setup.wLengthH;
        ((BYTE *)&Ep0Stage.wResidue)[1] = Setup.wLengthL;
        switch (Setup.bmRequestType & REQUEST_MASK)
        {
case STANDARD_REQUEST:
            switch (Setup.bRequest)
            {
case SET_ADDRESS:
                WriteReg(FADDR, Setup.wValueL);
                break;
case SET_CONFIG:
                WriteReg(INDEX, 1);
                WriteReg(INCSR2, INMODEIN);
                WriteReg(INMAXP, 8);
                WriteReg(INDEX, 1);
                WriteReg(INCSR2, INMODEOUT);
                WriteReg(OUTMAXP, 8);
                WriteReg(INDEX, 0);
                break;
case GET_DESCRIPTOR:
                Ep0Stage.bStage = EPDATAIN;
                switch (Setup.wValueH)
                {
case DESC_DEVICE:
                    Ep0Stage.pData = DEVIDDESC;
                    len = sizeof(DEVIDDESC);
                    break;
case DESC_CONFIG:
                    Ep0Stage.pData = CONFIGDESC;
                    len = sizeof(CONFIGDESC);
                    break;
case DESC_STRING:
                    switch (Setup.wValueL)
                    {
case 0:
                        Ep0Stage.pData = LANGIDDESC;
                        len = sizeof(LANGIDDESC);
                        break;
case 1:
                        Ep0Stage.pData = MANUFACTDESC;
                        len = sizeof(MANUFACTDESC);
                        break;
case 2:

```



```

        Ep0Stage.pData = PRODUCTDESC;
        len = sizeof(PRODUCTDESC);
        break;
    default:
        Ep0Stage.bStage = EPSTALL;
        break;
    }
    break;
case DESC_HIDREPORT:
    Ep0Stage.pData = HIDREPORTDESC;
    len = sizeof(HIDREPORTDESC);
    break;
default:
    Ep0Stage.bStage = EPSTALL;
    break;
}
if (len < Ep0Stage.wResidue)
{
    Ep0Stage.wResidue = len;
}
break;
default:
    Ep0Stage.bStage = EPSTALL;
    break;
}
break;
case CLASS_REQUEST:
    switch (Setup.bRequest)
    {
    case GET_REPORT:
        Ep0Stage.pData = HidFreature;
        Ep0Stage.bStage = EPDATAIN;
        break;
    case SET_REPORT:
        Ep0Stage.pData = HidFreature;
        Ep0Stage.bStage = EPDATAOUT;
        break;
    case SET_IDLE:
        break;
    case GET_IDLE:
    case GET_PROTOCOL:
    case SET_PROTOCOL:
    default:
        Ep0Stage.bStage = EPSTALL;
        break;
    }
    break;
default:
    Ep0Stage.bStage = EPSTALL;
    break;
}

switch (Ep0Stage.bStage)
{
case EPDATAIN:
    WriteReg(CSR0, SOPRDY);
    goto L_Ep0SendData;
    break;
case EPDATAOUT:

```

```

        WriteReg(CSR0, SOPRDY);
        break;
    case EPSTATUS:
        WriteReg(CSR0, SOPRDY / DATEND);
        Ep0Stage.bStage = EPIDLE;
        break;
    case EPSTALL:
        WriteReg(CSR0, SOPRDY / SDSTL);
        Ep0Stage.bStage = EPIDLE;
        break;
    }
}
break;
case EPDATAIN:
    if (!(csr & IPRDY))
    {
L_Ep0SendData:
        cnt = Ep0Stage.wResidue > 64 ? 64 : Ep0Stage.wResidue;
        WriteFifo(FIFO0, Ep0Stage.pData, cnt);
        Ep0Stage.wResidue -= cnt;
        Ep0Stage.pData += cnt;
        if (Ep0Stage.wResidue == 0)
        {
            WriteReg(CSR0, IPRDY / DATEND);
            Ep0Stage.bStage = EPIDLE;
        }
        else
        {
            WriteReg(CSR0, IPRDY);
        }
    }
    break;
case EPDATAOUT:
    if (csr & OPRDY)
    {
        cnt = ReadFifo(FIFO0, Ep0Stage.pData);
        Ep0Stage.wResidue -= cnt;
        Ep0Stage.pData += cnt;
        if (Ep0Stage.wResidue == 0)
        {
            WriteReg(CSR0, SOPRDY / DATEND);
            Ep0Stage.bStage = EPIDLE;
        }
        else
        {
            WriteReg(CSR0, SOPRDY);
        }
    }
    break;
}

if (intrin & EPIINIF)
{
    WriteReg(INDEX, 1);
    csr = ReadReg(INCSRI);
    if (csr & INSTSTL)
    {
        WriteReg(INCSRI, INCLRDT);
    }
}

```

```

    }
    if (csr & INUNDRUN)
    {
        WriteReg(INCSR1, 0);
    }
}

if (introut & EPIOUTFIF)
{
    WriteReg(INDEX, 1);
    csr = ReadReg(OUTCSR1);
    if (csr & OUTSTSTL)
    {
        WriteReg(OUTCSR1, OUTCLRDT);
    }
    if (csr & OUTOPRDY)
    {
        ReadFifo(FIFO1, HidOutput);
        WriteReg(OUTCSR1, 0);

        WriteReg(INDEX, 1);
        WriteFifo(FIFO1, HidOutput, 64);
        WriteReg(INCSR1, INIPRDY);
    }
}
}

char code DEVIDEDESC[18] =
{
    0x12, //bLength(18);
    0x01, //bDescriptorType(Device);
    0x00,0x02, //bcdUSB(2.00);
    0x00, //bDeviceClass(0);
    0x00, //bDeviceSubClass0);
    0x00, //bDeviceProtocol(0);
    0x40, //bMaxPacketSize0(64);
    0xbf,0x34, //idVendor(34bf);
    0x01,0xf0, //idProduct(f001);
    0x00,0x01, //bcdDevice(1.00);
    0x01, //iManufacturer(1);
    0x02, //iProduct(2);
    0x00, //iSerialNumber(0);
    0x01, //bNumConfigurations(1);
};

char code CONFIGDESC[41] =
{
    0x09, //bLength(9);
    0x02, //bDescriptorType(Configuration);
    0x29,0x00, //wTotalLength(41);
    0x01, //bNumInterfaces(1);
    0x01, //bConfigurationValue(1);
    0x00, //iConfiguration(0);
    0x80, //bmAttributes(BUSPower);
    0x32, //MaxPower(100mA);

    0x09, //bLength(9);
    0x04, //bDescriptorType(Interface);
    0x00, //bInterfaceNumber(0);

```

```

    0x00, //bAlternateSetting(0);
    0x02, //bNumEndpoints(2);
    0x03, //bInterfaceClass(HID);
    0x00, //bInterfaceSubClass(0);
    0x00, //bInterfaceProtocol(0);
    0x00, //iInterface(0);

    0x09, //bLength(9);
    0x21, //bDescriptorType(HID);
    0x01,0x01, //bcdHID(1.01);
    0x00, //bCountryCode(0);
    0x01, //bNumDescriptors(1);
    0x22, //bDescriptorType(HID Report);
    0x1b,0x00, //wDescriptorLength(27);

    0x07, //bLength(7);
    0x05, //bDescriptorType(Endpoint);
    0x81, //bEndpointAddress(EndPoint1 as IN);
    0x03, //bmAttributes(Interrupt);
    0x40,0x00, //wMaxPacketSize(64);
    0x01, //bInterval(10ms);

    0x07, //bLength(7);
    0x05, //bDescriptorType(Endpoint);
    0x01, //bEndpointAddress(EndPoint1 as OUT);
    0x03, //bmAttributes(Interrupt);
    0x40,0x00, //wMaxPacketSize(64);
    0x01, //bInterval(10ms);
};

char code HIDREPORTDESC[27] =
{
    0x05,0x0c, //USAGE_PAGE(Consumer);
    0x09,0x01, //USAGE(Consumer Control);
    0xa1,0x01, //COLLECTION(Application);
    0x15,0x00, // LOGICAL_MINIMUM(0);
    0x25,0xff, // LOGICAL_MAXIMUM(255);
    0x75,0x08, // REPORT_SIZE(8);
    0x95,0x40, // REPORT_COUNT(64);
    0x09,0x01, // USAGE(Consumer Control);
    0xb1,0x02, // FEATURE(Data,Variable);
    0x09,0x01, // USAGE(Consumer Control);
    0x81,0x02, // INPUT(Data,Variable);
    0x09,0x01, // USAGE(Consumer Control);
    0x91,0x02, // OUTPUT(Data,Variable);
    0xc0, //END_COLLECTION;
};

char code LANGIDDESC[4] =
{
    0x04,0x03,
    0x09,0x04,
};

char code MANUFACTDESC[8] =
{
    0x08,0x03,
    'S',0,
    'T',0,

```

```
    'C',0,  
};  
  
char code PRODUCTDESC[30] =  
{  
    0x1e,0x03,  
    'S',0,  
    'T',0,  
    'C',0,  
    ' ',0,  
    'U',0,  
    'S',0,  
    'B',0,  
    ' ',0,  
    'D',0,  
    'e',0,  
    'v',0,  
    'i',0,  
    'c',0,  
    'e',0,  
};
```

---

---

STC MCU

## 24.4.2 HID(Human Interface Device)协议范例

将代码下载到实验箱后, 可使用最新的 STC-ISP 下载软件中的 HID 助手检测测试

详细代码请参考官网上的“STC8H8K64U 实验箱-V9.3 程序包”包中的“60—HID(Human Interface Device) 协议范例”

## 24.4.3 CDC(Communication Device Class)协议范例

WIN10 以下的操作系统需要安装 sys 目录中的驱动程序, WIN10 和 WIN11 免安装驱动

将代码下载到实验箱后, 在 PC 端可识别为 USB 转串口的设备

使用实验箱上的 DB9 接口即可与其它串口进行通讯

串口的数据位只支持 8 位, 停止位只支持 1 位

校验位可支持: 无校验、奇校验、偶校验、1 校验和 0 校验

波特率最高可支持 460800, 且支持自定义波特率

详细代码请参考官网上的“STC8H8K64U 实验箱-V9.3 程序包”包中的“61—CDC(Communication Device Class)协议范例”

## 24.4.4 基于 HID 协议的 USB 键盘范例

将代码下载到实验箱后即可实现 USB 键盘的基本功能

跑马灯中的 LED17 为 NumLock 灯、LED16 为 CapsLock 灯、LED15 为 ScrollLock 灯

矩阵按键中的 KEY0~KEY7 分别为键盘中的 1~8

详细代码请参考官网上的“STC8H8K64U 实验箱-V9.3 程序包”包中的“62—基于 HID 协议的 USB 键盘范例”

## 24.4.5 基于 HID 协议的 USB 鼠标范例

将代码下载到实验箱后即可实现 USB 鼠标的基本功能

矩阵按键中的 KEY0 为鼠标左键, KEY1 为鼠标中键, KEY2 为鼠标右键

矩阵按键中的 KEY4 为左移, KEY5 为右移, KEY6 为上移, KEY7 为下移

详细代码请参考官网上的“STC8H8K64U 实验箱-V9.3 程序包”包中的“63—基于 HID 协议的 USB 鼠标范例”

## 24.4.6 基于 WINUSB 协议的范例

WIN10 以下的操作系统需要安装 sys 目录中的驱动程序, WIN10 和 WIN11 免安装驱动

可使用 exe 目录下的“STC\_WINUSB.exe”进行测试

详细代码请参考官网上的“STC8H8K64U 实验箱-V9.3 程序包”包中的“64—基于 WINUSB 协议的范例”

## 24.4.7 MSC(Mass Storage Class)协议范例

将代码下载到实验箱后，在 PC 端可识别为一个 512K 的 U 盘

U 盘存储器为实验箱上的外挂 512K 的串行 FLASH

详细代码请参考官网上的“STC8H8K64U 实验箱-V9.3 程序包”包中的“65—MSC(Mass Storage Class) 协议范例”

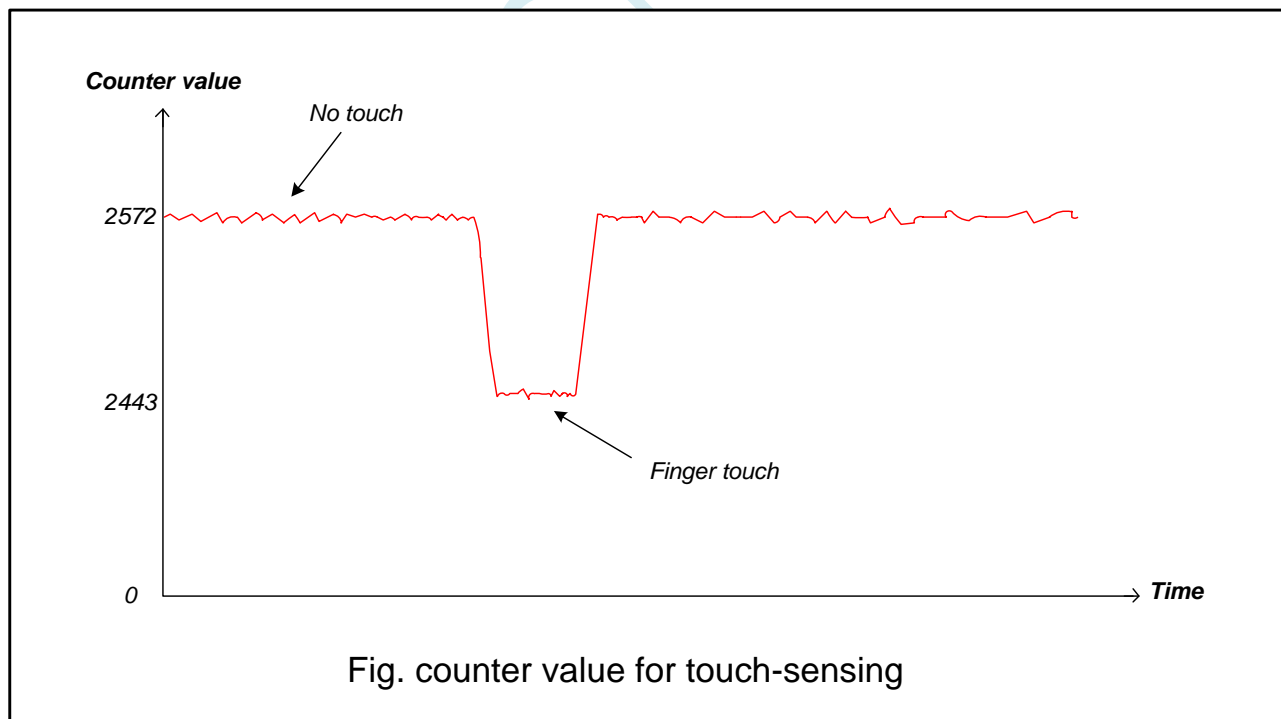
STC MCU

## 25 触摸按键控制器，先工程样片，可少量送样

| 产品线              | 触摸按键 |
|------------------|------|
| STC8H1K08 系列     |      |
| STC8H1K28 系列     |      |
| STC8H3K64S4 系列   |      |
| STC8H3K64S2 系列   |      |
| STC8H8K64U 系列    |      |
| STC8H4K64TL 系列   | ●    |
| STC8H4K64TLCD 系列 | ●    |
| STC8H1K08T 系列    | ●    |

STC8H 系列部分型号的单片机内部集成了一个触摸按键控制器 (TSU)，最大能连接 16 个按键，能够侦测手指触摸于按键电极后导致的微小电容变化，并将之量化为 16-bit 的数字。从原理上看来，TSU 模块与 16-bit ADC 雷同，差异仅是 ADC 为侦测及量化仿真电压或电流，而 TSU 是侦测及量化电容大小。触摸按键的感测，应用时需要外加一个约 10nF 至 47nF 之间的固定电容当参考，如果应用环境温度变化快速且激烈的环境下，应选用低温飘特性电容，才能保持 TSU 输出数据不会有太大变化，避免软件误判。

下图为一说明，对某一个按键做连续感测，在没有手指碰触时，计数器输出值为 2572 附近，当有手指触摸后，计数器输出变成 2443 附近，有无手指触摸的计数器输出值差异约 5%



TSU 模块最大能连接 16 个按键，通过 TSCHEN1 及 TSCHEN2 两个八位寄存器能对每一个触摸按键口独立使能，不做触摸按键的 I/O 口，仍能维持其原本 GPIO 或特殊功能如 LED。开关电容电路的频率，透过 SCR[2:0]来选择，建议频率不要超过 12.5MHz，能避免产生较大的误差，TSU 模块内部比较器的参



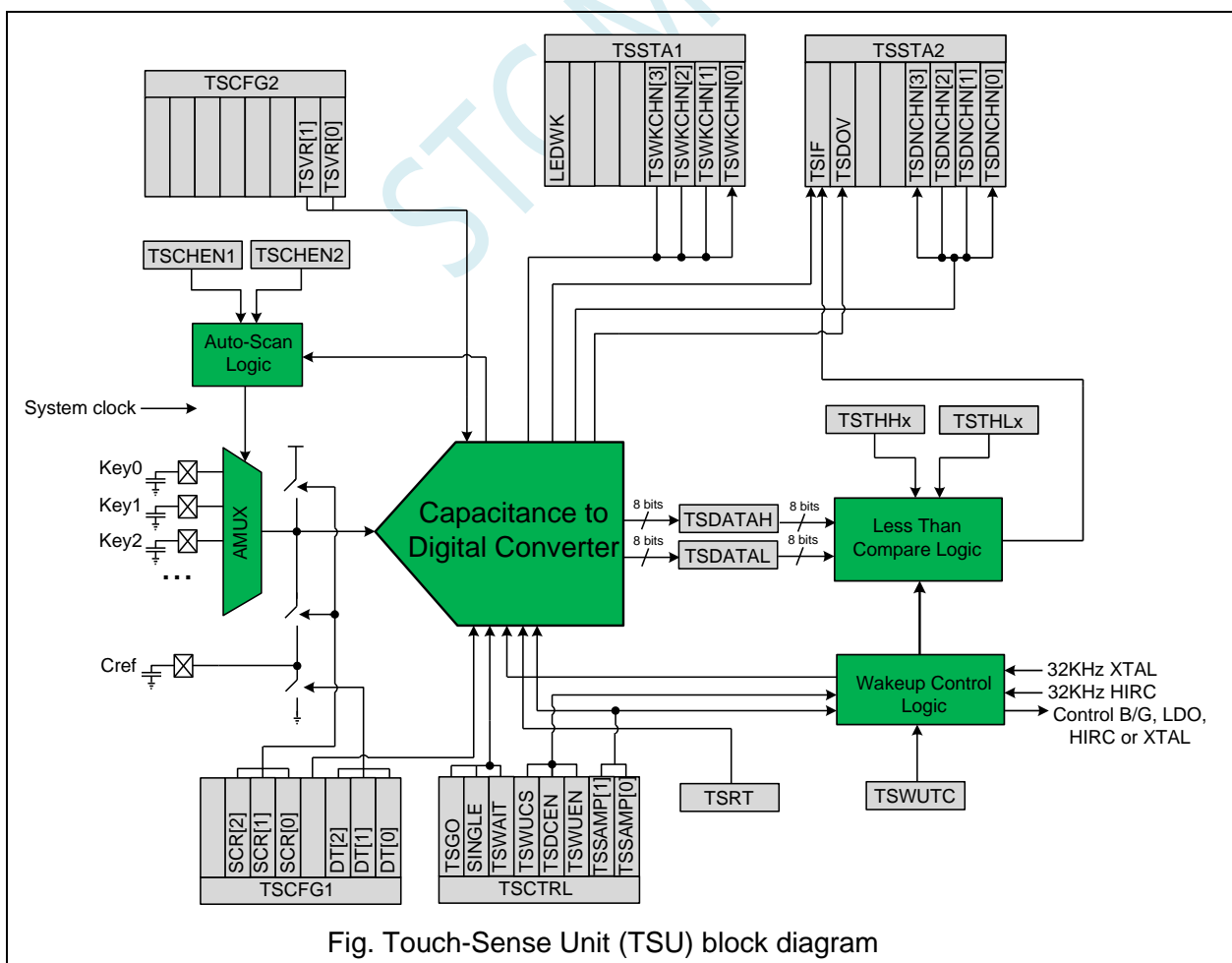
考电压, 透过 TSVR[1:0]来选取, 共有四段可选, 改变参考电压会改变触摸感测时间及灵敏度。

触摸按键扫描可配置成连续不断地循环扫描或是扫描已使能键一轮后就停止, 这功能由 SINGLE 位来控制, 配置寄存器 TSSAMP[1:0]允许 TSU 模块对同一个通道做最多连续 4 次的扫描采样并计算数据的平均值, 这有硬件滤波的效果。而 WAIT 位, 则能让 TSU 模块在 TSIF 标志位为 1 时, 进入等待模式, 直到软件将 WAIT 位清除为 0, TSU 才会接续做按键扫描, 这有助于负担沉重的 CPU 有足够的时间做处理。外部电容 Cref 值推荐 10nF~47nF 范围, Cref 的起始放电至零的时间必需充足, 这可以透过 DT[2:0]弹性调整放电时间。

当完成一个按键的扫描时, 16 位计数器的输出值会被写入 TSDATAH 及 TSDATAL 中, 硬件也同时将标志位 TSIF 设成 1, 同时会把已扫描完成的触摸信道号码写入 TSDNCHN[3:0], 若模块外部中断控制器使能, TSIF 能向 CPU 发出中断请求, 软件能读取 TSDNCHN[3:0]寄存器内容以判别发出 TSIF 中断的是哪一个触摸通道, 而软件读取 TSWKCHN[3:0]与 TSGO 则能实时了解 TSU 模块正在扫描的状态与正在扫描的按键号。如果 16 位计数器发生溢位情形, 硬件会将 TSDOV 标志位设成 1。

TSU 模块能与 LED 驱动电路做 I/O 口分时复用, 当 TSRT 寄存器内容不为零时, 即代表当 TSGO 使能 TSU 工作后是要一起使能 LED 驱动电路并分时复用 I/O 口。因此在 TSGO 使能前, 软件必需先依步配置好 TSU 模块与 LED 驱动电路的相关寄存器, 最后才使能 TSGO。在 TSU/LED 分时多任务模式下, 为了维持 LED 固定的 frame rate, 若分配给按键扫描的时间已到达但正扫描的按键尚未完成, 则该按键将于 LED 分时工作时间终止后, 由重新扫描该未完成按键开始新一轮按键扫描。

## 25.1 触摸按键控制器内部框架图



## 25.2 低功耗触摸唤醒

TSU 模块内部有一专用计时与控制电路,能接受外部 32KHz 晶振或内部 32KHz RC 振荡电路为时钟源,每隔一段时间唤醒 TSU 模块做按键扫描,以 duty control 方式实现低功耗触摸唤醒功能。每一个触摸通道有一个专属的 16 位阈值寄存器{TSTHHx,TSTHLx},当唤醒使能位 TSWUEN 置 1 及数字比较器使能位 TSDCEN 置 1 后 MCU 进入 power-down 状态,整个芯片进入低功耗触摸唤醒模式,TSU 模块能重复不断地规则性短暂自我唤醒做按键扫描以便唤醒 CPU,当按键扫描的数据结果(或者数据平均结果)小于设定的阈值时,硬件会将 TSIF 置 1 并唤醒 CPU 离开 power-down 状态。TSU 模块内部有一硬件平均电路,能够对同一通道最多连续四次的扫描采样数据结果做平均计算,这透过 TSSAMP[1:0]寄存器来配置采样的次数,而{TSDATAH,TSDATAL}则存放着数据平均值。

## 25.3 仅做触摸按键功能时的操作步骤

1. 选择需要扫描的信道,寄存器为 TSCHEN1 及 TSCHEN2
2. 设置 TSRT 内容为 0x00,代表没有开启 LED 驱动分时多任务功能
3. 依据 Cref 大小及触摸按键电容大小配置开关频率 SCR[2:0]与放电时间 DT[2:0],根据所需要的扫描时间及灵敏度选择内部比较器参考电压 TSVR[1:0]
4. 配置 SINGLE 位来决定是扫描一轮自动停止或者是连续不停扫描,配置 TSSAMP[1:0]能让一通道的重复采样最多四次,若 CPU 任务繁重,配置 TSWAIT 以利用 TSIF 状态来延时下一个信道的扫描
5. 必要的话,配置 TSDCEN 以使能内部数字比较功能
6. 设置 TSGO=1,触摸按键开始扫描,软件读取 TSWKCHN[3:0]来了解当下在扫描那一个信道,每扫完一个通道,硬件会将 TSIF 设置成 1,同时完成的信道号码会写入 TSDNCHN[3:0]中,若有溢位发生,TSOV 也会被设置成 1,软件应读取这些寄存器以决定接下来怎么做。TSIF 及 TSOV 仅能由硬件设置及软件清除
7. 若 SINGLE=1,一轮扫完后硬件将自动将 TSGO 清除并结束扫描,否则 TSGO 维持在 1,继续新一轮扫描。
8. 若软件想停止触摸按键扫描,可随时将 TSGO 设置成 0。

## 25.4 低功耗触摸按键唤醒功能的操作步骤

1. 选择需要扫描的信道,寄存器为 TSCHEN1 及 TSCHEN2
2. 务必设置 TSRT 内容为 0x00,此时不能开启 LED 驱动分时多任务功能
3. 配置开关频率 SCR[2:0],放电时间 DT[2:0]及选择内部比较器参考电压 TSVR[1:0]
4. 配置 SINGLE 位为 0 做连续不停扫描,配置 TSWAIT 为 0。
5. 配置 TSWUCS,决定唤醒控制器的时钟来源是外部 32KHz 晶振或内部 32KHz IRC。
6. 配置 TSWUTC,以决定 TSU 每隔多久需要起来干活,干活完就自动进入省电模式
7. 配置 TSSAMP[1:0]以决定每一个信道的扫描采样次数,配置 TSDCEN=1 使能内部数字比较功能
8. 配置好每一个信道的唤醒阈值{TSTHHx,TSTHLx},这阈值会与扫描结果的平均值做比较
9. 使能 TSWUEN=1,设置 TSPD=1 以关闭 TSU 模块模拟电源,以及使能 TSIF 能唤醒 CPU,然后让 MCU 进入 power-down 状态。MCU 一旦进入 power-down 状态,TSU 内部的唤醒控制器即开始工作,周期性控制 TSU 模块的电源开关、按键扫描、数据比较等等。
10. 当数据结果低于所设定的阈值,硬件会将 TSIF 设成 1,将该按键号码写入 TSDNCHN[3:0]中 CPU 被唤醒,低功耗触摸唤醒模式结束。

11. CPU 被唤醒后,除了可以直接读取 TSDNCHN[3:0]认定是那个按键有触摸外,也可以在正常工作模式再施行按键扫描以确认是否唤醒是因为噪声干扰所致。

## 25.5 触摸按键相关的寄存器

| 符号      | 描述             | 地址    | 位地址与符号 |          |        |        |              |         |             |       | 复位值       |
|---------|----------------|-------|--------|----------|--------|--------|--------------|---------|-------------|-------|-----------|
|         |                |       | B7     | B6       | B5     | B4     | B3           | B2      | B1          | B0    |           |
| TSCHEN1 | 触摸按键使能寄存器 1    | FB40H | TKEN7  | TKEN6    | TKEN5  | TKEN4  | TKEN3        | TKEN2   | TKEN1       | TKEN0 | 0000,0000 |
| TSCHEN2 | 触摸按键使能寄存器 2    | FB41H | TKEN15 | TKEN14   | TKEN13 | TKEN12 | TKEN11       | TKEN10  | TKEN9       | TKEN8 | 0000,0000 |
| TSCFG1  | 触摸按键配置寄存器 1    | FB42H | -      | SCR[2:0] |        |        | -            | DT[2:0] |             |       | x000,x000 |
| TSCFG2  | 触摸按键配置寄存器 2    | FB43H | -      | -        | -      | -      | -            | -       | TSVR[1:0]   |       | xxxx,xx00 |
| TSWUTC  | 触摸按键唤醒控制寄存器    | FB44H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSCTRL  | 触摸按键控制寄存器      | FB45H | TSGO   | SINGLE   | TSWAIT | TSWUCS | TSDCEN       | TSWUEN  | TSSAMP[1:0] |       | 0000,0000 |
| TSSTA1  | 触摸按键状态寄存器 1    | FB46H | LEDWK  | -        | -      | -      | TSWKCHN[3:0] |         |             |       | 0xxx,0000 |
| TSSTA2  | 触摸按键状态寄存器 2    | FB47H | TSIF   | TSDOV    | -      | -      | TSDNCHN[3:0] |         |             |       | 00xx,0000 |
| TSRT    | 触摸按键时间控制寄存器    | FB48H |        |          |        |        |              |         |             |       | 0000,0001 |
| TSDATH  | 触摸按键数据高字节      | FB49H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSDATL  | 触摸按键数据低字节      | FB4AH |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH00H | 触摸按键 0 门阈值高字节  | FB50H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH00L | 触摸按键 0 门阈值低字节  | FB51H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH01H | 触摸按键 1 门阈值高字节  | FB52H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH01L | 触摸按键 1 门阈值低字节  | FB53H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH02H | 触摸按键 2 门阈值高字节  | FB54H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH02L | 触摸按键 2 门阈值低字节  | FB55H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH03H | 触摸按键 3 门阈值高字节  | FB56H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH03L | 触摸按键 3 门阈值低字节  | FB57H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH04H | 触摸按键 4 门阈值高字节  | FB58H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH04L | 触摸按键 4 门阈值低字节  | FB59H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH05H | 触摸按键 5 门阈值高字节  | FB5AH |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH05L | 触摸按键 5 门阈值低字节  | FB5BH |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH06H | 触摸按键 6 门阈值高字节  | FB5CH |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH06L | 触摸按键 6 门阈值低字节  | FB5DH |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH07H | 触摸按键 7 门阈值高字节  | FB5EH |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH07L | 触摸按键 7 门阈值低字节  | FB5FH |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH08H | 触摸按键 8 门阈值高字节  | FB60H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH08L | 触摸按键 8 门阈值低字节  | FB61H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH09H | 触摸按键 9 门阈值高字节  | FB62H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH09L | 触摸按键 9 门阈值低字节  | FB63H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH10H | 触摸按键 10 门阈值高字节 | FB64H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH10L | 触摸按键 10 门阈值低字节 | FB65H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH11H | 触摸按键 11 门阈值高字节 | FB66H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH11L | 触摸按键 11 门阈值低字节 | FB67H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH12H | 触摸按键 12 门阈值高字节 | FB68H |        |          |        |        |              |         |             |       | 0000,0000 |
| TSTH12L | 触摸按键 12 门阈值低字节 | FB69H |        |          |        |        |              |         |             |       | 0000,0000 |

|         |               |       |  |           |
|---------|---------------|-------|--|-----------|
| TSTH13H | 触摸按键 13 阈值高字节 | FB6AH |  | 0000,0000 |
| TSTH13L | 触摸按键 13 阈值低字节 | FB6BH |  | 0000,0000 |
| TSTH14H | 触摸按键 14 阈值高字节 | FB6CH |  | 0000,0000 |
| TSTH14L | 触摸按键 14 阈值低字节 | FB6DH |  | 0000,0000 |
| TSTH15H | 触摸按键 15 阈值高字节 | FB6EH |  | 0000,0000 |
| TSTH15L | 触摸按键 15 阈值低字节 | FB6FH |  | 0000,0000 |

### 25.5.1 触摸按键使能寄存器 (TSCHENn)

| 符号      | 地址    | B7     | B6     | B5     | B4     | B3     | B2     | B1    | B0    |
|---------|-------|--------|--------|--------|--------|--------|--------|-------|-------|
| TSCHEN1 | FB40H | TKEN7  | TKEN6  | TKEN5  | TKEN4  | TKEN3  | TKEN2  | TKEN1 | TKEN0 |
| TSCHEN2 | FB41H | TKEN15 | TKEN14 | TKEN13 | TKEN12 | TKEN11 | TKEN10 | TKEN9 | TKEN8 |

TKENn: 触摸按键使能位 (n=0~15)

0: 相应的 TKn 管脚为 GPIO

1: 相应的 TKn 管脚为触摸按键

### 25.5.2 触摸按键配置寄存器 (TSCFGn)

| 符号     | 地址    | B7 | B6       | B5 | B4 | B3 | B2      | B1        | B0 |
|--------|-------|----|----------|----|----|----|---------|-----------|----|
| TSCFG1 | FB42H | -  | SCR[2:0] |    |    | -  | DT[2:0] |           |    |
| TSCFG2 | FB43H | -  | -        | -  | -  | -  | -       | TSVR[1:0] |    |

SCR: 配置触摸按键控制器内部的开关电容工作频率 (频率越高充电时间越短)

$$\text{开关电容工作频率} = \frac{\text{系统工作频率}}{2 * (\text{SCR}[2:0] + 1)}$$

DT[2:0]: 配置触摸按键控制器内部的Cref的起始对地放电时间

| DT[2:0] | 放电时间      |
|---------|-----------|
| 000     | 125个系统时钟  |
| 001     | 250个系统时钟  |
| 010     | 500个系统时钟  |
| 011     | 1000个系统时钟 |
| 100     | 2000个系统时钟 |
| 101     | 2500个系统时钟 |
| 110     | 5000个系统时钟 |
| 111     | 7500个系统时钟 |

TSVR[1:0]: 配置触摸按键控制器内部的参考电压

| TSVR[1:0] | 参考电压     |
|-----------|----------|
| 00        | 1/4 AVcc |

|    |                      |
|----|----------------------|
| 01 | 1/2 AV <sub>CC</sub> |
| 10 | 5/8 AV <sub>CC</sub> |
| 11 | 3/4 AV <sub>CC</sub> |

### 25.5.3 触摸按键低电模式唤醒时间控制寄存器 (TSWUTC)

| 符号     | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-------|----|----|----|----|----|----|----|----|
| TSWUTC | FB44H |    |    |    |    |    |    |    |    |

TSWUTC 寄存器用于配置每隔多久唤醒一次触摸按键控制器

$$\text{唤醒频率} = \frac{F_{32K}}{32 * 8 * \text{TSWUTC}[7:0]}$$

例如：若外挂 32.768KHz 的晶振，且 TSWUTC = 0x80，

则触摸按键控制器的唤醒频率则为 32768 / (32 \* 8 \* 0x80) = 1Hz，即 1 秒唤醒一次。

**注意：**若设置的唤醒频率太快，唤醒的时间不足以完成一轮按键扫描，则触摸按键控制器会持续不停的扫描，而无法进入省电模式。

### 25.5.4 触摸按键控制寄存器 (TSCTRL)

| 符号     | 地址    | B7   | B6     | B5     | B4     | B3     | B2     | B1          | B0 |
|--------|-------|------|--------|--------|--------|--------|--------|-------------|----|
| TSCTRL | FB45H | TSGO | SINGLE | TSWAIT | TSWUCS | TSDCEN | TSWUEN | TSSAMP[1:0] |    |

TSGO：触摸按键控制器启动控制

- 0：触摸按键控制器不工作
- 1：触摸按键控制器开始启动。

SINGLE：单次扫描模式控制

- 0：重复扫描模式
- 1：单次扫描模式。当 TSGO=1 时，当完成一轮按键扫描后，硬件自动将 TSGO 清 0 终止扫描。

**注意：**当 TSGO=1，且 TSRT 不为 0 时，代表触摸按键控制器与 LED 驱动器为共享 GPIO 且分时多工，此时 SINGLE 控制位无效。

TSWAIT：触摸按键控制器等待控制

- 0：触摸按键控制器自动重复扫描
- 1：完成一轮扫描后，TSIF 被硬件置 1，此时触摸按键控制器会暂停扫描，直到 TSIF 标志位被清 0 后才开始下一轮的扫描。

TSWUCS：低功耗模式下触摸按键控制器时钟源选择

- 0：低功耗模式下触摸按键控制器时钟源为内部 32K 的 IRC
- 1：低功耗模式下触摸按键控制器时钟源为外部 32K 晶振

TSDCEN：触摸按键控制器内部 16 位数字比较器控制位

- 0：关闭触摸按键控制器内部 16 位数字比较器
- 1：使能触摸按键控制器内部 16 位数字比较器

**注意：**当 TSU 内部数字比较器被使能，只有当触摸感测数据结果 { TSDATAH, TSDATAL } 小于该

对应通道所设定的阈值 { TSTHH<sub>x</sub>, TSTHL<sub>x</sub> } 时, TSIF 才会被设置 1。此功能用于低功耗模式下可实行触摸唤醒 CPU

TSWUEN: 触摸按键控制器低功耗唤醒使能位

0: 关闭触摸按键控制器低功耗唤醒功能

1: 使能触摸按键控制器低功耗唤醒功能。使能后当 MCU 进入 power-down 状态, 旋即进入触摸按键低功耗唤醒 MCU 模式。在此模式下, 触摸按键控制器内部低功耗时序控制电路将周期性使能 TSU 做按键扫描, 以 duty control 维持极低的平均电流。TSWUEN 只有在 MCU 进入 power-down mode 以及 TSIF 为 0 时才会真正有作用。

**注意:** 此芯片内部有两个 32K 的振荡器, 一个是外部的 32K 晶振, 另一个是内部的 IRC32K 振荡器。在非 STOP-mode 的情况下: 内部 IRC32K 的使能机制单纯是透过 XFR:IRC32KCR[7]; 外部的 X32K 的使能机制单纯是透过 XFR:X2KCR[7]。在 STOP-mode 的情况下: 内部 IRC32K 的使能机制除了 XFR:IRC32KCR[7] 必须被设 1 外, 还必须 SFR:ENWKT 或是 XFR:TSWUEN 被设 1; 外部的 X32K 的使能机制除了 XFR:X32KCR[7] 必须被设 1 外, 还必须 XFR:TSWUEN 被设 1。强调的重点是: TSWUEN 在 STOP-mode 的情况下还扮演启动或是不启动 32K 振荡器的角色。

TSSAMP[1:0]: 单一触摸通道重复扫描次

| TSSAMP [1:0] | 重复扫描次 |
|--------------|-------|
| 00           | 1次    |
| 01           | 2次    |
| 10           | 3次    |
| 11           | 4次    |

**注意:** 中断标志位 TSIF 只有当扫描同一按键的次数到达 TSSAMP 的配置时才会被置 1, 这时 {TSDATAH, TSDATAL} 里是写入的是数次结果的平均值。但若有任一次溢位发生, 硬件会将 TSDOV 置 1。在 TSWUEN 使能的情况下, 平均值必须小于阈值内容才会导致中断标志位 TSIF 置 1, 才能唤醒 CPU。

## 25.5.5 触摸按键状态寄存器 1 (TSSTA1)

| 符号     | 地址    | B7    | B6 | B5 | B4 | B3           | B2 | B1 | B0 |
|--------|-------|-------|----|----|----|--------------|----|----|----|
| TSSTA1 | FB46H | LEDWK | -  | -  | -  | TSWKCHN[3:0] |    |    |    |

LEDWK: 触摸按键控制器与LED驱动器分时多工时的工作状态

0: LED 驱动器处于等待状态, 触摸按键控制器处于工作状态

1: LED 驱动器处于工作状态, 触摸按键控制器处于等待状态

TSWKCHN [3:0]: 触摸通道扫描状态

| TSWKCHN [3:0] | 触摸通道状态       |
|---------------|--------------|
| 0000          | 触摸通道 0 正在被扫描 |
| 0001          | 触摸通道 1 正在被扫描 |
| 0010          | 触摸通道 2 正在被扫描 |
| 0011          | 触摸通道 3 正在被扫描 |
| 0100          | 触摸通道 4 正在被扫描 |



|      |               |
|------|---------------|
| 0101 | 触摸通道 5 正在被扫描  |
| 0110 | 触摸通道 6 正在被扫描  |
| 0111 | 触摸通道 7 正在被扫描  |
| 1000 | 触摸通道 8 正在被扫描  |
| 1001 | 触摸通道 9 正在被扫描  |
| 1010 | 触摸通道 10 正在被扫描 |
| 1011 | 触摸通道 11 正在被扫描 |
| 1100 | 触摸通道 12 正在被扫描 |
| 1101 | 触摸通道 13 正在被扫描 |
| 1110 | 触摸通道 14 正在被扫描 |
| 1111 | 触摸通道 15 正在被扫描 |

### 25.5.6 触摸按键状态寄存器 2 (TSSTA2)

| 符号     | 地址    | B7   | B6    | B5 | B4 | B3           | B2 | B1 | B0 |
|--------|-------|------|-------|----|----|--------------|----|----|----|
| TSSTA2 | FB47H | TSIF | TSDOV | -  | -  | TSDNCHN[3:0] |    |    |    |

TSIF: 触摸按键通道扫描完成标志 (软件写1清0)

0: 还未完成扫描

1: 当完成 TSSAMP 所设定的扫描次数时, 硬件会将 TSIF 置 1, TSIF 可以对 CPU 发出中断请求。

如果是在低功耗唤醒模式下, 还必需加上扫描数据结果值低于设定的门槛, TSIF 才会被设成 1。

**注意: TSIF 只能由硬件置 1, 软件无法将 TSIF 设成 1。特别注意的是软件写 1 至 TSIF 会将 TSIF 清为 0, 软件写 0 至 TSIF 则对内容毫无影响。当 TSWAIT=1 时且 TSIF 为 1 时, TSU 处于暂停等待状态, 等待 CPU 忙完后将 TSIF 清为 0 后才会继续下一个按键的扫描。**

TSDOV: 按键扫描数据溢出标志 (软件写1清0)

0: 按键扫描数据没有溢出, 扫描数据小于或等于 0xFFFF

1: 按键扫描数据发生溢出, 扫描数据大于 0xFFFF。此时软件配置(如 TSVR)或系统硬件必需调整以免溢位发生。TSDOV 仅能由硬件设 1, 软件必需对 TSDOV 写入 1 方能将其清除, 软件对此位写 0 无任何作用。

TSDNCHN [3:0]: 触摸通道完成状态

| TSDNCHN [3:0] | 触摸通道状态      |
|---------------|-------------|
| 0000          | 触摸通道 0 完成扫描 |
| 0001          | 触摸通道 1 完成扫描 |
| 0010          | 触摸通道 2 完成扫描 |
| 0011          | 触摸通道 3 完成扫描 |
| 0100          | 触摸通道 4 完成扫描 |
| 0101          | 触摸通道 5 完成扫描 |
| 0110          | 触摸通道 6 完成扫描 |
| 0111          | 触摸通道 7 完成扫描 |
| 1000          | 触摸通道 8 完成扫描 |

|      |              |
|------|--------------|
| 1001 | 触摸通道 9 完成扫描  |
| 1010 | 触摸通道 10 完成扫描 |
| 1011 | 触摸通道 11 完成扫描 |
| 1100 | 触摸通道 12 完成扫描 |
| 1101 | 触摸通道 13 完成扫描 |
| 1110 | 触摸通道 14 完成扫描 |
| 1111 | 触摸通道 15 完成扫描 |

### 25.5.7 触摸按键时间控制寄存器 (TSRT)

| 符号   | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------|-------|----|----|----|----|----|----|----|----|
| TSRT | FB48H |    |    |    |    |    |    |    |    |

TSRT 寄存器用于配置触摸按键控制器和 LED 驱动器分时多工。

当 TSRT 不为 00 时, 表示触摸按键控制器和 LED 驱动器为分时多工模式。触摸按键控制器分得的时间长度为  $TSRT * T_{LED}$  ( $T_{LED}$  请参考 LED 驱动说明部分)

### 25.5.8 触摸按键数据寄存器 (TSDAT)

| 符号     | 地址    | B7          | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-------|-------------|----|----|----|----|----|----|----|
| TSDATH | FB49H | TSDAT[15:8] |    |    |    |    |    |    |    |
| TSDATL | FB4AH | TSDAT[7:0]  |    |    |    |    |    |    |    |

TSDAT[15:0]: 触摸按键扫描的数据

### 25.5.9 触摸按键阈值寄存器 (TSTH)

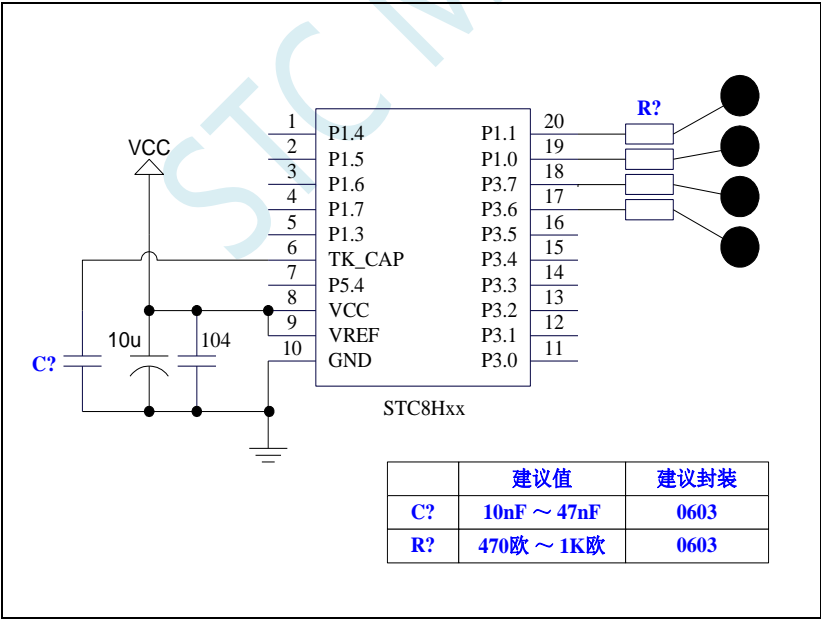
| 符号      | 地址    | B7           | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------|-------|--------------|----|----|----|----|----|----|----|
| TSTH00H | FB50H | TSTH00[15:8] |    |    |    |    |    |    |    |
| TSTH00L | FB51H | TSTH00[7:0]  |    |    |    |    |    |    |    |
| TSTH01H | FB52H | TSTH01[15:8] |    |    |    |    |    |    |    |
| TSTH01L | FB53H | TSTH01[7:0]  |    |    |    |    |    |    |    |
| TSTH02H | FB54H | TSTH02[15:8] |    |    |    |    |    |    |    |
| TSTH02L | FB55H | TSTH02[7:0]  |    |    |    |    |    |    |    |
| TSTH03H | FB56H | TSTH03[15:8] |    |    |    |    |    |    |    |
| TSTH03L | FB57H | TSTH03[7:0]  |    |    |    |    |    |    |    |
| TSTH04H | FB58H | TSTH04[15:8] |    |    |    |    |    |    |    |
| TSTH04L | FB59H | TSTH04[7:0]  |    |    |    |    |    |    |    |
| TSTH05H | FB5AH | TSTH05[15:8] |    |    |    |    |    |    |    |
| TSTH05L | FB5BH | TSTH05[7:0]  |    |    |    |    |    |    |    |
| TSTH06H | FB5CH | TSTH06[15:8] |    |    |    |    |    |    |    |
| TSTH06L | FB5DH | TSTH06[7:0]  |    |    |    |    |    |    |    |
| TSTH07H | FB5EH | TSTH07[15:8] |    |    |    |    |    |    |    |
| TSTH07L | FB5FH | TSTH07[7:0]  |    |    |    |    |    |    |    |
| TSTH08H | FB60H | TSTH08[15:8] |    |    |    |    |    |    |    |



|         |       |              |
|---------|-------|--------------|
| TSTH08L | FB61H | TSTH08[7:0]  |
| TSTH09H | FB62H | TSTH09[15:8] |
| TSTH09L | FB63H | TSTH09[7:0]  |
| TSTH10H | FB64H | TSTH10[15:8] |
| TSTH10L | FB65H | TSTH10[7:0]  |
| TSTH11H | FB66H | TSTH11[15:8] |
| TSTH11L | FB67H | TSTH11[7:0]  |
| TSTH12H | FB68H | TSTH12[15:8] |
| TSTH12L | FB69H | TSTH12[7:0]  |
| TSTH13H | FB6AH | TSTH13[15:8] |
| TSTH13L | FB6BH | TSTH13[7:0]  |
| TSTH14H | FB6CH | TSTH14[15:8] |
| TSTH14L | FB6DH | TSTH14[7:0]  |
| TSTH15H | FB6EH | TSTH15[15:8] |
| TSTH15L | FB6FH | TSTH15[7:0]  |

TSTHn[15:0]: 触摸按键扫描数据门坎值。当数字比较器被使能后, 只有当扫描数据低于此门坎值, TSIF 才会被硬件置 1。

## 25.6 基本参考电路图以及注意事项



注意: 参考线路图中, C?为触摸按键的灵敏度调节电容 (不要使用陶瓷电容, 强烈建议使用温度误差较小的独石电容), R?为ESD 保护电阻, 在PCB 布板时, C?与R?请务必尽量靠近IC 管脚。

## 25.7 范例程序

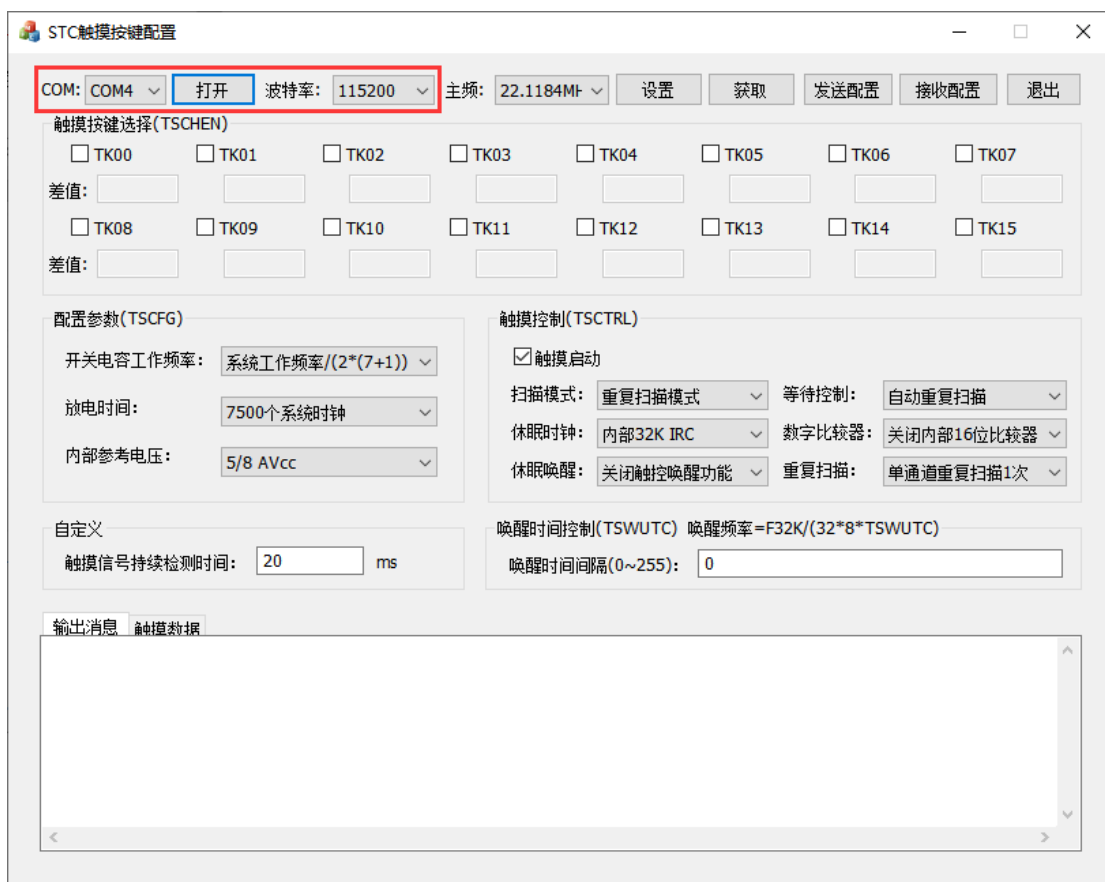
### 25.7.1 触摸按键配置软件介绍

#### 1. 串口设置

“COM”：选择串口号；

“波特率”：设置串口通信的波特率，默认‘115200, N, 8, 1’。如果 MCU 系统主频设置为 5.5296MHz 以下需降低波特率才能正常通信，推荐 9600。MCU 代码需同步修改“Baudrate”定义。

“打开”：设置完串口参数，点击该按钮打开串口，按钮名称变为“关闭”；再次点击则关闭串口，按钮名称变为“打开”。

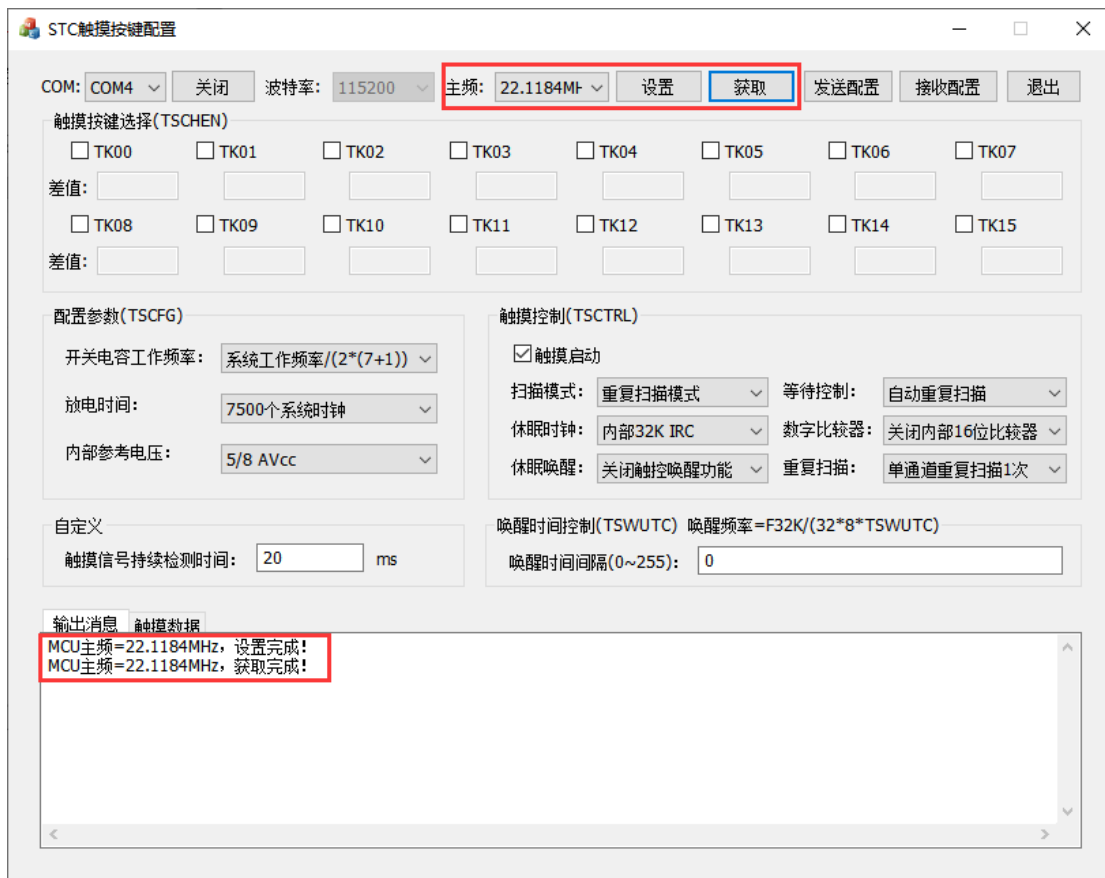


## 2. MCU 主频设置

“主频”：下拉框选择需要设置的 MCU 主频；

“设置”：按钮发送主频设置指令给 MCU，同时输出消息框显示设置主频的消息；

“获取”：按钮发送主频获取指令给 MCU，接收 MCU 返回的主频参数，并在输出消息框显示。



### 3. 触摸配置设置

“**触摸按键选择**”：将需要设置为触摸按键功能的通道勾选，“差值”为触摸按键按下前后的键值差，MCU 以差值的一半作为阈值，判断触摸按键是否按下；

“**配置参数**”、“**触摸控制**”：通过下拉框、勾选框设置对应的寄存器参数；

“**自定义**”：可设置触摸信号持续检测时间与唤醒时间间隔；

“**发送配置**”：点击该按钮发送触摸按键配置参数给 MCU，同时输出消息框显示触摸按键寄存器参数配置的消息；

“**接收配置**”：点击该按钮发送获取触摸按键配置给 MCU，接收 MCU 返回的配置参数，并在输出消息框显示获取的触摸按键寄存器参数配置的消息。

STC 触摸按键配置

COM: COM4 关闭 波特率: 115200 主频: 22.1184MHz 设置 获取 发送配置 接收配置 退出

2 发送配置参数指令

触摸按键选择 (TSCHEN)

☒ TK00 ☒ TK01 ☐ TK02 ☐ TK03 ☐ TK04 ☐ TK05 ☐ TK06 ☐ TK07

差值: 940 890

☐ TK08 ☐ TK09 ☐ TK10 ☐ TK11 ☐ TK12 ☐ TK13 ☐ TK14 ☒ TK15

差值: 1854

配置参数 (TSCFG)

开关电容工作频率: 系统工作频率/(2\*(7+1))

放电时间: 7500个系统时钟

内部参考电压: 5/8 AVcc

自定义

触摸信号持续检测时间: 20 ms

触摸控制 (TSCTRL)

1 设置触摸按键参数

☒ 触摸启动

扫描模式: 重复扫描模式 等待控制: 自动重复扫描

休眠时钟: 内部32K IRC 数字比较器: 关闭内部16位比较器

休眠唤醒: 关闭触控唤醒功能 重复扫描: 单通道重复扫描1次

唤醒时间控制 (TSWUTC) 唤醒频率=F32K/(32\*8\*TSWUTC)

唤醒时间间隔(0~255): 0

输出消息 触摸数据

3 输出配置参数消息

TSCHEN1 = 0x03; TSCHEN2 = 0x80; TSCFG1 = 0x7F; TSCFG2 = 0x02; TSCTRL = 0x80; TSRT = 0x00; TSWUTC = 0; 检测时间 = 20

TK00差值 = 940; TK01差值 = 890; TK02差值 = 0; TK03差值 = 0; TK04差值 = 0; TK05差值 = 0; TK06差值 = 0; TK07差值 = 0;

TK08差值 = 0; TK09差值 = 0; TK10差值 = 0; TK11差值 = 0; TK12差值 = 0; TK13差值 = 0; TK14差值 = 0; TK15差值 = 1854;

设置完成!

TSCHEN1 = 0x03; TSCHEN2 = 0x80; TSCFG1 = 0x7F; TSCFG2 = 0x02; TSCTRL = 0x80; TSRT = 0x00; TSWUTC = 0; 检测时间 = 20

TK00差值 = 940; TK01差值 = 890; TK02差值 = 800; TK03差值 = 800; TK04差值 = 800; TK05差值 = 800; TK06差值 = 800; TK07差值 = 800;

TK08差值 = 800; TK09差值 = 800; TK10差值 = 800; TK11差值 = 800; TK12差值 = 800; TK13差值 = 800; TK14差值 = 800; TK15差值 = 800;

获取完成!

## 26 LED 驱动器

| 产品线              | LED 驱动 |
|------------------|--------|
| STC8H1K08 系列     |        |
| STC8H1K28 系列     |        |
| STC8H3K64S4 系列   |        |
| STC8H3K64S2 系列   |        |
| STC8H8K64U 系列    |        |
| STC8H4K64TL 系列   | ●      |
| STC8H4K64TLCD 系列 |        |
| STC8H1K08T 系列    |        |

STC8H 系列部分型号的单片机内部集成了一个 LED 驱动器。

LED 驱动电路包含一个时序控制器，8 个 COM 输出引脚及 16 个 SEGMENT 输出引脚。每一个引脚有一个对应的寄存器使能位，能独立控制该引脚使能与否，没使能的引脚能当做 GPIO 或其它功能的引脚。

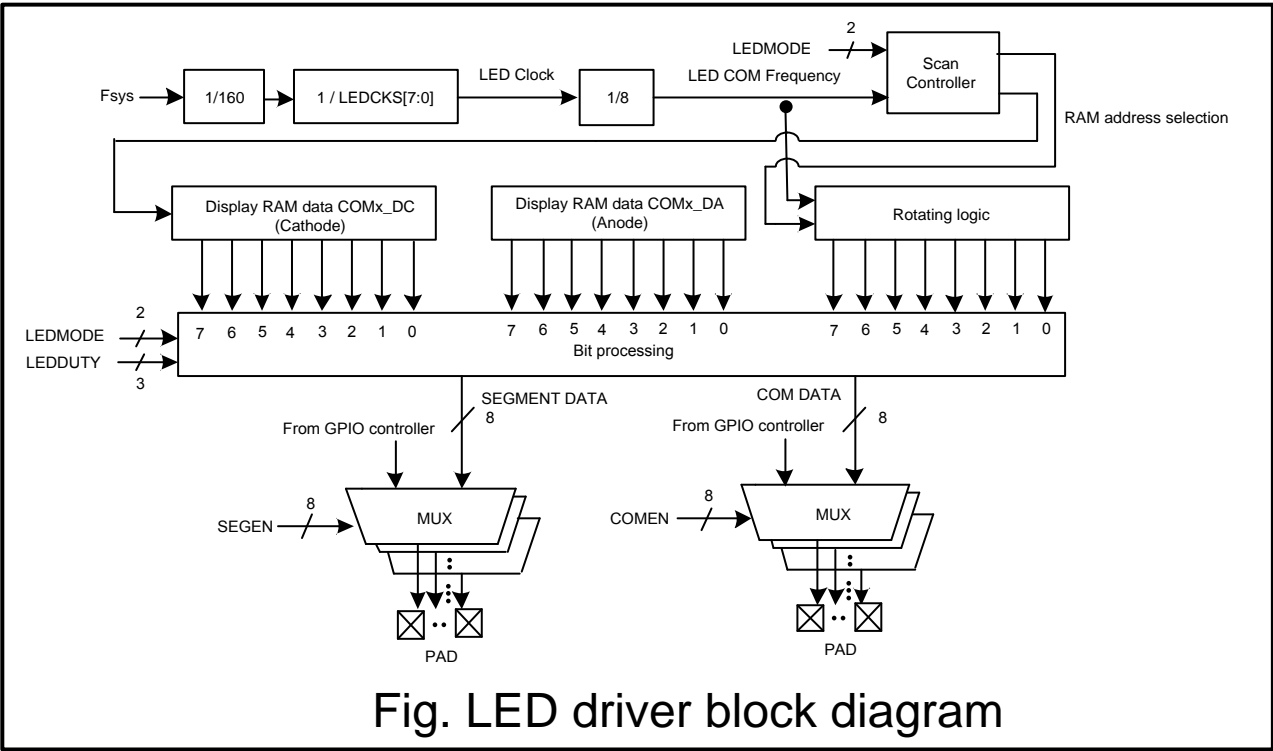
**16 个 SEGMENT 和 8 个 COM，可以分时当共阴/共阳使用，相当于 16 个 COM，可以最高支持到 32 个 8-LED 数码管**

LED 驱动支持共阴、共阳、共阴/共阳三种模式，同时能选择 1/8 ~ 8/8 占空比来调节灰度，因此仅需透过软件即可调节 LED 及数码管的亮度。

上电复位后，LEDON 使能位为 0，LED 驱动电路关闭。配置 LEDON 为 1 使能 LED 驱动电路，当 LEDMODE=00 时，驱动电路工作于共阴模式，此时被选中的 COM 输出低电平，被选中的 SEGMENT 且希望 LED 发亮则输出高电平，因此 SEGMENT 与 COM 两端点间的 LED 顺向偏压导通发亮。同理，当 LEDMODE=01 时，驱动电路工作于共阳模式，此时被选中的 COM 输出高电平，被选中的 SEGMENT 且希望 LED 发亮则输出低电平，因此 SEGMENT 与 COM 两端点间的 LED 顺向偏压导通发亮。当 LEDMODE=10 时，驱动电路工作于共阴/共阳分时驱动模式，COM 的电平为低电平与高电平时交错，LED 导通发亮原理与共阴、共阳是相同的。

共阴模式及共阳模式下 display RAM 是独立的地址，共阴/共阳分时多任务模式下 display RAM 的位置也是从独立的地址去读取。

## 26.1 LED 驱动器内部框架图



## 26.2 LED 驱动相关的寄存器

| 符号        | 描述          | 地址    | 位地址与符号 |       |              |       |       |              |      |      | 复位值       |
|-----------|-------------|-------|--------|-------|--------------|-------|-------|--------------|------|------|-----------|
|           |             |       | B7     | B6    | B5           | B4    | B3    | B2           | B1   | B0   |           |
| COMEN     | COM 使能寄存器   | FB00H | C7EN   | C6EN  | C5EN         | C4EN  | C3EN  | C2EN         | C1EN | C0EN | 0000,0000 |
| SEGENL    | SEG 使能寄存器   | FB01H | S7EN   | S6EN  | S5EN         | S4EN  | S3EN  | S2EN         | S1EN | S0EN | 0000,0000 |
| SEGENH    | SEG 使能寄存器   | FB02H | S15EN  | S14EN | S13EN        | S12EN | S11EN | S10EN        | S9EN | S8EN | 0000,0000 |
| LEDCTRL   | LED 控制寄存器   | FB03H | LEDON  | -     | LEDMODE[1:0] |       | -     | LEDDUTY[2:0] |      |      | 0x00,x000 |
| LEDCKS    | LED 时钟分频寄存器 | FB04H |        |       |              |       |       |              |      |      | 0000,0001 |
| COM0_DA_L | 共阳模式显示数据    | FB10H |        |       |              |       |       |              |      |      | 0000,0000 |
| COM1_DA_L | 共阳模式显示数据    | FB11H |        |       |              |       |       |              |      |      | 0000,0000 |
| COM2_DA_L | 共阳模式显示数据    | FB12H |        |       |              |       |       |              |      |      | 0000,0000 |
| COM3_DA_L | 共阳模式显示数据    | FB13H |        |       |              |       |       |              |      |      | 0000,0000 |
| COM4_DA_L | 共阳模式显示数据    | FB14H |        |       |              |       |       |              |      |      | 0000,0000 |
| COM5_DA_L | 共阳模式显示数据    | FB15H |        |       |              |       |       |              |      |      | 0000,0000 |
| COM6_DA_L | 共阳模式显示数据    | FB16H |        |       |              |       |       |              |      |      | 0000,0000 |
| COM7_DA_L | 共阳模式显示数据    | FB17H |        |       |              |       |       |              |      |      | 0000,0000 |
| COM0_DA_H | 共阳模式显示数据    | FB18H |        |       |              |       |       |              |      |      | 0000,0000 |
| COM1_DA_H | 共阳模式显示数据    | FB19H |        |       |              |       |       |              |      |      | 0000,0000 |
| COM2_DA_H | 共阳模式显示数据    | FB1AH |        |       |              |       |       |              |      |      | 0000,0000 |
| COM3_DA_H | 共阳模式显示数据    | FB1BH |        |       |              |       |       |              |      |      | 0000,0000 |

|           |          |       |  |           |
|-----------|----------|-------|--|-----------|
| COM4_DA_H | 共阳模式显示数据 | FB1CH |  | 0000,0000 |
| COM5_DA_H | 共阳模式显示数据 | FB1DH |  | 0000,0000 |
| COM6_DA_H | 共阳模式显示数据 | FB1EH |  | 0000,0000 |
| COM7_DA_H | 共阳模式显示数据 | FB1FH |  | 0000,0000 |
| COM0_DC_L | 共阴模式显示数据 | FB20H |  | 0000,0000 |
| COM1_DC_L | 共阴模式显示数据 | FB21H |  | 0000,0000 |
| COM2_DC_L | 共阴模式显示数据 | FB22H |  | 0000,0000 |
| COM3_DC_L | 共阴模式显示数据 | FB23H |  | 0000,0000 |
| COM4_DC_L | 共阴模式显示数据 | FB24H |  | 0000,0000 |
| COM5_DC_L | 共阴模式显示数据 | FB25H |  | 0000,0000 |
| COM6_DC_L | 共阴模式显示数据 | FB26H |  | 0000,0000 |
| COM7_DC_L | 共阴模式显示数据 | FB27H |  | 0000,0000 |
| COM0_DC_H | 共阴模式显示数据 | FB28H |  | 0000,0000 |
| COM1_DC_H | 共阴模式显示数据 | FB29H |  | 0000,0000 |
| COM2_DC_H | 共阴模式显示数据 | FB2AH |  | 0000,0000 |
| COM3_DC_H | 共阴模式显示数据 | FB2BH |  | 0000,0000 |
| COM4_DC_H | 共阴模式显示数据 | FB2CH |  | 0000,0000 |
| COM5_DC_H | 共阴模式显示数据 | FB2DH |  | 0000,0000 |
| COM6_DC_H | 共阴模式显示数据 | FB2EH |  | 0000,0000 |
| COM7_DC_H | 共阴模式显示数据 | FB2FH |  | 0000,0000 |

### 26.2.1 COM 使能寄存器 (COMEN)

| 符号    | 地址    | B7   | B6   | B5   | B4   | B3   | B2   | B1   | B0   |
|-------|-------|------|------|------|------|------|------|------|------|
| COMEN | FB00H | C7EN | C6EN | C5EN | C4EN | C3EN | C2EN | C1EN | C0EN |

CnEN: COMn使能控制位 (n=0~7)

0: 不使能 COMn, 仍然为 GPIO

1: 使能 COMn, 对应的 I/O 在 LEDON=1 时输出 COM 的驱动波形。

### 26.2.2 SEG 使能寄存器 (SEGEN)

| 符号     | 地址    | B7    | B6    | B5    | B4    | B3    | B2    | B1   | B0   |
|--------|-------|-------|-------|-------|-------|-------|-------|------|------|
| SEGENL | FB01H | S7EN  | S6EN  | S5EN  | S4EN  | S3EN  | S2EN  | S1EN | S0EN |
| SEGENH | FB02H | S15EN | S14EN | S13EN | S12EN | S11EN | S10EN | S9EN | S8EN |

SnEN: SEGn使能控制位 (n=0~15)

0: 不使能 SEGn, 仍然为 GPIO

1: 使能 SEGn, 对应的 I/O 在 LEDON=1 时输出 SEG 的驱动波形。

### 26.2.3 LED 控制寄存器 (LEDCTRL)

| 符号      | 地址    | B7    | B6 | B5           | B4 | B3 | B2           | B1 | B0 |
|---------|-------|-------|----|--------------|----|----|--------------|----|----|
| LEDCTRL | FB03H | LEDON | -  | LEDMODE[1:0] |    | -  | LEDDUTY[2:0] |    |    |

LEDON: LED驱动器使能控制

0: 关闭 LED 驱动器

1: 使能 LED 驱动器。

LEDMODE[1:0]: LED驱动模式

| LEDMODE[1:0] | 驱动模式    |
|--------------|---------|
| 00           | 共阴模式    |
| 01           | 共阳模式    |
| 10           | 共阴/共阳模式 |
| 11           | 保留      |

LEDDUTY[2:0]: LED灰度调节

| LEDDUTY[2:0] | LED占空比 | LED亮度 |
|--------------|--------|-------|
| 000          | 8/8    | 100%  |
| 001          | 7/8    | 87.5% |
| 010          | 6/8    | 75%   |
| 011          | 5/8    | 62.5% |
| 100          | 4/8    | 50%   |
| 101          | 3/8    | 37.5% |
| 110          | 2/8    | 25%   |
| 111          | 1/8    | 12.5% |

26.2.4 LED 时钟分频寄存器（LEDCKS）

| 符号     | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------|-------|----|----|----|----|----|----|----|----|
| LEDCKS | FB04H |    |    |    |    |    |    |    |    |

LEDCKS: LED时钟分频控制。

SYSclk

LED工作频率 =  $\frac{\hspace{1.5cm}}{160 * LEDCKS[7:0]}$

寄存器 LEDCKS 的值若设置太大，会导致 LED 产生闪烁现象，一般 LED 刷新频率大于或等于 75Hz 则不会有明显的闪烁

建议值:  $\frac{\hspace{1.5cm}}{160 * 8 * N_{COM} * LEDCKS[7:0]}$

$\geq 75Hz$

N<sub>COM</sub>: 使能的COM数，若为共阴/共阳模式，则为COM数的两倍

例如：若单片机的工作频率为 11.0592MHz，LED 的工作模式为共阴/共阳模式，COMEN 设置为 0FFH，即使能 8 个 COM，则 11059200 / 160 / 8 / 16 / LEDCKS ≥ 75，得到 LEDCKS ≤ 7.2，所以 LEDCKS 建议设置为 7。



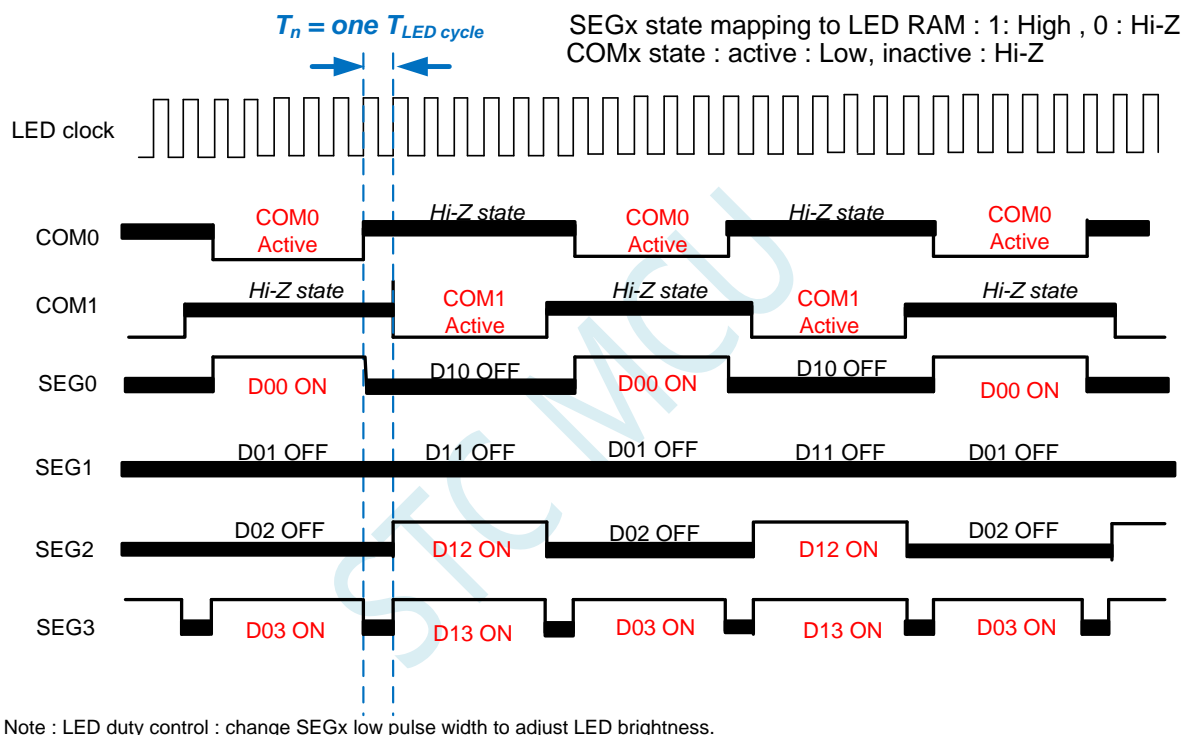
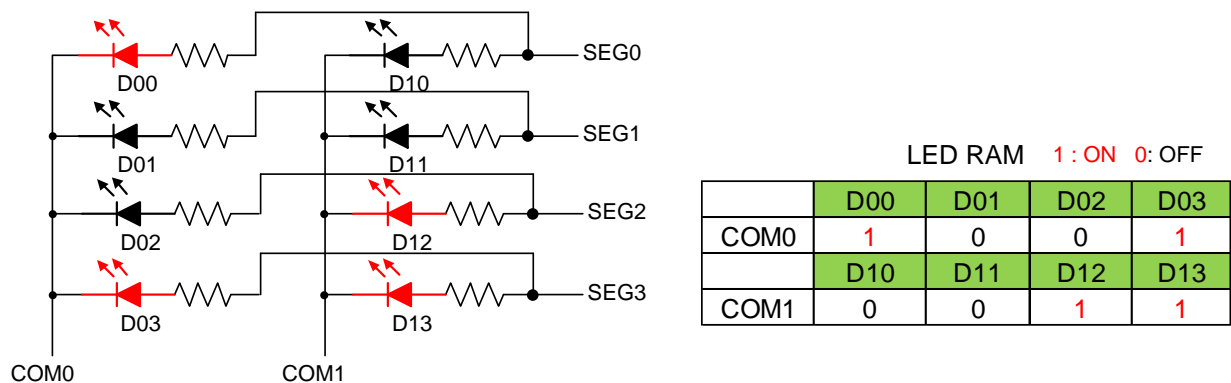
## 26.2.5 LED 共阳模式数据寄存器 (COMn\_DA)

| 符号        | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----------|-------|----|----|----|----|----|----|----|----|
| COM0_DA_L | FB10H |    |    |    |    |    |    |    |    |
| COM1_DA_L | FB11H |    |    |    |    |    |    |    |    |
| COM2_DA_L | FB12H |    |    |    |    |    |    |    |    |
| COM3_DA_L | FB13H |    |    |    |    |    |    |    |    |
| COM4_DA_L | FB14H |    |    |    |    |    |    |    |    |
| COM5_DA_L | FB15H |    |    |    |    |    |    |    |    |
| COM6_DA_L | FB16H |    |    |    |    |    |    |    |    |
| COM7_DA_L | FB17H |    |    |    |    |    |    |    |    |
| COM0_DA_H | FB18H |    |    |    |    |    |    |    |    |
| COM1_DA_H | FB19H |    |    |    |    |    |    |    |    |
| COM2_DA_H | FB1AH |    |    |    |    |    |    |    |    |
| COM3_DA_H | FB1BH |    |    |    |    |    |    |    |    |
| COM4_DA_H | FB1CH |    |    |    |    |    |    |    |    |
| COM5_DA_H | FB1DH |    |    |    |    |    |    |    |    |
| COM6_DA_H | FB1EH |    |    |    |    |    |    |    |    |
| COM7_DA_H | FB1FH |    |    |    |    |    |    |    |    |

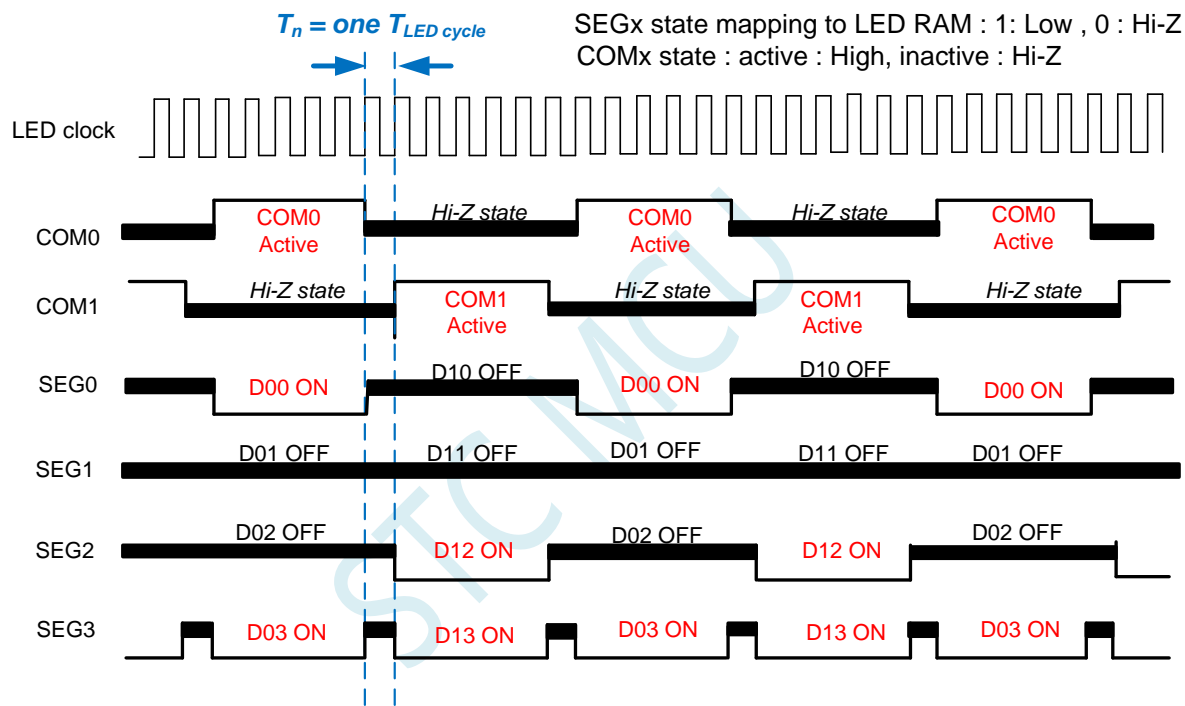
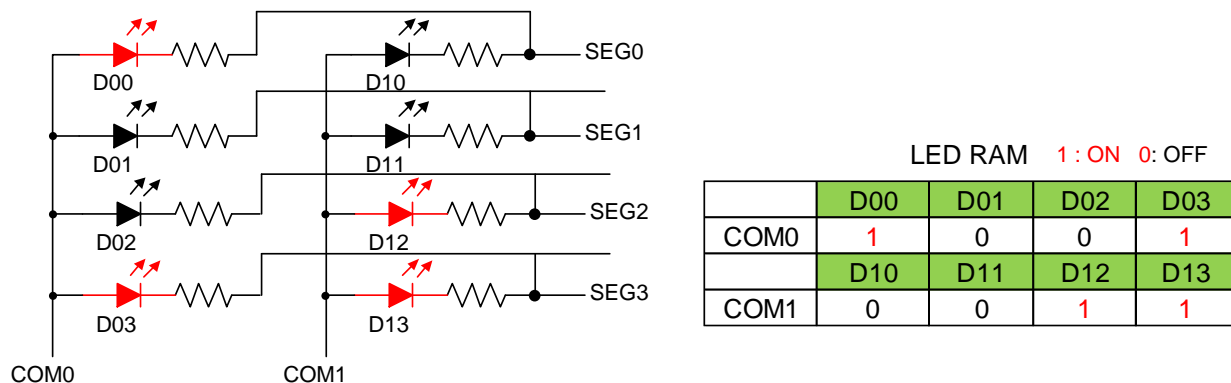
## 26.2.6 LED 共阴模式数据寄存器 (COMn\_DC)

| 符号        | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----------|-------|----|----|----|----|----|----|----|----|
| COM0_DC_L | FB20H |    |    |    |    |    |    |    |    |
| COM1_DC_L | FB21H |    |    |    |    |    |    |    |    |
| COM2_DC_L | FB22H |    |    |    |    |    |    |    |    |
| COM3_DC_L | FB23H |    |    |    |    |    |    |    |    |
| COM4_DC_L | FB24H |    |    |    |    |    |    |    |    |
| COM5_DC_L | FB25H |    |    |    |    |    |    |    |    |
| COM6_DC_L | FB26H |    |    |    |    |    |    |    |    |
| COM7_DC_L | FB27H |    |    |    |    |    |    |    |    |
| COM0_DC_H | FB28H |    |    |    |    |    |    |    |    |
| COM1_DC_H | FB29H |    |    |    |    |    |    |    |    |
| COM2_DC_H | FB2AH |    |    |    |    |    |    |    |    |
| COM3_DC_H | FB2BH |    |    |    |    |    |    |    |    |
| COM4_DC_H | FB2CH |    |    |    |    |    |    |    |    |
| COM5_DC_H | FB2DH |    |    |    |    |    |    |    |    |
| COM6_DC_H | FB2EH |    |    |    |    |    |    |    |    |
| COM7_DC_H | FB2FH |    |    |    |    |    |    |    |    |

## 26.3 LED 共阴模式 (LEDMODE = 00)



## 26.4 LED 共阳模式 (LEDMODE = 01)

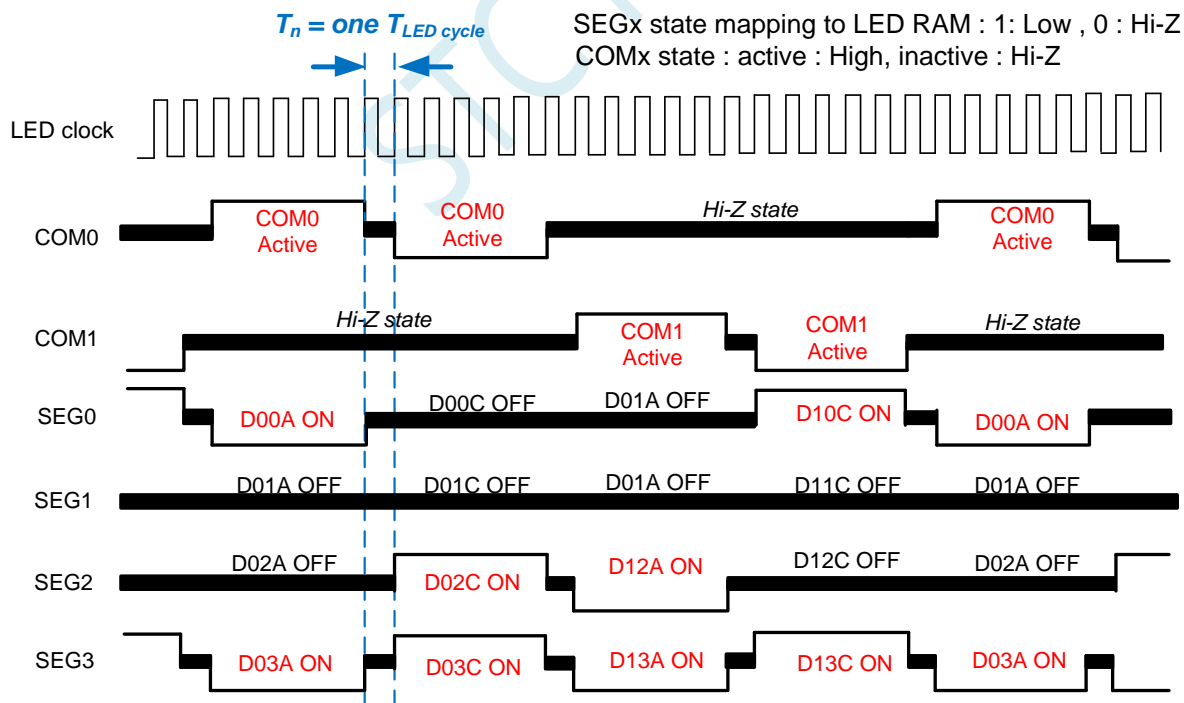
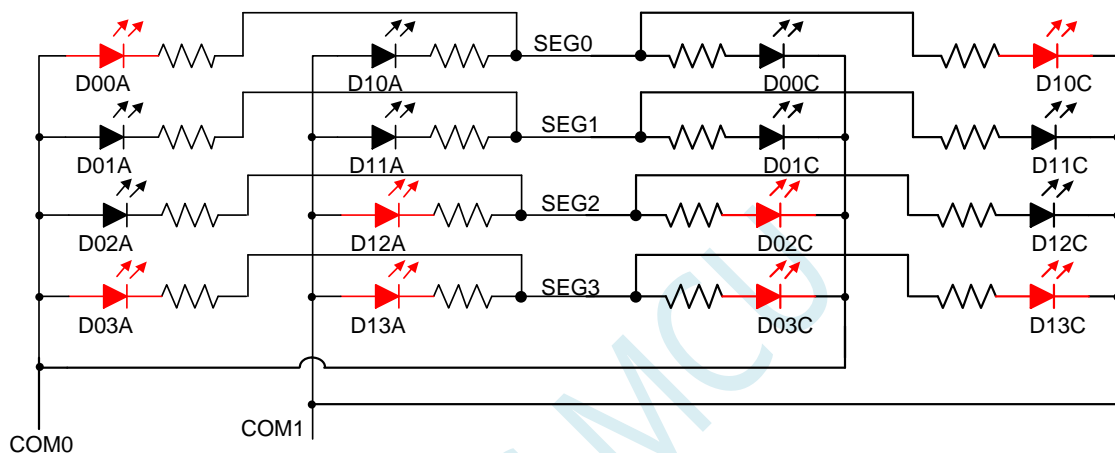


Note: LED duty control: change SEGx low pulse width to adjust LED brightness.

## 26.5 LED 共阴/共阳模式 (LEDMODE = 10)

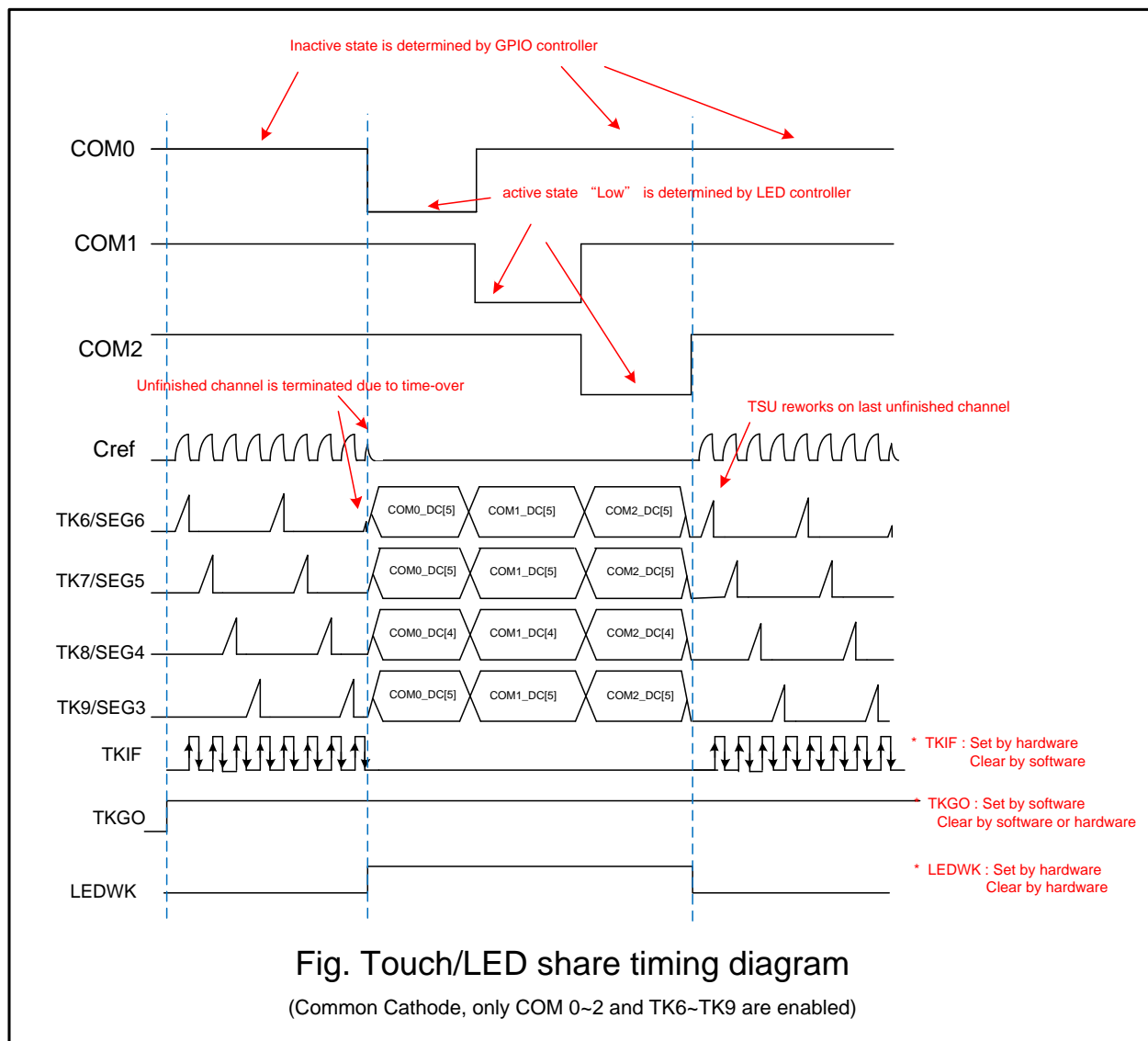
LED RAM 1: ON 0: OFF

|      | D00A | D01A | D02A | D03A |
|------|------|------|------|------|
| COM0 | 1    | 0    | 0    | 1    |
|      | D00C | D01C | D02C | D03C |
| COM0 | 0    | 0    | 1    | 1    |
|      | D10A | D11A | D12A | D13A |
| COM1 | 0    | 0    | 1    | 1    |
|      | D10C | D11C | D12C | D13C |
| COM1 | 1    | 0    | 0    | 1    |



Note : LED duty control : change SEGx low pulse width to adjust LED brightness.

## 26.6 触摸按键与 LED 分时复用 I/O



### 操作步骤:

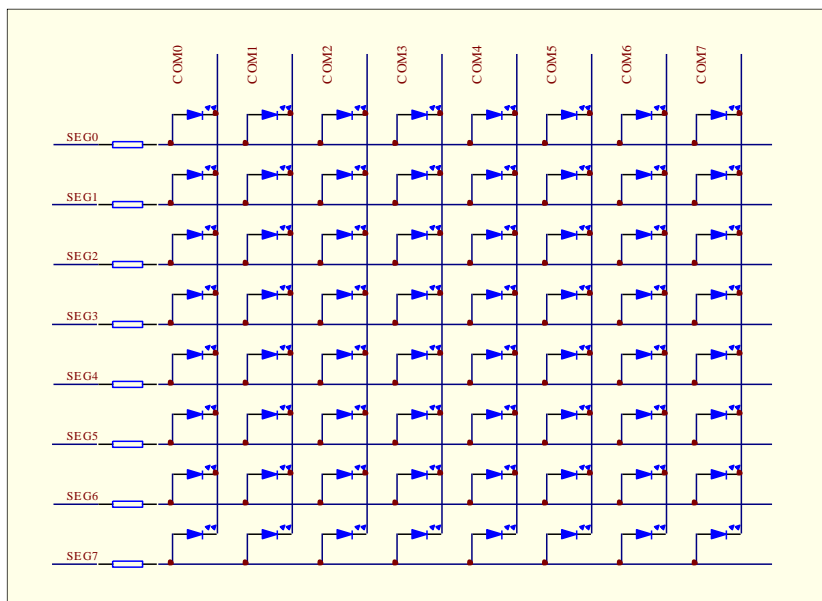
1. 选择需要扫描的触摸按键通道，寄存器为 TSCHEN1 及 TSCHEN2。
2. 配置开关频率 SCR[2:0]，放电时间 DT[2:0]及选择内部比较器参考电压 TSVR[1:0]。
3. 配置 TSSAMP[1:0]以决定同一信道的重复扫描次数，若 CPU 任务繁重，配置 TSWAIT 以利用 TSIF 状态来延时下一个信道的扫描。
4. 必要的话，配置 TSDCEN 以使能内部数字比较功能。
5. 设置 TSRT 内容，当 TSRT 内容不为 0x00 代表没有将开启 LED 驱动分时多任务功能。
6. 配置 SEGEN、COMEN 寄存器。
7. 配置 LEDCKS 以决定每一个 COM 作用的时间长度，这要与 TKRT 寄存器一起考虑以计算出 LED 的刷新率。
8. 根据 LED 所需的工作模式及辉度配置 LEDMODE[1:0]及 LEDDUTY。
9. 写入数据至 LED 的数据寄存器 COMx\_DC 及 COM\_DA 中。
10. 设置 TSGO=1，触摸按键开始扫描。

- a) 软件可读取 TSWKCHN[3:0]来了解当下那一个信道正被扫描, 每扫完一个通道, 硬件会将 TSIF 设置成 1, 同时完成的信道号码会写入 TSDNCHN[3:0]中, 若有溢位发生, TSOV 也会被设置成 1, 软件应读取这些寄存器以决定接下来怎么做。TSIF 及 TSOV 仅能由硬件设置及软件清除。
  - b) 当切换成 LED 工作时间时, 软件读取 LEDWK 位会得到 1, 用以判断当下是触摸按键控制器正在工作或 LED 驱动电路正在工作
  - c) 软件依实际需要, 持续更新 LED 数据寄存器
11. 若软件想终止触摸按键与 LED 分时多任务模式, 需写入 TSGO=0, 则多任务模式终止, 触摸按键与 LED 皆不工作, I/O 口的控制权回归 GPIO 控制器

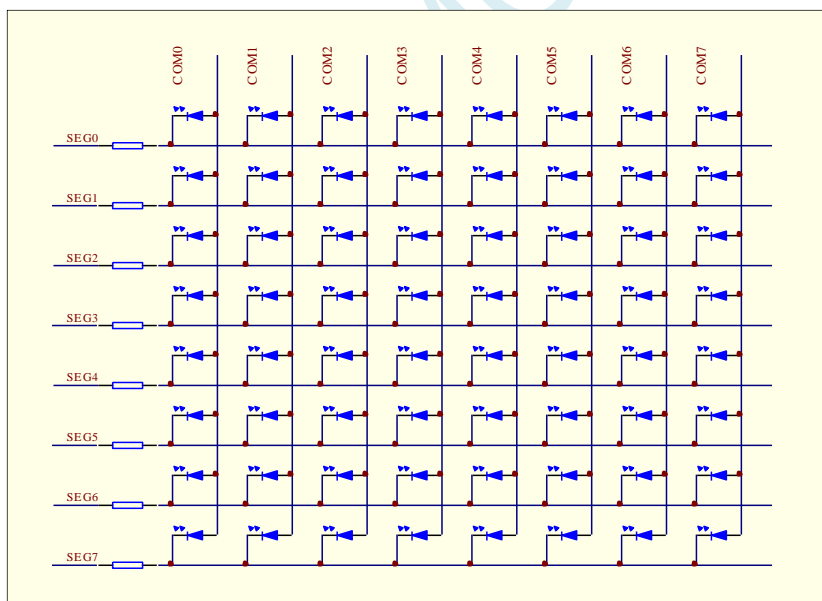
**注意:**

1. SINGLE 控制位在触摸按键与 LED 分时多任务模式下无效, 只有软件写入 TSGO 才能控制模块的开启和关闭
2. 由于触摸按键扫描时间终止并转换为 LED 作用时间时, 最后的那一个按键几乎都没完成, 这时硬件的处理是不会发出 TSIF 也不会更新触摸数据相关的寄存器, 但是硬件会记住该信道号码, 等 LED 作用期间终止并转换为触摸按键扫描时间时, 会由该信道重新扫描来起始新一轮扫描。
3. LED 与触摸复用的电路图如上, 要注意的是 LED 的发光颜色不同, 代表的 LED 等效电容大小会不同, 电容愈大对触摸按键愈不利, 灵敏度会下降。一般而言, 红光 LED 电容值可能为 35pF, 但是黄光会高达 100pF, 这时如果要提高触摸按键的灵敏度, 可以串一个 1N4148 二极管。1N4148 本身电容仅有 4pF, 1N4148 串一个 100pF 的黄光 LED, 对该键的寄生电容会比 4pF 略小

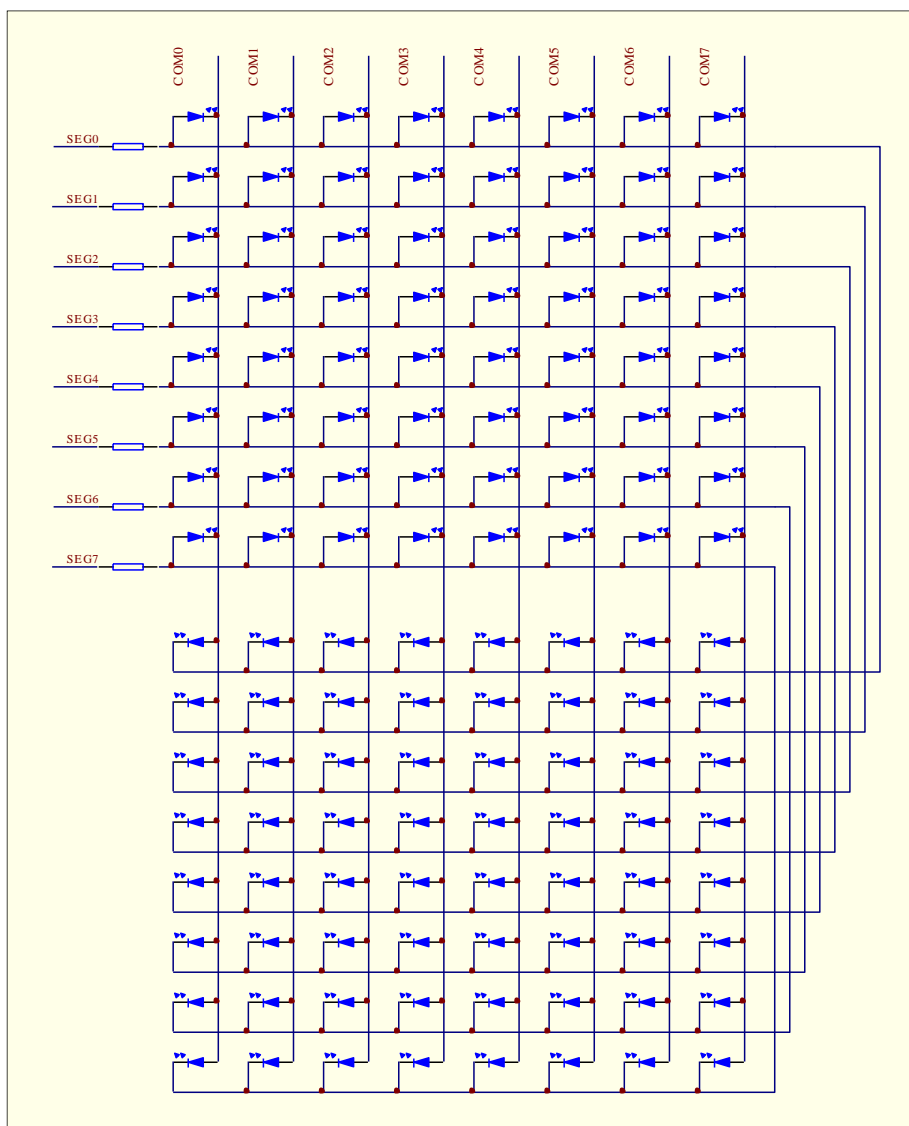
## 26.7 共阴模式参考电路图



## 26.8 共阳模式参考电路图

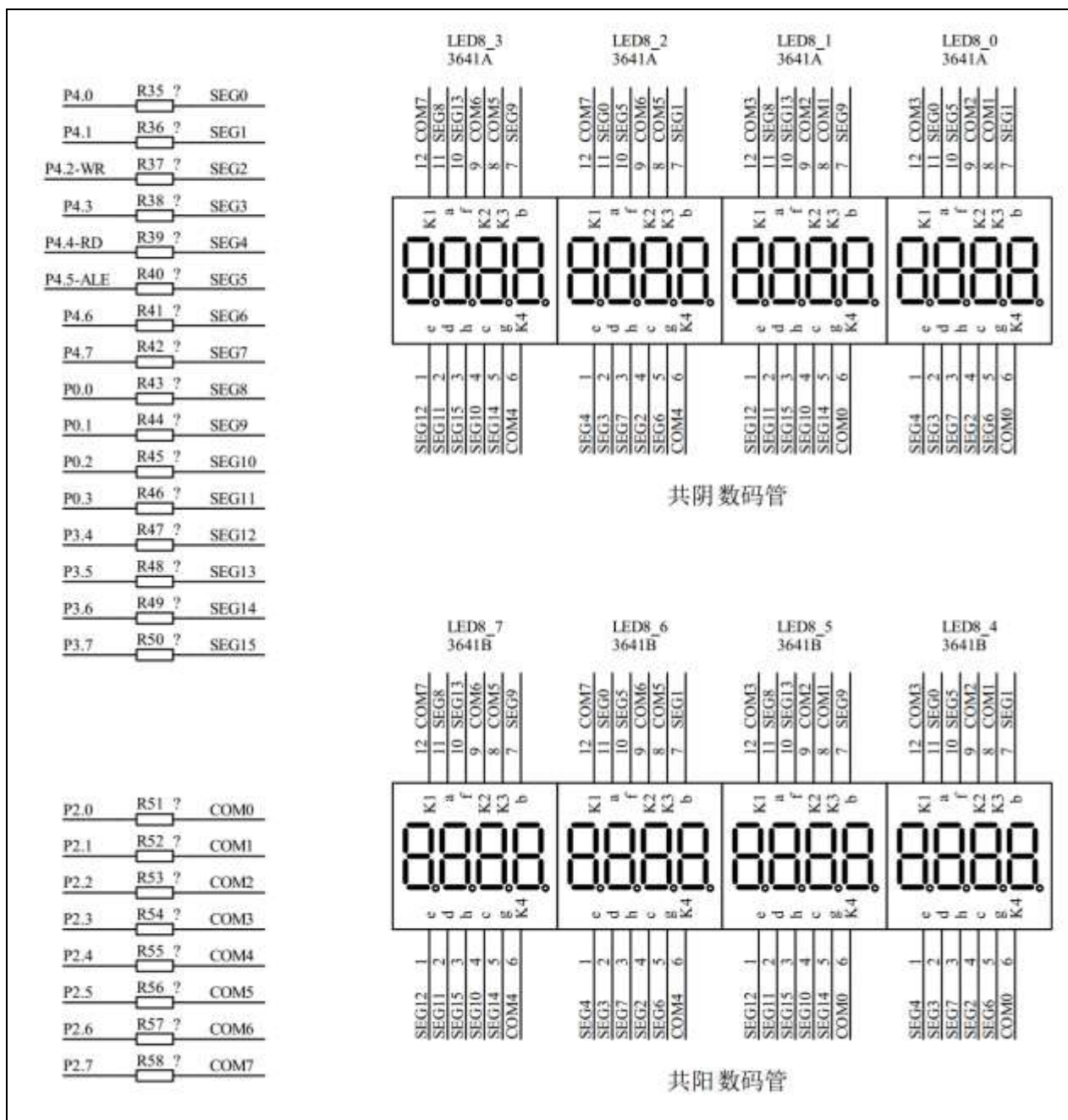


## 26.9 共阴/共阳模式参考电路图 1



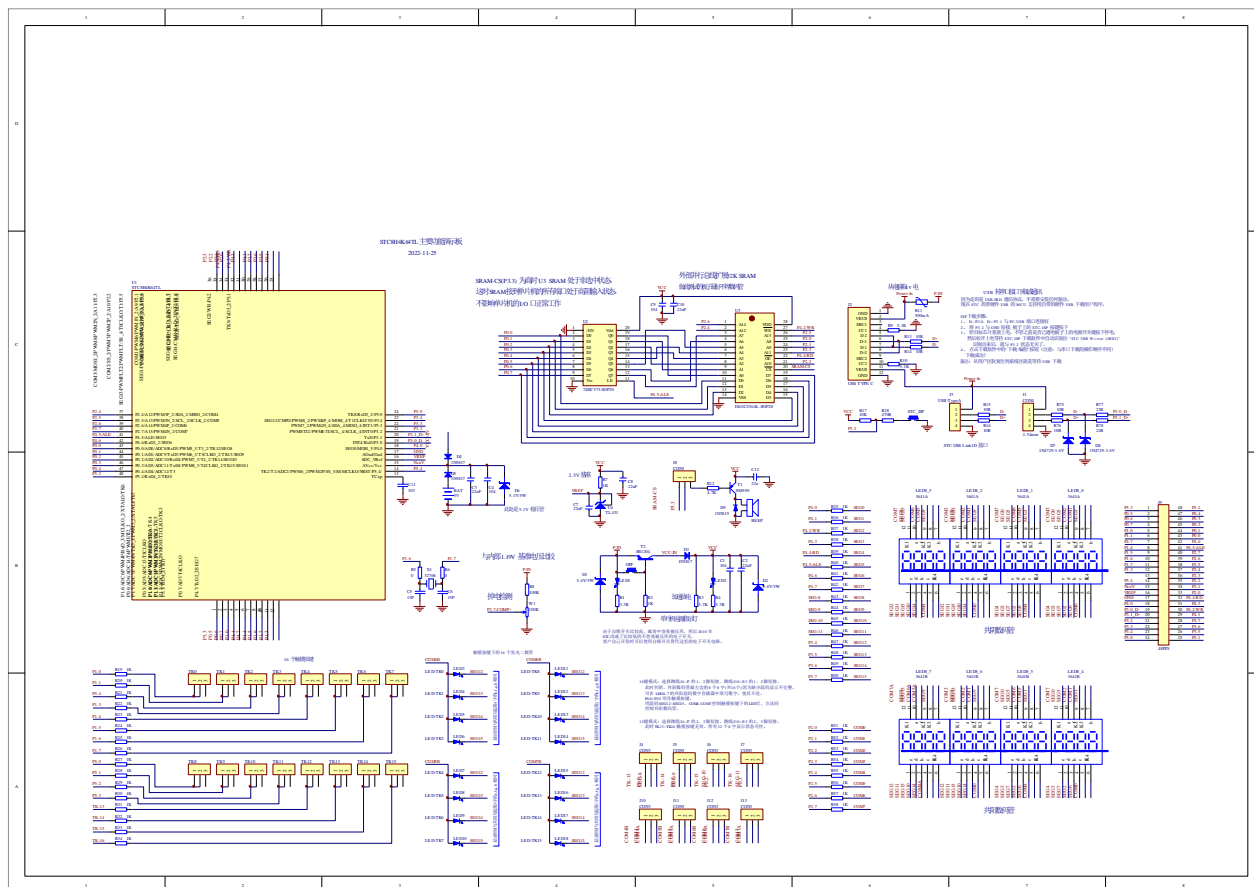


### 26.10 共阴/共阳模式参考电路图 2



## 26.11 范例程序

### 26.11.1 共阴/共阳模式驱动 16 个 7 段数码管



## C 语言代码

//测试工作频率为11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
char code PATTERN[16] =
{
    0x3f,          //0
    0x06,          //1
    0x5b,          //2
    0x4f,          //3
    0x66,          //4
    0x6d,          //5
    0x7d,          //6
    0x27,          //7
    0x7f,          //8
    0x6f,          //9
    0x77,          //A
    0x7c,          //b
    0x39,          //C

```

```
    0x5E,        //d
    0x79,        //E
    0x71,        //F
};

void main()
{
    P_SW2 /= 0x80;                //使能访问 XFR

    P1M0 = 0xff;
    P1M1 = 0x00;
    P3M0 = 0xff;
    P3M1 = 0x00;
    P5M0 = 0x10;
    P5M1 = 0x00;

    COMEN = 0xff;                //使能 COM0~COM7
    SEGENL = 0xff;              //使能 SEG0~SEG7
    LEDCTRL = 0x20;             //LED 驱动共阴/共阳模式
    LEDCKS = 7;                //设置 LED 刷新频率

    COM0_DA = PATTERN[0];        //设置 LED 显示内容
    COM1_DA = PATTERN[1];
    COM2_DA = PATTERN[2];
    COM3_DA = PATTERN[3];
    COM4_DA = PATTERN[4];
    COM5_DA = PATTERN[5];
    COM6_DA = PATTERN[6];
    COM7_DA = PATTERN[7];

    COM0_DC = PATTERN[8];
    COM1_DC = PATTERN[9];
    COM2_DC = PATTERN[10];
    COM3_DC = PATTERN[11];
    COM4_DC = PATTERN[12];
    COM5_DC = PATTERN[13];
    COM6_DC = PATTERN[14];
    COM7_DC = PATTERN[15];

    LEDCTRL /= 0x80;            //启动 LED 驱动

    while (1);
}
```

## 27 RTC 实时时钟

| 产品线                    | RTC |
|------------------------|-----|
| STC8H1K08 系列           |     |
| STC8H1K28 系列           |     |
| STC8H3K64S4 系列         |     |
| STC8H3K64S2 系列         |     |
| STC8H8K64U 系列 A 版本     |     |
| STC8H8K64U 系列 B/C/D 版本 | ●   |
| STC8H4K64TL 系列         | ●   |
| STC8H4K64TLCD 系列       | ●   |
| STC8H1K08T 系列          | ●   |

STC8H 系列部分单片机内部集成一个实时时钟控制电路，主要有如下特性：

- 低功耗：RTC 模块工作电流低至 **2uA@VCC=3.3V**、**3uA@VCC=5.0V**（典型值）
- 长时间跨度：支持 2000 年~2099 年，并自动判断闰年
- 闹钟：支持一组闹钟设置
- 支持多个中断：闹钟中断、日中断、小时中断、分钟中断、秒中断、1/2 秒中断、1/8 秒中断、1/32 秒中断
- 支持掉电唤醒

### 27.1 RTC 相关的寄存器

| 符号       | 描述               | 地址    | 位地址与符号 |       |        |       |       |        |        |         | 复位值       |
|----------|------------------|-------|--------|-------|--------|-------|-------|--------|--------|---------|-----------|
|          |                  |       | B7     | B6    | B5     | B4    | B3    | B2     | B1     | B0      |           |
| RTCCR    | RTC 控制寄存器        | FE60H | -      | -     | -      | -     | -     | -      | -      | RUNRTC  | xxxx,xxx0 |
| RTCCFG   | RTC 配置寄存器        | FE61H | -      | -     | -      | -     | -     | -      | RTCKKS | SETRTC  | xxxx,xx00 |
| RTCEN    | RTC 中断使能寄存器      | FE62H | EALAI  | EDAYI | EHOURI | EMINI | ESECI | ESEC2I | ESEC8I | ESEC32I | 0000,0000 |
| RTCIF    | RTC 中断请求寄存器      | FE63H | ALAIIF | DAYIF | HOURIF | MINIF | SECIF | SEC2IF | SEC8IF | SEC32IF | 0000,0000 |
| ALAHOUR  | RTC 闹钟的小时值       | FE64H | -      | -     | -      |       |       |        |        |         | xxx0,0000 |
| ALAMIN   | RTC 闹钟的分钟值       | FE65H | -      | -     |        |       |       |        |        |         | xx00,0000 |
| ALASEC   | RTC 闹钟的秒值        | FE66H | -      | -     |        |       |       |        |        |         | xx00,0000 |
| ALASSEC  | RTC 闹钟的 1/128 秒值 | FE67H | -      |       |        |       |       |        |        |         | x000,0000 |
| INIYEAR  | RTC 年初初始化        | FE68H | -      |       |        |       |       |        |        |         | x000,0000 |
| INIMONTH | RTC 月初初始化        | FE69H | -      | -     | -      | -     |       |        |        |         | xxxx,0000 |
| INIDAY   | RTC 日初始化         | FE6AH | -      | -     | -      |       |       |        |        |         | xxx0,0000 |
| INIHOUR  | RTC 小时初始化        | FE6BH | -      | -     | -      |       |       |        |        |         | xxx0,0000 |
| INIMIN   | RTC 分钟初始化        | FE6CH | -      | -     |        |       |       |        |        |         | xx00,0000 |
| INISEC   | RTC 秒初始化         | FE6DH | -      | -     |        |       |       |        |        |         | xx00,0000 |
| INISSEC  | RTC1/128 秒初始化    | FE6EH | -      |       |        |       |       |        |        |         | x000,0000 |

|       |                  |       |   |   |   |   |  |           |
|-------|------------------|-------|---|---|---|---|--|-----------|
| YEAR  | RTC 的年计数值        | FE70H | - |   |   |   |  | x000,0000 |
| MONTH | RTC 的月计数值        | FE71H | - | - | - | - |  | xxxx,0000 |
| DAY   | RTC 的日计数值        | FE72H | - | - | - |   |  | xxx0,0000 |
| HOURL | RTC 的小时计数值       | FE73H | - | - | - |   |  | xxx0,0000 |
| MIN   | RTC 的分钟计数值       | FE74H | - | - |   |   |  | xx00,0000 |
| SEC   | RTC 的秒计数值        | FE75H | - | - |   |   |  | xx00,0000 |
| SSEC  | RTC 的 1/128 秒计数值 | FE76H | - |   |   |   |  | x000,0000 |

STC MCU

### 27.1.1 RTC 控制寄存器 (RTCCR)

| 符号    | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0     |
|-------|-------|----|----|----|----|----|----|----|--------|
| RTCCR | FE60H | -  | -  | -  | -  | -  | -  | -  | RUNRTC |

RUNRTC: RTC 模块控制位

- 0: 关闭 RTC, RTC 停止计数
- 1: 使能 RTC, 并开始 RTC 计数

### 27.1.2 RTC 配置寄存器 (RTCCFG)

| 符号     | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1     | B0     |
|--------|-------|----|----|----|----|----|----|--------|--------|
| RTCCFG | FE61H | -  | -  | -  | -  | -  | -  | RTCKKS | SETRTC |

RTCKKS: RTC 时钟源选择

- 0: 选择外部 32.768KHz 时钟源 (需先软件启动外部 32K 晶振)
- 1: 选择内部 32K 时钟源 (需先软件启动内部 32K 振荡器)

SETRTC: 设置 RTC 初始值

- 0: 无意义
- 1: 触发 RTC 寄存器初始化。当 SETRTC 设置为 1 时, 硬件会自动将寄存器 INIYEAR、INIMONTH、INIDAY、INIHOUR、INIMIN、INISEC、INISSEC 中的值复制到寄存器 YEAR、MONTH、DAY、HOUR、MIN、SEC、SSEC 中。初始完成后, 硬件会自动将 SETRTC 位清 0。

### 27.1.3 RTC 中断使能寄存器 (RTCIEN)

| 符号     | 地址    | B7    | B6    | B5     | B4    | B3    | B2     | B1     | B0      |
|--------|-------|-------|-------|--------|-------|-------|--------|--------|---------|
| RTCIEN | FE62H | EALAI | EDAYI | EHOURI | EMINI | ESECI | ESEC2I | ESEC8I | ESEC32I |

EALAI: 闹钟中断使能位

- 0: 关闭闹钟中断
- 1: 使能闹钟中断

EDAYI: 一日 (24 小时) 中断使能位

- 0: 关闭一日中断
- 1: 使能一日中断

EHOURI: 一小时 (60 分钟) 中断使能位

- 0: 关闭小时中断
- 1: 使能小时中断

EMINI: 一分钟 (60 秒) 中断使能位

- 0: 关闭分钟中断
- 1: 使能分钟中断

ESECI: 一秒中断使能位

- 0: 关闭秒中断
- 1: 使能秒中断

ESEC2I: 1/2 秒中断使能位

- 0: 关闭 1/2 秒中断
- 1: 使能 1/2 秒中断

ESEC8I: 1/8 秒中断使能位

0: 关闭 1/8 秒中断

1: 使能 1/8 秒中断

ESEC32I: 1/32 秒中断使能位

0: 关闭 1/32 秒中断

1: 使能 1/32 秒中断

## 27.1.4 RTC 中断请求寄存器 (RTCIF)

| 符号    | 地址    | B7    | B6    | B5      | B4    | B3    | B2     | B1     | B0      |
|-------|-------|-------|-------|---------|-------|-------|--------|--------|---------|
| RTCIF | FE63H | ALAIF | DAYIF | HOURLIF | MINIF | SECIF | SEC2IF | SEC8IF | SEC32IF |

ALAIF: 闹钟中断请求位。需软件清 0, 软件写 1 无效。

DAYIF: 一日 (24 小时) 中断请求位。需软件清 0, 软件写 1 无效。

HOURLIF: 一小时 (60 分钟) 中断请求位。需软件清 0, 软件写 1 无效。

MINIF: 一分钟 (60 秒) 中断请求位。需软件清 0, 软件写 1 无效。

SECIF: 一秒中断请求位。需软件清 0, 软件写 1 无效。

SEC2IF: 1/2 秒中断请求位。需软件清 0, 软件写 1 无效。

SEC8IF: 1/8 秒中断请求位。需软件清 0, 软件写 1 无效。

SEC32IF: 1/32 秒中断请求位。需软件清 0, 软件写 1 无效。

## 27.1.5 RTC 闹钟设置寄存器

| 符号      | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------|-------|----|----|----|----|----|----|----|----|
| ALAHOUR | FE64H | -  | -  | -  |    |    |    |    |    |
| ALAMIN  | FE65H | -  | -  |    |    |    |    |    |    |
| ALASEC  | FE66H | -  | -  |    |    |    |    |    |    |
| ALASSEC | FE67H | -  |    |    |    |    |    |    |    |

ALAHOUR: 设置每天闹钟的小时值。

**注意: 设置的值不是 BCD 码, 而是 HEX 码, 比如需要设置小时值 20 到 ALAHOUR, 则需使用如下代码进行设置**

**MOV DPTR,#ALAHOUR**

**MOV A,#14H**

**MOVX @DPTR,A**

ALAMIN: 设置每天闹钟的分钟值。数字编码与 ALAHOUR 相同。

ALASEC: 设置每天闹钟的秒值。数字编码与 ALAHOUR 相同。

ALASSEC: 设置每天闹钟的 1/128 秒值。数字编码与 ALAHOUR 相同。

## 27.1.6 RTC 实时时钟初始值设置寄存器

| 符号       | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----------|-------|----|----|----|----|----|----|----|----|
| INIYEAR  | FE68H | -  |    |    |    |    |    |    |    |
| INIMONTH | FE69H |    |    |    |    |    |    |    |    |
| INIDAY   | FE6AH |    |    |    |    |    |    |    |    |
| INIHOUR  | FE6BH | -  | -  | -  |    |    |    |    |    |

|         |       |   |   |  |
|---------|-------|---|---|--|
| INIMIN  | FE6CH | - | - |  |
| INISEC  | FE6DH | - | - |  |
| INISSEC | FE6EH | - |   |  |

INIYEAR: 设置当前实时时间的年值。有效值范围 00~99。对应 2000 年~2099 年

**注意: 设置的值不是 BCD 码, 而是 HEX 码, 比如需要设置 20 到 INIYEAR, 则需使用如下代码进行设置**

**MOV DPTR,#INIYEAR**

**MOV A,#14H**

**MOVBX @DPTR,A**

INIMONTH: 设置当前实时时间的月值。有效值范围 1~12。数字编码与 INIYEAR 相同。

INIDAY: 设置当前实时时间的日值。有效值范围 1~31。数字编码与 INIYEAR 相同。

INIHOUR: 设置当前实时时间的小时值。有效值范围 00~23。数字编码与 INIYEAR 相同。

INIMIN: 设置当前实时时间的分钟值。有效值范围 00~59。数字编码与 INIYEAR 相同。

INISEC: 设置当前实时时间的秒值。有效值范围 00~59。数字编码与 INIYEAR 相同。

INISSEC: 设置当前实时时间的 1/128 秒值。有效值范围 00~127。数字编码与 INIYEAR 相同。

当用户设置完成上面的初始值寄存器后, 用户还需要向 SETRTC 位 (RTCCFG.0) 写 1 来触发硬件将初始值装载到 RTC 实时计数器中

**另需注意: 硬件不会对初始化数据的有效性进行检查, 需要用户在设置初始值时, 必须保证数据的有效性, 不能超出其有效范围。**

## 27.1.7 RTC 实时时钟计数寄存器

| 符号    | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------|-------|----|----|----|----|----|----|----|----|
| YEAR  | FE70H | -  |    |    |    |    |    |    |    |
| MONTH | FE71H |    |    |    |    |    |    |    |    |
| DAY   | FE72H |    |    |    |    |    |    |    |    |
| HOUR  | FE73H | -  | -  | -  |    |    |    |    |    |
| MIN   | FE74H | -  | -  |    |    |    |    |    |    |
| SEC   | FE75H | -  | -  |    |    |    |    |    |    |
| SSEC  | FE76H | -  |    |    |    |    |    |    |    |

YEAR: 当前实时时间的年值。**注意: 寄存器的值不是 BCD 码, 而是 HEX 码**

MONTH: 当前实时时间的月值。数字编码与 YEAR 相同。

DAY: 当前实时时间的日值。数字编码与 YEAR 相同。

HOUR: 当前实时时间的小时值。数字编码与 YEAR 相同。

MIN: 当前实时时间的分钟值。数字编码与 YEAR 相同。

SEC: 当前实时时间的秒值。数字编码与 YEAR 相同。

SSEC: 当前实时时间的 1/128 秒值。数字编码与 YEAR 相同。

**注意: YEAR、MONTH、DAY、HOUR、MIN、SEC 和 SSEC 均为只读寄存器, 若需要对这些寄存器执行写操作, 必须通过寄存器 INIYEAR、INIMONTH、INIDAT、INIHOU、INIMIN、INISEC、INISSEC 和 SETRTC 来实现。**

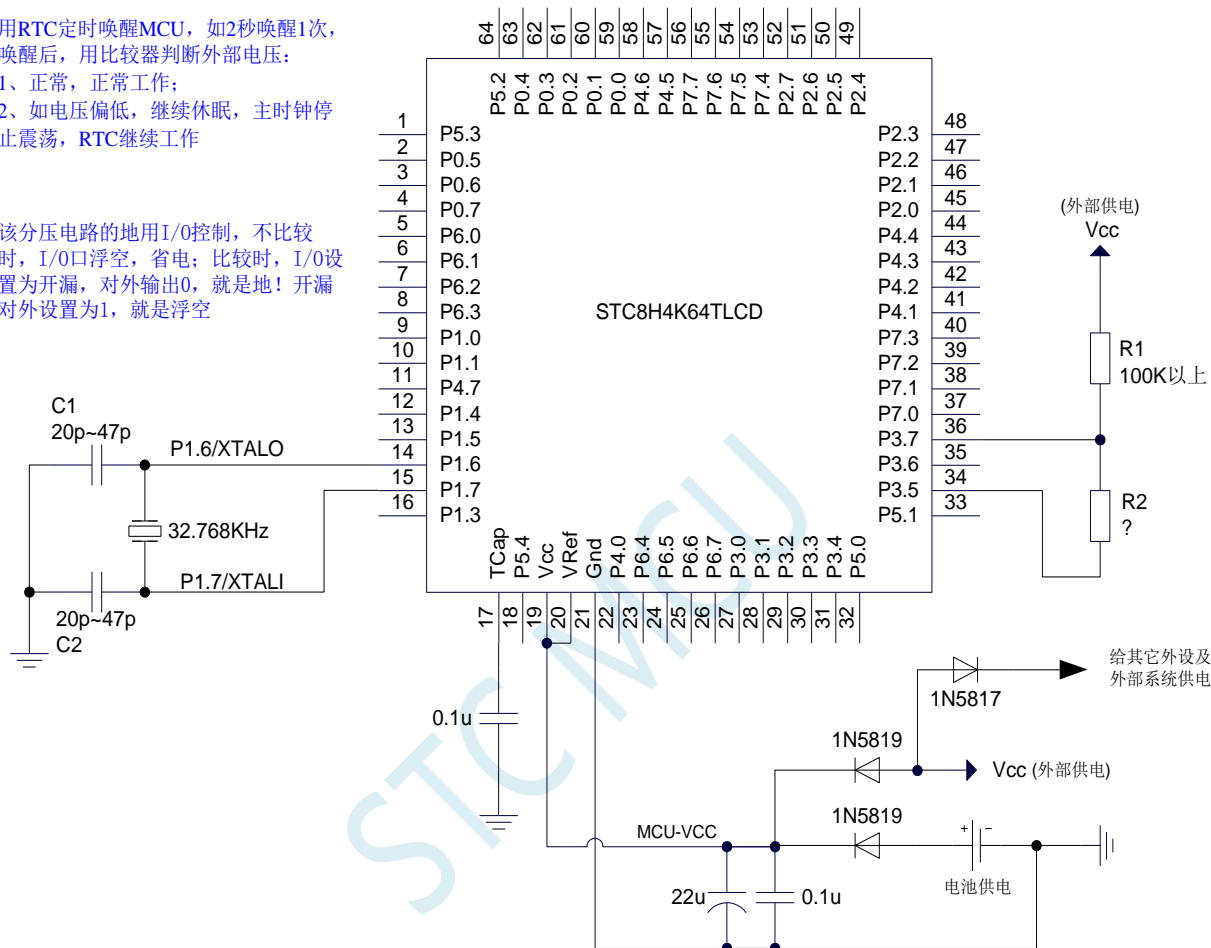


## 27.2 RTC 实战线路图

用RTC定时唤醒MCU，如2秒唤醒1次，  
唤醒后，用比较器判断外部电压：

- 1、正常，正常工作；
- 2、如电压偏低，继续休眠，主时钟停止震荡，RTC继续工作

该分压电路的地用I/O控制，不比较时，I/O口浮空，省电；比较时，I/O设置为开漏，对外输出0，就是地！开漏对外设置为1，就是浮空



## 27.3 范例程序

### 27.3.1 串口打印 RTC 时钟范例

#### C 语言代码

//测试工作频率为22.1184MHz, 需要将C语言代码文件与下面的汇编代码文件加载到同一个项目里使用

```
#include "stc8h.h"
#include "intrins.h"
#include "stdio.h"

#define MAIN_Fosc      22118400L
#define Baudrate       115200L
#define TM              (65536 -(MAIN_Fosc/Baudrate+2)/4)
//加2 操作是为了让 Keil 编译器
//自动实现四舍五入运算

bit      BIS_Flag;

void RTC_config(void);

void UartInit(void)
{
    SCON = (SCON & 0x3f) | 0x40;
    TL2  = TM;
    TH2  = TM>>8;
    AUXR |= 0x15;
}

void UartPutc(unsigned char dat)
{
    SBUF = dat;
    while(TI==0);
    TI = 0;
}

char putchar(char c)
{
    UartPutc(c);
    return c;
}

void RTC_Isr() interrupt 13
{
    if(RTCIF & 0x08)    //判断是否秒中断
    {
        RTCIF &= ~0x08;    //清中断标志
        BIS_Flag = 1;
    }
}

void main(void)
{
    P_SW2 |= 0x80;    //使能 XFR 访问

    P0M1 = 0;    P0M0 = 0;    //设置为准双向口
    P1M1 = 0;    P1M0 = 0;    //设置为准双向口
```

```

P2M1 = 0;   P2M0 = 0;   // 设置为双向口
P3M1 = 0;   P3M0 = 0;   // 设置为双向口
P4M1 = 0;   P4M0 = 0;   // 设置为双向口
P5M1 = 0;   P5M0 = 0;   // 设置为双向口

UartInit();
RTC_config();
EA = 1;
printf("RTC Test Programme!\r\n");           // UART 发送一个字符串

while (1)
{
    if(BIS_Flag)
    {
        BIS_Flag = 0;

        printf("Year=20%bd   ", YEAR);
        printf("Month=%bd   ", MONTH);
        printf("Day=%bd   ", DAY);
        printf("Hour=%bd   ", HOUR);
        printf("Minute=%bd   ", MIN);
        printf("Second=%bd   ", SEC);
        printf("\r\n");
    }
}

void RTC_config(void)
{
    // 选择内部 32K
    IRC32KCR = 0x80;           // 启动内部 32K 振荡器
    while (!(IRC32KCR & 0x01)); // 等待时钟稳定
    RTCCFG |= 0x02;           // 选择内部 32K 作为 RTC 时钟源

    // 选择外部 32K
    // X32KCR = 0xc0;           // 启动外部 32K 晶振
    // while (!(X32KCR & 0x01)); // 等待时钟稳定
    // RTCCFG &= ~0x02;         // 选择外部 32K 作为 RTC 时钟源

    INIYEAR = 21;              // Y:2021
    INIMONTH = 12;             // M:12
    INIDAY = 31;               // D:31
    INIHOUR = 23;              // H:23
    INIMIN = 59;               // M:59
    INISEC = 50;               // S:50
    INISSEC = 0;               // S/128:0
    RTCCFG |= 0x01;           // 触发 RTC 寄存器初始化

    RTCIF = 0;                 // 清中断标志
    RTCIEN = 0x08;             // 使能 RTC 秒中断
    RTCCR = 0x01;              // RTC 使能
}

```

## 汇编代码

; 将以下代码保存为 ASM 格式文件，一起加载到项目里，例如：isr.asm

```

CSEG    AT    0123H
JMP     006BH

```

*END*

---

---

STC MCU

## 27.3.2 利用 ISP 软件的用户接口实现不停电下载保持 RTC 参数

### C 语言代码

//测试工作频率为 11.0592MHz

/\*\*\*\*\*\* 功能说明 \*\*\*\*\*

现有单片机系列的 RTC 模块, 在单片机复位后 RTC 相关的特殊功能寄存器也会复位

本例程主要用于解决 ISP 下载后用户的 RTC 参数丢失的问题

解决思路: ISP 下载前, 先将 RTC 相关参数通过 ISP 下载软件的用户接口上传到 PC 保存, 等待 ISP 下载完成后, 下载软件再将保存的相关参数写入到 FLASH 的指定地址 (范例中指定的地址为 FE00H)。ISP 下载完成后会立即运行用户代码, 用户程序在初始化 RTC 寄存器时, 可从 FLASH 的指定地址中读取之前上传的 RTC 相关参数对 RTC 寄存器进行初始化, 即可实现不停电下载保持 RTC 参数的目的。

下载时, 选择时钟 11.0592MHZ

\*\*\*\*\*/

```
#include "stc8h.h"
#include "intrins.h"
#include "stdio.h"
```

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC 11059200UL
#define BAUD (65536 - FOSC/4/115200)
```

```
typedef bit BOOL;
typedef unsigned char BYTE;
typedef unsigned int WORD;
```

```
struct RTC_INIT
{
```

```
    BYTE bValidTag;           //数据有效标志(0x5a)
    BYTE bIniYear;            //年(RTC 初始化值)
    BYTE bIniMonth;           //月
    BYTE bIniDay;             //日
    BYTE bIniHour;            //时
    BYTE bIniMinute;          //分
    BYTE bIniSecond;          //秒
    BYTE bIniSSSecond;        //次秒
    BYTE bAlaHour;            //时(RTC 闹钟设置值)
    BYTE bAlaMinute;          //分
    BYTE bAlaSecond;          //秒
    BYTE bAlaSSSecond;        //次秒
```

```
};
```

```
struct RTC_INIT code InitBlock _at_ 0xfe00;
```

```
void SysInit();
void UartInit();
void RTCInit();
void SendUart(BYTE dat);
void UnpackCmd(BYTE dat);
void IapProgram(WORD addr, BYTE dat);
```

```

BOOL fUartBusy;
BOOL fFetchRtc;
BOOL fReset2Isp;
BYTE bUartStage;

BYTE bDump[7];

void main()
{
    SysInit();                //系统初始化
    UartInit();
    RTCInit();
    EA = 1;

    fUartBusy = 0;
    fFetchRtc = 0;
    fReset2Isp = 0;
    bUartStage = 0;

    while (1)
    {
        if (fFetchRtc)        //获取 RTC 数据请求
        {
            fFetchRtc = 0;

            RTCCR = 0;        //上传当前的 RTC 值时,必须临时停止 RTC
                               //以免发生进位错误
            bDump[0] = YEAR;  //快速将当前的 RTC 值缓存,
                               //以缩短 RTC 暂停的时间,减小误差

            bDump[1] = MONTH;
            bDump[2] = DAY;
            bDump[3] = HOUR;
            bDump[4] = MIN;
            bDump[5] = SEC;
            bDump[6] = SSEC;
            RTCCR = 1;

            SendUart(0x5a);    //上传 12 字节 RTC 参数
            SendUart(bDump[0]);
            SendUart(bDump[1]);
            SendUart(bDump[2]);
            SendUart(bDump[3]);
            SendUart(bDump[4]);
            SendUart(bDump[5]);
            SendUart(bDump[6]);
            SendUart(ALAHOUR);
            SendUart(ALAMIN);
            SendUart(ALASEC);
            SendUart(ALASSEC);
        }

        if (fReset2Isp)      //重启请求
        {
            fReset2Isp = 0;

            IAP_CONTR = 0x60; //软件触发复位到系统 ISP 区
        }
    }
}

```

```

void uart_isr() interrupt UART1_VECTOR
{
    BYTE dat;

    if (TI)
    {
        TI = 0;

        fUartBusy = 0;
    }

    if (RI)
    {
        RI = 0;

        dat = SBUF;
        switch (bUartStage++) //解析串口命令
        {
            default:
            case 0:
L_Check1st:
                if (dat == '@') bUartStage = 1;
                else bUartStage = 0;
                break;
            case 1:
                if (dat == 'F') bUartStage = 2;
                else if (dat == 'R') bUartStage = 7;
                else goto L_Check1st;
                break;
            case 2:
                if (dat != 'E') goto L_Check1st;
                break;
            case 3:
                if (dat != 'T') goto L_Check1st;
                break;
            case 4:
                if (dat != 'C') goto L_Check1st;
                break;
            case 5:
                if (dat != 'H') goto L_Check1st;
                break;
            case 6:
                if (dat != '#') goto L_Check1st;
                bUartStage = 0;
                fFetchRtc = 1; //当前命令序列为获取RTC 数据命令:"@FETCH#"
                break;
            case 7:
                if (dat != 'E') goto L_Check1st;
                break;
            case 8:
                if (dat != 'B') goto L_Check1st;
                break;
            case 9:
            case 10:
                if (dat != 'O') goto L_Check1st;
                break;
            case 11:
                if (dat != 'T') goto L_Check1st;

```

```

        break;
    case 12:
        if (dat != '#') goto L_Check1st;
        bUartStage = 0;
        fReset2Isp = 1;                                     //当前命令序列为重启命令:"@REBOOT#"
        break;
    }
}

void rtc_isr() interrupt RTC_VECTOR                       //RTC 中断复位程序
{
    RTCIF = 0x00;                                         //清 RTC 中断标志

    P20 = !P20;                                           //P2.0 口每秒闪烁一次,测试用
}

void SysInit()
{
    P_SW2 /= 0x80;

    P0M0 = 0x00; P0M1 = 0x00;
    P1M0 = 0x00; P1M1 = 0x00;
    P2M0 = 0x00; P2M1 = 0x00;
    P3M0 = 0x00; P3M1 = 0x00;
    P4M0 = 0x00; P4M1 = 0x00;
    P5M0 = 0x00; P5M1 = 0x00;
    P6M0 = 0x00; P6M1 = 0x00;
    P7M0 = 0x00; P7M1 = 0x00;
}

void UartInit()                                          //串口初始化函数
{
    SCON = 0x50;
    AUXR = 0x40;
    TMOD = 0x00;
    TL1 = BAUD;
    TH1 = BAUD >> 8;
    TR1 = 1;
    ES = 1;
}

void RTCInit()                                          //RTC 初始化函数
{
    IRC32KCR = 0x80;
    while (!(IRC32KCR & 0x01));
    RTCCFG /= 0x02;                                       //选择内部 32K 为 RTC 时钟源

    // X32KCR = 0xc0;
    // while (!(X32KCR & 0x01));
    // RTCCFG &= ~0x02;                                    //选择外部部 32K 为 RTC 时钟源

    if (InitBlock.bValidTag == 0x5a)
    {
        INIYEAR = InitBlock.bIniYear;
        INIMONTH = InitBlock.bIniMonth;
        INIDAY = InitBlock.bIniDay;
        INIHOURL = InitBlock.bIniHour;
        INIMIN = InitBlock.bIniMinute;
        //如果初始化数据块有效,则使用数据块初始化 RTC
    }
}

```



```

    INISEC = InitBlock.bIniSecond;
    INISSEC = InitBlock.bIniSSecond;
    ALAHOUR = InitBlock.bAlaHour;
    ALAMIN = InitBlock.bAlaMinute;
    ALASEC = InitBlock.bAlaSecond;
    ALASSEC = InitBlock.bAlaSSecond;

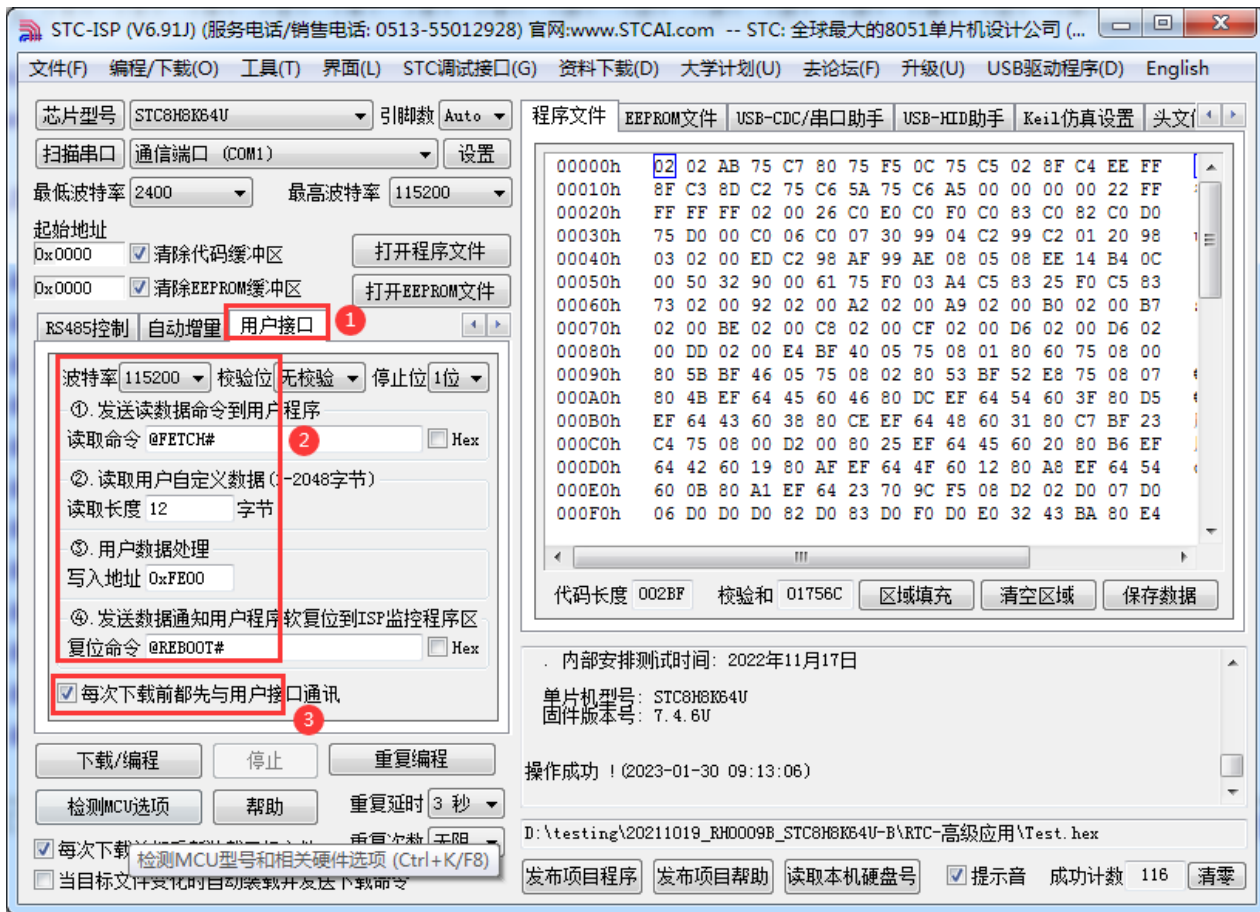
    IapProgram(0x0000, 0x00);           //销毁初始化数据块,以免重复初始化
}
else
{
    INIYEAR = 23;                       //否则初始化 RTC 为默认值
    INIMONTH = 1;
    INIDAY = 29;
    INIHOUR = 12;
    INIMIN = 0;
    INISEC = 0;
    INISSEC = 0;
    ALAHOUR = 0;
    ALAMIN = 0;
    ALASEC = 0;
    ALASSEC = 0;
}
RTCCFG |= 0x01;                        //写入 RTC 初始值
RTCCR = 0x01;                          //RTC 开始运行
while (RTCCFG & 0x01);                 //等待 RTC 初始化完成
RTCIF = 0x00;
RTCIEN = 0x08;                        //使能 RTC 秒中断
}

void SendUart(BYTE dat)                //串口发送函数
{
    while (fUartBusy);
    SBUF = dat;
    fUartBusy = 1;
}

void IapProgram(WORD addr, BYTE dat)   //EEPROM 编程函数
{
    IAP_CONTR = 0x80;
    IAP_TPS = 12;
    IAP_CMD = 2;
    IAP_ADDRL = addr;
    IAP_ADDRH = addr >> 8;
    IAP_DATA = dat;
    IAP_TRIG = 0x5a;
    IAP_TRIG = 0xa5;
    _nop_();
    _nop_();
    _nop_();
    _nop_();
}

```

ISP 下载软件中“用户接口”的设置如下: (注意, 首次下载不能使能用户接口)



## 27.3.3 内部 RTC 时钟低功耗休眠唤醒-比较器检测电压程序

/\*\*\*\*\*\* 本程序功能说明 \*\*\*\*\*/

本例程基于 STC8H8K64U 为主控芯片的实验箱 9 进行编写测试, STC8H 系列带 RTC 模块的芯片可通用参考。

读写芯片内部集成的 RTC 模块

电路连接参考规格书 RTC 章节-RTC 实战线路图

用 RTC 定时唤醒 MCU, 如 1 秒唤醒 1 次, 唤醒后用比较器判断外部电压: 1, 正常, 正常工作; 2, 如电压偏低, 继续休眠, 主时钟停止震荡, RTC 继续工作

比较器正极通过电阻分压后输入到 P3.7 口, 比较器负极使用内部 1.19V 参考电压

该分压电路的地用 I/O(P3.5)控制, I/O 设置为开漏, 不比较时, 对外设置为 1, I/O 口浮空, 省电; 比较时, 对外输出 0, 就是地!

下载时, 选择时钟 24MHZ (用户可自行修改频率).

\*\*\*\*\*/

```
#include "STC8H.h"
```

```
#include "stdio.h"
```

```
#include "intrins.h"
```

```
typedef unsigned char u8;
```

```
typedef unsigned int u16;
```

```
typedef unsigned long u32;
```

/\*\*\*\*\*\* 用户定义宏 \*\*\*\*\*/

```
#define MAIN_Fosc 24000000L
```

//定义主时钟

```
#define PrintUart 2
```

//1:printf 使用 UART1; 2:printf 使用 UART2

```
#define Baudrate 115200L
```

```
#define TM (65536 -(MAIN_Fosc/Baudrate/4))
```

\*\*\*\*\*/

/\*\*\*\*\*\* 本地常量声明 \*\*\*\*\*/

```
u8 code ledNum[]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
```

/\*\*\*\*\*\* 本地变量声明 \*\*\*\*\*/

```
bit B_Is;
```

```
bit B_Alarm;
```

//闹钟标志

```
u8 ledIndex;
```

/\*\*\*\*\*\* 本地函数声明 \*\*\*\*\*/

```
void RTC_config(void);
```

```
void CMP_config(void);
```

```
void Ext_Vcc_Det(void);
```

/\*\*\*\*\*\* 串口打印函数 \*\*\*\*\*/

```
void UartInit(void)
```

```
{
```

```
  #if(PrintUart == 1)
```

```
    SCON = (SCON & 0x3f) | 0x40;
```

```
    AUXR |= 0x40;
```

//定时器时钟 1T 模式

```
    AUXR &= 0xFE;
```

//串口 1 选择定时器 1 为波特率发生器

```

    TL1 = TM;
    TH1 = TM>>8;
    TR1 = 1;                                     //定时器1 开始计时

//    SCON = (SCON & 0x3f) | 0x40;
//    T2L = TM;
//    T2H = TM>>8;
//    AUXR |= 0x15;                             //串口1 选择定时器2 为波特率发生器
#else
    P_SW2 /= 1;
    S2CON &= ~(1<<7);
    T2L = TM;
    T2H = TM>>8;
    AUXR |= 0x14;                             //定时器2 时钟 1T 模式,开始计时
#endif
}

void UartPutc(unsigned char dat)
{
    #if(PrintUart == 1)
        SBUF = dat;
        while(TI == 0);
        TI = 0;
    #else
        S2BUF = dat;
        while((S2CON & 2) == 0);
        S2CON &= ~2;                          //Clear Tx flag
    #endif
}

char putchar(char c)
{
    UartPutc(c);
    return c;
}

/*****
void main(void)
{
    P_SW2 /= 0x80;                             //扩展寄存器(XFR)访问使能

    P0M1 = 0x00;    P0M0 = 0x00;              //设置为准双向口
    P1M1 = 0x00;    P1M0 = 0x00;              //设置为准双向口
    P2M1 = 0x00;    P2M0 = 0x00;              //设置为准双向口
    P3M1 = 0xa0;    P3M0 = 0x20;              //设置为准双向口
                                           // P3.5 设置开漏输出, P3.7 设置高阻输入
    P4M1 = 0x00;    P4M0 = 0x00;              //设置为准双向口
    P5M1 = 0x00;    P5M0 = 0x00;              //设置为准双向口
    P6M1 = 0x00;    P6M0 = 0x00;              //设置为准双向口
    P7M1 = 0x00;    P7M0 = 0x00;              //设置为准双向口

    UartInit();
    CMP_config();
    RTC_config();
    EA = 1;                                     //打开总中断

    while(1)
    {
        if(B_Is)

```

```

    {
        B_Is = 0;
        printf("Year=20%bd,Month=%bd,Day=%bd,Hour=%bd,Minute=%bd,Second=%bd\r\n",
            YEAR,MONTH,DAY,HOUR,MIN,SEC);

        Ext_Vcc_Det(); //每秒钟检测一次外部电源,
                       //如果外部电源连接则工作,
                       //外部电源断开则进入休眠模式

    }

    if(B_Alarm)
    {
        B_Alarm = 0;
        printf("RTC Alarm!\r\n");
    }
}

//=====
// 函数: void Ext_Vcc_Det(void)
// 描述: 外部电源检测函数。
// 参数: 无
// 返回: 无
// 版本: V1.0, 2022-10-10
//=====
void Ext_Vcc_Det(void)
{
    P35 = 0; //比较时, 对外输出0, 做比较电路的地线
    CMPCR1 |= 0x80; //使能比较器模块
    _nop_();
    _nop_();
    _nop_();
    if(CMPCR1 & 0x01) //判断是否 CMP+ 电平高于 CMP-, 外部电源连接
    {
        P40 = 0; //LED Power On
        P6 = ~ledNum[ledIndex]; //输出低驱动
        ledIndex++;
        if(ledIndex > 7)
        {
            ledIndex = 0;
        }
    }
    else
    {
        CMPCR1 &= ~0x80; //关闭比较器模块
        P35 = 1; //不比较时, 对外设置为1, I/O 口浮空, 省电
        P40 = 1; //LED Power Off
        _nop_();
        _nop_();
        PCON = 0x02; //STC8H8K64U B 版本芯片使用内部 32K 时钟,
                     //休眠无法唤醒

        _nop_();
        _nop_();
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

```

```

//=====
// 函数: void CMP_config(void)
// 描述: 比较器初始化函数。
// 参数: 无
// 返回: 无
// 版本: V1.0, 2022-10-10
//=====
void CMP_config(void)
{
    CMPEXCFG = 0x00;
    // CMPEXCFG |= 0x40; //比较器DC 迟滞输入选择, 0:0mV;
    //                                //0x40:10mV; 0x80:20mV; 0xc0:30mV

    // CMPEXCFG &= ~0x04; //P3.6 为CMP-输入脚
    // CMPEXCFG |= 0x04; //内部1.19V 参考电压为CMP-输入脚

    CMPEXCFG &= ~0x03; //P3.7 为CMP+输入脚
    // CMPEXCFG |= 0x01; //P5.0 为CMP+输入脚
    // CMPEXCFG |= 0x02; //P5.1 为CMP+输入脚
    // CMPEXCFG |= 0x03; //ADC 输入脚为CMP+输入脚

    CMPCR2 = 0x00;
    CMPCR2 &= ~0x80; //比较器正向输出
    // CMPCR2 |= 0x80; //比较器反向输出
    CMPCR2 &= ~0x40; //使能0.1us 滤波
    // CMPCR2 |= 0x40; //禁止0.1us 滤波
    CMPCR2 &= ~0x3f; //比较器结果直接输出
    // CMPCR2 |= 0x10; //比较器结果经过16 个去抖时钟后输出

    CMPCR1 = 0x00;
    // CMPCR1 |= 0x30; //使能比较器边沿中断
    // CMPCR1 &= ~0x20; //禁止比较器上升沿中断
    // CMPCR1 |= 0x20; //使能比较器上升沿中断
    // CMPCR1 &= ~0x10; //禁止比较器下降沿中断
    // CMPCR1 |= 0x10; //使能比较器下降沿中断

    CMPCR1 &= ~0x02; //禁止比较器输出
    // CMPCR1 |= 0x02; //使能比较器输出

    P_SW2 &= ~0x08; //选择P3.4 作为比较器输出脚
    // P_SW2 |= 0x08; //选择P4.1 作为比较器输出脚
    CMPCR1 |= 0x80; //使能比较器模块
}

//=====
// 函数: void RTC_config(void)
// 描述: RTC 初始化函数。
// 参数: 无
// 返回: 无
// 版本: V1.0, 2022-10-10
//=====
void RTC_config(void)
{
    INIYEAR = 21; //Y:2021
    INIMONTH = 12; //M:12
    INIDAY = 31; //D:31
    INIhour = 23; //H:23

```

```

    INIMIN = 59;                //M:59
    INISEC = 50;                //S:50
    INISSEC = 0;                //S/128:0

    ALAHOURL = 0;               //闹钟小时
    ALAMIN = 0;                 //闹钟分钟
    ALASEC = 0;                 //闹钟秒
    ALASSECL = 0;               //闹钟 1/128 秒

//STC8H8K64U B 版本芯片使用内部 32K 时钟, 休眠无法唤醒
//    IRC32KCR = 0x80;          //启动内部 32K 晶振
//    while(!(IRC32KCR & 1));    //等待时钟稳定
//    RTCCFG = 0x03;            //选择内部 32K 时钟源, 触发 RTC 寄存器初始化

    X32KCR = 0x80 + 0x40;       //启动外部 32K 晶振, 低增益+0x00, 高增益+0x40.
    while(!(X32KCR & 1));       //等待时钟稳定
    RTCCFG = 0x01;              //选择外部 32K 时钟源, 触发 RTC 寄存器初始化

    RTCIF = 0x00;               //清中断标志
    RTCIEN = 0x88;              //中断使能, 0x80: 闹钟中断, 0x40: 日中断, 0x20: 小时中断,
                                //0x10: 分钟中断, 0x08: 秒中断, 0x04: 1/2 秒中断,
                                //0x02: 1/8 秒中断, 0x01: 1/32 秒中断
    RTCCR = 0x01;               //RTC 使能

    while(RTCCFG & 0x01);       //等待初始化完成, 需要在 "RTC 使能" 之后判断
                                //设置 RTC 时间需要 32768Hz 的 1 个周期时间,
                                //大约 30.5us./ 由于同步, 所以实际等待时间是 0~30.5us.
                                //如果不等待设置完成就睡眠, 则 RTC 会由于设置没完成
                                //停止计数, 唤醒后才继续完成设置并继续计数
}

/***** RTC 中断函数 *****/
void RTC_Isr() interrupt 13
{
    if(RTCIF & 0x80)             //闹钟中断
    {
        //    P01 = !P01;
        RTCIF &= ~0x80;
        B_Alarm = 1;
    }

    if(RTCIF & 0x08)             //秒中断
    {
        //    P00 = !P00;
        RTCIF &= ~0x08;
        B_Is = 1;
    }
}

/*****
//如果开启了比较器中断就需要编写对应的中断函数
void CMP_Isr() interrupt 21
{
    CMPCRI &= ~0x40;             //清中断标志
    //    P10 = CMPCRI & 0x01;    //中断方式读取比较器比较结果
}
*****/

```

## 28 LCD 液晶驱动，即传统的段式/笔画式屏驱动

| 产品线              | LCD 驱动 |
|------------------|--------|
| STC8H1K08 系列     |        |
| STC8H1K28 系列     |        |
| STC8H3K64S4 系列   |        |
| STC8H3K64S2 系列   |        |
| STC8H8K64U 系列    |        |
| STC8H4K64TL 系列   |        |
| STC8H4K64TLCD 系列 | ●      |
| STC8H1K08T 系列    |        |

STC8H 系列的部分单片机内部集成了一个 LCD 驱动器，可用于驱动液晶屏。最多可驱动 4COM\*40SEG 点阵的液晶屏

COM 线对应的端口：

| COM7 | COM6 | COM5 | COM4 | COM3 | COM2 | COM1 | COM0 |
|------|------|------|------|------|------|------|------|
| -    | -    | -    | -    | P3.6 | P3.5 | P5.1 | P5.0 |

SEG 线对应的端口：

| SEG39 | SEG38 | SEG37 | SEG36 | SEG35 | SEG34     | SEG33     | SEG32     |
|-------|-------|-------|-------|-------|-----------|-----------|-----------|
| P6.3  | P6.2  | P6.1  | P6.0  | P7.0  | P7.1      | P7.2      | P7.3      |
| SEG31 | SEG30 | SEG29 | SEG28 | SEG27 | SEG26     | SEG25     | SEG24     |
| P3.7  | P4.1  | P4.2  | P4.3  | P4.4  | P2.0      | P2.1      | P2.2      |
| SEG23 | SEG22 | SEG21 | SEG20 | SEG19 | SEG18     | SEG17     | SEG16     |
| P2.3  | P2.4  | P2.5  | P2.6  | P2.7  | P4.5      | P4.6      | P0.0      |
| SEG15 | SEG14 | SEG13 | SEG12 | SEG11 | SEG10     | SEG9      | SEG8      |
| P0.1  | P0.2  | P0.3  | P0.4  | P5.2  | P5.3      | P0.5      | P0.6      |
| SEG7  | SEG6  | SEG5  | SEG4  | SEG3  | SEG2      | SEG1      | SEG0      |
| P0.7  | P1.0  | P1.1  | P4.7  | P7.4  | P1.3/P7.5 | P1.4/P7.6 | P1.5/P7.7 |

### 28.1 LCD 功能脚切换

| 符号      | 地址    | B7 | B6 | B5 | B4 | B3     | B2     | B1     | B0     |
|---------|-------|----|----|----|----|--------|--------|--------|--------|
| LCDCFG2 | FB81H | -  | -  | -  | -  | SEG3PS | SEG2PS | SEG1PS | SEG0PS |

SEG3PS: SEG3 控制脚选择位

| SEG3PS | SEG3 |
|--------|------|
| 0      | -    |
| 1      | P7.4 |

SEG2PS: SEG2 控制脚选择位

| SEG2PS | SEG2 |
|--------|------|
| 0      | P1.3 |



|   |      |
|---|------|
| 1 | P7.5 |
|---|------|

SEG1PS: SEG1 控制脚选择位

| SEG1PS | SEG1 |
|--------|------|
| 0      | P1.4 |
| 1      | P7.6 |

SEG0PS: SEG0 控制脚选择位

| SEG0PS | SEG0 |
|--------|------|
| 0      | P1.5 |
| 1      | P7.7 |

## 28.2 LCD 相关的寄存器

| 符号        | 描述               | 地址    | 位地址与符号         |       |       |       |               |              |        |        |           | 复位值       |
|-----------|------------------|-------|----------------|-------|-------|-------|---------------|--------------|--------|--------|-----------|-----------|
|           |                  |       | B7             | B6    | B5    | B4    | B3            | B2           | B1     | B0     |           |           |
| LCDCFG    | LCD 配置寄存器        | FB80H | CLKSEL         | -     | -     | -     | VRLPSEL       | VLCDSEL[2:0] |        |        | 0xxx,0000 |           |
| LCDCFG2   | LCD 配置寄存器 2      | FB81H | -              | -     | -     | -     | SEG3PS        | SEG2PS       | SEG1PS | SEG0PS | xxxx,0000 |           |
| DBLEN     | 死区时间长度配置         | FB82H | -              | -     | -     | -     | -             | DBLEN[2:0]   |        |        | xxxx,x000 |           |
| COMLENL   | COM 时间长度配置低位     | FB83H | COMLEN[7:0]    |       |       |       |               |              |        |        |           | 0000,0000 |
| COMLENM   | COM 时间长度配置中位     | FB84H | COMLEN[15:8]   |       |       |       |               |              |        |        |           | 0000,0000 |
| COMLENH   | COM 时间长度配置高位     | FB85H | -              | -     | -     | -     | COMLEN[19:16] |              |        |        | xxxx,0000 |           |
| BLINKRATE | 闪烁率配置寄存器         | FB86H | BLANKRATE[7:0] |       |       |       |               |              |        |        |           | 1000,0000 |
| LCDCR     | LCD 控制寄存器        | FB87H | -              | -     | -     | -     | -             | ACTMODE[1:0] |        | ENLCD  | xxxx,x000 |           |
| COMON     | COM 线使能寄存器       | FB88H | -              | -     | -     | -     | COM3          | COM2         | COM1   | COM0   | xxxx,0000 |           |
| SEGON1    | SEG 线使能寄存器 1     | FB8AH | SEG7           | SEG6  | SEG5  | SEG4  | SEG3          | SEG2         | SEG1   | SEG0   | 0000,0000 |           |
| SEGON2    | SEG 线使能寄存器 2     | FB8BH | SEG15          | SEG14 | SEG13 | SEG12 | SEG11         | SEG10        | SEG9   | SEG8   | 0000,0000 |           |
| SEGON3    | SEG 线使能寄存器 3     | FB8CH | SEG23          | SEG22 | SEG21 | SEG20 | SEG19         | SEG18        | SEG17  | SEG16  | 0000,0000 |           |
| SEGON4    | SEG 线使能寄存器 4     | FB8DH | SEG31          | SEG30 | SEG29 | SEG28 | SEG27         | SEG26        | SEG25  | SEG24  | 0000,0000 |           |
| SEGON5    | SEG 线使能寄存器 5     | FB8EH | SEG39          | SEG38 | SEG37 | SEG36 | SEG35         | SEG34        | SEG33  | SEG32  | 0000,0000 |           |
| C0SEGV0   | C0SEG7_0 数据寄存器   | FB90H | C0S7           | C0S6  | C0S5  | C0S4  | C0S3          | C0S2         | C0S1   | C0S0   | 0000,0000 |           |
| C0SEGV1   | C0SEG15_8 数据寄存器  | FB91H | C0S15          | C0S14 | C0S13 | C0S12 | C0S11         | C0S10        | C0S9   | C0S8   | 0000,0000 |           |
| C0SEGV2   | C0SEG23_16 数据寄存器 | FB92H | C0S23          | C0S22 | C0S21 | C0S20 | C0S19         | C0S18        | C0S17  | C0S16  | 0000,0000 |           |
| C0SEGV3   | C0SEG31_24 数据寄存器 | FB93H | C0S31          | C0S30 | C0S29 | C0S28 | C0S27         | C0S26        | C0S25  | C0S24  | 0000,0000 |           |
| C0SEGV4   | C0SEG39_32 数据寄存器 | FB94H | C0S39          | C0S38 | C0S37 | C0S36 | C0S35         | C0S34        | C0S33  | C0S32  | 0000,0000 |           |
| C1SEGV0   | C1SEG7_0 数据寄存器   | FB98H | C1S7           | C1S6  | C1S5  | C1S4  | C1S3          | C1S2         | C1S1   | C1S0   | 0000,0000 |           |
| C1SEGV1   | C1SEG15_8 数据寄存器  | FB99H | C1S15          | C1S14 | C1S13 | C1S12 | C1S11         | C1S10        | C1S9   | C1S8   | 0000,0000 |           |
| C1SEGV2   | C1SEG23_16 数据寄存器 | FB9AH | C1S23          | C1S22 | C1S21 | C1S20 | C1S19         | C1S18        | C1S17  | C1S16  | 0000,0000 |           |
| C1SEGV3   | C1SEG31_24 数据寄存器 | FB9BH | C1S31          | C1S30 | C1S29 | C1S28 | C1S27         | C1S26        | C1S25  | C1S24  | 0000,0000 |           |
| C1SEGV4   | C1SEG39_32 数据寄存器 | FB9CH | C1S39          | C1S38 | C1S37 | C1S36 | C1S35         | C1S34        | C1S33  | C1S32  | 0000,0000 |           |
| C2SEGV0   | C2SEG7_0 数据寄存器   | FBA0H | C2S7           | C2S6  | C2S5  | C2S4  | C2S3          | C2S2         | C2S1   | C2S0   | 0000,0000 |           |
| C2SEGV1   | C2SEG15_8 数据寄存器  | FBA1H | C2S15          | C2S14 | C2S13 | C2S12 | C2S11         | C2S10        | C2S9   | C2S8   | 0000,0000 |           |
| C2SEGV2   | C2SEG23_16 数据寄存器 | FBA2H | C2S23          | C2S22 | C2S21 | C2S20 | C2S19         | C2S18        | C2S17  | C2S16  | 0000,0000 |           |
| C2SEGV3   | C2SEG31_24 数据寄存器 | FBA3H | C2S31          | C2S30 | C2S29 | C2S28 | C2S27         | C2S26        | C2S25  | C2S24  | 0000,0000 |           |
| C2SEGV4   | C2SEG39_32 数据寄存器 | FBA4H | C2S39          | C2S38 | C2S37 | C2S36 | C2S35         | C2S34        | C2S33  | C2S32  | 0000,0000 |           |
| C3SEGV0   | C3SEG7_0 数据寄存器   | FBA8H | C3S7           | C3S6  | C3S5  | C3S4  | C3S3          | C3S2         | C3S1   | C3S0   | 0000,0000 |           |

|         |                  |       |       |       |       |       |       |       |       |       |           |
|---------|------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-----------|
| C3SEGV1 | C3SEG15_8 数据寄存器  | FBA9H | C3S15 | C3S14 | C3S13 | C3S12 | C3S11 | C3S10 | C3S9  | C3S8  | 0000,0000 |
| C3SEGV2 | C3SEG23_16 数据寄存器 | FBAAH | C3S23 | C3S22 | C3S21 | C3S20 | C3S19 | C3S18 | C3S17 | C3S16 | 0000,0000 |
| C3SEGV3 | C3SEG31_24 数据寄存器 | FBABH | C3S31 | C3S30 | C3S29 | C3S28 | C3S27 | C3S26 | C3S25 | C3S24 | 0000,0000 |
| C3SEGV4 | C3SEG39_32 数据寄存器 | FBACH | C3S39 | C3S38 | C3S37 | C3S36 | C3S35 | C3S34 | C3S33 | C3S32 | 0000,0000 |

STC MCU

## 28.2.1 LCD 配置寄存器 (LCDCFG)

| 符号     | 地址    | B7     | B6 | B5 | B4 | B3      | B2           | B1 | B0 |
|--------|-------|--------|----|----|----|---------|--------------|----|----|
| LCDCFG | FB80H | CLKSEL | -  | -  | -  | VRLPSEL | VLCDSEL[2:0] |    |    |

CLKSEL: LCD 时钟源选择

0: 选择 CPU 时钟为 LCD 时钟源

1: 选择外部 32K 晶振为 LCD 时钟源

VRLPSEL: 暂时不用, 建议设置为 0

VLCDSEL: VLCD 电压选择位

| VLCDSEL[2:0] | VLCD     |
|--------------|----------|
| 000          | 0.65*VCC |
| 001          | 0.70*VCC |
| 010          | 0.75*VCC |
| 011          | 0.80*VCC |
| 100          | 0.85*VCC |
| 101          | 0.90*VCC |
| 110          | 0.95*VCC |
| 111          | 1.00*VCC |

例如:

若 VCC 工作电压为 3.3V, LCD 屏为 3.0V 的规格, 则 VLCDSEL 可设置为 101B, 即设置 VLCD 电压为  $3.3V \times 0.90 = 2.97V$ ;

若 VCC 工作电压为 5.0V, LCD 屏为 3.0V 的规格, 则 VLCDSEL 可设置为 000B, 即设置 VLCD 电压为  $5.0V \times 0.65 = 3.25V$ ;

## 28.2.2 LCD 配置寄存器 2 (LCDCFG2)

| 符号      | 地址    | B7 | B6 | B5 | B4 | B3     | B2     | B1     | B0     |
|---------|-------|----|----|----|----|--------|--------|--------|--------|
| LCDCFG2 | FB81H | -  | -  | -  | -  | SEG3PS | SEG2PS | SEG1PS | SEG0PS |

SEG3PS: SEG3 控制脚选择位

| SEG3PS | SEG3 |
|--------|------|
| 0      | -    |
| 1      | P7.4 |

SEG2PS: SEG2 控制脚选择位

| SEG2PS | SEG2 |
|--------|------|
| 0      | P1.3 |
| 1      | P7.5 |

SEG1PS: SEG1 控制脚选择位

| SEG1PS | SEG1 |
|--------|------|
| 0      | P1.4 |
| 1      | P7.6 |

SEG0PS: SEG0 控制脚选择位

| SEG0PS | SEG0 |
|--------|------|
| 0      | P1.5 |

|   |      |
|---|------|
| 1 | P7.7 |
|---|------|

### 28.2.3 死区时间长度配置寄存器 (DBLEN)

| 符号    | 地址    | B7 | B6 | B5 | B4 | B3 | B2         | B1 | B0 |
|-------|-------|----|----|----|----|----|------------|----|----|
| DBLEN | FB82H | -  | -  | -  | -  | -  | DBLEN[2:0] |    |    |

DBLEN[2:0]: 设置 LCD 显示时的死区时间长度

### 28.2.4 COM 时间长度配置寄存器 (COMLEN<sub>x</sub>)

| 符号      | 地址    | B7           | B6 | B5 | B4 | B3            | B2 | B1 | B0 |
|---------|-------|--------------|----|----|----|---------------|----|----|----|
| COMLENL | FB83H | COMLEN[7:0]  |    |    |    |               |    |    |    |
| COMLENM | FB84H | COMLEN[15:8] |    |    |    |               |    |    |    |
| COMLENH | FB85H | -            | -  | -  | -  | COMLEN[19:16] |    |    |    |

COMLEN[19:0]: 设置 LCD 显示时的 COM 时间长度

### 28.2.5 闪烁率配置寄存器 (BLANKRATE)

| 符号        | 地址    | B7             | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----------|-------|----------------|----|----|----|----|----|----|----|
| BLANKRATE | FB86H | BLANKRATE[7:0] |    |    |    |    |    |    |    |

BLANKRATE[7:0]: 设置闪烁模式下, LCD 的闪烁率

### 28.2.6 LCD 控制寄存器 (LCDCR)

| 符号   | 地址    | B7 | B6 | B5 | B4 | B3 | B2           | B1 | B0    |
|------|-------|----|----|----|----|----|--------------|----|-------|
| LDCR | FB87H | -  | -  | -  | -  | -  | ACTMODE[1:0] |    | ENLCD |

ACTMODE[1:0]: 设置 LCD 显示模式

| ACTMODE[1:0] | 模式     |
|--------------|--------|
| 00           | 普通显示模式 |
| 01           | 长亮模式   |
| 10           | 长暗模式   |
| 11           | 闪烁模式   |

ENLCD: LCD 模块使能控制位

0: 禁止 LCD 模块

1: 使能 LCD 模块

### 28.2.7 COM 线使能寄存器 (COMON)

| 符号    | 地址    | B7 | B6 | B5 | B4 | B3   | B2   | B1   | B0   |
|-------|-------|----|----|----|----|------|------|------|------|
| COMON | FB88H | -  | -  | -  | -  | COM3 | COM2 | COM1 | COM0 |

COM<sub>n</sub>: COM<sub>n</sub> 使能控制位 (n=0~3)

0: 禁止 COM<sub>n</sub> (相应端口为普通 IO)

1: 允许 COM<sub>n</sub>

## 28.2.8 SEG 线使能寄存器 (SEGON<sub>x</sub>)

| 符号     | 地址    | B7    | B6    | B5    | B4    | B3    | B2    | B1    | B0    |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| SEGON1 | FB8AH | SEG7  | SEG6  | SEG5  | SEG4  | SEG3  | SEG2  | SEG1  | SEG0  |
| SEGON2 | FB8BH | SEG15 | SEG14 | SEG13 | SEG12 | SEG11 | SEG10 | SEG9  | SEG8  |
| SEGON3 | FB8CH | SEG23 | SEG22 | SEG21 | SEG20 | SEG19 | SEG18 | SEG17 | SEG16 |
| SEGON4 | FB8DH | SEG31 | SEG30 | SEG29 | SEG28 | SEG27 | SEG26 | SEG25 | SEG24 |
| SEGON5 | FB8EH | SEG39 | SEG38 | SEG37 | SEG36 | SEG35 | SEG34 | SEG33 | SEG32 |

SEGN: SEGN 使能控制位 (n=0~39)

0: 禁止 SEGN (相应端口为普通 IO)

1: 允许 SEGN

## 28.2.9 LCD 数据寄存器 (CxSEGV<sub>x</sub>)

| 符号      | 地址    | B7    | B6    | B5    | B4    | B3    | B2    | B1    | B0    |
|---------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| C0SEGV0 | FB90H | C0S7  | C0S6  | C0S5  | C0S4  | C0S3  | C0S2  | C0S1  | C0S0  |
| C0SEGV1 | FB91H | C0S15 | C0S14 | C0S13 | C0S12 | C0S11 | C0S10 | C0S9  | C0S8  |
| C0SEGV2 | FB92H | C0S23 | C0S22 | C0S21 | C0S20 | C0S19 | C0S18 | C0S17 | C0S16 |
| C0SEGV3 | FB93H | C0S31 | C0S30 | C0S29 | C0S28 | C0S27 | C0S26 | C0S25 | C0S24 |
| C0SEGV4 | FB94H | C0S39 | C0S38 | C0S37 | C0S36 | C0S35 | C0S34 | C0S33 | C0S32 |
| C1SEGV0 | FB98H | C1S7  | C1S6  | C1S5  | C1S4  | C1S3  | C1S2  | C1S1  | C1S0  |
| C1SEGV1 | FB99H | C1S15 | C1S14 | C1S13 | C1S12 | C1S11 | C1S10 | C1S9  | C1S8  |
| C1SEGV2 | FB9AH | C1S23 | C1S22 | C1S21 | C1S20 | C1S19 | C1S18 | C1S17 | C1S16 |
| C1SEGV3 | FB9BH | C1S31 | C1S30 | C1S29 | C1S28 | C1S27 | C1S26 | C1S25 | C1S24 |
| C1SEGV4 | FB9CH | C1S39 | C1S38 | C1S37 | C1S36 | C1S35 | C1S34 | C1S33 | C1S32 |
| C2SEGV0 | FBA0H | C2S7  | C2S6  | C2S5  | C2S4  | C2S3  | C2S2  | C2S1  | C2S0  |
| C2SEGV1 | FBA1H | C2S15 | C2S14 | C2S13 | C2S12 | C2S11 | C2S10 | C2S9  | C2S8  |
| C2SEGV2 | FBA2H | C2S23 | C2S22 | C2S21 | C2S20 | C2S19 | C2S18 | C2S17 | C2S16 |
| C2SEGV3 | FBA3H | C2S31 | C2S30 | C2S29 | C2S28 | C2S27 | C2S26 | C2S25 | C2S24 |
| C2SEGV4 | FBA4H | C2S39 | C2S38 | C2S37 | C2S36 | C2S35 | C2S34 | C2S33 | C2S32 |
| C3SEGV0 | FBA8H | C3S7  | C3S6  | C3S5  | C3S4  | C3S3  | C3S2  | C3S1  | C3S0  |
| C3SEGV1 | FBA9H | C3S15 | C3S14 | C3S13 | C3S12 | C3S11 | C3S10 | C3S9  | C3S8  |
| C3SEGV2 | FBAAH | C3S23 | C3S22 | C3S21 | C3S20 | C3S19 | C3S18 | C3S17 | C3S16 |
| C3SEGV3 | FBABH | C3S31 | C3S30 | C3S29 | C3S28 | C3S27 | C3S26 | C3S25 | C3S24 |
| C3SEGV4 | FBACH | C3S39 | C3S38 | C3S37 | C3S36 | C3S35 | C3S34 | C3S33 | C3S32 |

CmSn: LCD 矩阵中 COMm-SEGN 的显示数据 (m=0~3, n=0~39)

## 28.3 LCD 显示相关配置

### 28.3.1 配置 LCD 刷新率 (帧速率)

LCD 的刷新率的配置是指设置将整个 LCD 所有的点全部扫描一遍的时间。一般将 LCD 的刷新率配置为 55~60Hz, 显示效果最好。

STC8H 的 LCD 模块中, 刷新率主要有 DBLEN、COMLENL、COMLENM、COMLENH 和 COMON 这几个寄存器进行设置。

$$\text{LCD刷新率} = \frac{\text{LCD时钟频率}}{(\text{DBLEN}[2:0] + \text{COMLEN}[19:0] + 1) * \text{COM数}} \quad (\text{Hz})$$

例如: CPU 工作频率为 24MHz, 选择 CPU 时钟为 LCD 时钟源, 使能 COM0~COM3, 则可将 DBLEN 设置为 2, COMLEN 设置为 99997 (即 COMLENH=0x01, COMLENM=0x86, COMLENL=0x9D), 从而可得到 LCD 刷新率为  $24000000/(4*(2+99997+1))=60\text{Hz}$

### 28.3.2 配置 LCD 闪烁率

当 LCD 的显示模式被配置为闪烁模式时, LCD 的闪烁率由 BLANKRATE 寄存器进行设置。

$$\text{LCD闪烁率} = \frac{\text{LCD刷新率}}{\text{BLANKRATE}[7:0] * 2} \quad (\text{Hz})$$

## 28.4 范例程序

### 28.4.1 LCD 显示 RTC 时钟范例

[5 位 LCD 液晶显示屏显示 RTC 实时时钟的范例](http://www.stcmcu.com/OpenSource/STC8H4K64TLCD-LCD-RTC-Demo-20220810.zip)

(<http://www.stcmcu.com/OpenSource/STC8H4K64TLCD-LCD-RTC-Demo-20220810.zip>)

直接点击上面的超链接下载完整的 Keil 项目

STC MCU

# 29 LCM 接口（8/16 位彩屏模块 I8080/M6800 接口）

| 产品线                    | LCM |
|------------------------|-----|
| STC8H1K08 系列           |     |
| STC8H1K28 系列           |     |
| STC8H3K64S4 系列         |     |
| STC8H3K64S2 系列         |     |
| STC8H8K64U 系列 A 版本     |     |
| STC8H8K64U 系列 B/C/D 版本 | ●   |
| STC8H4K64TL 系列         | ●   |
| STC8H4K64TLCD 系列       | ●   |
| STC8H1K08T 系列          |     |

STC8H 系列的部分单片机内部集成了一个 LCM 接口控制器，可用于驱动目前流行的液晶显示屏模块。可驱动 I8080 接口和 M6800 接口彩屏，支持 8 位和 16 位数据宽度



## 29.1 LCM 接口功能脚切换

| 符号        | 地址    | B7      | B6            | B5           | B4          | B3            | B2 | B1         | B0      |
|-----------|-------|---------|---------------|--------------|-------------|---------------|----|------------|---------|
| LCMIFCFG  | FE50H | LCMIFIE | -             | LCMIFIP[1:0] |             | LCMIFDPS[1:0] |    | D16_D8     | M68_I80 |
| LCMIFCFG2 | FE51H |         | LCMIFCPS[1:0] |              | SETUPT[2:0] |               |    | HOLDT[1:0] |         |

LCMIFCPS[1:0]: LCM 接口控制脚选择位

| LCMIFCPS [1:0] | RS   | I8080 的读信号 RD<br>M6800 的使能信号 E | I8080 的写信号 WR<br>M6800 的读写信号 RW |
|----------------|------|--------------------------------|---------------------------------|
| 00             | P4.5 | P4.4                           | P4.2                            |
| 01             | P4.5 | P3.7                           | P3.6                            |
| 10             | P4.0 | P4.4                           | P4.2                            |
| 11             | P4.0 | P3.7                           | P3.6                            |

LCMIFDPS[1:0]: 8 位数据位 LCM 接口数据脚选择位

| LCMIFDPS [1:0] | D16_D8 | 数据字节 DAT[7:0] |
|----------------|--------|---------------|
| 00             | 0      | P2[7:0]       |
| 01             | 0      | P6[7:0]       |
| 10             | 0      | P2[7:0]       |
| 11             | 0      | P6[7:0]       |

LCMIFDPS[1:0]: 16 位数据位 LCM 接口数据脚选择位

| LCMIFDPS [1:0] | D16_D8 | 数据高字节 DAT[15:8] | 数据低字节 DAT[7:0]                    |
|----------------|--------|-----------------|-----------------------------------|
| 00             | 1      | P2[7:0]         | P0[7:0]                           |
| 01             | 1      | P6[7:0]         | P2[7:0]                           |
| 10             | 1      | P2[7:0]         | {P0[7:4],P4[7],P4[6],P4[3],P4[1]} |
| 11             | 1      | P6[7:0]         | P7[7:0]                           |

## 29.2 LCM 相关的寄存器

| 符号        | 描述            | 地址    | 位地址与符号         |               |              |             |               |          |            |         | 复位值       |
|-----------|---------------|-------|----------------|---------------|--------------|-------------|---------------|----------|------------|---------|-----------|
|           |               |       | B7             | B6            | B5           | B4          | B3            | B2       | B1         | B0      |           |
| LCMIFCFG  | LCM 接口配置寄存器   | FE50H | LCMIFIE        | -             | LCMIFIP[1:0] |             | LCMIFDPS[1:0] |          | D16_D8     | M68_I80 | 0x00,0000 |
| LCMIFCFG2 | LCM 接口配置寄存器 2 | FE51H | -              | LCMIFCPS[1:0] |              | SETUPT[2:0] |               |          | HOLDT[1:0] |         | 0000,0000 |
| LCMIFCR   | LCM 接口控制寄存器   | FE52H | ENLCMIF        | -             | -            | -           | -             | CMD[2:0] |            |         | 0xxx,x000 |
| LCMIFSTA  | LCM 接口状态寄存器   | FE53H | -              | -             | -            | -           | -             | -        | -          | LCMIFIF | xxxx,xxx0 |
| LCMIDDATL | LCM 接口低字节数据   | FE54H | LCMIFDAT[7:0]  |               |              |             |               |          |            |         | 0000,0000 |
| LCMIDDATH | LCM 接口高字节数据   | FE55H | LCMIFDAT[15:8] |               |              |             |               |          |            |         | 0000,0000 |

### 29.2.1 LCM 接口配置寄存器 (LCMIFCFG)

| 符号       | 地址    | B7      | B6 | B5           | B4 | B3            | B2 | B1     | B0      |
|----------|-------|---------|----|--------------|----|---------------|----|--------|---------|
| LCMIFCFG | FE50H | LCMIFIE | -  | LCMIFIP[1:0] |    | LCMIFDPS[1:0] |    | D16_D8 | M68_I80 |

LCMIFIE: LCM 接口中断使能控制位

0: 禁止 LCM 接口中断

1: 允许 LCM 接口中断

LCMIFIP[1:0]: LCM 接口中断优先级控制位

| LCMIFIP[1:0] | 中断优先级   |
|--------------|---------|
| 00           | 最低级 (0) |
| 01           | 较低级 (1) |
| 10           | 较高级 (2) |
| 11           | 最高级 (3) |

LCMIFDPS[1:0]: LCM 接口数据脚选择位

| LCMIFDPS [1:0] | D16_D8 | 数据高字节 DAT[15:8] | 数据低字节 DAT[7:0]                    |
|----------------|--------|-----------------|-----------------------------------|
| 00             | 0      | N/A             | P2[7:0]                           |
| 01             | 0      | N/A             | P6[7:0]                           |
| 10             | 0      | N/A             | P2[7:0]                           |
| 11             | 0      | N/A             | P6[7:0]                           |
| 00             | 1      | P2[7:0]         | P0[7:0]                           |
| 01             | 1      | P6[7:0]         | P2[7:0]                           |
| 10             | 1      | P2[7:0]         | {P0[7:4],P4[7],P4[6],P4[3],P4[1]} |
| 11             | 1      | P6[7:0]         | P7[7:0]                           |

D16\_D8: LCM 接口数据宽度控制位

0: 8 位数据宽度

1: 16 位数据宽度

M68\_I80: LCM 接口模式选择位

0: I8080 模式

1: M6800 模式

## 29.2.2 LCM 接口配置寄存器 2 (LCMIFCFG2)

| 符号        | 地址    | B7 | B6            | B5 | B4          | B3 | B2 | B1         | B0 |
|-----------|-------|----|---------------|----|-------------|----|----|------------|----|
| LCMIFCFG2 | FE51H | -  | LCMIFCPS[1:0] |    | SETUPT[2:0] |    |    | HOLDT[1:0] |    |

LCMIFCPS[1:0]: LCM 接口控制脚选择位

| LCMIFCPS [1:0] | RS   | I8080 的读信号 RD<br>M6800 的使能信号 E | I8080 的写信号 WR<br>M6800 的读写信号 RW |
|----------------|------|--------------------------------|---------------------------------|
| 00             | P4.5 | P4.4                           | P4.2                            |
| 01             | P4.5 | P3.7                           | P3.6                            |
| 10             | P4.0 | P4.4                           | P4.2                            |
| 11             | P4.0 | P3.7                           | P3.6                            |

SETUPT[2:0]: LCM 接口通讯的数据建立时间控制位 (详见后续章节的时序图)

HOLDT[1:0]: LCM 接口通讯的数据保持时间控制位 (详见后续章节的时序图)

## 29.2.3 LCM 接口控制寄存器 (LCMIFCR)

| 符号      | 地址    | B7      | B6 | B5 | B4 | B3 | B2       | B1 | B0 |
|---------|-------|---------|----|----|----|----|----------|----|----|
| LCMIFCR | FE52H | ENLCMIF | -  | -  | -  | -  | CMD[2:0] |    |    |

ENLCMIF: LCM 接口使能控制位

0: 禁止 LCM 接口功能

1: 允许 LCM 接口功能

CMD[2:0]: LCM 接口触发命令

| CMD[2:0] | 触发命令   |
|----------|--------|
| 100      | 写命令    |
| 101      | 写数据    |
| 110      | 读命令/状态 |
| 111      | 读数据    |

## 29.2.4 LCM 接口状态寄存器 (LCMIFSTA)

| 符号       | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0      |
|----------|-------|----|----|----|----|----|----|----|---------|
| LCMIFSTA | FE53H | -  | -  | -  | -  | -  | -  | -  | LCMIFIF |

LCMIFIF: LCM 接口中断请求标志, 需软件清 0

## 29.2.5 LCM 接口数据寄存器 (LCMIFDATL, LCMIFDATH)

| 符号        | 地址    | B7             | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----------|-------|----------------|----|----|----|----|----|----|----|
| LCMIFDATL | FE54H | LCMIFDAT[7:0]  |    |    |    |    |    |    |    |
| LCMIFDATH | FE55H | LCMIFDAT[15:8] |    |    |    |    |    |    |    |

LCMIFDAT: LCM 接口数据寄存器。

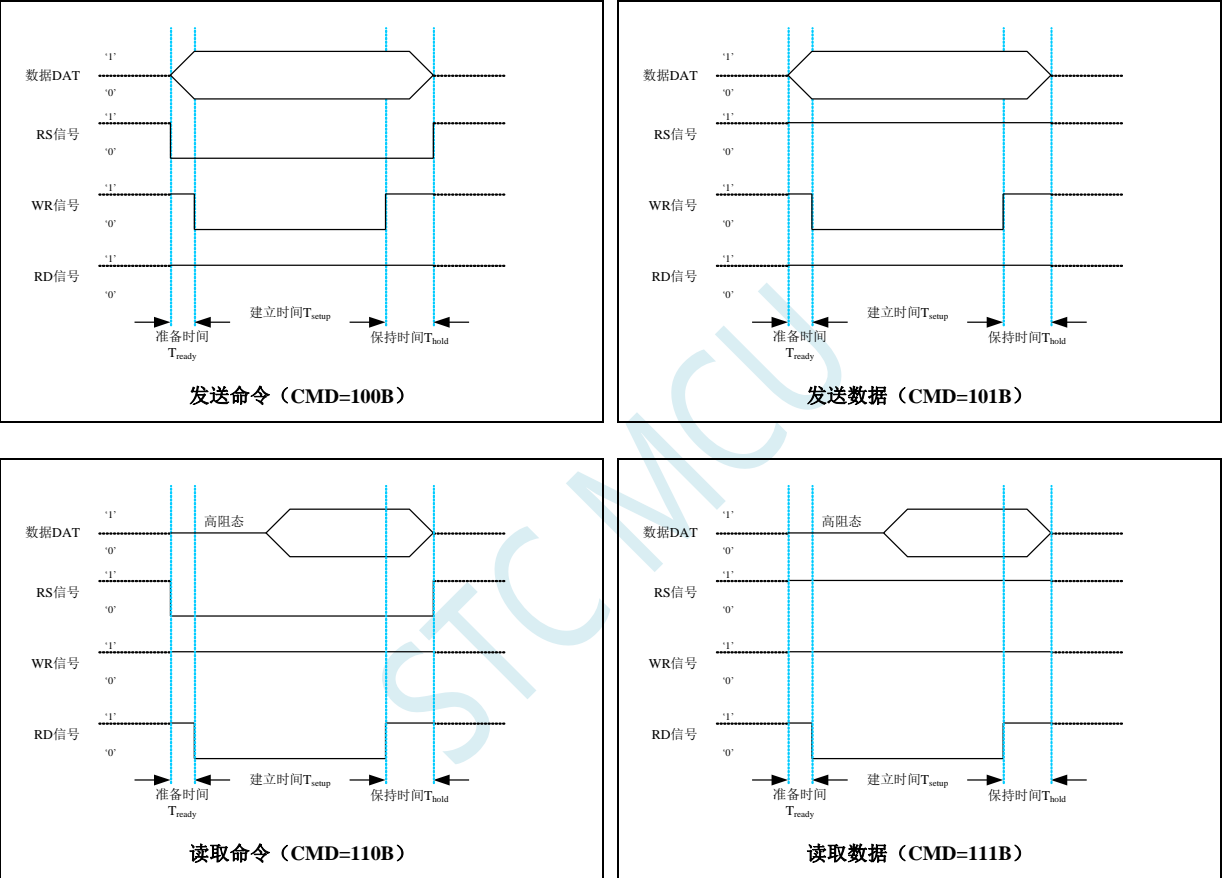
当数据宽度为 8 位数据时, 只有 LCMIFDATL 数据有效;

当数据宽度为 16 位数据时, 由 LCMIFDATL 和 LCMIFDATH 共同组合成 16 位数据

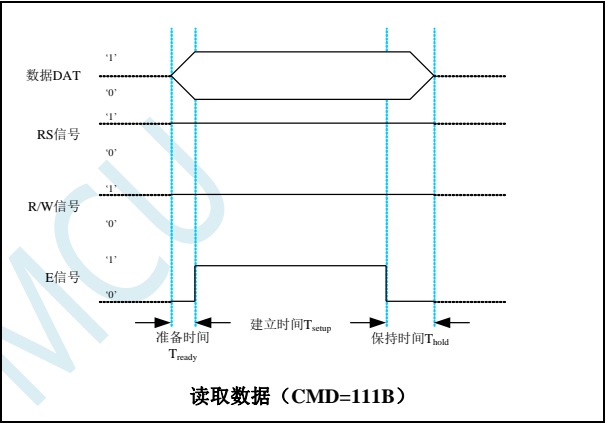
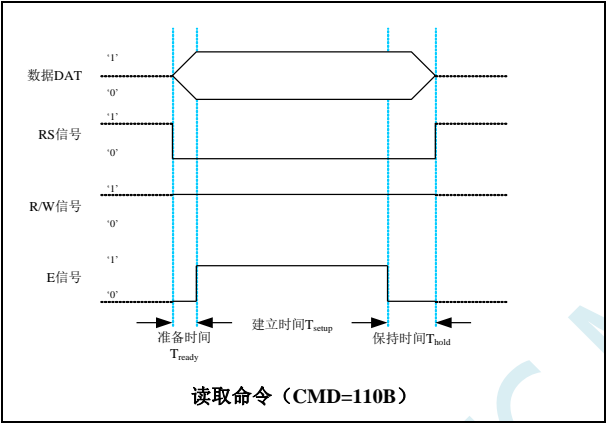
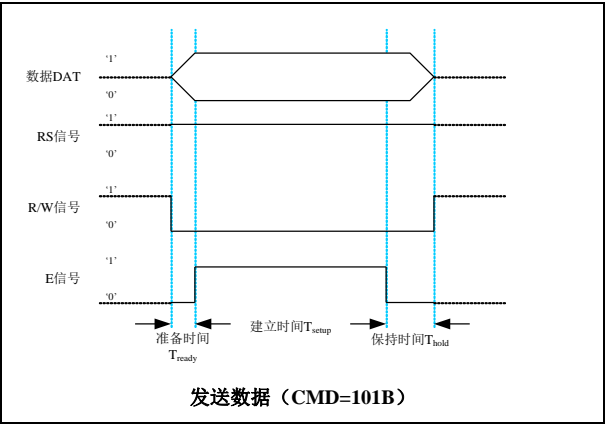
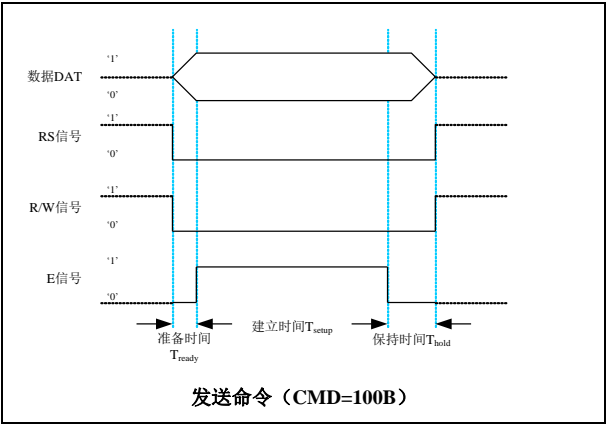
## 29.3 I8080/M6800 模式 LCM 接口时序图

注:  $T_{\text{ready}} = 1$  个系统时钟  
 $T_{\text{setup}} = (\text{SETUPT} + 1)$  个系统时钟  
 $T_{\text{hold}} = (\text{HOLDT} + 1)$  个系统时钟

### 29.3.1 I8080 模式



### 29.3.2 M6800 模式



## 30 DMA（批量数据传输）

| 产品线                    | DMA |
|------------------------|-----|
| STC8H1K08 系列           |     |
| STC8H1K28 系列           |     |
| STC8H3K64S4 系列         |     |
| STC8H3K64S2 系列         |     |
| STC8H8K64U 系列 A 版本     |     |
| STC8H8K64U 系列 B/C/D 版本 | ●   |
| STC8H4K64TL 系列         | ●   |
| STC8H4K64TLCD 系列       | ●   |
| STC8H1K08T 系列          | ●   |

STC8H 系列的部分单片机支持批量数据存储功能，即传统的 DMA。

支持如下几种 DMA 操作：

- M2M\_DMA：XRAM 存储器到 XRAM 存储器的数据读写
- ADC\_DMA：自动扫描使能的 ADC 通道并将转换的 ADC 数据自动存储到 XRAM 中
- SPI\_DMA：自动将 XRAM 中的数据和 SPI 外设之间进行数据交换
- UR1T\_DMA：自动将 XRAM 中的数据通过串口 1 发送出去
- UR1R\_DMA：自动将串口 1 接收到的数据存储到 XRAM 中
- UR2T\_DMA：自动将 XRAM 中的数据通过串口 2 发送出去
- UR2R\_DMA：自动将串口 2 接收到的数据存储到 XRAM 中
- UR3T\_DMA：自动将 XRAM 中的数据通过串口 3 发送出去
- UR3R\_DMA：自动将串口 3 接收到的数据存储到 XRAM 中
- UR4T\_DMA：自动将 XRAM 中的数据通过串口 4 发送出去
- UR4R\_DMA：自动将串口 4 接收到的数据存储到 XRAM 中
- LCM\_DMA：自动将 XRAM 中的数据和 LCM 设备之间进行数据交换

每次 DMA 数据传输最大数据量为 256 字节。

每种 DMA 对 XRAM 的读写操作都可设置 4 级访问优先级，硬件自动进行 XRAM 总线的访问仲裁，不会影响 CPU 的 XRAM 的访问。相同优先级下，不同 DMA 对 XRAM 的访问顺序如下：SPI\_DMA，UR1R\_DMA，UR1T\_DMA，UR2R\_DMA，UR2T\_DMA，UR3R\_DMA，UR3T\_DMA，UR4R\_DMA，UR4T\_DMA，LCM\_DMA，M2M\_DMA，ADC\_DMA

### 30.1 DMA 相关的寄存器

| 符号          | 描述            | 地址    | 位地址与符号 |      |       |       |            |    |             |    | 复位值       |
|-------------|---------------|-------|--------|------|-------|-------|------------|----|-------------|----|-----------|
|             |               |       | B7     | B6   | B5    | B4    | B3         | B2 | B1          | B0 |           |
| DMA_M2M_CFG | M2M_DMA 配置寄存器 | FA00H | M2MIE  | -    | TXACO | RXACO | M2MIP[1:0] |    | M2MPTY[1:0] |    | 0x00,0000 |
| DMA_M2M_CR  | M2M_DMA 控制寄存器 | FA01H | ENM2M  | TRIG | -     | -     | -          | -  | -           | -  | 00xx,xxxx |

|               |                  |       |        |        |        |      |                |       |              |        |           |
|---------------|------------------|-------|--------|--------|--------|------|----------------|-------|--------------|--------|-----------|
| DMA_M2M_STA   | M2M_DMA 状态寄存器    | FA02H | -      | -      | -      | -    | -              | -     | -            | M2MIF  | xxxx,xxx0 |
| DMA_M2M_AMT   | M2M_DMA 传输总字节数   | FA03H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_M2M_DONE  | M2M_DMA 传输完成字节数  | FA04H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_M2M_TXAH  | M2M_DMA 发送高地址    | FA05H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_M2M_TXAL  | M2M_DMA 发送低地址    | FA06H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_M2M_RXAH  | M2M_DMA 接收高地址    | FA07H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_M2M_RXAL  | M2M_DMA 接收低地址    | FA08H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_ADC_CFG   | ADC_DMA 配置寄存器    | FA10H | ADCIE  | -      | -      | -    | ADCMIP[1:0]    |       | ADCPTY[1:0]  |        | 0xxx,0000 |
| DMA_ADC_CR    | ADC_DMA 控制寄存器    | FA11H | ENADC  | TRIG   | -      | -    | -              | -     | -            | -      | 00xx,xxxx |
| DMA_ADC_STA   | ADC_DMA 状态寄存器    | FA12H | -      | -      | -      | -    | -              | -     | -            | ADCIF  | xxxx,xxx0 |
| DMA_ADC_RXAH  | ADC_DMA 接收高地址    | FA17H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_ADC_RXAL  | ADC_DMA 接收低地址    | FA18H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_ADC_CFG2  | ADC_DMA 配置寄存器 2  | FA19H | -      | -      | -      | -    | CVTIMESEL[3:0] |       |              |        | xxxx,0000 |
| DMA_ADC_CHSW0 | ADC_DMA 通道使能     | FA1AH | CH7    | CH6    | CH5    | CH4  | CH3            | CH2   | CH1          | CH0    | 0000,0001 |
| DMA_ADC_CHSW1 | ADC_DMA 通道使能     | FA1BH | CH15   | CH14   | CH13   | CH12 | CH11           | CH10  | CH9          | CH8    | 1000,0000 |
| DMA_SPI_CFG   | SPI_DMA 配置寄存器    | FA20H | SPIIE  | ACT_TX | ACT_RX | -    | SPIIP[1:0]     |       | SPIPTY[1:0]  |        | 000x,0000 |
| DMA_SPI_CR    | SPI_DMA 控制寄存器    | FA21H | ENSPI  | TRIG_M | TRIG_S | -    | -              | -     | -            | CLRIFO | 000x,xxx0 |
| DMA_SPI_STA   | SPI_DMA 状态寄存器    | FA22H | -      | -      | -      | -    | -              | TXOVW | RXLOSS       | SPIIF  | xxxx,x000 |
| DMA_SPI_AMT   | SPI_DMA 传输总字节数   | FA23H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_SPI_DONE  | SPI_DMA 传输完成字节数  | FA24H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_SPI_TXAH  | SPI_DMA 发送高地址    | FA25H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_SPI_TXAL  | SPI_DMA 发送低地址    | FA26H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_SPI_RXAH  | SPI_DMA 接收高地址    | FA27H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_SPI_RXAL  | SPI_DMA 接收低地址    | FA28H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_SPI_CFG2  | SPI_DMA 配置寄存器 2  | FA29H | -      | -      | -      | -    | -              | WRPSS | SSS[1:0]     |        | xxxx,x000 |
| DMA_UR1T_CFG  | UR1T_DMA 配置寄存器   | FA30H | UR1TIE | -      | -      | -    | UR1TIP[1:0]    |       | UR1TPTY[1:0] |        | 0xxx,0000 |
| DMA_UR1T_CR   | UR1T_DMA 控制寄存器   | FA31H | ENUR1T | TRIG   | -      | -    | -              | -     | -            | -      | 00xx,xxxx |
| DMA_UR1T_STA  | UR1T_DMA 状态寄存器   | FA32H | -      | -      | -      | -    | -              | TXOVW | -            | UR1TIF | xxxx,x0x0 |
| DMA_UR1T_AMT  | UR1T_DMA 传输总字节数  | FA33H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_UR1T_DONE | UR1T_DMA 传输完成字节数 | FA34H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_UR1T_TXAH | UR1T_DMA 发送高地址   | FA35H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_UR1T_TXAL | UR1T_DMA 发送低地址   | FA36H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_UR1R_CFG  | UR1R_DMA 配置寄存器   | FA38H | UR1RIE | -      | -      | -    | UR1RIP[1:0]    |       | UR1RPTY[1:0] |        | 0xxx,0000 |
| DMA_UR1R_CR   | UR1R_DMA 控制寄存器   | FA39H | ENUR1R | -      | TRIG   | -    | -              | -     | -            | CLRIFO | 0x0x,xxx0 |
| DMA_UR1R_STA  | UR1R_DMA 状态寄存器   | FA3AH | -      | -      | -      | -    | -              | -     | RXLOSS       | UR1RIF | xxxx,xx00 |
| DMA_UR1R_AMT  | UR1R_DMA 传输总字节数  | FA3BH |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_UR1R_DONE | UR1R_DMA 传输完成字节数 | FA3CH |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_UR1R_RXAH | UR1R_DMA 接收高地址   | FA3DH |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_UR1R_RXAL | UR1R_DMA 接收低地址   | FA3EH |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_UR2T_CFG  | UR2T_DMA 配置寄存器   | FA40H | UR2TIE | -      | -      | -    | UR2TIP[1:0]    |       | UR2TPTY[1:0] |        | 0xxx,0000 |
| DMA_UR2T_CR   | UR2T_DMA 控制寄存器   | FA41H | ENUR2T | TRIG   | -      | -    | -              | -     | -            | -      | 00xx,xxxx |
| DMA_UR2T_STA  | UR2T_DMA 状态寄存器   | FA42H | -      | -      | -      | -    | -              | TXOVW | -            | UR2TIF | xxxx,x0x0 |
| DMA_UR2T_AMT  | UR2T_DMA 传输总字节数  | FA43H |        |        |        |      |                |       |              |        | 0000,0000 |
| DMA_UR2T_DONE | UR2T_DMA 传输完成字节数 | FA44H |        |        |        |      |                |       |              |        | 0000,0000 |

|               |                  |       |        |        |        |        |             |       |              |         |           |
|---------------|------------------|-------|--------|--------|--------|--------|-------------|-------|--------------|---------|-----------|
| DMA_UR2T_TXAH | UR2T_DMA 发送高地址   | FA45H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR2T_TXAL | UR2T_DMA 发送低地址   | FA46H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR2R_CFG  | UR2R_DMA 配置寄存器   | FA48H | UR2RIE | -      | -      | -      | UR2RIP[1:0] |       | UR2RPTY[1:0] |         | 0xxx,0000 |
| DMA_UR2R_CR   | UR2R_DMA 控制寄存器   | FA49H | ENUR2R | -      | TRIG   | -      | -           | -     | -            | CLRFIFO | 0x0x,xxx0 |
| DMA_UR2R_STA  | UR2R_DMA 状态寄存器   | FA4AH | -      | -      | -      | -      | -           | -     | RXLOSS       | UR2RIF  | xxxx,xx00 |
| DMA_UR2R_AMT  | UR2R_DMA 传输总字节数  | FA4BH |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR2R_DONE | UR2R_DMA 传输完成字节数 | FA4CH |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR2R_RXAH | UR2R_DMA 接收高地址   | FA4DH |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR2R_RXAL | UR2R_DMA 接收低地址   | FA4EH |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR3T_CFG  | UR3T_DMA 配置寄存器   | FA50H | UR3TIE | -      | -      | -      | UR3TIP[1:0] |       | UR3TPTY[1:0] |         | 0xxx,0000 |
| DMA_UR3T_CR   | UR3T_DMA 控制寄存器   | FA51H | ENUR3T | TRIG   | -      | -      | -           | -     | -            | -       | 00xx,xxxx |
| DMA_UR3T_STA  | UR3T_DMA 状态寄存器   | FA52H | -      | -      | -      | -      | -           | TXOVW | -            | UR3TIF  | xxxx,x0x0 |
| DMA_UR3T_AMT  | UR3T_DMA 传输总字节数  | FA53H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR3T_DONE | UR3T_DMA 传输完成字节数 | FA54H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR3T_TXAH | UR3T_DMA 发送高地址   | FA55H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR3T_TXAL | UR3T_DMA 发送低地址   | FA56H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR3R_CFG  | UR3R_DMA 配置寄存器   | FA58H | UR3RIE | -      | -      | -      | UR3RIP[1:0] |       | UR3RPTY[1:0] |         | 0xxx,0000 |
| DMA_UR3R_CR   | UR3R_DMA 控制寄存器   | FA59H | ENUR3R | -      | TRIG   | -      | -           | -     | -            | CLRFIFO | 0x0x,xxx0 |
| DMA_UR3R_STA  | UR3R_DMA 状态寄存器   | FA5AH | -      | -      | -      | -      | -           | -     | RXLOSS       | UR3RIF  | xxxx,xx00 |
| DMA_UR3R_AMT  | UR3R_DMA 传输总字节数  | FA5BH |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR3R_DONE | UR3R_DMA 传输完成字节数 | FA5CH |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR3R_RXAH | UR3R_DMA 接收高地址   | FA5DH |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR3R_RXAL | UR3R_DMA 接收低地址   | FA5EH |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR4T_CFG  | UR4T_DMA 配置寄存器   | FA60H | UR4TIE | -      | -      | -      | UR4TIP[1:0] |       | UR4TPTY[1:0] |         | 0xxx,0000 |
| DMA_UR4T_CR   | UR4T_DMA 控制寄存器   | FA61H | ENUR4T | TRIG   | -      | -      | -           | -     | -            | -       | 00xx,xxxx |
| DMA_UR4T_STA  | UR4T_DMA 状态寄存器   | FA62H | -      | -      | -      | -      | -           | TXOVW | -            | UR4TIF  | xxxx,x0x0 |
| DMA_UR4T_AMT  | UR4T_DMA 传输总字节数  | FA63H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR4T_DONE | UR4T_DMA 传输完成字节数 | FA64H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR4T_TXAH | UR4T_DMA 发送高地址   | FA65H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR4T_TXAL | UR4T_DMA 发送低地址   | FA66H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR4R_CFG  | UR4R_DMA 配置寄存器   | FA68H | UR4RIE | -      | -      | -      | UR4RIP[1:0] |       | UR4RPTY[1:0] |         | 0xxx,0000 |
| DMA_UR4R_CR   | UR4R_DMA 控制寄存器   | FA69H | ENUR4R | -      | TRIG   | -      | -           | -     | -            | CLRFIFO | 0x0x,xxx0 |
| DMA_UR4R_STA  | UR4R_DMA 状态寄存器   | FA6AH | -      | -      | -      | -      | -           | -     | RXLOSS       | UR4RIF  | xxxx,xx00 |
| DMA_UR4R_AMT  | UR4R_DMA 传输总字节数  | FA6BH |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR4R_DONE | UR4R_DMA 传输完成字节数 | FA6CH |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR4R_RXAH | UR4R_DMA 接收高地址   | FA6DH |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_UR4R_RXAL | UR4R_DMA 接收低地址   | FA6EH |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_LCM_CFG   | LCM_DMA 配置寄存器    | FA70H | LCMIE  | -      | -      | -      | LCMIP[1:0]  |       | LCMPTY[1:0]  |         | 0xxx,0000 |
| DMA_LCM_CR    | LCM_DMA 控制寄存器    | FA71H | ENLCM  | TRIGWC | TRIGWD | TRIGRC | TRIGRD      | -     | -            | -       | 0000,0xxx |
| DMA_LCM_STA   | LCM_DMA 状态寄存器    | FA72H | -      | -      | -      | -      | -           | -     | TXOVW        | LCMIF   | xxxx,xx00 |
| DMA_LCM_AMT   | LCM_DMA 传输总字节数   | FA73H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_LCM_DONE  | LCM_DMA 传输完成字节数  | FA74H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_LCM_TXAH  | LCM_DMA 发送高地址    | FA75H |        |        |        |        |             |       |              |         | 0000,0000 |
| DMA_LCM_TXAL  | LCM_DMA 发送低地址    | FA76H |        |        |        |        |             |       |              |         | 0000,0000 |



|              |               |       |  |           |
|--------------|---------------|-------|--|-----------|
| DMA_LCM_RXAH | LCM_DMA 接收高地址 | FA77H |  | 0000,0000 |
| DMA_LCM_RXAL | LCM_DMA 接收低地址 | FA78H |  | 0000,0000 |

## 30.2 存储器与存储器之间的数据读写 (M2M\_DMA)

### 30.2.1 M2M\_DMA 配置寄存器 (DMA\_M2M\_CFG)

| 符号          | 地址    | B7    | B6 | B5    | B4    | B3         | B2 | B1          | B0 |
|-------------|-------|-------|----|-------|-------|------------|----|-------------|----|
| DMA_M2M_CFG | FA00H | M2MIE | -  | TXACO | RXACO | M2MIP[1:0] |    | M2MPTY[1:0] |    |

M2MIE: M2M\_DMA 中断使能控制位

- 0: 禁止 M2M\_DMA 中断
- 1: 允许 M2M\_DMA 中断

TXACO: M2M\_DMA 源地址 (读取地址) 改变方向

- 0: 数据读取完成后地址自动递增
- 1: 数据读取完成后地址自动递减

RXACO: M2M\_DMA 目标地址 (写入地址) 改变方向

- 0: 数据写入完成后地址自动递增
- 1: 数据写入完成后地址自动递减

M2MIP[1:0]: M2M\_DMA 中断优先级控制位

| M2MIP[1:0] | 中断优先级   |
|------------|---------|
| 00         | 最低级 (0) |
| 01         | 较低级 (1) |
| 10         | 较高级 (2) |
| 11         | 最高级 (3) |

M2MPTY[1:0]: M2M\_DMA 数据总线访问优先级控制位

| M2MPTY [1:0] | 总线优先级   |
|--------------|---------|
| 00           | 最低级 (0) |
| 01           | 较低级 (1) |
| 10           | 较高级 (2) |
| 11           | 最高级 (3) |

### 30.2.2 M2M\_DMA 控制寄存器 (DMA\_M2M\_CR)

| 符号         | 地址    | B7    | B6   | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|-------|------|----|----|----|----|----|----|
| DMA_M2M_CR | FA01H | ENM2M | TRIG | -  | -  | -  | -  | -  | -  |

ENM2M: M2M\_DMA 功能使能控制位

- 0: 禁止 M2M\_DMA 功能
- 1: 允许 M2M\_DMA 功能

TRIG: M2M\_DMA 数据读写触发控制位

- 0: 写 0 无效
- 1: 写 1 开始 M2M\_DMA 操作,

### 30.2.3 M2M\_DMA 状态寄存器 (DMA\_M2M\_STA)

| 符号 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----|----|----|----|----|----|----|----|----|----|
|----|----|----|----|----|----|----|----|----|----|

|             |       |   |   |   |   |   |   |   |       |
|-------------|-------|---|---|---|---|---|---|---|-------|
| DMA_M2M_STA | FA02H | - | - | - | - | - | - | - | M2MIF |
|-------------|-------|---|---|---|---|---|---|---|-------|

M2MIF: M2M\_DMA 中断请求标志位, 当 M2M\_DMA 操作完成后, 硬件自动将 M2MIF 置 1, 若使能 M2M\_DMA 中断则进入中断服务程序。标志位需软件清零

### 30.2.4 M2M\_DMA 传输总字节寄存器 (DMA\_M2M\_AMT)

| 符号          | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------------|-------|----|----|----|----|----|----|----|----|
| DMA_M2M_AMT | FA03H |    |    |    |    |    |    |    |    |

DMA\_M2M\_AMT: 设置需要进行数据读写的字节数。

**注:** 实际读写的字节数为 (DMA\_M2M\_AMT+1), 即当 DMA\_M2M\_AMT 设置为 0 时, 读写 1 字节, 当 DMA\_M2M\_AMT 设置 255 时, 读写 256 字节

### 30.2.5 M2M\_DMA 传输完成字节寄存器 (DMA\_M2M\_DONE)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|----|----|----|----|----|----|----|----|
| DMA_M2M_DONE | FA04H |    |    |    |    |    |    |    |    |

DMA\_M2M\_DONE: 当前已经读写完成的字节数。

### 30.2.6 M2M\_DMA 发送地址寄存器 (DMA\_M2M\_TXA<sub>x</sub>)

| 符号           | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|------------|----|----|----|----|----|----|----|
| DMA_M2M_TXAH | FA05H | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_M2M_TXAL | FA06H | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_M2M\_TXA: 设置进行数据读写时的源地址。执行 M2M\_DMA 操作时会从这个地址开始读数据。

### 30.2.7 M2M\_DMA 接收地址寄存器 (DMA\_M2M\_RXA<sub>x</sub>)

| 符号           | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|------------|----|----|----|----|----|----|----|
| DMA_M2M_RXAH | FA07H | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_M2M_RXAL | FA08H | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_M2M\_RXA: 设置进行数据读写时的目标地址。执行 M2M\_DMA 操作时会从这个地址开始写入数据。

## 30.3 ADC 数据自动存储 (ADC\_DMA)

### 30.3.1 ADC\_DMA 配置寄存器 (DMA\_ADC\_CFG)

| 符号          | 地址    | B7    | B6 | B5 | B4 | B3         | B2 | B1          | B0 |
|-------------|-------|-------|----|----|----|------------|----|-------------|----|
| DMA_ADC_CFG | FA10H | ADCIE | -  |    |    | ADCIP[1:0] |    | ADCPTY[1:0] |    |

ADCIE: ADC\_DMA 中断使能控制位

0: 禁止 ADC\_DMA 中断

1: 允许 ADC\_DMA 中断

ADCIP[1:0]: ADC\_DMA 中断优先级控制位

| ADCIP[1:0] | 中断优先级   |
|------------|---------|
| 00         | 最低级 (0) |
| 01         | 较低级 (1) |
| 10         | 较高级 (2) |
| 11         | 最高级 (3) |

ADCPTY[1:0]: ADC\_DMA 数据总线访问优先级控制位

| ADCPTY [1:0] | 总线优先级   |
|--------------|---------|
| 00           | 最低级 (0) |
| 01           | 较低级 (1) |
| 10           | 较高级 (2) |
| 11           | 最高级 (3) |

### 30.3.2 ADC\_DMA 控制寄存器 (DMA\_ADC\_CR)

| 符号         | 地址    | B7    | B6   | B5 | B4 | B3 | B2 | B1 | B0 |
|------------|-------|-------|------|----|----|----|----|----|----|
| DMA_ADC_CR | FA11H | ENADC | TRIG | -  | -  | -  | -  | -  | -  |

ENADC: ADC\_DMA 功能使能控制位

0: 禁止 ADC\_DMA 功能

1: 允许 ADC\_DMA 功能

TRIG: ADC\_DMA 操作触发控制位

0: 写 0 无效

1: 写 1 开始 ADC\_DMA 操作,

### 30.3.3 ADC\_DMA 状态寄存器 (DMA\_ADC\_STA)

| 符号          | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0    |
|-------------|-------|----|----|----|----|----|----|----|-------|
| DMA_ADC_STA | FA12H | -  | -  | -  | -  | -  | -  | -  | ADCIF |

ADCIF: ADC\_DMA 中断请求标志位, 当 ADC\_DMA 完成扫描所有使能的 ADC 通道后, 硬件自动将 ADCIF 置 1, 若使能 ADC\_DMA 中断则进入中断服务程序。标志位需软件清零

### 30.3.4 ADC\_DMA 接收地址寄存器 (DMA\_ADC\_RXAx)

| 符号           | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|------------|----|----|----|----|----|----|----|
| DMA_ADC_RXAH | FA17H | ADDR[15:8] |    |    |    |    |    |    |    |

|              |       |           |
|--------------|-------|-----------|
| DMA_ADC_RXAL | FA18H | ADDR[7:0] |
|--------------|-------|-----------|

DMA\_ADC\_RXA: 设置进行 ADC\_DMA 操作时 ADC 转换数据的存储地址。

### 30.3.5 ADC\_DMA 配置寄存器 2 (DMA\_ADC\_CFG2)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3             | B2 | B1 | B0 |
|--------------|-------|----|----|----|----|----------------|----|----|----|
| DMA_ADC_CFG2 | FA19H | -  | -  | -  | -  | CVTIMESEL[3:0] |    |    |    |

CVTIMESEL[3:0]: 设置进行 ADC\_DMA 操作时, 对每个 ADC 通道进行 ADC 转换的次数

| CVTIMESEL[3:0] | 转换次数  |
|----------------|-------|
| 0xxx           | 1 次   |
| 1000           | 2 次   |
| 1001           | 4 次   |
| 1010           | 8 次   |
| 1011           | 16 次  |
| 1100           | 32 次  |
| 1101           | 64 次  |
| 1110           | 128 次 |
| 1111           | 256 次 |

### 30.3.6 ADC\_DMA 通道使能寄存器 (DMA\_ADC\_CHSW<sub>x</sub>)

| 符号            | 地址    | B7   | B6   | B5   | B4   | B3   | B2   | B1  | B0  |
|---------------|-------|------|------|------|------|------|------|-----|-----|
| DMA_ADC_CHSW0 | FA1AH | CH7  | CH6  | CH5  | CH4  | CH3  | CH2  | CH1 | CH0 |
| DMA_ADC_CHSW1 | FA1BH | CH15 | CH14 | CH13 | CH12 | CH11 | CH10 | CH9 | CH8 |

CH<sub>n</sub>: 设置 ADC\_DMA 操作时, 自动扫描的 ADC 通道。通道扫描总是从编号小的通道开始。

### 30.3.7 ADC\_DMA 的数据存储格式

注: ADC 转换速度和转换结果的对齐方式均由 ADC 相关寄存器进行设置

XRAM[DMA\_ADC\_RXA+0] = 使能的第 1 通道的第 1 次 ADC 转换结果的高字节;

XRAM[DMA\_ADC\_RXA+1] = 使能的第 1 通道的第 1 次 ADC 转换结果的低字节;

XRAM[DMA\_ADC\_RXA+2] = 使能的第 1 通道的第 2 次 ADC 转换结果的高字节;

XRAM[DMA\_ADC\_RXA+3] = 使能的第 1 通道的第 2 次 ADC 转换结果的低字节;

...

XRAM[DMA\_ADC\_RXA+2n-2] = 使能的第 1 通道的第 n 次 ADC 转换结果的高字节;

XRAM[DMA\_ADC\_RXA+2n-1] = 使能的第 1 通道的第 n 次 ADC 转换结果的低字节;

XRAM[DMA\_ADC\_RXA+2n] = 第 1 通道的 ADC 通道号;

XRAM[DMA\_ADC\_RXA+2n+1] = 第 1 通道 n 次 ADC 转换结果取完平均值之后的余数;

XRAM[DMA\_ADC\_RXA+2n+2] = 第 1 通道 n 次 ADC 转换结果平均值的高字节;

XRAM[DMA\_ADC\_RXA+2n+3] = 第 1 通道 n 次 ADC 转换结果平均值的低字节;

XRAM[DMA\_ADC\_RXA+(2n+4)+0] = 使能的第 2 通道的第 1 次 ADC 转换结果的高字节;

XRAM[DMA\_ADC\_RXA+(2n+4)+1] = 使能的第 2 通道的第 1 次 ADC 转换结果的低字节;

XRAM[DMA\_ADC\_RXA+(2n+4)+2] = 使能的第 2 通道的第 2 次 ADC 转换结果的高字节;

XRAM[DMA\_ADC\_RXA+(2n+4)+3] = 使能的第 2 通道的第 2 次 ADC 转换结果的低字节;

...

XRAM[DMA\_ADC\_RXA+(2n+4)+2n-2] = 使能的第 2 通道的第 n 次 ADC 转换结果的高字节;

XRAM[DMA\_ADC\_RXA+(2n+4)+2n-1] = 使能的第 2 通道的第 n 次 ADC 转换结果的低字节;

XRAM[DMA\_ADC\_RXA+(2n+4)+2n] = 第 2 通道的 ADC 通道号;

XRAM[DMA\_ADC\_RXA+(2n+4)+2n+1] = 第 2 通道的 n 次 ADC 转换结果取完平均值之后的余数;

XRAM[DMA\_ADC\_RXA+(2n+4)+2n+2] = 第 2 通道的 n 次 ADC 转换结果平均值的高字节;

XRAM[DMA\_ADC\_RXA+(2n+4)+2n+3] = 第 2 通道的 n 次 ADC 转换结果平均值的低字节;

...

XRAM[DMA\_ADC\_RXA+(m-1)(2n+4)+0] = 使能的第 m 通道的第 1 次 ADC 转换结果的高字节;

XRAM[DMA\_ADC\_RXA+(m-1)(2n+4)+1] = 使能的第 m 通道的第 1 次 ADC 转换结果的低字节;

XRAM[DMA\_ADC\_RXA+(m-1)(2n+4)+2] = 使能的第 m 通道的第 2 次 ADC 转换结果的高字节;

XRAM[DMA\_ADC\_RXA+(m-1)(2n+4)+3] = 使能的第 m 通道的第 2 次 ADC 转换结果的低字节;

...

XRAM[DMA\_ADC\_RXA+(m-1)(2n+4)+2n-2] = 使能的第 m 通道的第 n 次 ADC 转换结果的高字节;

XRAM[DMA\_ADC\_RXA+(m-1)(2n+4)+2n-1] = 使能的第 m 通道的第 n 次 ADC 转换结果的低字节;

XRAM[DMA\_ADC\_RXA+(m-1)(2n+4)+2n] = 第 m 通道的 ADC 通道号;

XRAM[DMA\_ADC\_RXA+(m-1)(2n+4)+2n+1] = 第 m 通道的 n 次 ADC 转换结果取完平均值之后的余数;

XRAM[DMA\_ADC\_RXA+(m-1)(2n+4)+2n+2] = 第 m 通道的 n 次 ADC 转换结果平均值的高字节;

XRAM[DMA\_ADC\_RXA+(m-1)(2n+4)+2n+3] = 第 m 通道的 n 次 ADC 转换结果平均值的低字节;

表格形式如下:

| ADC 通道   | 偏移地址                 | 数据                                |
|----------|----------------------|-----------------------------------|
| 第 1 通道   | 0                    | 使能的第 1 通道的第 1 次 ADC 转换结果的高字节      |
|          | 1                    | 使能的第 1 通道的第 1 次 ADC 转换结果的低字节      |
|          | 2                    | 使能的第 1 通道的第 2 次 ADC 转换结果的高字节      |
|          | 3                    | 使能的第 1 通道的第 2 次 ADC 转换结果的低字节      |
|          | ...                  | ...                               |
|          | $2n-2$               | 使能的第 1 通道的第 $n$ 次 ADC 转换结果的高字节    |
|          | $2n-1$               | 使能的第 1 通道的第 $n$ 次 ADC 转换结果的低字节    |
|          | $2n$                 | 第 1 通道的 ADC 通道号                   |
|          | $2n+1$               | 第 1 通道 $n$ 次 ADC 转换结果取完平均值之后的余数   |
|          | $2n+2$               | 第 1 通道 $n$ 次 ADC 转换结果平均值的高字节      |
|          | $2n+3$               | 第 1 通道 $n$ 次 ADC 转换结果平均值的低字节      |
| 第 2 通道   | $(2n+4) + 0$         | 使能的第 2 通道的第 1 次 ADC 转换结果的高字节      |
|          | $(2n+4) + 1$         | 使能的第 2 通道的第 1 次 ADC 转换结果的低字节      |
|          | $(2n+4) + 2$         | 使能的第 2 通道的第 2 次 ADC 转换结果的高字节      |
|          | $(2n+4) + 3$         | 使能的第 2 通道的第 2 次 ADC 转换结果的低字节      |
|          | ...                  | ...                               |
|          | $(2n+4) + 2n-2$      | 使能的第 2 通道的第 $n$ 次 ADC 转换结果的高字节    |
|          | $(2n+4) + 2n-1$      | 使能的第 2 通道的第 $n$ 次 ADC 转换结果的低字节    |
|          | $(2n+4) + 2n$        | 第 2 通道的 ADC 通道号                   |
|          | $(2n+4) + 2n+1$      | 第 2 通道 $n$ 次 ADC 转换结果取完平均值之后的余数   |
|          | $(2n+4) + 2n+2$      | 第 2 通道 $n$ 次 ADC 转换结果平均值的高字节      |
|          | $(2n+4) + 2n+3$      | 第 2 通道 $n$ 次 ADC 转换结果平均值的低字节      |
| 第 $m$ 通道 | ...                  | ...                               |
|          | $(m-1)(2n+4) + 0$    | 使能的第 $m$ 通道的第 1 次 ADC 转换结果的高字节    |
|          | $(m-1)(2n+4) + 1$    | 使能的第 $m$ 通道的第 1 次 ADC 转换结果的低字节    |
|          | $(m-1)(2n+4) + 2$    | 使能的第 $m$ 通道的第 2 次 ADC 转换结果的高字节    |
|          | $(m-1)(2n+4) + 3$    | 使能的第 $m$ 通道的第 2 次 ADC 转换结果的低字节    |
|          | ...                  | ...                               |
|          | $(m-1)(2n+4) + 2n-2$ | 使能的第 $m$ 通道的第 $n$ 次 ADC 转换结果的高字节  |
|          | $(m-1)(2n+4) + 2n-1$ | 使能的第 $m$ 通道的第 $n$ 次 ADC 转换结果的低字节  |
|          | $(m-1)(2n+4) + 2n$   | 第 $m$ 通道的 ADC 通道号                 |
|          | $(m-1)(2n+4) + 2n+1$ | 第 $m$ 通道 $n$ 次 ADC 转换结果取完平均值之后的余数 |
|          | $(m-1)(2n+4) + 2n+2$ | 第 $m$ 通道 $n$ 次 ADC 转换结果平均值的高字节    |
|          | $(m-1)(2n+4) + 2n+3$ | 第 $m$ 通道 $n$ 次 ADC 转换结果平均值的低字节    |

## 30.4 SPI 与存储器之间的数据交换 (SPI\_DMA)

### 30.4.1 SPI\_DMA 配置寄存器 (DMA\_SPI\_CFG)

| 符号          | 地址    | B7    | B6     | B5     | B4 | B3         | B2 | B1          | B0 |
|-------------|-------|-------|--------|--------|----|------------|----|-------------|----|
| DMA_SPI_CFG | FA20H | SPIIE | ACT_TX | ACT_RX | -  | SPIIP[1:0] |    | SPIPTY[1:0] |    |

SPIIE: SPI\_DMA 中断使能控制位

- 0: 禁止 SPI\_DMA 中断
- 1: 允许 SPI\_DMA 中断

ACT\_TX: SPI\_DMA 发送数据控制位

- 0: 禁止 SPI\_DMA 发送数据。主机模式时, SPI 只发送时钟到 SCLK 端口, 但不从 XRAM 读取数据, 也不向 MOSI 端口上发送数据; 从机模式时, SPI 不从 XRAM 读取数据, 也不向 MISO 端口上发送数据
- 1: 允许 SPI\_DMA 发送数据。主机模式时, SPI 发送时钟到 SCLK 端口, 同时从 XRAM 读取数据, 并将数据发送到 MOSI 端口; 从机模式时, SPI 从 XRAM 读取数据, 并将数据发送到 MISO 端口

ACT\_RX: SPI\_DMA 接收数据控制位

- 0: 禁止 SPI\_DMA 接收数据。主机模式时, SPI 只发送时钟到 SCLK 端口, 但不从 MISO 端口读取数据, 也不向 XRAM 写数据; 从机模式时, SPI 不从 MOSI 端口读取数据, 也不向 XRAM 写数据。
- 1: 允许 SPI\_DMA 接收数据。主机模式时, SPI 发送时钟到 SCLK 端口, 同时从 MISO 端口读取数据, 并将数据写入 XRAM; 从机模式时, SPI 从 MOSI 端口读取数据, 并写入 XRAM。

SPIIP[1:0]: SPI\_DMA 中断优先级控制位

| SPIIP[1:0] | 中断优先级   |
|------------|---------|
| 00         | 最低级 (0) |
| 01         | 较低级 (1) |
| 10         | 较高级 (2) |
| 11         | 最高级 (3) |

SPIPTY[1:0]: SPI\_DMA 数据总线访问优先级控制位

| SPIPTY [1:0] | 总线优先级   |
|--------------|---------|
| 00           | 最低级 (0) |
| 01           | 较低级 (1) |
| 10           | 较高级 (2) |
| 11           | 最高级 (3) |

### 30.4.2 SPI\_DMA 控制寄存器 (DMA\_SPI\_CR)

| 符号         | 地址    | B7    | B6     | B5     | B4 | B3 | B2 | B1 | B0     |
|------------|-------|-------|--------|--------|----|----|----|----|--------|
| DMA_SPI_CR | FA21H | ENSPI | TRIG_M | TRIG_S | -  | -  | -  | -  | CLRIFO |

ENSPI: SPI\_DMA 功能使能控制位

- 0: 禁止 SPI\_DMA 功能
- 1: 允许 SPI\_DMA 功能

TRIG\_M: SPI\_DMA 主机模式触发控制位

- 0: 写 0 无效
- 1: 写 1 开始 SPI\_DMA 主机模式操作,



TRIG\_S: SPI\_DMA 从机模式触发控制位

0: 写 0 无效

1: 写 1 开始 SPI\_DMA 从机模式操作,

CLRIFO: 清除 SPI\_DMA 接收 FIFO 控制位

0: 写 0 无效

1: 开始 SPI\_DMA 操作前, 先清空 SPI\_DMA 内置的 FIFO

### 30.4.3 SPI\_DMA 状态寄存器 (DMA\_SPI\_STA)

| 符号          | 地址    | B7 | B6 | B5 | B4 | B3 | B2    | B1     | B0    |
|-------------|-------|----|----|----|----|----|-------|--------|-------|
| DMA_SPI_STA | FA22H | -  | -  | -  | -  | -  | TXOVW | RXLOSS | SPIIF |

SPIIF: SPI\_DMA 中断请求标志位, 当 SPI\_DMA 数据交换完成后, 硬件自动将 SPIIF 置 1, 若使能 SPI\_DMA 中断则进入中断服务程序。标志位需软件清零

RXLOSS: SPI\_DMA 接收数据丢失标志位。SPI\_DMA 操作过程中, 当 XRAM 总线过于繁忙, 来不及清空 SPI\_DMA 的接收 FIFO 导致 SPI\_DMA 接收的数据自动丢失时, 硬件自动将 RXLOSS 置 1。标志位需软件清零

TXOVW: SPI\_DMA 数据覆盖标志位。SPI\_DMA 正在数据传输过程中, 主机模式的 SPI 写 SPDAT 寄存器再次触发 SPI 数据传输时, 会导致数据传输失败, 此时硬件自动将 TXOVW 置 1。标志位需软件清零

### 30.4.4 SPI\_DMA 传输总字节寄存器 (DMA\_SPI\_AMT)

| 符号          | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------------|-------|----|----|----|----|----|----|----|----|
| DMA_SPI_AMT | FA23H |    |    |    |    |    |    |    |    |

DMA\_SPI\_AMT: 设置需要进行数据读写的字节数。

注: 实际读写的字节数为 (DMA\_SPI\_AMT+1), 即当 DMA\_SPI\_AMT 设置为 0 时, 传输 1 字节, 当 DMA\_SPI\_AMT 设置 255 时, 传输 256 字节

### 30.4.5 SPI\_DMA 传输完成字节寄存器 (DMA\_SPI\_DONE)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|----|----|----|----|----|----|----|----|
| DMA_SPI_DONE | FA24H |    |    |    |    |    |    |    |    |

DMA\_SPI\_DONE: 当前已经传输完成的字节数。

### 30.4.6 SPI\_DMA 发送地址寄存器 (DMA\_SPI\_TXAx)

| 符号           | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|------------|----|----|----|----|----|----|----|
| DMA_SPI_TXAH | FA25H | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_SPI_TXAL | FA26H | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_SPI\_TXA: 设置进行数据传输时的源地址。执行 SPI\_DMA 操作时会从这个地址开始读数据。

### 30.4.7 SPI\_DMA 接收地址寄存器 (DMA\_SPI\_RXAx)

| 符号           | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|------------|----|----|----|----|----|----|----|
| DMA_SPI_RXAH | FA27H | ADDR[15:8] |    |    |    |    |    |    |    |



|              |       |           |
|--------------|-------|-----------|
| DMA_SPI_RXAL | FA28H | ADDR[7:0] |
|--------------|-------|-----------|

DMA\_SPI\_RXA: 设置进行数据传输时的目标地址。执行 SPI\_DMA 操作时会从这个地址开始写入数据。

### 30.4.8 SPI\_DMA 配置寄存 2 器 (DMA\_SPI\_CFG2)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2    | B1       | B0 |
|--------------|-------|----|----|----|----|----|-------|----------|----|
| DMA_SPI_CFG2 | FA29H | -  | -  | -  | -  | -  | WRPSS | SSS[1:0] |    |

WRPSS: SPI\_DMA 过程中使能 SS 脚控制位

0: SPI\_DMA 传输过程中, 不自动控制 SS 脚

1: SPI\_DMA 传输过程中, 自动拉低 SS 脚, 传输完成后, 自动恢复原始状态

SSS[1:0]: SPI\_DMA 过程中, 自动控制 SS 选择位

| SSS[1:0] | SS 脚 |
|----------|------|
| 00       | P1.2 |
| 01       | P2.2 |
| 10       | P7.4 |
| 11       | P3.5 |

## 30.5 串口 1 与存储器之间的数据交换 (UR1T\_DMA, UR1R\_DMA)

### 30.5.1 UR1T\_DMA 配置寄存器 (DMA\_UR1T\_CFG)

| 符号           | 地址    | B7     | B6 | B5 | B4 | B3          | B2 | B1           | B0 |
|--------------|-------|--------|----|----|----|-------------|----|--------------|----|
| DMA_UR1T_CFG | FA30H | UR1TIE | -  | -  | -  | UR1TIP[1:0] |    | UR1TPTY[1:0] |    |

UR1TIE: UR1T\_DMA 中断使能控制位

0: 禁止 UR1T\_DMA 中断

1: 允许 UR1T\_DMA 中断

UR1TIP[1:0]: UR1T\_DMA 中断优先级控制位

| UR1TIP[1:0] | 中断优先级   |
|-------------|---------|
| 00          | 最低级 (0) |
| 01          | 较低级 (1) |
| 10          | 较高级 (2) |
| 11          | 最高级 (3) |

UR1TPTY[1:0]: UR1T\_DMA 数据总线访问优先级控制位

| UR1TPTY [1:0] | 总线优先级   |
|---------------|---------|
| 00            | 最低级 (0) |
| 01            | 较低级 (1) |
| 10            | 较高级 (2) |
| 11            | 最高级 (3) |

### 30.5.2 UR1T\_DMA 控制寄存器 (DMA\_UR1T\_CR)

| 符号          | 地址    | B7     | B6   | B5 | B4 | B3 | B2 | B1 | B0 |
|-------------|-------|--------|------|----|----|----|----|----|----|
| DMA_UR1T_CR | FA31H | ENUR1T | TRIG | -  | -  | -  | -  | -  | -  |

ENUR1T: UR1T\_DMA 功能使能控制位

0: 禁止 UR1T\_DMA 功能

1: 允许 UR1T\_DMA 功能

TRIG: UR1T\_DMA 串口 1 发送触发控制位

0: 写 0 无效

1: 写 1 开始 UR1T\_DMA 自动发送数据

### 30.5.3 UR1T\_DMA 状态寄存器 (DMA\_UR1T\_STA)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2    | B1 | B0     |
|--------------|-------|----|----|----|----|----|-------|----|--------|
| DMA_UR1T_STA | FA32H | -  | -  | -  | -  | -  | TXOVW | -  | UR1TIF |

UR1TIF: UR1T\_DMA 中断请求标志位, 当 UR1T\_DMA 数据发送完成后, 硬件自动将 UR1TIF 置 1, 若使能 UR1T\_DMA 中断则进入中断服务程序。标志位需软件清零

TXOVW: UR1T\_DMA 数据覆盖标志位。UR1T\_DMA 正在数据传输过程中, 串口写 SBUF 寄存器再次触发串口发送数据时, 会导致数据传输失败, 此时硬件自动将 TXOVW 置 1。标志位需软件清零

### 30.5.4 UR1T\_DMA 传输总字节寄存器 (DMA\_UR1T\_AMT)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR1T_AMT | FA33H |    |    |    |    |    |    |    |    |

DMA\_UR1T\_AMT: 设置需要自动发送数据的字节数。

注: 实际的字节数为 (DMA\_UR1T\_AMT+1), 即当 DMA\_UR1T\_AMT 设置为 0 时, 传输 1 字节, 当 DMA\_UR1T\_AMT 设置 255 时, 传输 256 字节

### 30.5.5 UR1T\_DMA 传输完成字节寄存器 (DMA\_UR1T\_DONE)

| 符号            | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR1T_DONE | FA34H |    |    |    |    |    |    |    |    |

DMA\_UR1T\_DONE: 当前已经发送完成的字节数。

### 30.5.6 UR1T\_DMA 发送地址寄存器 (DMA\_UR1T\_TXAx)

| 符号            | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|------------|----|----|----|----|----|----|----|
| DMA_UR1T_TXAH | FA35H | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_UR1T_TXAL | FA36H | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_UR1T\_TXA: 设置自动发送数据的源地址。执行 UR1T\_DMA 操作时会从这个地址开始读数据。

### 30.5.7 UR1R\_DMA 配置寄存器 (DMA\_UR1R\_CFG)

| 符号           | 地址    | B7     | B6 | B5 | B4 | B3          | B2 | B1           | B0 |
|--------------|-------|--------|----|----|----|-------------|----|--------------|----|
| DMA_UR1R_CFG | FA38H | UR1RIE | -  | -  | -  | UR1RIP[1:0] |    | UR1RPTY[1:0] |    |

UR1RIE: UR1R\_DMA 中断使能控制位

0: 禁止 UR1R\_DMA 中断

1: 允许 UR1R\_DMA 中断

UR1RIP[1:0]: UR1R\_DMA 中断优先级控制位

| UR1RIP[1:0] | 中断优先级   |
|-------------|---------|
| 00          | 最低级 (0) |
| 01          | 较低级 (1) |
| 10          | 较高级 (2) |
| 11          | 最高级 (3) |

UR1RPTY[1:0]: UR1R\_DMA 数据总线访问优先级控制位

| UR1RPTY [1:0] | 总线优先级   |
|---------------|---------|
| 00            | 最低级 (0) |
| 01            | 较低级 (1) |
| 10            | 较高级 (2) |
| 11            | 最高级 (3) |

### 30.5.8 UR1R\_DMA 控制寄存器 (DMA\_UR1R\_CR)

| 符号          | 地址    | B7     | B6 | B5   | B4 | B3 | B2 | B1 | B0      |
|-------------|-------|--------|----|------|----|----|----|----|---------|
| DMA_UR1R_CR | FA39H | ENUR1R | -  | TRIG | -  | -  | -  | -  | CLRFIFO |

ENUR1R: UR1R\_DMA 功能使能控制位

0: 禁止 UR1R\_DMA 功能

1: 允许 UR1R\_DMA 功能

TRIG: UR1R\_DMA 串口 1 接收触发控制位

0: 写 0 无效

1: 写 1 开始 UR1R\_DMA 自动接收数据

CLRFIFO: 清除 UR1R\_DMA 接收 FIFO 控制位

0: 写 0 无效

1: 开始 UR1R\_DMA 操作前, 先清空 UR1R\_DMA 内置的 FIFO

### 30.5.9 UR1R\_DMA 状态寄存器 (DMA\_UR1R\_STA)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1     | B0     |
|--------------|-------|----|----|----|----|----|----|--------|--------|
| DMA_UR1R_STA | FA3AH | -  | -  | -  | -  | -  | -  | RXLOSS | UR1RIF |

UR1RIF: UR1R\_DMA 中断请求标志位, 当 UR1R\_DMA 接收数据完成后, 硬件自动将 UR1RIF 置 1, 若使能 UR1R\_DMA 中断则进入中断服务程序。标志位需软件清零

RXLOSS: UR1R\_DMA 接收数据丢失标志位。UR1R\_DMA 操作过程中, 当 XRAM 总线过于繁忙, 来不及清空 UR1R\_DMA 的接收 FIFO 导致 UR1R\_DMA 接收的数据自动丢失时, 硬件自动将 RXLOSS 置 1。标志位需软件清零

### 30.5.10 UR1R\_DMA 传输总字节寄存器 (DMA\_UR1R\_AMT)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR1R_AMT | FA3BH |    |    |    |    |    |    |    |    |

DMA\_UR1R\_AMT: 设置需要自动接收数据的字节数。

注: 实际的字节数为 (DMA\_UR1R\_AMT+1), 即当 DMA\_UR1R\_AMT 设置为 0 时, 传输 1 字节, 当 DMA\_UR1R\_AMT 设置 255 时, 传输 256 字节

### 30.5.11 UR1R\_DMA 传输完成字节寄存器 (DMA\_UR1R\_DONE)

| 符号            | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR1R_DONE | FA3CH |    |    |    |    |    |    |    |    |

DMA\_UR1R\_DONE: 当前已经接收完成的字节数。

### 30.5.12 UR1R\_DMA 接收地址寄存器 (DMA\_UR1R\_RXAx)

| 符号            | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|------------|----|----|----|----|----|----|----|
| DMA_UR1R_RXAH | FA3DH | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_UR1R_RXAL | FA3EH | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_UR1R\_RXA: 设置自动接收数据的目标地址。执行 UR1R\_DMA 操作时会从这个地址开始写数据。

## 30.6 串口 2 与存储器之间的数据交换 (UR2T\_DMA, UR2R\_DMA)

### 30.6.1 UR2T\_DMA 配置寄存器 (DMA\_UR2T\_CFG)

| 符号           | 地址    | B7     | B6 | B5 | B4 | B3          | B2 | B1           | B0 |
|--------------|-------|--------|----|----|----|-------------|----|--------------|----|
| DMA_UR2T_CFG | FA40H | UR2TIE | -  | -  | -  | UR2TIP[1:0] |    | UR2TPTY[1:0] |    |

UR2TIE: UR2T\_DMA 中断使能控制位

0: 禁止 UR2T\_DMA 中断

1: 允许 UR2T\_DMA 中断

UR2TIP[1:0]: UR2T\_DMA 中断优先级控制位

| UR2TIP[1:0] | 中断优先级   |
|-------------|---------|
| 00          | 最低级 (0) |
| 01          | 较低级 (1) |
| 10          | 较高级 (2) |
| 11          | 最高级 (3) |

UR2TPTY[1:0]: UR2T\_DMA 数据总线访问优先级控制位

| UR2TPTY [1:0] | 总线优先级   |
|---------------|---------|
| 00            | 最低级 (0) |
| 01            | 较低级 (1) |
| 10            | 较高级 (2) |
| 11            | 最高级 (3) |

### 30.6.2 UR2T\_DMA 控制寄存器 (DMA\_UR2T\_CR)

| 符号          | 地址    | B7     | B6   | B5 | B4 | B3 | B2 | B1 | B0 |
|-------------|-------|--------|------|----|----|----|----|----|----|
| DMA_UR2T_CR | FA41H | ENUR2T | TRIG | -  | -  | -  | -  | -  | -  |

ENUR2T: UR2T\_DMA 功能使能控制位

0: 禁止 UR2T\_DMA 功能

1: 允许 UR2T\_DMA 功能

TRIG: UR2T\_DMA 串口 1 发送触发控制位

0: 写 0 无效

1: 写 1 开始 UR2T\_DMA 自动发送数据

### 30.6.3 UR2T\_DMA 状态寄存器 (DMA\_UR2T\_STA)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2    | B1 | B0     |
|--------------|-------|----|----|----|----|----|-------|----|--------|
| DMA_UR2T_STA | FA42H | -  | -  | -  | -  | -  | TXOVW | -  | UR2TIF |

UR2TIF: UR2T\_DMA 中断请求标志位, 当 UR2T\_DMA 数据发送完成后, 硬件自动将 UR2TIF 置 1, 若使能 UR2T\_DMA 中断则进入中断服务程序。标志位需软件清零

TXOVW: UR2T\_DMA 数据覆盖标志位。UR2T\_DMA 正在数据传输过程中, 串口写 S2BUF 寄存器再次触发串口发送数据时, 会导致数据传输失败, 此时硬件自动将 TXOVW 置 1。标志位需软件清零

### 30.6.4 UR2T\_DMA 传输总字节寄存器 (DMA\_UR2T\_AMT)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR2T_AMT | FA43H |    |    |    |    |    |    |    |    |

DMA\_UR2T\_AMT: 设置需要自动发送数据的字节数。

注: 实际的字节数为 (DMA\_UR2T\_AMT+1), 即当 DMA\_UR2T\_AMT 设置为 0 时, 传输 1 字节, 当 DMA\_UR2T\_AMT 设置 255 时, 传输 256 字节

### 30.6.5 UR2T\_DMA 传输完成字节寄存器 (DMA\_UR2T\_DONE)

| 符号            | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR2T_DONE | FA44H |    |    |    |    |    |    |    |    |

DMA\_UR2T\_DONE: 当前已经发送完成的字节数。

### 30.6.6 UR2T\_DMA 发送地址寄存器 (DMA\_UR2T\_TXAx)

| 符号            | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|------------|----|----|----|----|----|----|----|
| DMA_UR2T_TXAH | FA45H | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_UR2T_TXAL | FA46H | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_UR2T\_TXA: 设置自动发送数据的源地址。执行 UR2T\_DMA 操作时会从这个地址开始读数据。

### 30.6.7 UR2R\_DMA 配置寄存器 (DMA\_UR2R\_CFG)

| 符号           | 地址    | B7     | B6 | B5 | B4 | B3          | B2 | B1           | B0 |
|--------------|-------|--------|----|----|----|-------------|----|--------------|----|
| DMA_UR2R_CFG | FA48H | UR2RIE | -  | -  | -  | UR2RIP[1:0] |    | UR2RPTY[1:0] |    |

UR2RIE: UR2R\_DMA 中断使能控制位

0: 禁止 UR2R\_DMA 中断

1: 允许 UR2R\_DMA 中断

UR2RIP[1:0]: UR2R\_DMA 中断优先级控制位

| UR2RIP[1:0] | 中断优先级   |
|-------------|---------|
| 00          | 最低级 (0) |
| 01          | 较低级 (1) |
| 10          | 较高级 (2) |
| 11          | 最高级 (3) |

UR2RPTY[1:0]: UR2R\_DMA 数据总线访问优先级控制位

| UR2RPTY [1:0] | 总线优先级   |
|---------------|---------|
| 00            | 最低级 (0) |
| 01            | 较低级 (1) |
| 10            | 较高级 (2) |
| 11            | 最高级 (3) |

### 30.6.8 UR2R\_DMA 控制寄存器 (DMA\_UR2R\_CR)

| 符号          | 地址    | B7     | B6 | B5   | B4 | B3 | B2 | B1 | B0      |
|-------------|-------|--------|----|------|----|----|----|----|---------|
| DMA_UR2R_CR | FA49H | ENUR2R | -  | TRIG | -  | -  | -  | -  | CLRFIFO |

ENUR2R: UR2R\_DMA 功能使能控制位

0: 禁止 UR2R\_DMA 功能

1: 允许 UR2R\_DMA 功能

TRIG: UR2R\_DMA 串口 1 接收触发控制位

0: 写 0 无效

1: 写 1 开始 UR2R\_DMA 自动接收数据

CLRFIFO: 清除 UR2R\_DMA 接收 FIFO 控制位

0: 写 0 无效

1: 开始 UR2R\_DMA 操作前, 先清空 UR2R\_DMA 内置的 FIFO

### 30.6.9 UR2R\_DMA 状态寄存器 (DMA\_UR2R\_STA)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1     | B0     |
|--------------|-------|----|----|----|----|----|----|--------|--------|
| DMA_UR2R_STA | FA4AH | -  | -  | -  | -  | -  | -  | RXLOSS | UR2RIF |

UR2RIF: UR2R\_DMA 中断请求标志位, 当 UR2R\_DMA 接收数据完成后, 硬件自动将 UR2RIF 置 1, 若使能 UR2R\_DMA 中断则进入中断服务程序。标志位需软件清零

RXLOSS: UR2R\_DMA 接收数据丢失标志位。UR2R\_DMA 操作过程中, 当 XRAM 总线过于繁忙, 来不及清空 UR2R\_DMA 的接收 FIFO 导致 UR2R\_DMA 接收的数据自动丢失时, 硬件自动将 RXLOSS 置 1。标志位需软件清零

### 30.6.10 UR2R\_DMA 传输总字节寄存器 (DMA\_UR2R\_AMT)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR2R_AMT | FA4BH |    |    |    |    |    |    |    |    |

DMA\_UR2R\_AMT: 设置需要自动接收数据的字节数。

注: 实际的字节数为 (DMA\_UR2R\_AMT+1), 即当 DMA\_UR2R\_AMT 设置为 0 时, 传输 1 字节, 当 DMA\_UR2R\_AMT 设置 255 时, 传输 256 字节

### 30.6.11 UR2R\_DMA 传输完成字节寄存器 (DMA\_UR2R\_DONE)

| 符号            | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR2R_DONE | FA4CH |    |    |    |    |    |    |    |    |

DMA\_UR2R\_DONE: 当前已经接收完成的字节数。

### 30.6.12 UR2R\_DMA 接收地址寄存器 (DMA\_UR2R\_RXAx)

| 符号            | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|------------|----|----|----|----|----|----|----|
| DMA_UR2R_RXAH | FA4DH | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_UR2R_RXAL | FA4EH | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_UR2R\_RXA: 设置自动接收数据的目标地址。执行 UR2R\_DMA 操作时会从这个地址开始写数据。

## 30.7 串口 3 与存储器之间的数据交换 (UR3T\_DMA, UR3R\_DMA)

### 30.7.1 UR3T\_DMA 配置寄存器 (DMA\_UR3T\_CFG)

| 符号           | 地址    | B7     | B6 | B5 | B4 | B3          | B2 | B1           | B0 |
|--------------|-------|--------|----|----|----|-------------|----|--------------|----|
| DMA_UR3T_CFG | FA50H | UR3TIE | -  | -  | -  | UR3TIP[1:0] |    | UR3TPTY[1:0] |    |

UR3TIE: UR3T\_DMA 中断使能控制位

0: 禁止 UR3T\_DMA 中断

1: 允许 UR3T\_DMA 中断

UR3TIP[1:0]: UR3T\_DMA 中断优先级控制位

| UR3TIP[1:0] | 中断优先级   |
|-------------|---------|
| 00          | 最低级 (0) |
| 01          | 较低级 (1) |
| 10          | 较高级 (2) |
| 11          | 最高级 (3) |

UR3TPTY[1:0]: UR3T\_DMA 数据总线访问优先级控制位

| UR3TPTY [1:0] | 总线优先级   |
|---------------|---------|
| 00            | 最低级 (0) |
| 01            | 较低级 (1) |
| 10            | 较高级 (2) |
| 11            | 最高级 (3) |

### 30.7.2 UR3T\_DMA 控制寄存器 (DMA\_UR3T\_CR)

| 符号          | 地址    | B7     | B6   | B5 | B4 | B3 | B2 | B1 | B0 |
|-------------|-------|--------|------|----|----|----|----|----|----|
| DMA_UR3T_CR | FA51H | ENUR3T | TRIG | -  | -  | -  | -  | -  | -  |

ENUR3T: UR3T\_DMA 功能使能控制位

0: 禁止 UR3T\_DMA 功能

1: 允许 UR3T\_DMA 功能

TRIG: UR3T\_DMA 串口 1 发送触发控制位

0: 写 0 无效

1: 写 1 开始 UR3T\_DMA 自动发送数据

### 30.7.3 UR3T\_DMA 状态寄存器 (DMA\_UR3T\_STA)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2    | B1 | B0     |
|--------------|-------|----|----|----|----|----|-------|----|--------|
| DMA_UR3T_STA | FA52H | -  | -  | -  | -  | -  | TXOVW | -  | UR3TIF |

UR3TIF: UR3T\_DMA 中断请求标志位, 当 UR3T\_DMA 数据发送完成后, 硬件自动将 UR3TIF 置 1, 若使能 UR3T\_DMA 中断则进入中断服务程序。标志位需软件清零

TXOVW: UR3T\_DMA 数据覆盖标志位。UR3T\_DMA 正在数据传输过程中, 串口写 S3BUF 寄存器再次触发串口发送数据时, 会导致数据传输失败, 此时硬件自动将 TXOVW 置 1。标志位需软件清零



### 30.7.4 UR3T\_DMA 传输总字节寄存器 (DMA\_UR3T\_AMT)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR3T_AMT | FA53H |    |    |    |    |    |    |    |    |

DMA\_UR3T\_AMT: 设置需要自动发送数据的字节数。

注: 实际的字节数为 (DMA\_UR3T\_AMT+1), 即当 DMA\_UR3T\_AMT 设置为 0 时, 传输 1 字节, 当 DMA\_UR3T\_AMT 设置 255 时, 传输 256 字节

### 30.7.5 UR3T\_DMA 传输完成字节寄存器 (DMA\_UR3T\_DONE)

| 符号            | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR3T_DONE | FA54H |    |    |    |    |    |    |    |    |

DMA\_UR3T\_DONE: 当前已经发送完成的字节数。

### 30.7.6 UR3T\_DMA 发送地址寄存器 (DMA\_UR3T\_TXAx)

| 符号            | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|------------|----|----|----|----|----|----|----|
| DMA_UR3T_TXAH | FA55H | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_UR3T_TXAL | FA56H | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_UR3T\_TXA: 设置自动发送数据的源地址。执行 UR3T\_DMA 操作时会从这个地址开始读数据。

### 30.7.7 UR3R\_DMA 配置寄存器 (DMA\_UR3R\_CFG)

| 符号           | 地址    | B7     | B6 | B5 | B4 | B3          | B2 | B1           | B0 |
|--------------|-------|--------|----|----|----|-------------|----|--------------|----|
| DMA_UR3R_CFG | FA58H | UR3RIE | -  | -  | -  | UR3RIP[1:0] |    | UR3RPTY[1:0] |    |

UR3RIE: UR3R\_DMA 中断使能控制位

0: 禁止 UR3R\_DMA 中断

1: 允许 UR3R\_DMA 中断

UR3RIP[1:0]: UR3R\_DMA 中断优先级控制位

| UR3RIP[1:0] | 中断优先级   |
|-------------|---------|
| 00          | 最低级 (0) |
| 01          | 较低级 (1) |
| 10          | 较高级 (2) |
| 11          | 最高级 (3) |

UR3RPTY[1:0]: UR3R\_DMA 数据总线访问优先级控制位

| UR3RPTY [1:0] | 总线优先级   |
|---------------|---------|
| 00            | 最低级 (0) |
| 01            | 较低级 (1) |
| 10            | 较高级 (2) |
| 11            | 最高级 (3) |

### 30.7.8 UR3R\_DMA 控制寄存器 (DMA\_UR3R\_CR)

| 符号          | 地址    | B7     | B6 | B5   | B4 | B3 | B2 | B1 | B0      |
|-------------|-------|--------|----|------|----|----|----|----|---------|
| DMA_UR3R_CR | FA59H | ENUR3R | -  | TRIG | -  | -  | -  | -  | CLRFIFO |

ENUR3R: UR3R\_DMA 功能使能控制位

0: 禁止 UR3R\_DMA 功能

1: 允许 UR3R\_DMA 功能

TRIG: UR3R\_DMA 串口 1 接收触发控制位

0: 写 0 无效

1: 写 1 开始 UR3R\_DMA 自动接收数据

CLRFIFO: 清除 UR3R\_DMA 接收 FIFO 控制位

0: 写 0 无效

1: 开始 UR3R\_DMA 操作前, 先清空 UR3R\_DMA 内置的 FIFO

### 30.7.9 UR3R\_DMA 状态寄存器 (DMA\_UR3R\_STA)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1     | B0     |
|--------------|-------|----|----|----|----|----|----|--------|--------|
| DMA_UR3R_STA | FA5AH | -  | -  | -  | -  | -  | -  | RXLOSS | UR3RIF |

UR3RIF: UR3R\_DMA 中断请求标志位, 当 UR3R\_DMA 接收数据完成后, 硬件自动将 UR3RIF 置 1, 若使能 UR3R\_DMA 中断则进入中断服务程序。标志位需软件清零

RXLOSS: UR3R\_DMA 接收数据丢失标志位。UR3R\_DMA 操作过程中, 当 XRAM 总线过于繁忙, 来不及清空 UR3R\_DMA 的接收 FIFO 导致 UR3R\_DMA 接收的数据自动丢失时, 硬件自动将 RXLOSS 置 1。标志位需软件清零

### 30.7.10 UR3R\_DMA 传输总字节寄存器 (DMA\_UR3R\_AMT)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR3R_AMT | FA5BH |    |    |    |    |    |    |    |    |

DMA\_UR3R\_AMT: 设置需要自动接收数据的字节数。

注: 实际的字节数为 (DMA\_UR3R\_AMT+1), 即当 DMA\_UR3R\_AMT 设置为 0 时, 传输 1 字节, 当 DMA\_UR3R\_AMT 设置 255 时, 传输 256 字节

### 30.7.11 UR3R\_DMA 传输完成字节寄存器 (DMA\_UR3R\_DONE)

| 符号            | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR3R_DONE | FA5CH |    |    |    |    |    |    |    |    |

DMA\_UR3R\_DONE: 当前已经接收完成的字节数。

### 30.7.12 UR3R\_DMA 接收地址寄存器 (DMA\_UR3R\_RXAx)

| 符号            | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|------------|----|----|----|----|----|----|----|
| DMA_UR3R_RXAH | FA5DH | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_UR3R_RXAL | FA5EH | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_UR3R\_RXA: 设置自动接收数据的目标地址。执行 UR3R\_DMA 操作时会从这个地址开始写数据。

## 30.8 串口 4 与存储器之间的数据交换 (UR4T\_DMA, UR4R\_DMA)

### 30.8.1 UR4T\_DMA 配置寄存器 (DMA\_UR4T\_CFG)

| 符号           | 地址    | B7     | B6 | B5 | B4 | B3          | B2 | B1           | B0 |
|--------------|-------|--------|----|----|----|-------------|----|--------------|----|
| DMA_UR4T_CFG | FA50H | UR4TIE | -  | -  | -  | UR4TIP[1:0] |    | UR4TPTY[1:0] |    |

UR4TIE: UR4T\_DMA 中断使能控制位

0: 禁止 UR4T\_DMA 中断

1: 允许 UR4T\_DMA 中断

UR4TIP[1:0]: UR4T\_DMA 中断优先级控制位

| UR4TIP[1:0] | 中断优先级   |
|-------------|---------|
| 00          | 最低级 (0) |
| 01          | 较低级 (1) |
| 10          | 较高级 (2) |
| 11          | 最高级 (3) |

UR4TPTY[1:0]: UR4T\_DMA 数据总线访问优先级控制位

| UR4TPTY [1:0] | 总线优先级   |
|---------------|---------|
| 00            | 最低级 (0) |
| 01            | 较低级 (1) |
| 10            | 较高级 (2) |
| 11            | 最高级 (3) |

### 30.8.2 UR4T\_DMA 控制寄存器 (DMA\_UR4T\_CR)

| 符号          | 地址    | B7     | B6   | B5 | B4 | B3 | B2 | B1 | B0 |
|-------------|-------|--------|------|----|----|----|----|----|----|
| DMA_UR4T_CR | FA51H | ENUR4T | TRIG | -  | -  | -  | -  | -  | -  |

ENUR4T: UR4T\_DMA 功能使能控制位

0: 禁止 UR4T\_DMA 功能

1: 允许 UR4T\_DMA 功能

TRIG: UR4T\_DMA 串口 1 发送触发控制位

0: 写 0 无效

1: 写 1 开始 UR4T\_DMA 自动发送数据

### 30.8.3 UR4T\_DMA 状态寄存器 (DMA\_UR4T\_STA)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2    | B1 | B0     |
|--------------|-------|----|----|----|----|----|-------|----|--------|
| DMA_UR4T_STA | FA62H | -  | -  | -  | -  | -  | TXOVW | -  | UR4TIF |

UR4TIF: UR4T\_DMA 中断请求标志位, 当 UR4T\_DMA 数据发送完成后, 硬件自动将 UR4TIF 置 1, 若使能 UR4T\_DMA 中断则进入中断服务程序。标志位需软件清零

TXOVW: UR4T\_DMA 数据覆盖标志位。UR4T\_DMA 正在数据传输过程中, 串口写 S4BUF 寄存器再次触发串口发送数据时, 会导致数据传输失败, 此时硬件自动将 TXOVW 置 1。标志位需软件清零

### 30.8.4 UR4T\_DMA 传输总字节寄存器 (DMA\_UR4T\_AMT)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR4T_AMT | FA53H |    |    |    |    |    |    |    |    |

DMA\_UR4T\_AMT: 设置需要自动发送数据的字节数。

注: 实际的字节数为 (DMA\_UR4T\_AMT+1), 即当 DMA\_UR4T\_AMT 设置为 0 时, 传输 1 字节, 当 DMA\_UR4T\_AMT 设置 255 时, 传输 256 字节

### 30.8.5 UR4T\_DMA 传输完成字节寄存器 (DMA\_UR4T\_DONE)

| 符号            | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR4T_DONE | FA54H |    |    |    |    |    |    |    |    |

DMA\_UR4T\_DONE: 当前已经发送完成的字节数。

### 30.8.6 UR4T\_DMA 发送地址寄存器 (DMA\_UR4T\_TXAx)

| 符号            | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|------------|----|----|----|----|----|----|----|
| DMA_UR4T_TXAH | FA55H | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_UR4T_TXAL | FA56H | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_UR4T\_TXA: 设置自动发送数据的源地址。执行 UR4T\_DMA 操作时会从这个地址开始读数据。

### 30.8.7 UR4R\_DMA 配置寄存器 (DMA\_UR4R\_CFG)

| 符号           | 地址    | B7     | B6 | B5 | B4 | B3          | B2 | B1           | B0 |
|--------------|-------|--------|----|----|----|-------------|----|--------------|----|
| DMA_UR4R_CFG | FA58H | UR4RIE | -  | -  | -  | UR4RIP[1:0] |    | UR4RPTY[1:0] |    |

UR4RIE: UR4R\_DMA 中断使能控制位

0: 禁止 UR4R\_DMA 中断

1: 允许 UR4R\_DMA 中断

UR4RIP[1:0]: UR4R\_DMA 中断优先级控制位

| UR4RIP[1:0] | 中断优先级   |
|-------------|---------|
| 00          | 最低级 (0) |
| 01          | 较低级 (1) |
| 10          | 较高级 (2) |
| 11          | 最高级 (3) |

UR4RPTY[1:0]: UR4R\_DMA 数据总线访问优先级控制位

| UR4RPTY [1:0] | 总线优先级   |
|---------------|---------|
| 00            | 最低级 (0) |
| 01            | 较低级 (1) |
| 10            | 较高级 (2) |
| 11            | 最高级 (3) |

### 30.8.8 UR4R\_DMA 控制寄存器 (DMA\_UR4R\_CR)

| 符号          | 地址    | B7     | B6 | B5   | B4 | B3 | B2 | B1 | B0      |
|-------------|-------|--------|----|------|----|----|----|----|---------|
| DMA_UR4R_CR | FA59H | ENUR4R | -  | TRIG | -  | -  | -  | -  | CLRFIFO |

ENUR4R: UR4R\_DMA 功能使能控制位

0: 禁止 UR4R\_DMA 功能

1: 允许 UR4R\_DMA 功能

TRIG: UR4R\_DMA 串口 1 接收触发控制位

0: 写 0 无效

1: 写 1 开始 UR4R\_DMA 自动接收数据

CLRFIFO: 清除 UR4R\_DMA 接收 FIFO 控制位

0: 写 0 无效

1: 开始 UR4R\_DMA 操作前, 先清空 UR4R\_DMA 内置的 FIFO

### 30.8.9 UR4R\_DMA 状态寄存器 (DMA\_UR4R\_STA)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1     | B0     |
|--------------|-------|----|----|----|----|----|----|--------|--------|
| DMA_UR4R_STA | FA5AH | -  | -  | -  | -  | -  | -  | RXLOSS | UR4RIF |

UR4RIF: UR4R\_DMA 中断请求标志位, 当 UR4R\_DMA 接收数据完成后, 硬件自动将 UR4RIF 置 1, 若使能 UR4R\_DMA 中断则进入中断服务程序。标志位需软件清零

RXLOSS: UR4R\_DMA 接收数据丢失标志位。UR4R\_DMA 操作过程中, 当 XRAM 总线过于繁忙, 来不及清空 UR4R\_DMA 的接收 FIFO 导致 UR4R\_DMA 接收的数据自动丢失时, 硬件自动将 RXLOSS 置 1。标志位需软件清零

### 30.8.10 UR4R\_DMA 传输总字节寄存器 (DMA\_UR4R\_AMT)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR4R_AMT | FA5BH |    |    |    |    |    |    |    |    |

DMA\_UR4R\_AMT: 设置需要自动接收数据的字节数。

注: 实际的字节数为 (DMA\_UR4R\_AMT+1), 即当 DMA\_UR4R\_AMT 设置为 0 时, 传输 1 字节, 当 DMA\_UR4R\_AMT 设置 255 时, 传输 256 字节

### 30.8.11 UR4R\_DMA 传输完成字节寄存器 (DMA\_UR4R\_DONE)

| 符号            | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|----|----|----|----|----|----|----|----|
| DMA_UR4R_DONE | FA5CH |    |    |    |    |    |    |    |    |

DMA\_UR4R\_DONE: 当前已经接收完成的字节数。

### 30.8.12 UR4R\_DMA 接收地址寄存器 (DMA\_UR4R\_RXAx)

| 符号            | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---------------|-------|------------|----|----|----|----|----|----|----|
| DMA_UR4R_RXAH | FA5DH | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_UR4R_RXAL | FA5EH | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_UR4R\_RXA: 设置自动接收数据的目标地址。执行 UR4R\_DMA 操作时会从这个地址开始写数据。

## 30.9 LCM 与存储器之间的数据读写 (LCM\_DMA)

### 30.9.1 LCM\_DMA 配置寄存器 (DMA\_LCM\_CFG)

| 符号          | 地址    | B7    | B6 | B5 | B4 | B3         | B2 | B1         | B0 |
|-------------|-------|-------|----|----|----|------------|----|------------|----|
| DMA_LCM_CFG | FA70H | LCMIE | -  | -  | -  | LCMIP[1:0] |    | LCMPY[1:0] |    |

LCMIE: LCM\_DMA 中断使能控制位

0: 禁止 LCM\_DMA 中断

1: 允许 LCM\_DMA 中断

LCMIP[1:0]: LCM\_DMA 中断优先级控制位

| LCMIP[1:0] | 中断优先级   |
|------------|---------|
| 00         | 最低级 (0) |
| 01         | 较低级 (1) |
| 10         | 较高级 (2) |
| 11         | 最高级 (3) |

LCMPY[1:0]: LCM\_DMA 数据总线访问优先级控制位

| LCMPY [1:0] | 总线优先级   |
|-------------|---------|
| 00          | 最低级 (0) |
| 01          | 较低级 (1) |
| 10          | 较高级 (2) |
| 11          | 最高级 (3) |

### 30.9.2 LCM\_DMA 控制寄存器 (DMA\_LCM\_CR)

| 符号         | 地址    | B7    | B6     | B5     | B4     | B3     | B2 | B1 | B0     |
|------------|-------|-------|--------|--------|--------|--------|----|----|--------|
| DMA_LCM_CR | FA71H | ENLCM | TRIGWC | TRIGWD | TRIGRC | TRIGRD | -  | -  | CLRIFO |

ENLCM: LCM\_DMA 功能使能控制位

0: 禁止 LCM\_DMA 功能

1: 允许 LCM\_DMA 功能

TRIGWC: LCM\_DMA 发送命令模式触发控制位

0: 写 0 无效

1: 写 1 开始 LCM\_DMA 发送命令模式操作

TRIGWD: LCM\_DMA 发送数据模式触发控制位

0: 写 0 无效

1: 写 1 开始 LCM\_DMA 发送数据模式操作

TRIGRC: LCM\_DMA 读取命令模式触发控制位

0: 写 0 无效

1: 写 1 开始 LCM\_DMA 读取命令模式操作

TRIGRD: LCM\_DMA 读取数据模式触发控制位

0: 写 0 无效

1: 写 1 开始 LCM\_DMA 读取数据模式操作

### 30.9.3 LCM\_DMA 状态寄存器 (DMA\_LCM\_STA)

| 符号          | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1    | B0    |
|-------------|-------|----|----|----|----|----|----|-------|-------|
| DMA_LCM_STA | FA72H | -  | -  | -  | -  | -  | -  | TXOVW | LCMIF |

LCMIF: LCM\_DMA 中断请求标志位, 当 LCM\_DMA 数据交换完成后, 硬件自动将 LCMIF 置 1, 若使能 LCM\_DMA 中断则进入中断服务程序。标志位需软件清零

TXOVW: LCM\_DMA 数据覆盖标志位。LCM\_DMA 正在数据传输过程中, LCMIF 写 LCMIFDATL 和 LCMIDDATH 寄存器时, 会导致数据传输失败, 此时硬件自动将 TXOVW 置 1。标志位需软件清零

### 30.9.4 LCM\_DMA 传输总字节寄存器 (DMA\_LCM\_AMT)

| 符号          | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-------------|-------|----|----|----|----|----|----|----|----|
| DMA_LCM_AMT | FA73H |    |    |    |    |    |    |    |    |

DMA\_LCM\_AMT: 设置需要进行数据读写的字节数。

注: 实际读写的字节数为 (DMA\_LCM\_AMT+1), 即当 DMA\_LCM\_AMT 设置为 0 时, 传输 1 字节, 当 DMA\_LCM\_AMT 设置 255 时, 传输 256 字节

### 30.9.5 LCM\_DMA 传输完成字节寄存器 (DMA\_LCM\_DONE)

| 符号           | 地址    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|----|----|----|----|----|----|----|----|
| DMA_LCM_DONE | FA74H |    |    |    |    |    |    |    |    |

DMA\_LCM\_DONE: 当前已经传输完成的字节数。

### 30.9.6 LCM\_DMA 发送地址寄存器 (DMA\_LCM\_TXAx)

| 符号           | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|------------|----|----|----|----|----|----|----|
| DMA_LCM_TXAH | FA75H | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_LCM_TXAL | FA76H | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_LCM\_TXA: 设置进行数据传输时的源地址。执行 LCM\_DMA 操作时会从这个地址开始读数据。

### 30.9.7 LCM\_DMA 接收地址寄存器 (DMA\_LCM\_RXAx)

| 符号           | 地址    | B7         | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|--------------|-------|------------|----|----|----|----|----|----|----|
| DMA_LCM_RXAH | FA77H | ADDR[15:8] |    |    |    |    |    |    |    |
| DMA_LCM_RXAL | FA78H | ADDR[7:0]  |    |    |    |    |    |    |    |

DMA\_LCM\_RXA: 设置进行数据传输时的目标地址。执行 LCM\_DMA 操作时会从这个地址开始写入数据。



## 30.10 范例程序

### 30.10.1 串口 1 中断模式与电脑收发测试 - DMA 接收超时中断

#### C 语言代码

---

//测试工作频率为 22.1184MHz

/\*\*\*\*\*\* 功能说明 \*\*\*\*\*

串口 1 全双工中断方式收发通讯程序。通过 PC 向 MCU 发送数据,MCU 将收到的数据自动存入 DMA 空间。当一次性接收的内容存满设置的 DMA 空间后,通过串口 1 的 DMA 自动发送功能把存储空间的数据输出。利用串口接收中断进行超时判断,超时没有收到新的数据,表示一串数据已经接收完毕,将已接收的内容输出,并清除 DMA 空间。用定时器做波特率发生器,建议使用 1T 模式(除非低波特率用 12T),并选择可被波特率整除的时钟频率,以提高精度。

下载时,选择时钟 22.1184MHz (用户可自行修改频率)。

\*\*\*\*\*/

#include "stdio.h"

#include "stc8h.h"

#define MAIN\_Fosc 22118400L

//定义主时钟(精确计算 115200 波特率)

#define Baudrate1 115200L

#define Timer0\_Reload (65536UL-(MAIN\_Fosc / 1000))

#define DMA\_AMT\_LEN 255

//设置传输总字节数(0~255) : DMA\_AMT\_LEN+1

bit B\_1ms;

//1ms 标志

bit DMATxFlag;

bit DMARxFlag;

bit BusyFlag;

u8 Rx\_cnt;

u8 RX1\_TimeOut;

u8 xdata DMABuffer[256];

void UART1\_config(u8 brt);

void DMA\_Config(void);

void UartPutc(unsigned char dat)

{

    BusyFlag = 1;

    SBUF = dat;

    while(BusyFlag);

}

char putchar(char c)

{

    UartPutc(c);

    return c;

}

void main(void)

{

    u16 i;



```

P_SW2 |= 0x80;    //使能 XFR 访问

P0M1 = 0x00;    P0M0 = 0x00;    //设置为双向口
P1M1 = 0x00;    P1M0 = 0x00;    //设置为双向口
P2M1 = 0x00;    P2M0 = 0x00;    //设置为双向口
P3M1 = 0x00;    P3M0 = 0x00;    //设置为双向口
P4M1 = 0x00;    P4M0 = 0x00;    //设置为双向口
P5M1 = 0x00;    P5M0 = 0x00;    //设置为双向口
P6M1 = 0x00;    P6M0 = 0x00;    //设置为双向口
P7M1 = 0x00;    P7M0 = 0x00;    //设置为双向口

for(i=0; i<256; i++)
{
    DMABuffer[i] = i;
}

AUXR = 0x80;    //Timer0 set as 1T, 16 bits timer auto-reload,
TH0 = (u8)(Timer0_Reload / 256);
TL0 = (u8)(Timer0_Reload % 256);
ET0 = 1;    //Timer0 interrupt enable
TR0 = 1;    //Timer0 run

UART1_config(1);    //使用Timer1 做波特率
DMA_Config();
EA = 1;    //允许总中断

printf("UART1 DMA Timeout Programme!\r\n");    //UART1 发送一个字符串
DMATxFlag = 0;
DMARxFlag = 0;

while (1)
{
    if((DMATxFlag) && (DMARxFlag))    //判断发送完成标志与接收完成标志
    {
        Rx_cnt = 0;
        RX1_TimeOut = 0;
        printf("\r\nUART1 DMA FULL!\r\n");    //UART1 发送一个字符串
        DMATxFlag = 0;
        DMA_UR1T_CR = 0xc0;    //bit7 1: 使能 UART1_DMA,
        //bit6 1: 开始 UART1_DMA 自动发送

        DMARxFlag = 0;
        DMA_UR1R_CR = 0xa1;    //bit7 1: 使能 UART1_DMA,
        //bit5 1: 开始 UART1_DMA 自动接收,
        //bit0 1: 清除 FIFO
    }

    if(B_1ms)    //1ms 到
    {
        B_1ms = 0;
        if(RX1_TimeOut > 0)    //超时计数
        {
            if(--RX1_TimeOut == 0)
            {
                DMA_UR1R_CR = 0x00;    //关闭 UART1_DMA
                printf("\r\nUART1 Timeout!\r\n");    //UART1 发送一个字符串

                for(i=0; i<Rx_cnt; i++) UartPutc(DMABuffer[i]);
                printf("\r\n");
            }
        }
    }
}

```

```

        Rx_cnt = 0;
        DMA_URIR_CR = 0xa1;           //bit7 1: 使能 UART1_DMA,
                                      //bit5 1: 开始 UART1_DMA 自动接收,
                                      //bit0 1: 清除 FIFO
    }
    }
}

void DMA_Config(void)
{
    DMA_URIT_CFG = 0x80;               //bit7 1: Enable Interrupt
    DMA_URIT_STA = 0x00;
    DMA_URIT_AMT = DMA_AMT_LEN;        //设置传输总字节数: n+1
    DMA_URIT_TXA = DMABuffer;
    DMA_URIT_CR = 0xc0;               //bit7 1: 使能 UART1_DMA,
                                      //bit6 1: 开始 UART1_DMA 自动发送

    DMA_URIR_CFG = 0x80;               //bit7 1: Enable Interrupt
    DMA_URIR_STA = 0x00;
    DMA_URIR_AMT = DMA_AMT_LEN;        //设置传输总字节数: n+1
    DMA_URIR_RXA = DMABuffer;
    DMA_URIR_CR = 0xa1;               //bit7 1: 使能 UART1_DMA,
                                      //bit5 1: 开始 UART1_DMA 自动接收,
                                      //bit0 1: 清除 FIFO
}

void SetTimer2Baudraye(u16 dat)
{
    AUXR &= ~(1<<4);                 //Timer stop
    AUXR &= ~(1<<3);                 //Timer2 set As Timer
    AUXR /= (1<<2);                  //Timer2 set as 1T mode
    T2H = dat / 256;
    T2L = dat % 256;
    IE2 &= ~(1<<2);
    AUXR /= (1<<4);
}

void UART1_config(u8 brt)
    //选择波特率:
    //2: 使用 Timer2 做波特率,
    //其它值: 使用 Timer1 做波特率

{
    /***** 波特率使用定时器2 *****/
    if(brt == 2)
    {
        AUXR /= 0x01;                //S1 BRT Use Timer2;
        SetTimer2Baudraye(65536UL - (MAIN_Fosc / 4) / Baudrate1);
    }

    /***** 波特率使用定时器1 *****/
    else
    {
        TR1 = 0;
        AUXR &= ~0x01;                //S1 BRT Use Timer1;
        AUXR /= (1<<6);                //Timer1 set as 1T mode
        TMOD &= ~(1<<6);               //Timer1 set As Timer
        TMOD &= ~0x30;                //Timer1_16bitAutoReload;
        TH1 = (u8)((65536UL - (MAIN_Fosc / 4) / Baudrate1) / 256);
    }
}

```

```

    TL1 = (u8)((65536UL - (MAIN_Fosc / 4) / Baudrate1) % 256);
    ET1 = 0;                                     //禁止中断
    INTCLKO &= ~0x02;                             //不输出时钟
    TRI = 1;
}
/*****/

SCON = (SCON & 0x3f) / 0x40;                    //UART1 模式:
                                                //0x00: 同步移位输出,
                                                //0x40: 8 位数据,可变波特率,
                                                //0x80: 9 位数据,固定波特率,
                                                //0xc0: 9 位数据,可变波特率
// PS = 1;                                     //高优先级中断
ES = 1;                                         //允许中断
REN = 1;                                       //允许接收
P_SW1 &= 0x3f;
P_SW1 /= 0x00;                                //UART1 switch to:
                                                //0x00: P3.0 P3.1,
                                                //0x40: P3.6 P3.7,
                                                //0x80: P1.6 P1.7,
                                                //0xc0: P4.3 P4.4

RX1_TimeOut = 0;
}

void UART1_int (void) interrupt 4
{
    if(RI)
    {
        RI = 0;
        Rx_cnt++;
        if(Rx_cnt >= DMA_AMT_LEN) Rx_cnt = 0;
        RX1_TimeOut = 5;                       //如果5ms 没收到新的数据, 判定一串数据接收完毕
    }

    if(TI)
    {
        TI = 0;
        BusyFlag = 0;
    }
}

void timer0 (void) interrupt 1
{
    B_1ms = 1;                                //1ms 标志
}

void UART1_DMA_Interrup(void) interrupt 13
{
    if (DMA_URIT_STA & 0x01)                    //发送完成
    {
        DMA_URIT_STA &= ~0x01;
        DMATxFlag = 1;
    }
    if (DMA_URIT_STA & 0x04)                    //数据覆盖
    {
        DMA_URIT_STA &= ~0x04;
    }
}

```

```

if (DMA_URIR_STA & 0x01)                //接收完成
{
    DMA_URIR_STA &= ~0x01;
    DMARxFlag = 1;
}
if (DMA_URIR_STA & 0x02)                //数据丢弃
{
    DMA_URIR_STA &= ~0x02;
}
}

```

//文件: ISR.ASM

//中断号大于 31 的中断, 需要进行中断入口地址重映射处理

```

CSEG AT 012BH                          ;P0INT_VECTOR
JMP P0INT_ISR
CSEG AT 0133H                          ;P1INT_VECTOR
JMP P1INT_ISR
CSEG AT 013BH                          ;P2INT_VECTOR
JMP P2INT_ISR
CSEG AT 0143H                          ;P3INT_VECTOR
JMP P3INT_ISR
CSEG AT 014BH                          ;P4INT_VECTOR
JMP P4INT_ISR
CSEG AT 0153H                          ;P5INT_VECTOR
JMP P5INT_ISR
CSEG AT 015BH                          ;P6INT_VECTOR
JMP P6INT_ISR
CSEG AT 0163H                          ;P7INT_VECTOR
JMP P7INT_ISR
CSEG AT 016BH                          ;P8INT_VECTOR
JMP P8INT_ISR
CSEG AT 0173H                          ;P9INT_VECTOR
JMP P9INT_ISR
CSEG AT 017BH                          ;M2MDMA_VECTOR
JMP M2MDMA_ISR
CSEG AT 0183H                          ;ADCDMA_VECTOR
JMP ADCDMA_ISR
CSEG AT 018BH                          ;SPIDMA_VECTOR
JMP SPIDMA_ISR
CSEG AT 0193H                          ;UITXDMA_VECTOR
JMP UITXDMA_ISR
CSEG AT 019BH                          ;UIRXDMA_VECTOR
JMP UIRXDMA_ISR
CSEG AT 01A3H                          ;U2TXDMA_VECTOR
JMP U2TXDMA_ISR
CSEG AT 01ABH                          ;U2RXDMA_VECTOR
JMP U2RXDMA_ISR
CSEG AT 01B3H                          ;U3TXDMA_VECTOR
JMP U3TXDMA_ISR
CSEG AT 01BBH                          ;U3RXDMA_VECTOR
JMP U3RXDMA_ISR
CSEG AT 01C3H                          ;U4TXDMA_VECTOR
JMP U4TXDMA_ISR
CSEG AT 01CBH                          ;U4RXDMA_VECTOR
JMP U4RXDMA_ISR
CSEG AT 01D3H                          ;LCMDMA_VECTOR
JMP LCMDMA_ISR
CSEG AT 01DBH                          ;LCMIF_VECTOR

```

*JMP**LCMIF\_ISR**P0INT\_ISR:**P1INT\_ISR:**P2INT\_ISR:**P3INT\_ISR:**P4INT\_ISR:**P5INT\_ISR:**P6INT\_ISR:**P7INT\_ISR:**P8INT\_ISR:**P9INT\_ISR:**M2MDMA\_ISR:**ADC DMA\_ISR:**SPIDMA\_ISR:**U1TXDMA\_ISR:**U1RXDMA\_ISR:**U2TXDMA\_ISR:**U2RXDMA\_ISR:**U3TXDMA\_ISR:**U3RXDMA\_ISR:**U4TXDMA\_ISR:**U4RXDMA\_ISR:**LCMDMA\_ISR:**LCMIF\_ISR:**JMP**006BH**END*

### 30.10.2 串口 1 中断模式与电脑收发测试 - DMA 数据校验

#### C 语言代码

---

//测试工作频率为 22.1184MHz

```

/***** 功能说明 *****/

```

串口 1 全双工中断方式收发通讯程序。通过 PC 向 MCU 发送数据, MCU 将收到的数据自动存入 DMA 空间。数据包的最后两个字节作为校验位, 例程以 `crc16_ccitt` 算法进行校验。当 DMA 空间存满设置大小的内容后, 对有效数据进行校验计算, 然后与最后两位校验位进行对比。通过串口 1 的 DMA 自动发送功能把存储空间的数据输出。用定时器做波特率发生器, 建议使用 1T 模式(除非低波特率用 12T), 并选择可被波特率整除的时钟频率, 以提高精度。

下载时, 选择时钟 22.1184MHz (用户可自行修改频率)。

```

*****/

```

```

#include "stdio.h"

```

```

#include "STC8h.h"

```

```

#include "crc16.h"

```

```

#define MAIN_Fosc      22118400L

```

```

//定义主时钟(精确计算 115200 波特率)

```

```

#define Baudrate1      115200L

```

```

#define DMA_AMT_LEN    255

```

```

//设置传输总字节数(0~255) : DMA_AMT_LEN+1

```

```
bit    DMATxFlag;
```

```
bit    DMARxFlag;
```

```
u8      xdata DMABuffer[256];
```

```
void UART1_config(u8 brt);
```

```
void DMA_Config(void);
```

```
void UartPutc(unsigned char dat)
```

```
{
```

```
    SBUF = dat;
```

```
    while(TI == 0);
```

```
    TI = 0;
```

```
}
```

```
char putchar(char c)
```

```
{
```

```
    UartPutc(c);
```

```
    return c;
```

```
}
```

```
/*****CRC 计算函数*****/
```

```
u16 crc16_ccitt(u8 *pbuf, u16 len)
```

```
{
```

```
    unsigned short code crc16_ccitt_table[256] =
```

```
{
```

```
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,
    0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6,
    0x9339, 0x8318, 0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,
    0x2462, 0x3443, 0x0420, 0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485,
    0xA56A, 0xB54B, 0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,
    0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,
    0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823,
    0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,
    0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12,
    0xDBFD, 0xCBDC, 0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,
    0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41,
    0xEDAE, 0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,
    0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51, 0x0E70,
    0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A, 0x9F59, 0x8F78,
    0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D, 0xF14E, 0xE16F,
    0x1080, 0x00A1, 0x30C2, 0x20E3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E,
    0x02B1, 0x1290, 0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D,
    0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E, 0xC71D, 0xD73C,
    0x26D3, 0x36F2, 0x0691, 0x16B0, 0x6657, 0x7676, 0x4615, 0x5634,
    0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9, 0xB98A, 0xA9AB,
    0x5844, 0x4865, 0x7806, 0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3,
    0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A,
    0x4A75, 0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,
    0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8, 0x8DC9,
    0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83, 0x1CE0, 0x0CC1,
    0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA, 0x8FD9, 0x9FF8,
    0x6E17, 0x7E36, 0x4E55, 0x5E74, 0x2E93, 0x3EB2, 0x0ED1, 0x1EF0
```

```

};

u16 crc16 = 0x0000;
u16 crc_h8, crc_l8;

while( len-- ) {
    crc_h8 = (crc16 >> 8);
    crc_l8 = (crc16 << 8);
    crc16 = crc_l8 ^ crc16_ccitt_table[crc_h8 ^ *pbuf];
    pbuf++;
}

return crc16;
}

void main(void)
{
    u16 i;
    u16 CheckSum;

    P_SW2 /= 0x80; //使能XFR 访问

    P0M1 = 0x00;    P0M0 = 0x00; //设置为准双向口
    P1M1 = 0x00;    P1M0 = 0x00; //设置为准双向口
    P2M1 = 0x00;    P2M0 = 0x00; //设置为准双向口
    P3M1 = 0x00;    P3M0 = 0x00; //设置为准双向口
    P4M1 = 0x00;    P4M0 = 0x00; //设置为准双向口
    P5M1 = 0x00;    P5M0 = 0x00; //设置为准双向口
    P6M1 = 0x00;    P6M0 = 0x00; //设置为准双向口
    P7M1 = 0x00;    P7M0 = 0x00; //设置为准双向口

    for(i=0; i<256; i++)
    {
        DMABuffer[i] = i;
    }

    DMA_URIT_STA = 0x00;
    UART1_config(1);
    printf("UART1 DMA CRC Programme!\r\n");

    DMA_Config();
    EA = 1; //允许总中断

    DMATxFlag = 0;
    DMARxFlag = 0;

    while (1)
    {
        if((DMATxFlag) && (DMARxFlag))
        {
            CheckSum = crc16_ccitt(DMABuffer,DMA_AMT_LEN-1);
            if(((u8)CheckSum == DMABuffer[DMA_AMT_LEN-1]) &&
            ((u8)(CheckSum>>8) == DMABuffer[DMA_AMT_LEN]))
            {
                printf("\r\nOK! CheckSum = %04x\r\n",CheckSum);
            }
            else
            {
                printf("\r\nERROR! CheckSum = %04x\r\n",CheckSum);
            }
        }
    }
}

```

```

    }
    DMATxFlag = 0;
    DMA_URIT_CR = 0xc0; //bit7 1: 使能 UART1_DMA,
                        //bit6 1: 开始 UART1_DMA 自动发送

    DMARxFlag = 0;
    DMA_URIR_CR = 0xa1; //bit7 1: 使能 UART1_DMA,
                        //bit5 1: 开始 UART1_DMA 自动接收,
                        //bit0 1: 清除 FIFO
    }
}

void DMA_Config(void)
{
    DMA_URIT_CFG = 0x80; //bit7 1: Enable Interrupt
    DMA_URIT_STA = 0x00;
    DMA_URIT_AMT = DMA_AMT_LEN; //设置传输总字节数: n+1
    DMA_URIT_TXA = DMABuffer;
    DMA_URIT_CR = 0xc0; //bit7 1: 使能 UART1_DMA,
                        //bit6 1: 开始 UART1_DMA 自动发送

    DMA_URIR_CFG = 0x80; //bit7 1: Enable Interrupt
    DMA_URIR_STA = 0x00;
    DMA_URIR_AMT = DMA_AMT_LEN; //设置传输总字节数: n+1
    DMA_URIR_RXA = DMABuffer;
    DMA_URIR_CR = 0xa1; //bit7 1: 使能 UART1_DMA,
                        //bit5 1: 开始 UART1_DMA 自动接收, bit0 1: 清除 FIFO
}

void SetTimer2Baudrate(u16 dat) //选择波特率:
//2: 使用 Timer2 做波特率,
//其它值: 使用 Timer1 做波特率
{
    AUXR &= ~(1<<4); //Timer stop
    AUXR &= ~(1<<3); //Timer2 set As Timer
    AUXR |= (1<<2); //Timer2 set as 1T mode
    T2H = dat / 256;
    T2L = dat % 256;
    IE2 &= ~(1<<2); //禁止中断
    AUXR |= (1<<4); //Timer run enable
}

void UART1_config(u8 brt) //选择波特率:
//2: 使用 Timer2 做波特率,
//其它值: 使用 Timer1 做波特率
{
    /***** 波特率使用定时器2 *****/
    if(brt == 2)
    {
        AUXR |= 0x01; //S1 BRT Use Timer2;
        SetTimer2Baudrate(65536UL - (MAIN_Fosc / 4) / Baudrate1);
    }

    /***** 波特率使用定时器1 *****/
    else
    {
        TRI = 0;
        AUXR &= ~0x01; //S1 BRT Use Timer1;
        AUXR |= (1<<6); //Timer1 set as 1T mode
    }
}

```



```

    TMOD &= ~(1<<6);                //Timer1 set As Timer
    TMOD &= ~0x30;                  //Timer1_16bitAutoReload;
    TH1 = (u8)((65536UL - (MAIN_Fosc / 4) / Baudrate1) / 256);
    TL1 = (u8)((65536UL - (MAIN_Fosc / 4) / Baudrate1) % 256);
    ET1 = 0;                        //禁止中断
    INTCLKO &= ~0x02;              //不输出时钟
    TRI = I;
}
/*****/

SCON = (SCON & 0x3f) / 0x40;      //UART1 模式
//0x00: 同步移位输出,
//0x40: 8 位数据,可变波特率,
//0x80: 9 位数据,固定波特率,
//0xc0: 9 位数据,可变波特率
// PS = I;                        //高优先级中断
// ES = I;                        //允许中断
REN = I;                          //允许接收
P_SW1 &= 0x3f;
P_SW1 /= 0x00;
}

void UART1_DMA_Interrup(void) interrupt 13
{
    if (DMA_URIT_STA & 0x01)        //发送完成
    {
        DMA_URIT_STA &= ~0x01;
        DMATxFlag = 1;
    }
    if (DMA_URIT_STA & 0x04)        //数据覆盖
    {
        DMA_URIT_STA &= ~0x04;
    }

    if (DMA_URIR_STA & 0x01)        //接收完成
    {
        DMA_URIR_STA &= ~0x01;
        DMARxFlag = 1;
    }
    if (DMA_URIR_STA & 0x02)        //数据丢弃
    {
        DMA_URIR_STA &= ~0x02;
    }
}

```

//文件: ISR.ASM

//中断号大于 31 的中断, 需要进行中断入口地址重映射处理

```

CSEG AT 012BH                      ;P0INT_VECTOR
JMP P0INT_ISR
CSEG AT 0133H                      ;P1INT_VECTOR
JMP P1INT_ISR
CSEG AT 013BH                      ;P2INT_VECTOR
JMP P2INT_ISR
CSEG AT 0143H                      ;P3INT_VECTOR
JMP P3INT_ISR
CSEG AT 014BH                      ;P4INT_VECTOR
JMP P4INT_ISR
CSEG AT 0153H                      ;P5INT_VECTOR

```

|             |           |                    |                        |
|-------------|-----------|--------------------|------------------------|
| <i>JMP</i>  |           | <i>P5INT_ISR</i>   |                        |
| <i>CSEG</i> | <i>AT</i> | <i>015BH</i>       | <i>;P6INT_VECTOR</i>   |
| <i>JMP</i>  |           | <i>P6INT_ISR</i>   |                        |
| <i>CSEG</i> | <i>AT</i> | <i>0163H</i>       | <i>;P7INT_VECTOR</i>   |
| <i>JMP</i>  |           | <i>P7INT_ISR</i>   |                        |
| <i>CSEG</i> | <i>AT</i> | <i>016BH</i>       | <i>;P8INT_VECTOR</i>   |
| <i>JMP</i>  |           | <i>P8INT_ISR</i>   |                        |
| <i>CSEG</i> | <i>AT</i> | <i>0173H</i>       | <i>;P9INT_VECTOR</i>   |
| <i>JMP</i>  |           | <i>P9INT_ISR</i>   |                        |
| <i>CSEG</i> | <i>AT</i> | <i>017BH</i>       | <i>;M2MDMA_VECTOR</i>  |
| <i>JMP</i>  |           | <i>M2MDMA_ISR</i>  |                        |
| <i>CSEG</i> | <i>AT</i> | <i>0183H</i>       | <i>;ADCDMA_VECTOR</i>  |
| <i>JMP</i>  |           | <i>ADCDMA_ISR</i>  |                        |
| <i>CSEG</i> | <i>AT</i> | <i>018BH</i>       | <i>;SPIDMA_VECTOR</i>  |
| <i>JMP</i>  |           | <i>SPIDMA_ISR</i>  |                        |
| <i>CSEG</i> | <i>AT</i> | <i>0193H</i>       | <i>;UITXDMA_VECTOR</i> |
| <i>JMP</i>  |           | <i>UITXDMA_ISR</i> |                        |
| <i>CSEG</i> | <i>AT</i> | <i>019BH</i>       | <i>;UIRXDMA_VECTOR</i> |
| <i>JMP</i>  |           | <i>UIRXDMA_ISR</i> |                        |
| <i>CSEG</i> | <i>AT</i> | <i>01A3H</i>       | <i>;U2TXDMA_VECTOR</i> |
| <i>JMP</i>  |           | <i>U2TXDMA_ISR</i> |                        |
| <i>CSEG</i> | <i>AT</i> | <i>01ABH</i>       | <i>;U2RXDMA_VECTOR</i> |
| <i>JMP</i>  |           | <i>U2RXDMA_ISR</i> |                        |
| <i>CSEG</i> | <i>AT</i> | <i>01B3H</i>       | <i>;U3TXDMA_VECTOR</i> |
| <i>JMP</i>  |           | <i>U3TXDMA_ISR</i> |                        |
| <i>CSEG</i> | <i>AT</i> | <i>01BBH</i>       | <i>;U3RXDMA_VECTOR</i> |
| <i>JMP</i>  |           | <i>U3RXDMA_ISR</i> |                        |
| <i>CSEG</i> | <i>AT</i> | <i>01C3H</i>       | <i>;U4TXDMA_VECTOR</i> |
| <i>JMP</i>  |           | <i>U4TXDMA_ISR</i> |                        |
| <i>CSEG</i> | <i>AT</i> | <i>01CBH</i>       | <i>;U4RXDMA_VECTOR</i> |
| <i>JMP</i>  |           | <i>U4RXDMA_ISR</i> |                        |
| <i>CSEG</i> | <i>AT</i> | <i>01D3H</i>       | <i>;LCMDMA_VECTOR</i>  |
| <i>JMP</i>  |           | <i>LCMDMA_ISR</i>  |                        |
| <i>CSEG</i> | <i>AT</i> | <i>01DBH</i>       | <i>;LCMIF_VECTOR</i>   |
| <i>JMP</i>  |           | <i>LCMIF_ISR</i>   |                        |

*P0INT\_ISR:**P1INT\_ISR:**P2INT\_ISR:**P3INT\_ISR:**P4INT\_ISR:**P5INT\_ISR:**P6INT\_ISR:**P7INT\_ISR:**P8INT\_ISR:**P9INT\_ISR:**M2MDMA\_ISR:**ADCDMA\_ISR:**SPIDMA\_ISR:**UITXDMA\_ISR:**UIRXDMA\_ISR:**U2TXDMA\_ISR:**U2RXDMA\_ISR:**U3TXDMA\_ISR:**U3RXDMA\_ISR:**U4TXDMA\_ISR:**U4RXDMA\_ISR:**LCMDMA\_ISR:**LCMIF\_ISR:*

JMP

006BH

END

## 代码测试方法

根据预定义的 DMA 数据包长度（例如：256 字节），通过串口工具发送一包数据（254 字节），并在最后加上 2 个字节的 CCITT-CRC16 校验码：



MCU 收到整包数据（256 字节）之后对前面 254 字节数据进行 CRC16 校验，得出来的校验码与最后两个字节进行比较，如果数值相等，打印“OK!”以及计算出来的校验码，然后输出 DMA 空间收取的内容。



如果校验码数值不相等，打印“ERROR!”以及计算出来的校验码。

# 31 增强型双数据指针

STC8H 系列的单片机内部集成了两组 16 位的数据指针。通过程序控制, 可实现数据指针自动递增或递减功能以及两组数据指针的自动切换功能

## 31.1 相关的特殊功能寄存器

| 符号   | 描述            | 地址  | 位地址与符号 |     |     |     |     |    |    |     | 复位值       |
|------|---------------|-----|--------|-----|-----|-----|-----|----|----|-----|-----------|
|      |               |     | B7     | B6  | B5  | B4  | B3  | B2 | B1 | B0  |           |
| DPL  | 数据指针 (低字节)    | 82H |        |     |     |     |     |    |    |     | 0000,0000 |
| DPH  | 数据指针 (高字节)    | 83H |        |     |     |     |     |    |    |     | 0000,0000 |
| DPL1 | 第二组数据指针 (低字节) | E4H |        |     |     |     |     |    |    |     | 0000,0000 |
| DPH1 | 第二组数据指针 (高字节) | E5H |        |     |     |     |     |    |    |     | 0000,0000 |
| DPS  | DPTR 指针选择器    | E3H | ID1    | ID0 | TSL | AU1 | AU0 | -  | -  | SEL | 0000,0xx0 |
| TA   | DPTR 时序控制寄存器  | AEH |        |     |     |     |     |    |    |     | 0000,0000 |

### 31.1.1 第 1 组 16 位数据指针寄存器 (DPTR0)

| 符号  | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----|-----|----|----|----|----|----|----|----|----|
| DPL | 82H |    |    |    |    |    |    |    |    |
| DPH | 83H |    |    |    |    |    |    |    |    |

DPL 为低 8 位数据 (低字节)

DPH 为高 8 位数据 (高字节)

DPL 和 DPH 组合为第一组 16 位数据指针寄存器 DPTR0

### 31.1.2 第 2 组 16 位数据指针寄存器 (DPTR1)

| 符号   | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|------|-----|----|----|----|----|----|----|----|----|
| DPL1 | E4H |    |    |    |    |    |    |    |    |
| DPH1 | E5H |    |    |    |    |    |    |    |    |

DPL1 为低 8 位数据 (低字节)

DPH1 为高 8 位数据 (高字节)

DPL1 和 DPH1 组合为第二组 16 位数据指针寄存器 DPTR1

### 31.1.3 数据指针控制寄存器 (DPS)

| 符号  | 地址  | B7  | B6  | B5  | B4  | B3  | B2 | B1 | B0  |
|-----|-----|-----|-----|-----|-----|-----|----|----|-----|
| DPS | E3H | ID1 | ID0 | TSL | AU1 | AU0 | -  | -  | SEL |

ID1: 控制 DPTR1 自动递增方式

0: DPTR1 自动递增

1: DPTR1 自动递减

ID0: 控制DPTR0自动递增方式

- 0: DPTR0 自动递增
- 1: DPTR0 自动递减

TSL: DPTR0/DPTR1自动切换控制（自动对SEL进行取反）

- 0: 关闭自动切换功能
- 1: 使能自动切换功能

当 TSL 位被置 1 后，每当执行完成相关指令后，系统会自动将 SEL 位取反。

与 TSL 相关的指令包括如下指令：

```
MOV    DPTR,#data16
INC     DPTR
MOVC    A,@A+DPTR
MOVX    A,@DPTR
MOVX    @DPTR,A
```

AU1/AU0: 使能DPTR1/DPTR0使用ID1/ID0控制位进行自动递增/递减控制

- 0: 关闭自动递增/递减功能
- 1: 使能自动递增/递减功能

注意：在写保护模式下，AU0 和 AU1 位无法直接单独使能，若单独使能 AU1 位，则 AU0 位也会被自动使能，若单独使能 AU0，没有效果。若需要单独使能 AU1 或者 AU0，则必须使用 TA 寄存器触发 DPS 的保护机制（参考 TA 寄存器的说明）。另外，只有执行下面的 3 条指令后才会对 DPTR0/DPTR1 进行自动递增/递减操作。3 条相关指令如下：

```
MOVC    A,@A+DPTR
MOVX    A,@DPTR
MOVX    @DPTR,A
```

SEL: 选择DPTR0/DPTR1作为当前的目标DPTR

- 0: 选择 DPTR0 作为目标 DPTR
- 1: 选择 DPTR1 作为目标 DPTR

SEL 选择目标 DPTR 对下面指令有效：

```
MOV     DPTR,#data16
INC      DPTR
MOVC     A,@A+DPTR
MOVX     A,@DPTR
MOVX     @DPTR,A
JMP      @A+DPTR
```

### 31.1.4 数据指针控制寄存器（TA）

| 符号 | 地址  | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|----|-----|----|----|----|----|----|----|----|----|
| TA | AEH |    |    |    |    |    |    |    |    |

TA寄存器是对DPS寄存器中的AU1和AU0进行写保护的。由于程序无法对DPS中的AU1和AU0进行单独的写入，所以当需要单独使能AU1或者AU0时，必须使用TA寄存器进行触发。TA寄存器是只写寄存器。当需要对AU1或者AU0进行单独使能时，必须按照如下的步骤进行操作：

```
CLR      EA      ;关闭中断（必需）
MOV      TA,#0AAH ;写入触发命令序列 1
           ;此处不能有其他任何指令
```

|             |                 |   |
|-------------|-----------------|---|
| <b>MOV</b>  | <b>TA,#55H</b>  | ;写入触发命令序列 2<br>;此处不能有其他任何指令                     |
| <b>MOV</b>  | <b>DPS,#xxH</b> | ;写保护暂时关闭,可向 <b>DPS</b> 中写入任何值<br>;DSP 再次进行写保护状态 |
| <b>SETB</b> | <b>EA</b>       | ;打开中断 (如有必要)                                    |

STC MCU

## 31.2 范例程序

### 31.2.1 示例代码 1

将程序空间 1000H~1003H 的 4 个字节数据反向复制到扩展 RAM 的 0100H~0103H 中, 即

C:1000H → X:0103H

C:1001H → X:0102H

C:1002H → X:0101H

C:1003H → X:0100H

#### 汇编代码

;测试工作频率为 11.0592MHz

```

P1M1      DATA      091H
P1M0      DATA      092H
P0M1      DATA      093H
P0M0      DATA      094H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

                ORG      0000H
                LJMP     MAIN

MAIN:          ORG      0100H

                MOV      SP, #5FH
                ORL      P_SW2, #80H                ;使能访问 XFR

                MOV      P0M0, #00H
                MOV      P0M1, #00H
                MOV      P1M0, #00H
                MOV      P1M1, #00H
                MOV      P2M0, #00H
                MOV      P2M1, #00H
                MOV      P3M0, #00H
                MOV      P3M1, #00H
                MOV      P4M0, #00H
                MOV      P4M1, #00H
                MOV      P5M0, #00H
                MOV      P5M1, #00H

                MOV      DPS, #00100000B            ;使能 TSL, 并选择 DPTR0
                MOV      DPTR, #1000H                ;将 1000H 写入 DPTR0 后选择 DPTR1 为 DPTR
                MOV      DPTR, #0103H                ;将 0103H 写入 DPTR1 中
                MOV      DPS, #10111000B            ;设置 DPTR1 为递减模式, DPTR0 为递加模式, 使能 TSL
                                                    ;AU0 和 AU1, 并选择 DPTR0 为当前的 DPTR
                MOV      R7, #4                        ;设置数据复制个数

COPY_NEXT:    CLR      A
                MOVC     A, @A+DPTR                ;从 DPTR0 所指的程序空间读取数据,

```

```

MOVX    @DPTR,A           ;完成后DPTR0 自动加1 并将DPTR1 设置为DPTR
DJNZ    R7,COPY_NEXT      ;将ACC 的数据写入到DPTR1 所指的XDATA 中,
SJMP    $                 ;完成后DPTR1 自动减1 并将DPTR0 设置为DPTR
;

END

```

### 31.2.2 示例代码 2

将扩展 RAM 的 0100H~0103H 中的数据依次发送到 P0 口

汇编代码

;测试工作频率为 11.0592MHz

```

P1M1    DATA    091H
P1M0    DATA    092H
P0M1    DATA    093H
P0M0    DATA    094H
P2M1    DATA    095H
P2M0    DATA    096H
P3M1    DATA    0B1H
P3M0    DATA    0B2H
P4M1    DATA    0B3H
P4M0    DATA    0B4H
P5M1    DATA    0C9H
P5M0    DATA    0CAH

        ORG       0000H
        LJMP      MAIN

MAIN:    ORG       0100H

        MOV       SP, #5FH
        ORL       P_SW2, #80H           ;使能访问 XFR

        MOV       P0M0, #00H
        MOV       P0M1, #00H
        MOV       P1M0, #00H
        MOV       P1M1, #00H
        MOV       P2M0, #00H
        MOV       P2M1, #00H
        MOV       P3M0, #00H
        MOV       P3M1, #00H
        MOV       P4M0, #00H
        MOV       P4M1, #00H
        MOV       P5M0, #00H
        MOV       P5M1, #00H

        CLR       EA                   ;关闭中断
        MOV       TA, #0AAH            ;写入 DPS 写保护触发命令 1
        MOV       TA, #55H            ;写入 DPS 写保护触发命令 2
        MOV       DPS, #00001000B      ;DPTR0 递增, 单独使能 AU0, 并选择 DPTR0
        SETB      EA                   ;打开中断
        MOV       DPTR, #0100H         ;将 0100H 写入 DPTR0 中
        MOVX      A, @DPTR             ;从 DPTR0 所指的 XRAM 读取数据后 DPTR0 自动加 1
        MOV       P0, A                ;数据输出到 P0 口
        MOVX      A, @DPTR             ;从 DPTR0 所指的 XRAM 读取数据后 DPTR0 自动加 1

```



---

|             |                |  |
|-------------|----------------|--|
| <i>MOV</i>  | <i>P0,A</i>    | <i>;数据输出到 P0 口</i>                         |
| <i>MOVX</i> | <i>A,@DPTR</i> | <i>;从 DPTR0 所指的 XRAM 读取数据后 DPTR0 自动加 1</i> |
| <i>MOV</i>  | <i>P0,A</i>    | <i>;数据输出到 P0 口</i>                         |
| <i>MOVX</i> | <i>A,@DPTR</i> | <i>;从 DPTR0 所指的 XRAM 读取数据后 DPTR0 自动加 1</i> |
| <i>MOV</i>  | <i>P0,A</i>    | <i>;数据输出到 P0 口</i>                         |
| <i>SJMP</i> | <i>\$</i>      |  |
| <i>END</i>  |                |  |

---

STC MCU

## 32 MDU16 硬件 16 位乘除法器

| 产品线              | MDU16 |
|------------------|-------|
| STC8H1K08 系列     |       |
| STC8H1K28 系列     |       |
| STC8H3K64S4 系列   | ●     |
| STC8H3K64S2 系列   | ●     |
| STC8H8K64U 系列    | ●     |
| STC8H4K64TL 系列   | ●     |
| STC8H4K64TLCD 系列 | ●     |
| STC8H1K08T 系列    | ●     |

STC8H 系列部分型号的单片机内部集成了 MDU16/16 位硬件乘除法器。

支持如下数据运算：

- 数据规格化（需要 3~20 个时钟的运算时间）
- 逻辑左移（需要 3~18 个时钟的运算时间）
- 逻辑右移（需要 3~18 个时钟的运算时间）
- 16 位乘以 16 位（需要 10 个时钟的运算时间）
- 16 位除以 16 位（需要 9 个时钟的运算时间）
- 32 位除以 16 位（需要 17 个时钟的运算时间）

所有的操作都是基于无符号整形数据类型。

### 32.1 相关的特殊功能寄存器

| 符号    | 描述          | 地址    | 位地址与符号    |      |    |    |         |    |     |      | 复位值       |
|-------|-------------|-------|-----------|------|----|----|---------|----|-----|------|-----------|
|       |             |       | B7        | B6   | B5 | B4 | B3      | B2 | B1  | B0   |           |
| MD3   | MDU 数据寄存器   | FCF0H | MD3[7:0]  |      |    |    |         |    |     |      | 0000,0000 |
| MD2   | MDU 数据寄存器   | FCF1H | MD2[7:0]  |      |    |    |         |    |     |      | 0000,0000 |
| MD1   | MDU 数据寄存器   | FCF2H | MD1[7:0]  |      |    |    |         |    |     |      | 0000,0000 |
| MD0   | MDU 数据寄存器   | FCF3H | MD0[7:0]  |      |    |    |         |    |     |      | 0000,0000 |
| MD5   | MDU 数据寄存器   | FCF4H | MD5[7:0]  |      |    |    |         |    |     |      | 0000,0000 |
| MD4   | MDU 数据寄存器   | FCF5H | MD4[7:0]  |      |    |    |         |    |     |      | 0000,0000 |
| ARCON | MDU 模式控制寄存器 | FCF6H | MODE[2:0] |      |    |    | SC[4:0] |    |     |      | 0000,0000 |
| OPCON | MDU 操作控制寄存器 | FCF7H | -         | MDOV | -  | -  | -       | -  | RST | ENOP | x0xx,xx00 |

#### 32.1.1 操作数 1 数据寄存器（MD0~MD3）

| 符号  | 地址    | B7       | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----|-------|----------|----|----|----|----|----|----|----|
| MD3 | FCF0H | MD3[7:0] |    |    |    |    |    |    |    |
| MD2 | FCF1H | MD2[7:0] |    |    |    |    |    |    |    |

|     |       |          |
|-----|-------|----------|
| MD1 | FCF2H | MD1[7:0] |
| MD0 | FCF3H | MD0[7:0] |

### 32.1.2 操作数 2 数据寄存器 (MD4~MD5)

| 符号  | 地址    | B7       | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|-----|-------|----------|----|----|----|----|----|----|----|
| MD5 | FCF4H | MD5[7:0] |    |    |    |    |    |    |    |
| MD4 | FCF5H | MD4[7:0] |    |    |    |    |    |    |    |

#### 32位除以16位除法:

被除数: {MD3,MD2,MD1,MD0}

除数: {MD5,MD4}

商: {MD3,MD2,MD1,MD0}

余数: {MD5,MD4}

#### 16位除以16位除法:

被除数: {MD1,MD0}

除数: {MD5,MD4}

商: {MD1,MD0}

余数: {MD5,MD4}

#### 16位乘以16位乘法:

被乘数: {MD1,MD0}

乘数: {MD5,MD4}

积: {MD3,MD2,MD1,MD0}

#### 32 位逻辑左移/逻辑右移

操作数: {MD3,MD2,MD1,MD0}

#### 32 位数据规格化:

操作数: {MD3,MD2,MD1,MD0}

### 32.1.3 MDU 模式控制寄存器 (ARCON), 运算所需时钟数

| 符号    | 地址    | B7        | B6 | B5 | B4 | B3      | B2 | B1 | B0 |
|-------|-------|-----------|----|----|----|---------|----|----|----|
| ARCON | FCF6H | MODE[2:0] |    |    |    | SC[4:0] |    |    |    |

MODE[2:0]: MDU模式选择

| MODE[2:0] | 模式      | 时钟数  | 操作说明   |
|-----------|---------|------|--|
| 1         | 逻辑右移    | 3~18 | 将{MD3,MD2,MD1,MD0}中的数据右移SC[4:0]位, MD3的高位补0                                 |
| 2         | 逻辑左移    | 3~18 | 将{MD3,MD2,MD1,MD0}中的数据左移SC[4:0]位, MD0的低位补0                                 |
| 3         | 数据规格化   | 3~20 | 对{MD3,MD2,MD1,MD0}中的数据进行逻辑左移, 将数据高位的0全部移出, 使MD3的最高位为1, 逻辑左移的位数被记录在SC[4:0]中 |
| 4         | 16位×16位 | 10   | {MD1,MD0} × {MD5,MD4} = {MD3,MD2,MD1,MD0}                                  |
| 5         | 16位÷16位 | 9    | {MD1,MD0} ÷ {MD5,MD4} =<br>{MD1,MD0}...{MD5,MD4}                           |
| 6         | 32位÷16位 | 17   | {MD3,MD2,MD1,MD0} ÷ {MD5,MD4} =  |

|    |    |  |                               |
|----|----|--|-------------------------------|
|    |    |  | {MD3,MD2,MD1,MD0}...{MD5,MD4} |
| 其他 | 无效 |  |                               |

SC[4:0]: 数据移动位数

当 MDU 为移动模式时, SC 用于设置左移/右移的位数

当 MDU 为数据规格化模式时, SC 为数据规格化后数据所移动的实际位数

32.1.4 MDU 操作控制寄存器 (OPCON)

| 符号    | 地址    | B7 | B6   | B5 | B4 | B3 | B2 | B1  | B0   |
|-------|-------|----|------|----|----|----|----|-----|------|
| OPCON | FCF7H | -  | MDOV | -  | -  | -  | -  | RST | ENOP |

MDOV: MDU溢出标志位 (只读标志位)

在如下几种情况时, MDOV 会被硬件自动置 1:

- 1、除数为 0 时;
- 2、乘法的积大于 0FFFFH 时;

当软件写 OPCON.0 (EN) 或者写 ARCON 时, 硬件会自动清除 MDOV

RST: 软件复位 MDU 乘除单元。写 1 触发软件复位, MDU 复位完成后硬件自动清零。

注: 软件复位 MDU 乘除单元时, ARCON 寄存器的值会被清除。

ENOP: MDU 模块使能。写 1 触发 MDU 模块开始计算, 当 MDU 计算完成后, 硬件自动将 ENOP 清零。

软件可以在对 ENOP 置 1 后, 循环的查询 ENOP, 当 ENOP 由 1 变 0 则表示计算完成。

## 32.2 范例程序

### C 语言代码

---

//测试工作频率为 11.0592MHz

#include "stc8h.h"  
#include "intrins.h"

////////////////////////////////////  
//16 位乘 16 位  
////////////////////////////////////

unsigned long res;  
unsigned int dat1, dat2;

P\_SW2 |= 0x80; //访问扩展寄存器 xsfr  
MD1U16 = dat1; //dat1 用户给定  
MD5U16 = dat2; //dat2 用户给定  
ARCON = 4 << 5; //16 位\*16 位,乘法模式  
OPCON = 1; //启动计算  
while((OPCON & 1) != 0); //等待计算完成  
res = MD3U32; //32 位结果

////////////////////////////////////  
//32 位除以 16 位  
////////////////////////////////////

unsigned long res;  
unsigned long dat1;  
unsigned int dat2;

P\_SW2 |= 0x80; //访问扩展寄存器 xsfr  
MD3U32 = dat1; //dat1 用户给定  
MD5U16 = dat2; //dat2 用户给定  
ARCON = 6 << 5; //32 位/16 位,除法模式  
OPCON = 1; //启动计算  
while((OPCON & 1) != 0); //等待计算完成  
res = MD3U32; //32 位商, 16 位余数在 MD5U16 中

---

////////////////////////////////////

//左移或右移:

---

////////////////////////////////////

unsigned long res;  
unsigned long dat1;  
unsigned char num; //移位的位数, 用户给定

MD3U32 = dat1; //dat1 用户给定  
ARCON = (2 << 5) + num; //32 位左移模式  
//ARCON = (1 << 5) + num; //32 位右移模式  
OPCON = 1; //启动计算  
while((OPCON & 1) != 0); //等待计算完成  
res = MD3U32; //32 位结果

---

# 附录A 编译器（汇编器）/仿真器/头文件使用指南

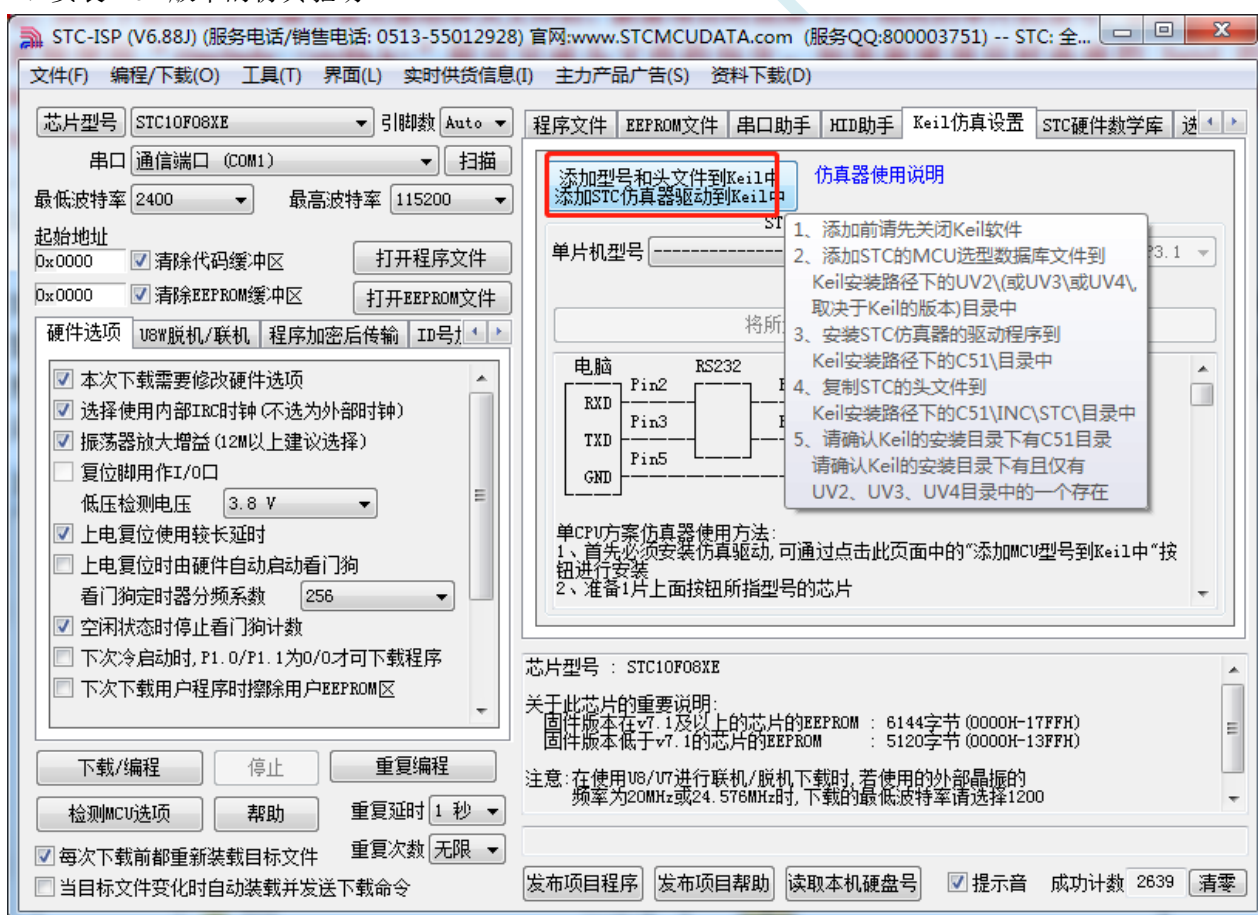
**A: STC 单片机应使用何种编译器/汇编器?**

**Q: 任何老式的 8051 编译器/汇编器都可以支持, 现流行使用 Keil C51**

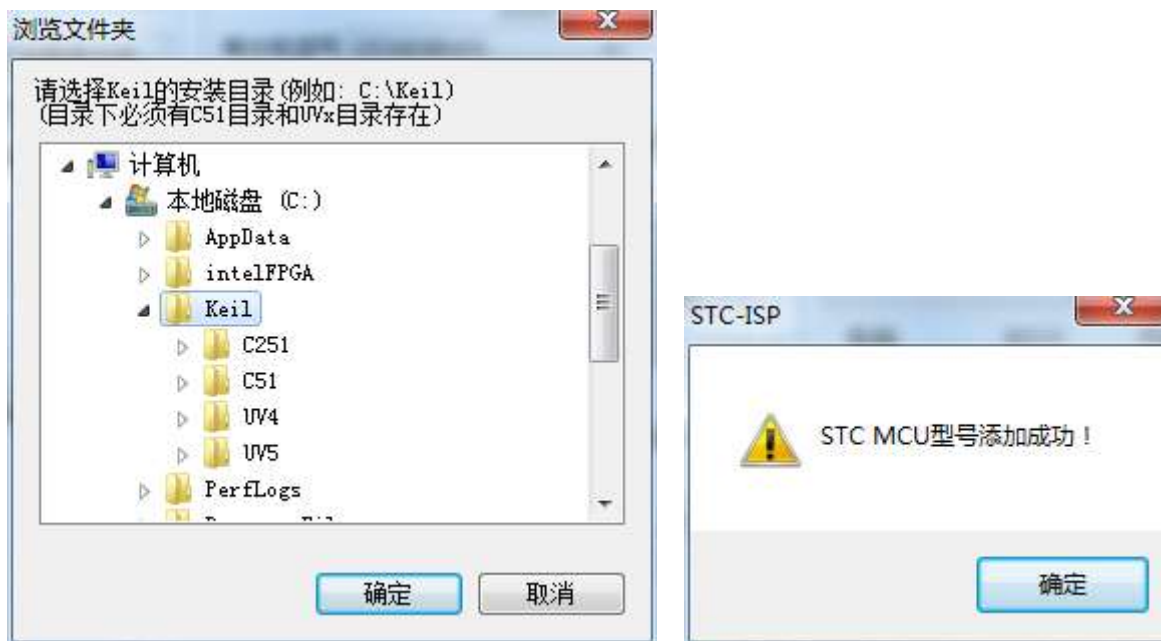
**A: Keil 环境中, 应如何包含头文件**

**Q: 按照下面图示的步骤安装完驱动和头文件后, 新建项目时选择 STC 相应的单片机型号, 在源文件中直接使用 “#include <stc8h.h>” 即可完成头文件的包含。如果新建项目时选择的 Intel 的 8052/87C52/87C54/87C58 或 Philips 的 P87C52/P87C54/P87C58 编译, 头文件包含<reg51.h>即可, 不过 STC 新增的特殊功能寄存器则需要用户自己声明。**

## 1、安装 Keil 版本的仿真驱动

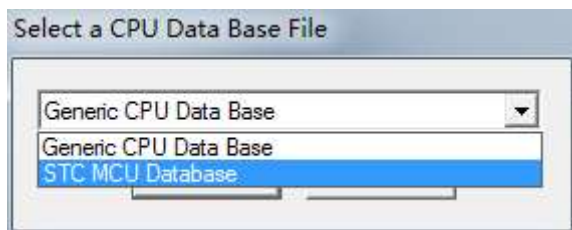


如上图, 首先选择“Keil 仿真设置”页面, 点击“添加 MCU 型号到 Keil 中”, 在出现的如下的目录选择窗口中, 定位到 Keil 的安装目录(一般可能为“C:\Keil\”), “确定”后出现下图中右边所示的提示信息, 表示安装成功。添加头文件的同时也会安装 STC 的 Monitor51 仿真驱动 STCMON51.DLL, 驱动与头文件的安装目录如上图所示。

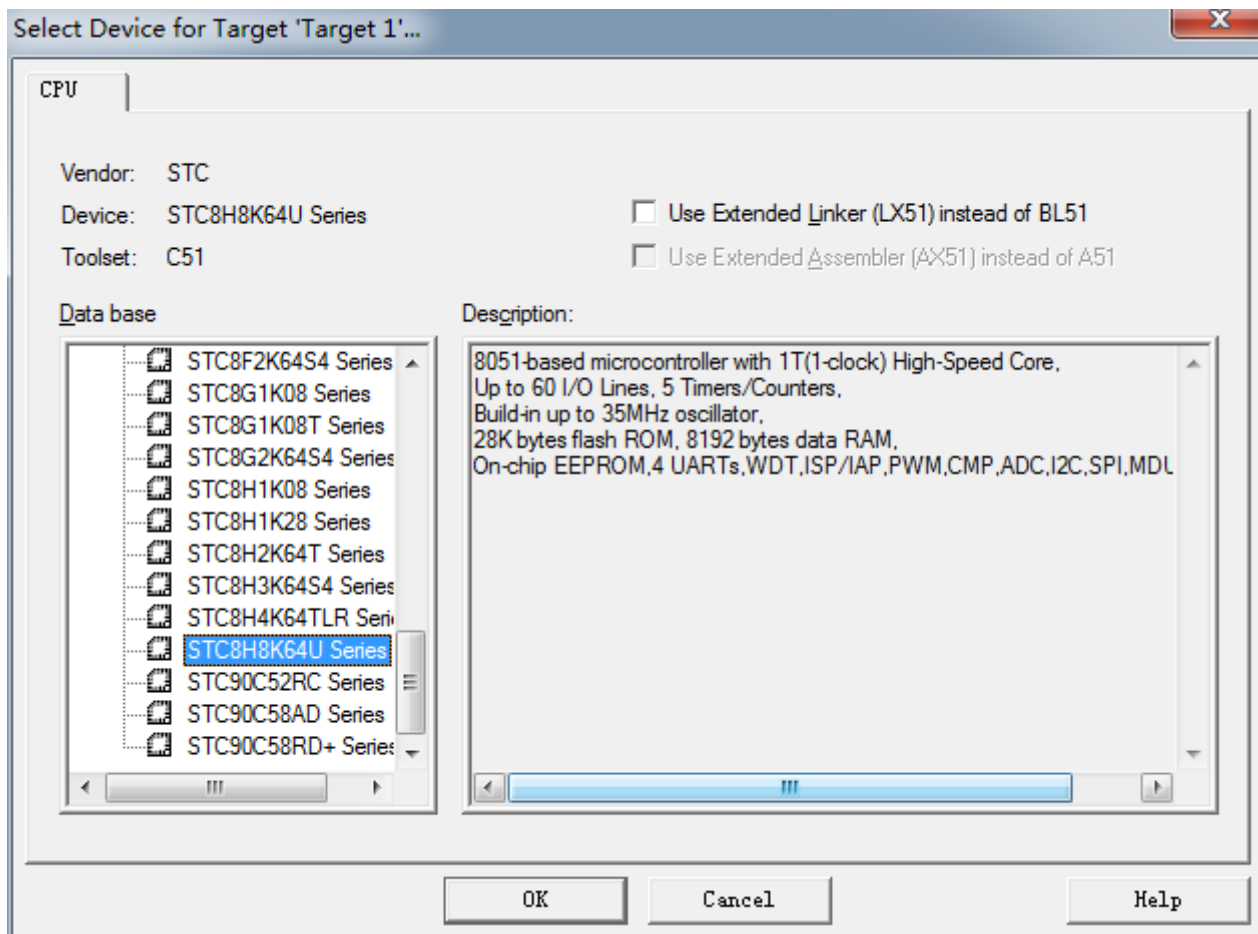


## 2、在 Keil 中创建项目

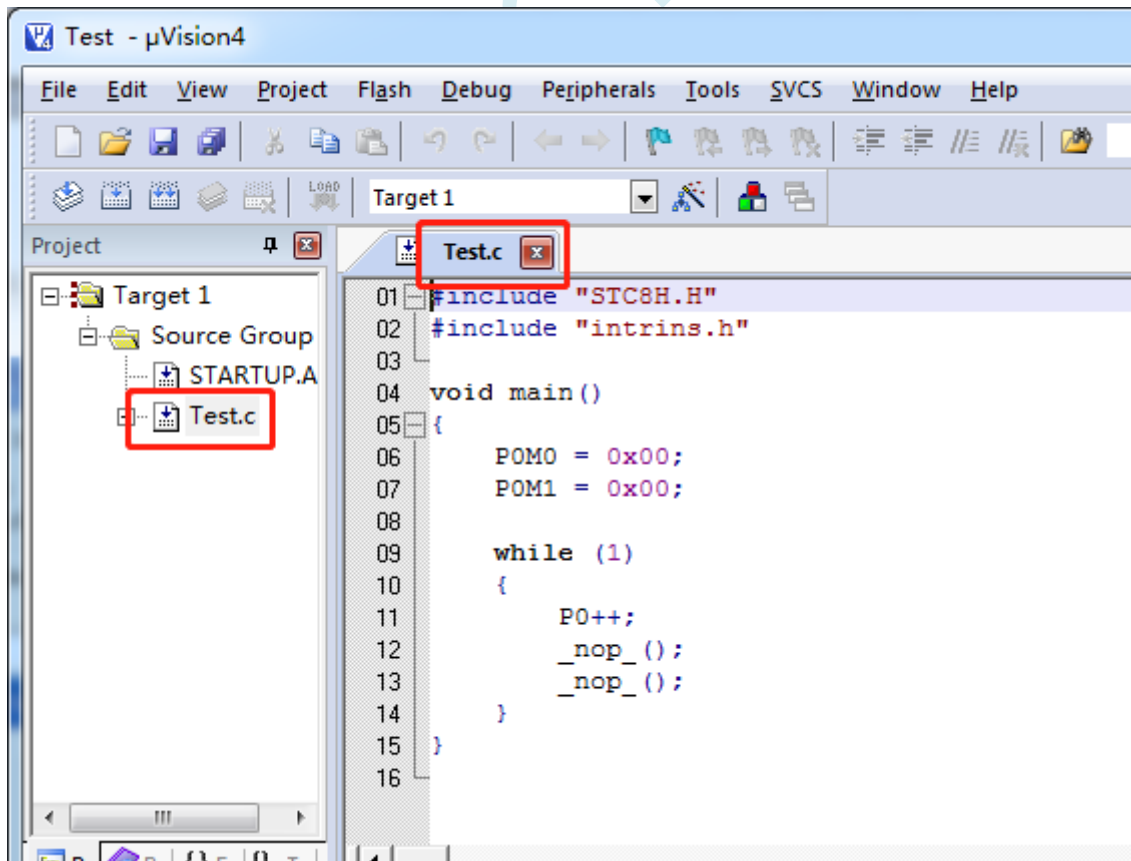
若第一步的驱动安装成功，则在 Keil 中新建项目时选择芯片型号时，便会有“STC MCU Database”的选择项，如下图



然后从列表中选择响应的 MCU 型号，我们在此选择“STC8H8K64U”的型号，点击“确定”完成选择



添加源代码文件到项目中，如下图：

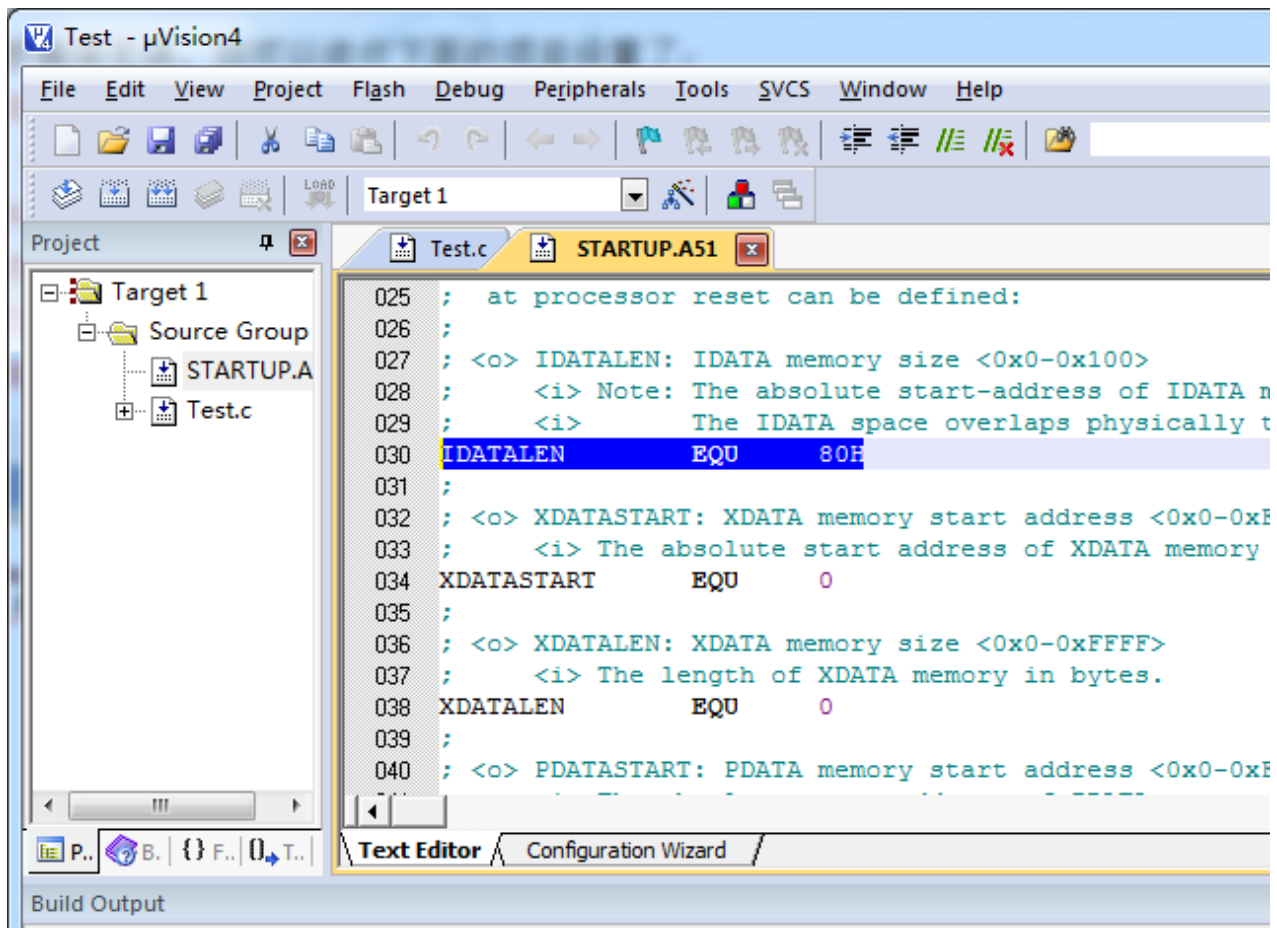


保存项目，若编译无误，则可以进行下面的项目设置了



附加说明一点:

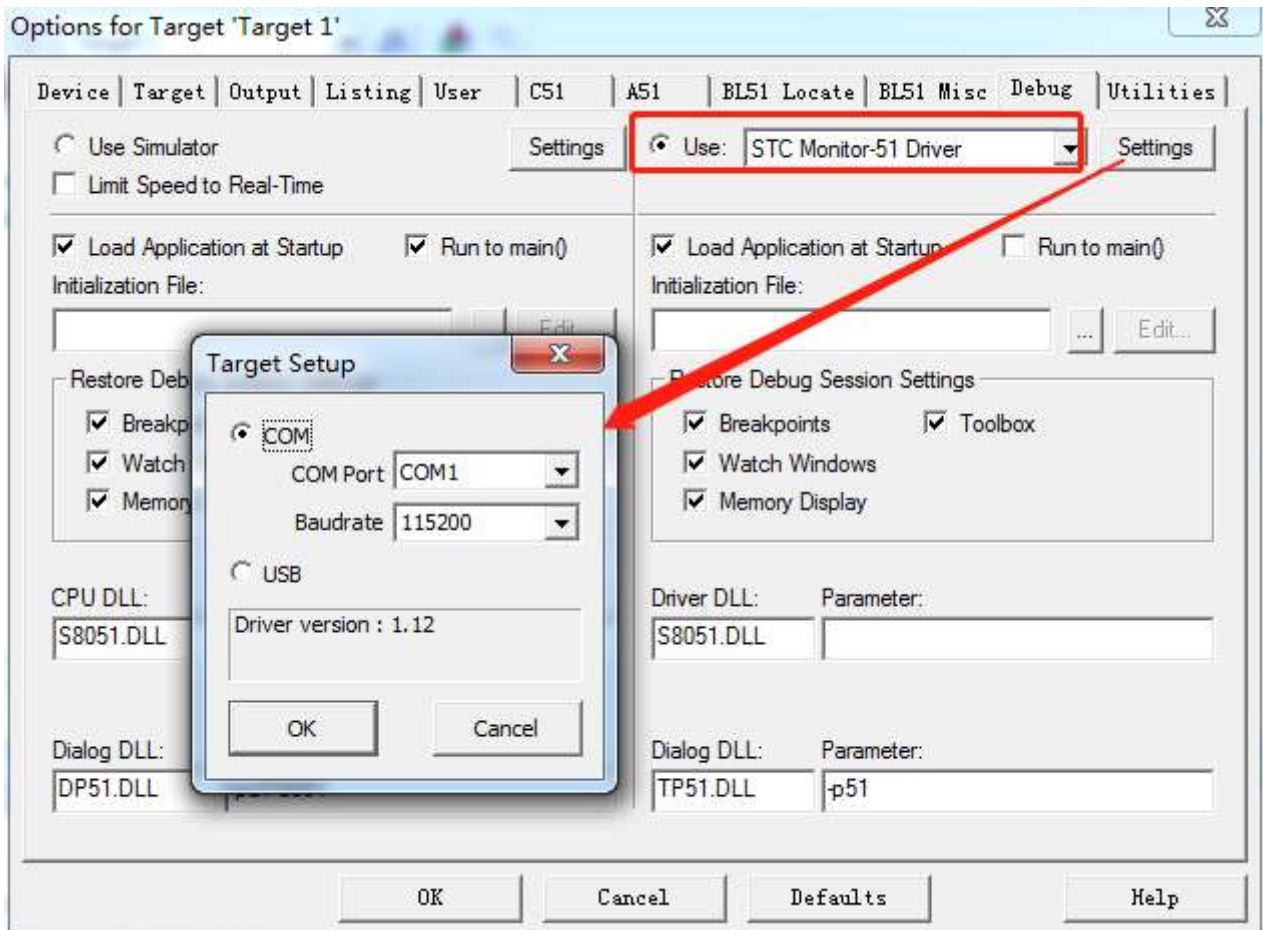
当创建的是 C 语言项目, 且有将启动文件 “STARTUP.A51” 添加到项目中时, 里面有一个命名为 “IDATALEN” 的宏定义, 它是用来定义 IDATA 大小的一个宏, 默认值是 128, 即十六进制的 80H, 同时它也是启动文件中需要初始化为 0 的 IDATA 的大小。所以当 IDATA 定义为 80H, 那么 STARTUP.A51 里面的代码则会将 IDATA 的 00-7F 的 RAM 初始化为 0; 同样若将 IDATA 定义为 0FFH, 则会将 IDATA 的 00-FF 的 RAM 初始化为 0。



虽然 STC8 系列的单片机的 IDATA 大小为 256 字节 (00-7F 的 DATA 和 80H-FFH 的 IDATA), 但由于在 RAM 的最后 17 个字节有写入 ID 号以及相关的测试参数, 若用户在程序中需要使用这一部分数据, 则一定不要将 IDATALEN 定义为 256。

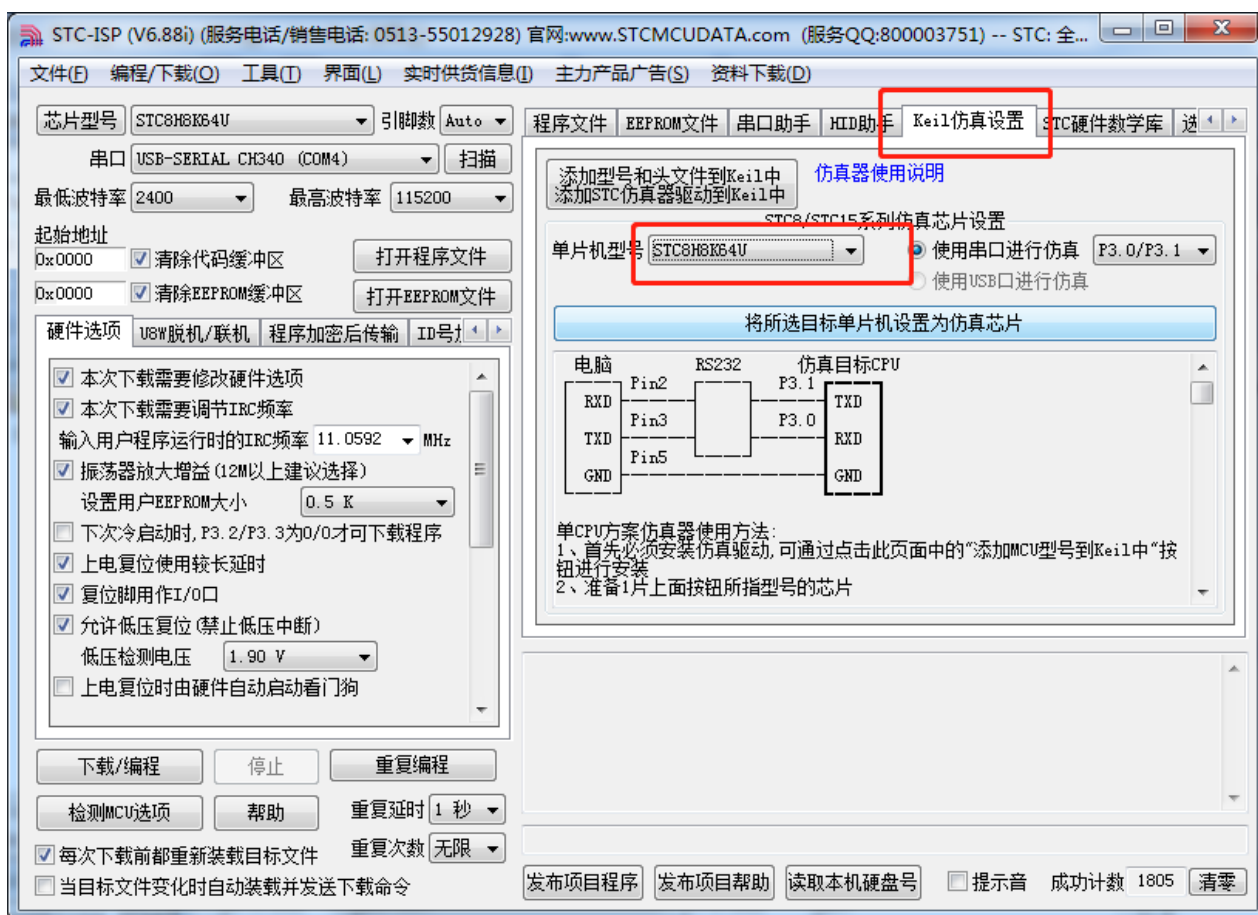
另外: **XDATALEN** 宏是用来定义 XDATA 大小的一个宏, 默认值是 0, STARTUP.A51 里面的代码会根据用户设置的 XDATALEN 长度来自动初始化 0~(XDATALEN-1)区域的 XDATA 为 0。一般保持 XDATALEN 为默认值 0 即可。若用户需要修改, 请根据实际的仿真的 XDATA 大小并保留 768 字节来设置此项。比如 STC8H8K64U 的 XDATA 为 8K, 则需要将 XDATALEN 设置为 8192-768

3、项目设置, 选择 STC 仿真驱动



如上图，首先进入到项目的设置页面，选择“Debug”设置页，第2步选择右侧的硬件仿真“Use ...”，第3步，在仿真驱动下拉列表中选择“STC Monitor-51 Driver”项，然后点击“Settings”按钮，进入下面的设置画面，对串口的端口号和波特率进行设置，波特率一般选择 115200。到此设置便完成了。

#### 4、创建仿真芯片



准备一颗 STC8A 系列或者 STC8F 系列的芯片, 并通过下载板连接到电脑的串口, 然后如上图, 选择正确的芯片型号, 然后进入到“Keil 仿真设置”页面, 点击相应型号的按钮, 当程序下载完成后仿真器便制作完成了。

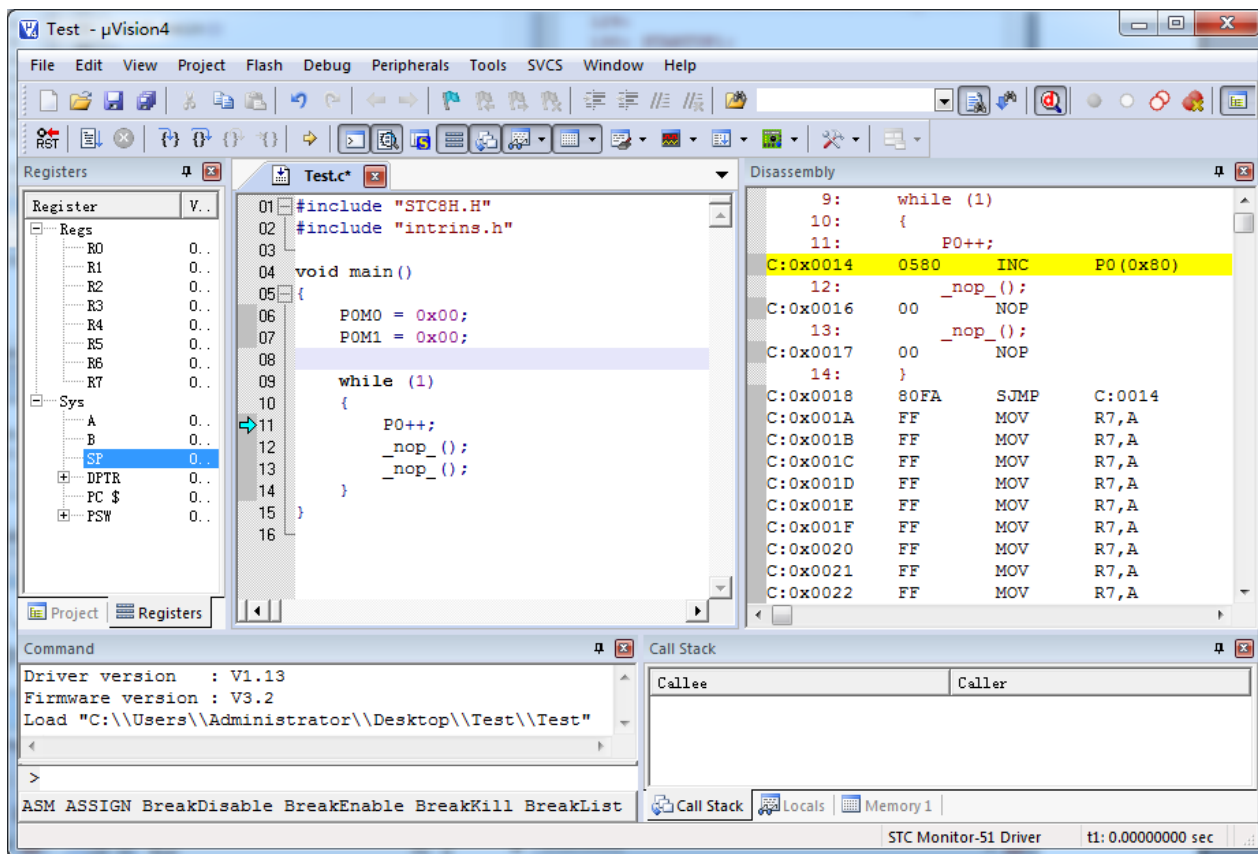
## 5、开始仿真

将制作完成的仿真芯片通过串口与电脑相连接。

将前面我们所创建的项目编译至没有错误后, 按“Ctrl+F5”开始调试。

若硬件连接无误的话, 将会进入到类似于下面的调试界面, 并在命令输出窗口显示当前的仿真驱动版本号和当前仿真监控代码固件的版本号

断点设置的个数目前最大允许 20 个 (理论上可设置任意个, 但是断点设置得过多会影响调试的速度)。



#### 仿真注意事项:

- 1、仿真监控程序占用 P3.0/P3.1 两个端口, 但不占用串口 1, 用户可以将串口 1 切换到 P3.6/P3.7 或者 P1.6/P1.7 再使用
- 2、仿真监控程序占用内部扩展 RAM(XDATA)的最后 768 字节, 用户不可对这个区域的 XDATA 进行写操作 (若有将 `STARTUP.A51` 文件加入到项目, 请注意: `XDATALEN` 宏是用来定义 XDATA 大小的一个宏, 默认值是 0, `STARTUP.A51` 里面的代码会根据用户设置的 `XDATALEN` 长度来自动初始化 0~(`XDATALEN`-1)区域的 XDATA 为 0。一般保持 `XDATALEN` 为默认值 0 即可。若用户需要修改, 请根据实际的仿真的 XDATA 大小并保留 768 字节来设置此项。比如 STC8H8K64U 的 XDATA 为 8K, 则需要将 `XDATALEN` 设置为 8192-768)

## 附录B STC-ISP 下载软件高级应用

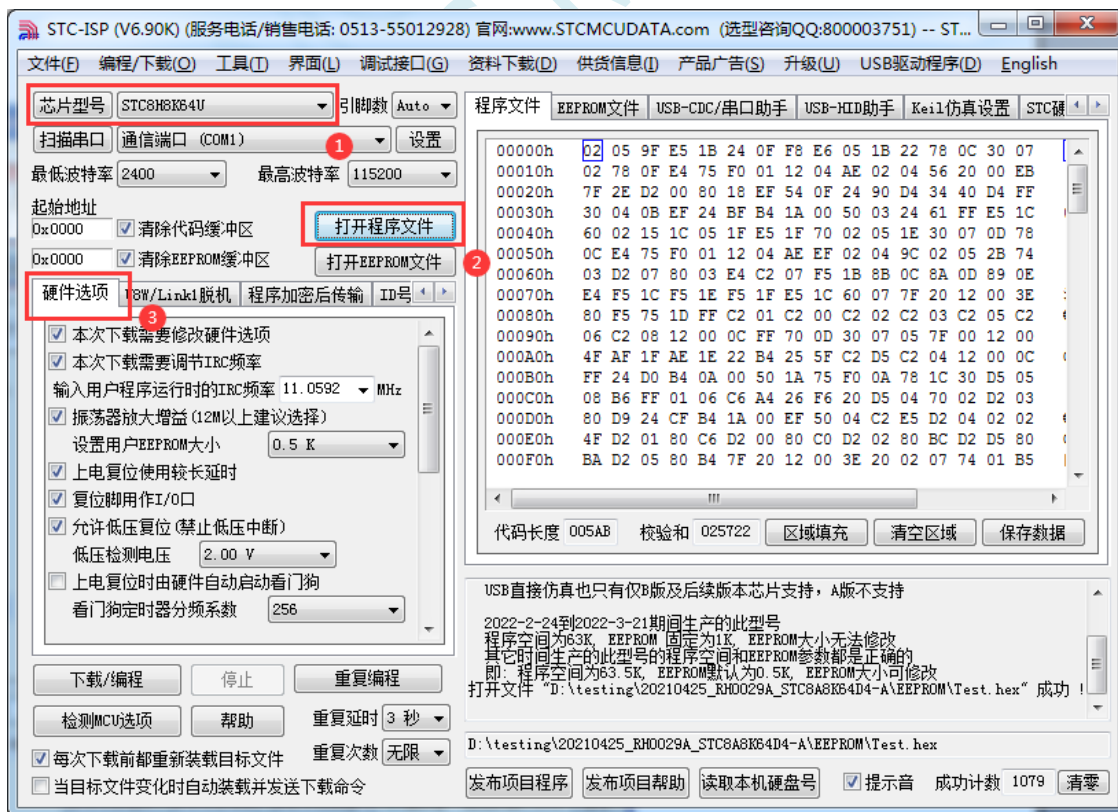
### B.1 发布项目程序

发布项目程序功能主要是将用户的程序代码与相关的选项设置打包成为一个可以直接对目标芯片进行下载编程的**超级简单的用户自己界面的可执行文件**。

关于界面, 用户可以自己进行定制 (用户可以自行修改发布项目程序的标题、按钮名称以及帮助信息), 同时用户还可以指定目标电脑的硬盘号和目标芯片的 ID 号, 指定目标电脑的硬盘号后, 便可以控制发布应用程序只能在指定的电脑上运行 (防止烧录人员将程序轻易从电脑盗走, 如通过网络发走, 如通过 U 盘拷走, 防不胜防, 当然盗走你的电脑那就没办法那, 所以 STC 的脱机下载工具比电脑烧录安全, 能限制可烧录芯片数量, 让前台文员小姐烧, 让老板娘烧都可以), 拷贝到其它电脑, 应用程序不能运行。同样的, 当指定了目标芯片的 ID 号后, 那么用户代码只能下载到具有相应 ID 号的目标芯片中 (对于一台设备要卖几千万的产品特别有用---坦克, 可以发给客户自己升级, 不需冒着生命危险跑到战火纷飞的伊拉克升级软件啦), 对于 ID 号不一致的其它芯片, 不能进行下载编程。

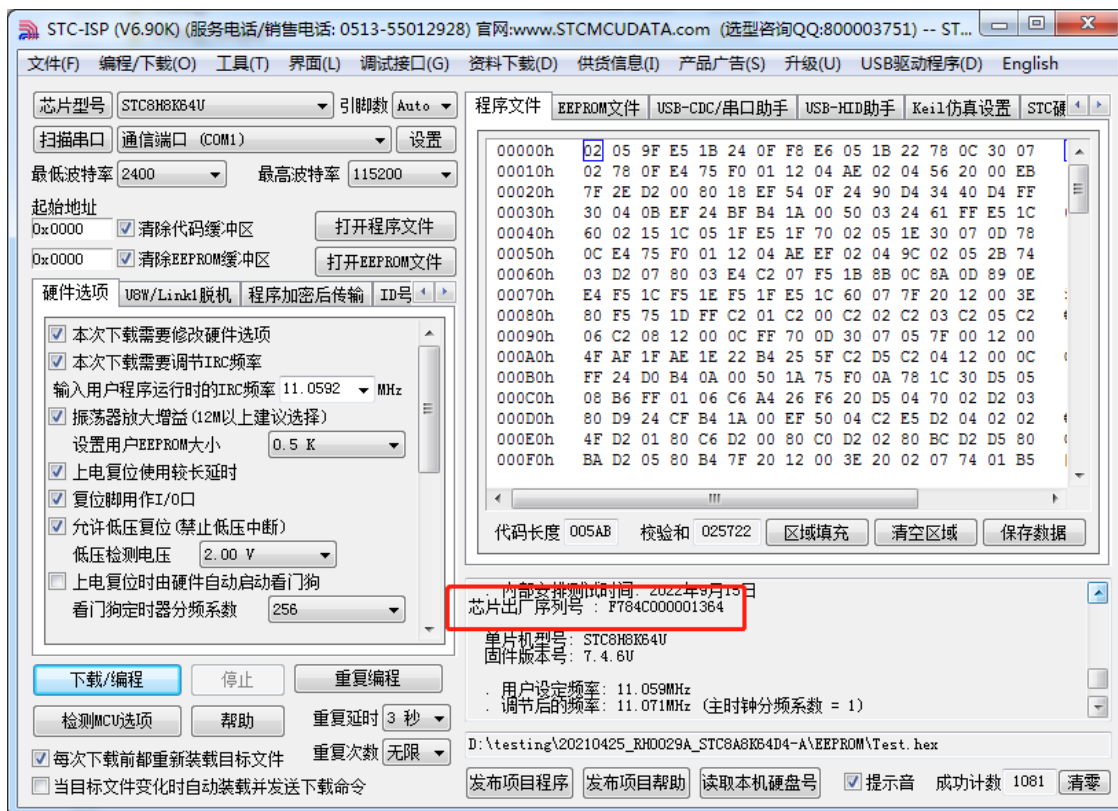
发布项目程序详细的操作步骤如下:

- 1、首先选择目标芯片的型号
- 2、打开程序代码文件
- 3、设置好相应的硬件选项

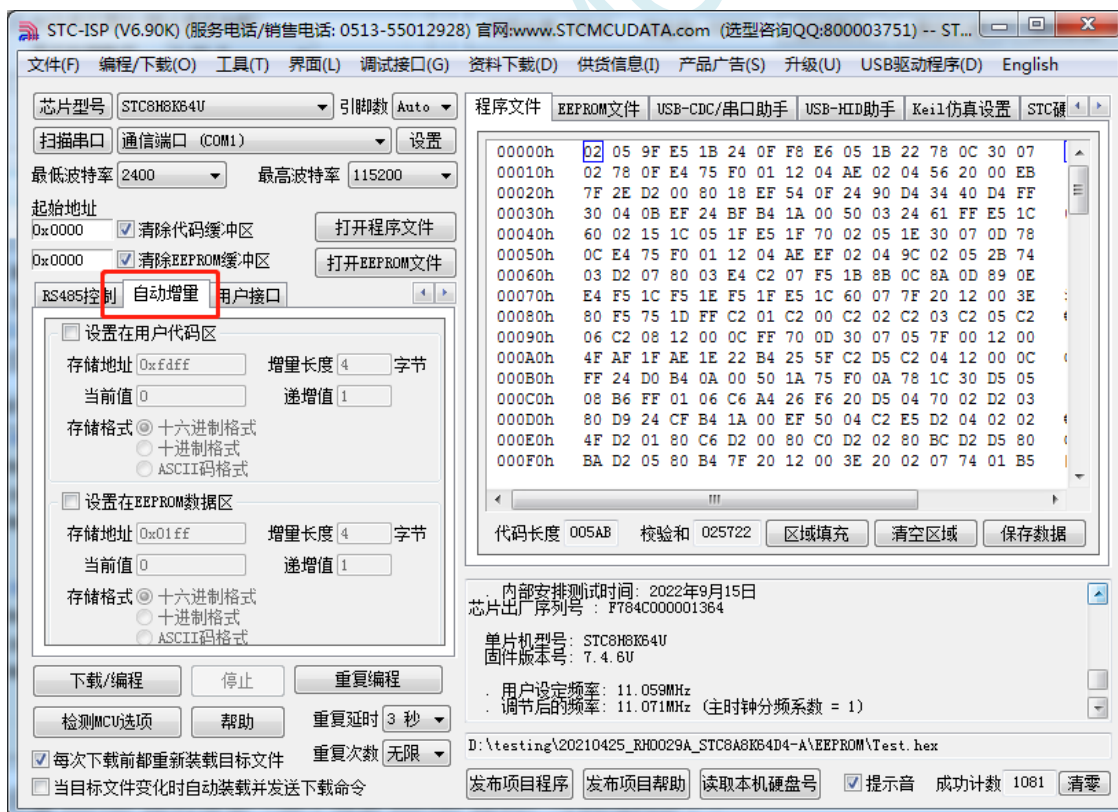


- 4、试烧一下芯片, 并记下目标芯片的 ID 号, 如下图所示, 该芯片的 ID 号即为“F784C000001364” (如不需要对目标芯片的 ID 号进行校验, 可跳过此步)

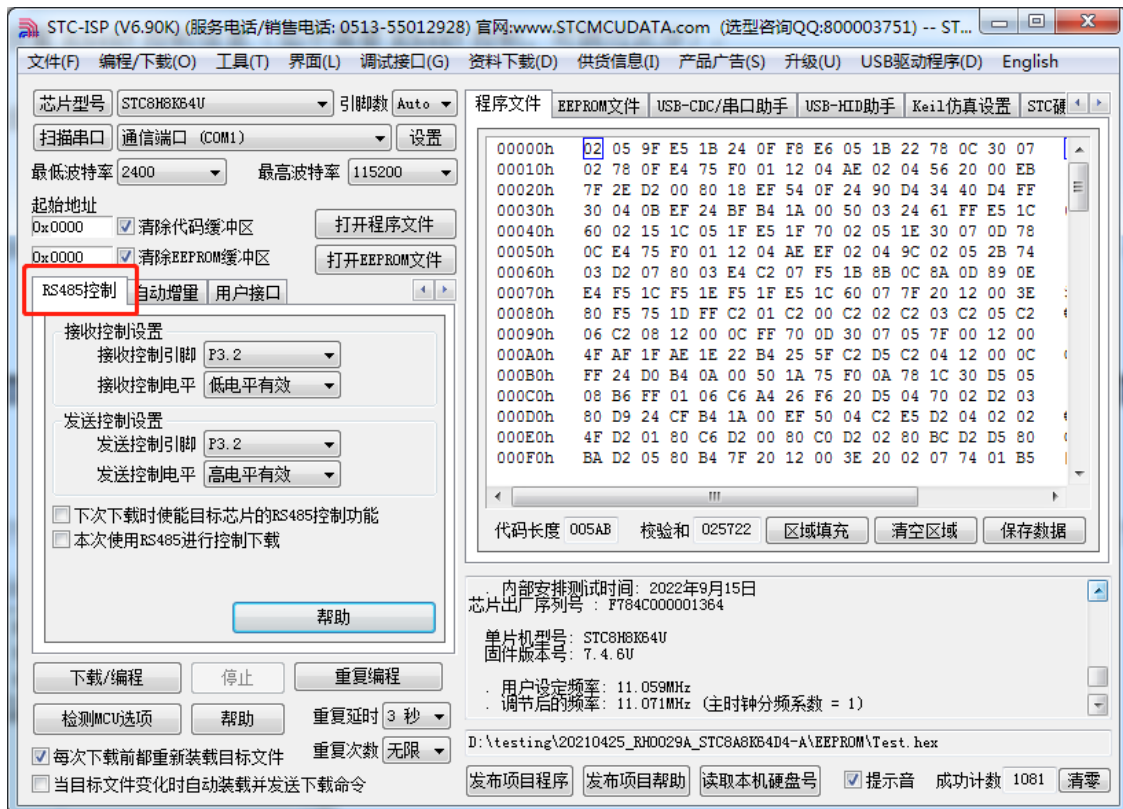




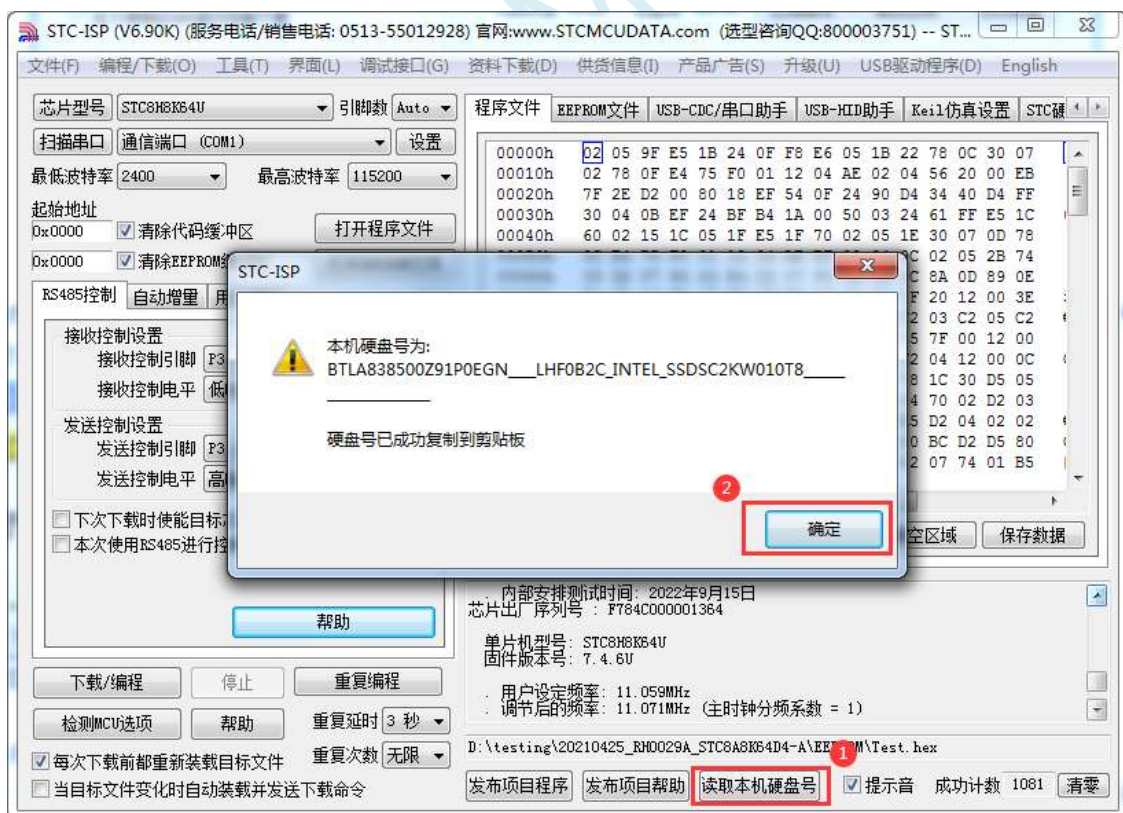
##### 5、设置自动增量（如不需要自动增量，可跳过此步）



##### 6、设置 RS485 控制信息（如不需要 RS485 控制，可跳过此步）



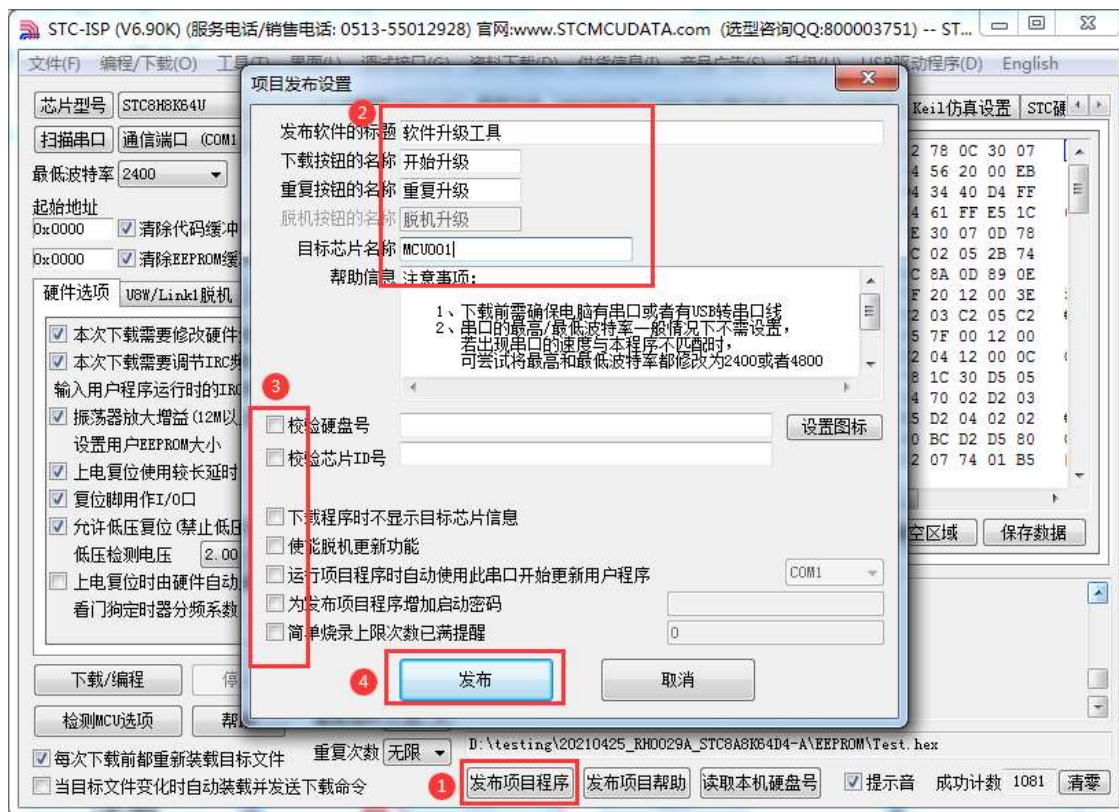
- 7、点击界面上的“读取本机硬盘号”按钮，并记下目标电脑的硬盘号（如不需要对目标电脑的硬盘号进行校验，可跳过此步）



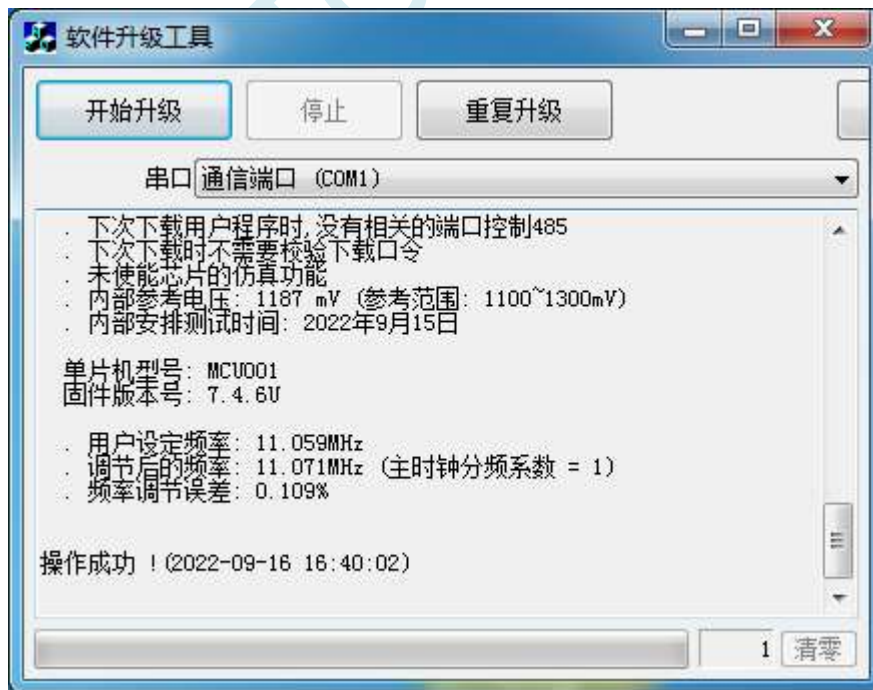
- 8、点击“发布项目程序”按钮，进入发布应用程序的设置界面。
- 9、根据各自的需要，修改发布软件的标题、下载按钮的名称、重复下载按钮的名称、自动增量的名称以及帮助信息
- 10、若需要校验目标电脑的硬盘号,则需要勾选上“校验硬盘号”，并在后面的文本框内输入前面所记

下的目标电脑的硬盘号

- 11、若需要校验目标芯片的 ID 号,则需要勾选上“校验芯片 ID 号”,并在后面的文本框内输入前面所记下的目标芯片的 ID 号



- 12、最后点击发布按钮,将项目发布程序保存,即可得到相应的可执行文件。发布的项目程序界面如下图





## B.2 程序加密后传输（防烧录时串口分析出程序）

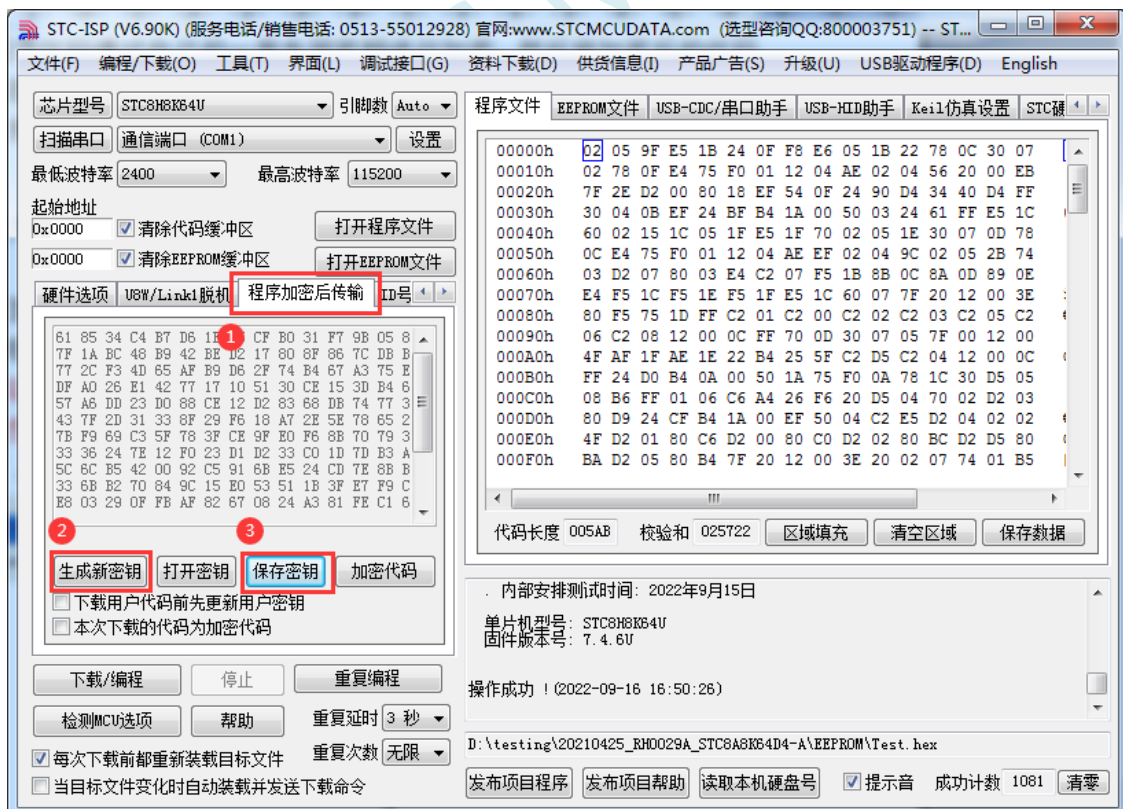
目前，所有的普通串口下载烧录编程都是采用明码通信的（电脑和目标芯片通信时，或脱机下载板和目标芯片通信时），问题：如果烧录人员通过分析下载烧录编程时串口通信的数据，高手是可以在烧录时在串口上引 2 根线出来，通过分析串口通信的数据分析出实际的用户程序代码的。当然用 STC 的脱机下载板烧程序总比用电脑烧程序强（防止烧录人员将程序轻易从电脑盗走，如通过网络发走，如通过 U 盘拷走，防不胜防，当然盗走你的电脑那就没办法那，所以 STC 的脱机下载工具比电脑烧录安全，让前台文员小姐烧，让老板娘烧都可以）。即使是 STC 全球首创的脱机下载工具，对于要防止天才的不法分子在脱机下载工具烧录的过程中通过分析串口通信的数据分析出实际的用户程序代码，也是没有办法达到要求的，这就需要用到最新的 STC 单片机所提供的程序加密后传输功能。

程序加密后传输下载是用户先将程序代码通过自己的一套专用密钥进行加密，然后将加密后的代码再通过串口下载，此时下载传输的是加密文件，通过串口分析出来的是加密后的乱码，如不通过派人潜入你公司盗窃你电脑里面的加密密钥，就无任何价值，便可起到防止在烧录程序时被烧录人员通过监测串口分析出代码的目的。

程序加密后传输功能的使用需要如下的几个步骤：

### 1、生成并保存新的密钥

如下图，进入到“程序加密后传输”页面，点击“生成新密钥”按钮，即可在缓冲区显示新生成的 256 字节的密钥。然后点击“保存密钥”按钮，即可将生成的新密钥保存为以“.K”为扩展名的密钥文件（注意：这个密钥文件一定要保存好，以后发布的代码文件都需要使用这个密钥加密，而且这个密钥的生成是非重复的，即任何时候都不可能生成两个完全相同的密钥，所以一旦密钥文件丢失将无法重新获得）。例如我们将密钥保存为“New.k”。

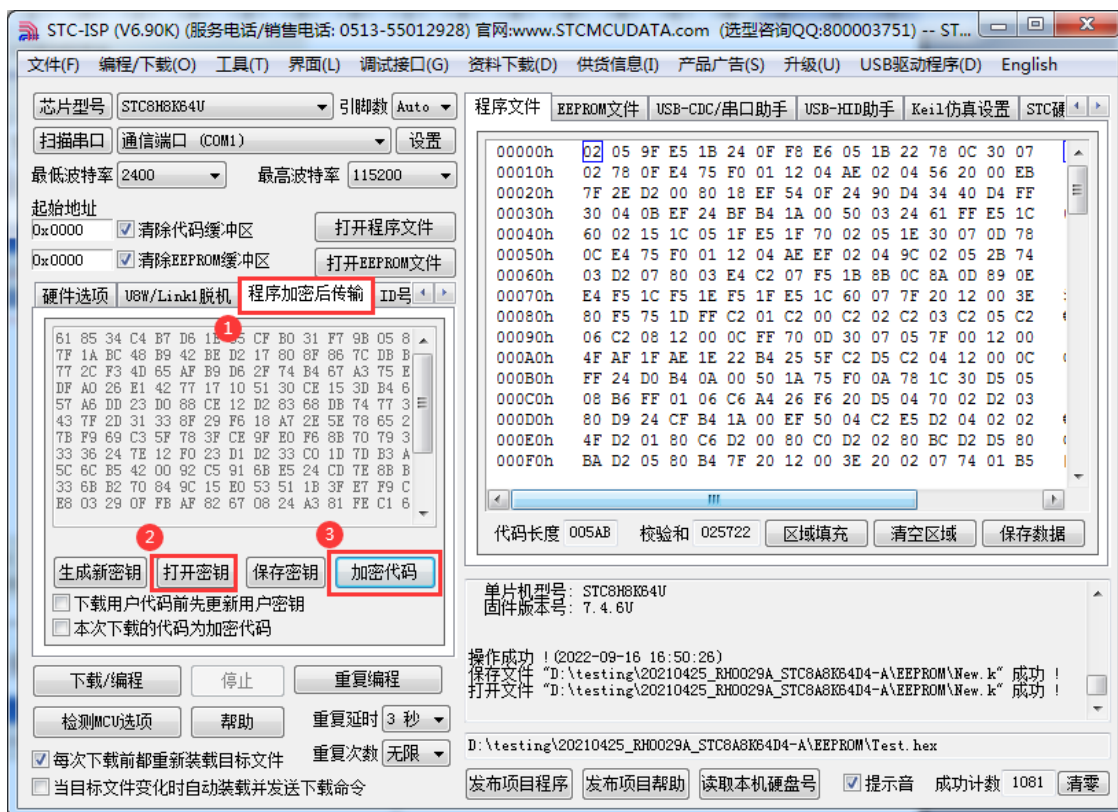


### 2、对代码文件加密

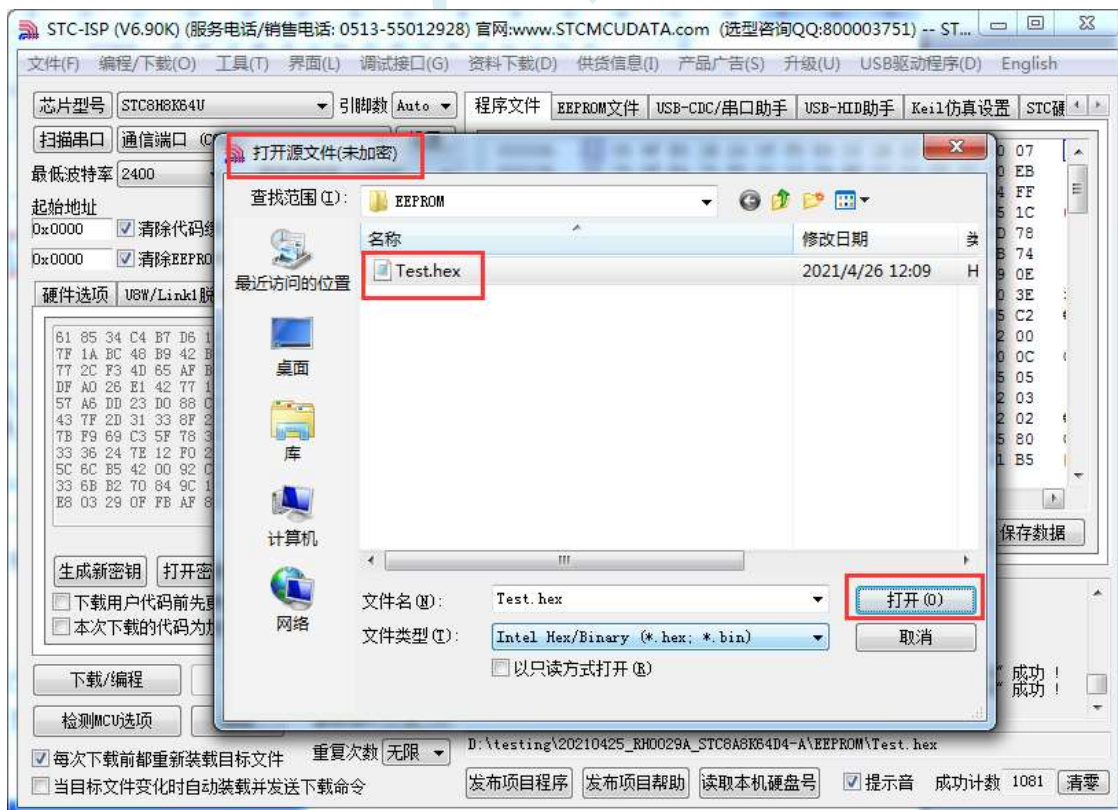
加密文件前，需要先打开我们自己的密钥。若缓冲区中存放的已经是我们的密钥，则不要再打开。

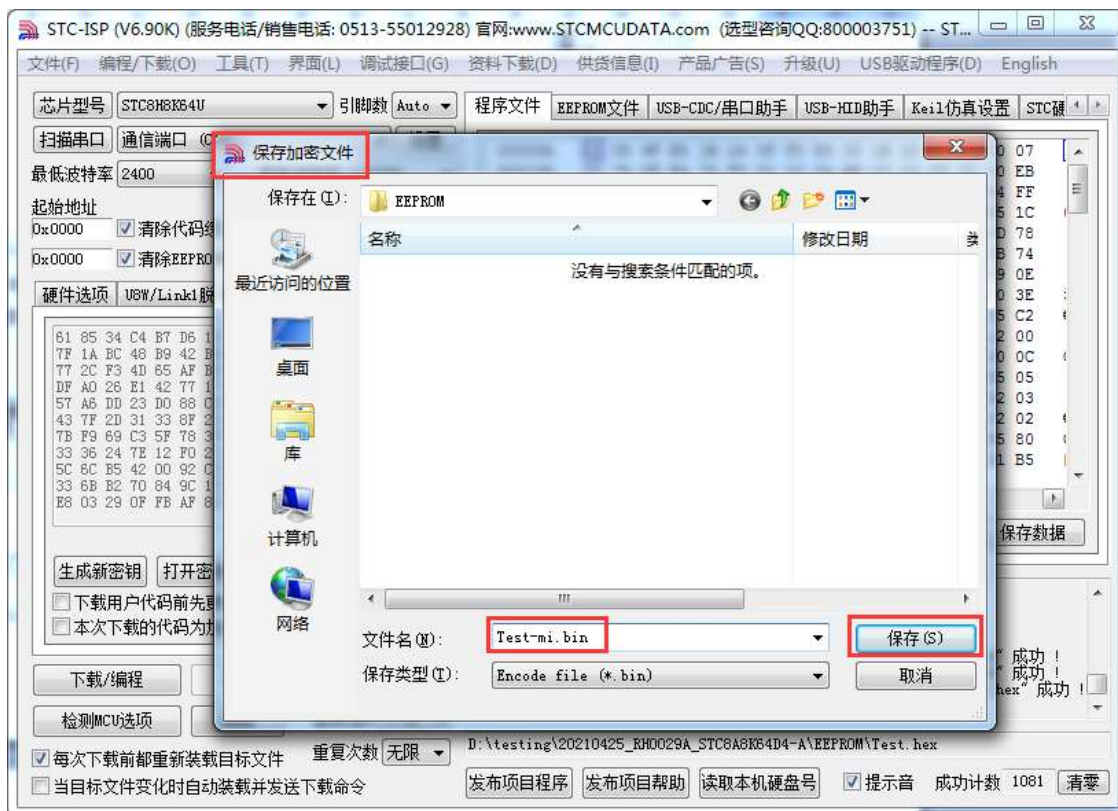
如下图，在“程序加密后传输”页面中点击“打开密钥”按钮，打开我们之前保存的密钥文件，

例如“New.k”，然后返回到“程序加密后传输”页面中点击“加密代码”按钮，如下图所示，首先会弹出“打开源文件（未加密）”的对话框，此时选择的是原始的未加密的代码文件



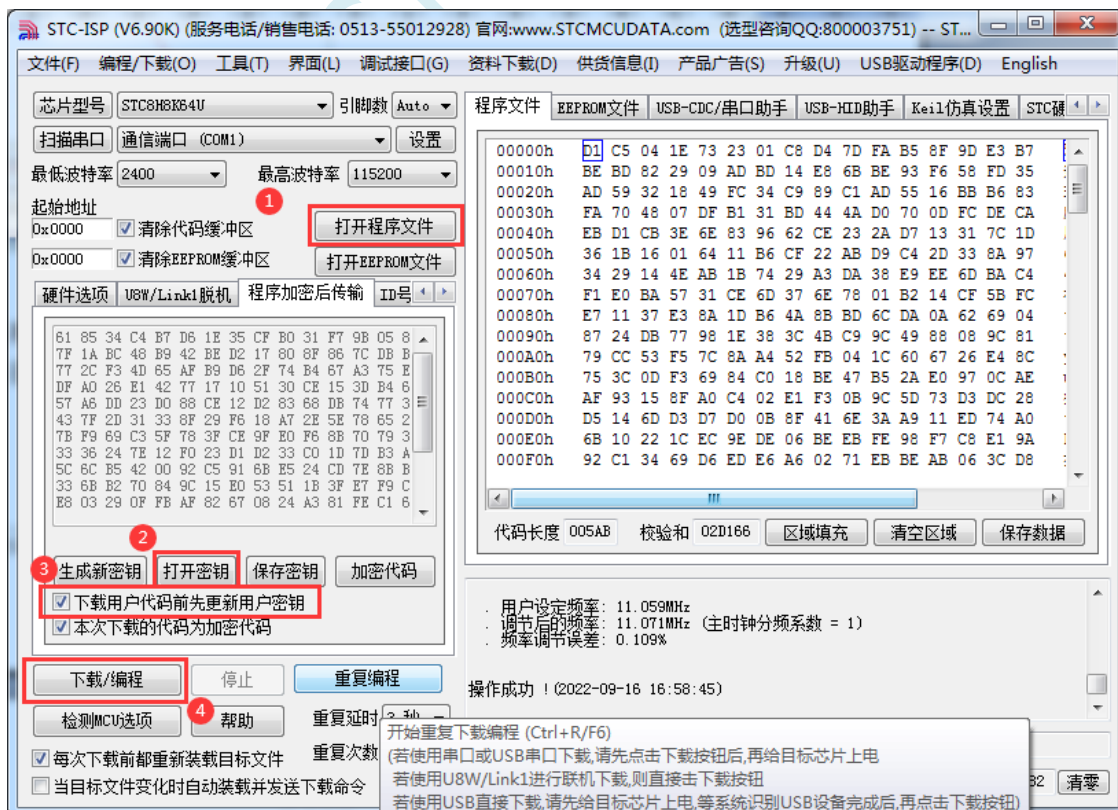
点击打开按钮后，马上会有会弹出一个类似的对话框，但此时是对加密后的文件进行保存的对话框。如下图所示，点击保存按钮即可保存加密后的文件。





### 3、将用户密钥更新到目标芯片中

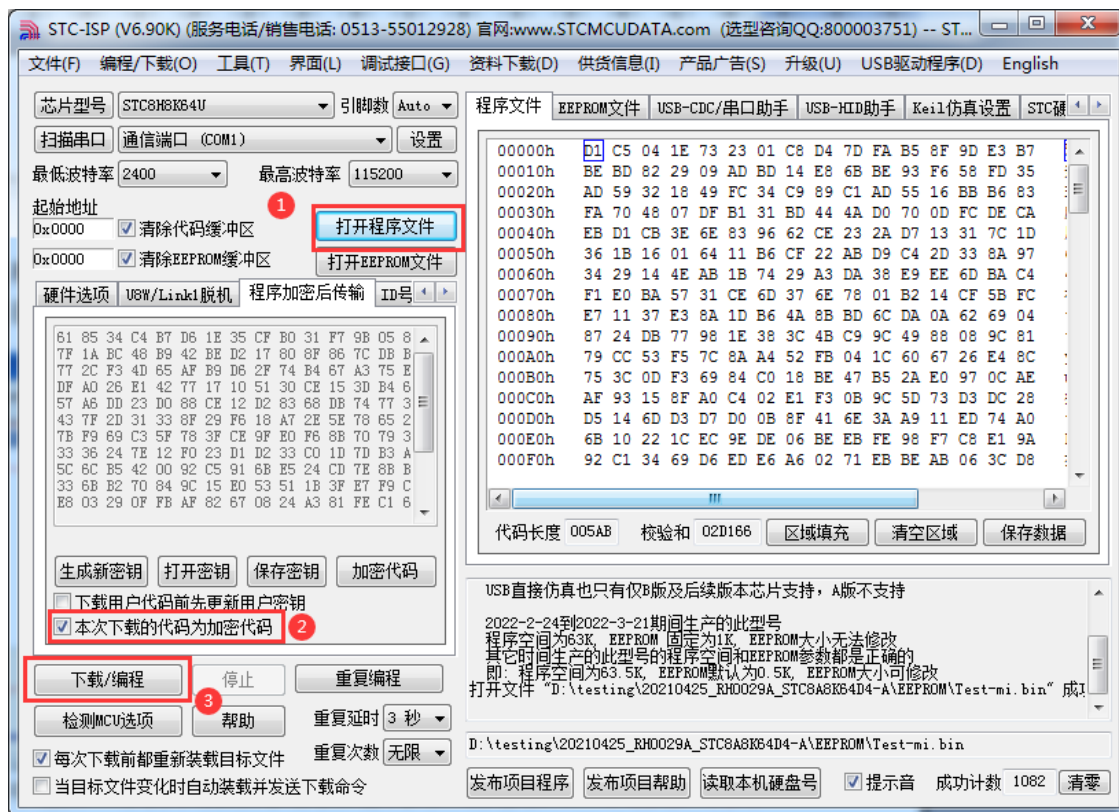
更新密钥前，需要先打开我们自己的密钥。若缓冲区中存放的已经是我们的密钥，则不要再打开。如下图，在“自定义加密下载”页面中点击“打开密钥”按钮，打开我们之前保存的密钥文件，例如“New.k”。密钥打开后，如下图所示，勾选上“下载用户代码前先更新用户密钥”选项和“本次下载的代码为加密码码”的选项，然后打开我们之前加密过后的文件，打开后点击界面左下角的“下载/编程”按钮，按正常方式对目标芯片下载完成即可更新用户密钥。





#### 4、加密更新用户代码

密钥更新成功后, 目标芯片便具有接收加密代码并还原的功能。此时若需要再次升级/更新代码, 则只需要参考第二步的方法, 将目标代码进行加密, 然后如下图



对于一片新的 STC 单片机, 可将步骤 3 和步骤 4 合并完成, 即将密钥更新到目标单片机的同时也可将加密后的代码一并下载到单片机中, 若已经执行过步骤 3 (即已经将密钥更新到目标芯片中了), 则后续的代码更新就只需要按照步骤 4, 只需要在“程序加密后传输”页面中选择“本次下载的代码为加密代码”的选项 (“下载用户代码前先更新用户密钥”选项不需要选了), 然后打开我们之前加过密后的文件, 打开后点击界面左下角的“下载/编程”按钮, 按正常方式对目标芯片下载即可完成用用户自己专用的加密文件更新用户代码的目的 (防止在烧录程序时被烧录人员通过监测串口分析出代码的目的)。

## B.3 发布项目程序+程序加密后传输结合使用

发布项目程序与程序加密后传输两项新的特殊功能可以结合在一起使用。首先程序加密后传输可以确保用户代码在烧录编程时串口通信传输过程当中的保密性，而发布项目程序可实现让最终使用者远程升级功能（方案公司的人员不需要亲自到场）。所以两项功能结合起来使用，非常适用于方案公司/生产商在软件需要更新时，让最终使用者自己对终端产品进行软件更新的目的，又确保现场烧录人员无法通过串口分析出有用程序，强烈建议方案公司使用。

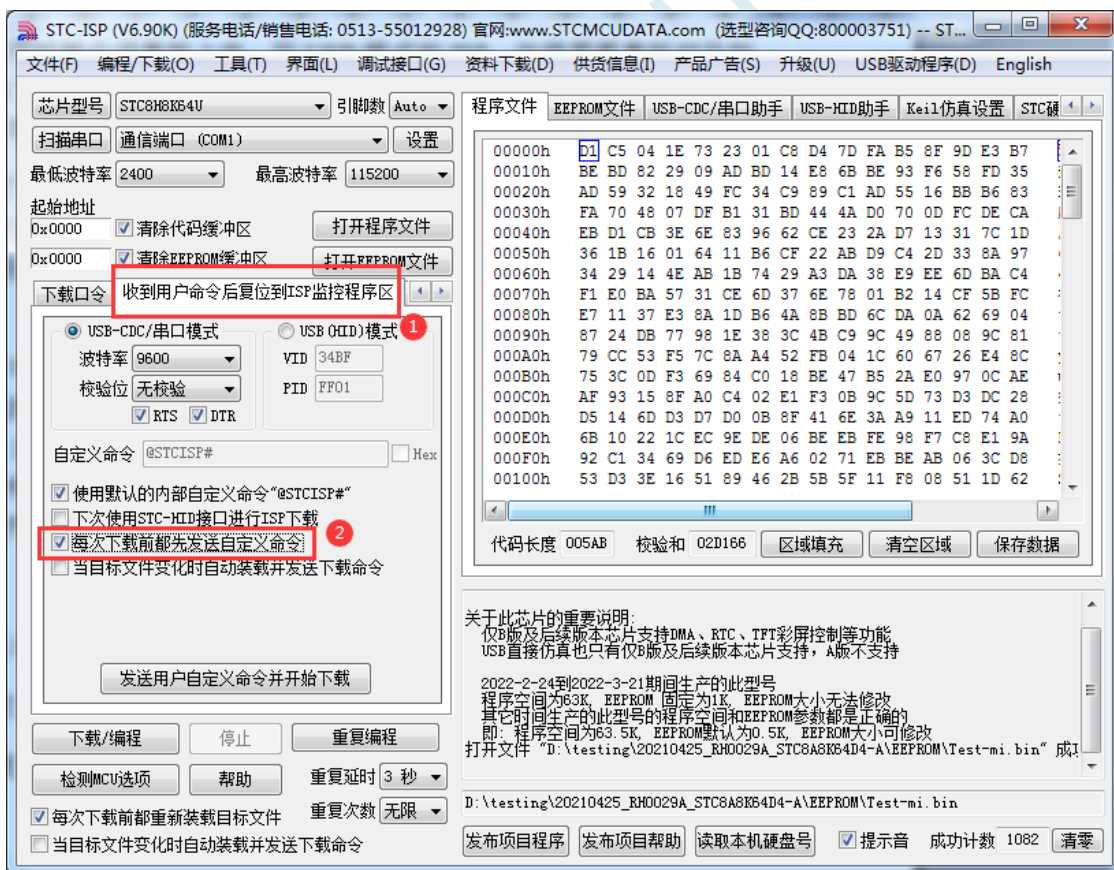
STC MCU

## B.4 用户自定义下载（实现不停电下载）

将用户的目标程序下载到STC单片机是通过执行单片机内部的ISP系统代码和上位机进行串口或者USB通讯来实现的。但STC单片机内部的ISP系统代码只有在每次重新停电再上电时才会被执行，这就要求用户每次需要对目标单片机更新程序时必须重新上电，而USB模式的ISP，处理需要重新对目标芯片上电外，还需要在上电时将P3.2口下拉到GND。对于处于开发阶段的项目，需要频繁的修改代码、更新代码，每次下载都需要重新上电会导致操作非常麻烦。

STC单片机在硬件设计时，增加了一个软复位寄存器（IAP\_CONTR），让用户可以通过设置此寄存器来决定CPU复位后重新执行用户代码还是复位到ISP区执行ISP系统代码。当向IAP\_CONTR寄存器写入0x20时，CPU复位后重新执行用户代码；当向IAP\_CONTR寄存器写入0x60时，CPU复位后复位到ISP区执行ISP系统代码。

要实现不停电进行ISP下载，用户可以在程序中设计一段代码，例如检测一个特殊的按键、或者监控串口等待一个特殊的串口命令，当检测到满足下载条件时，就通过软件触发软复位寄存器复位到ISP区执行ISP系统代码，从而实现不停电ISP下载。当触发条件是外部按键时，则在用户代码中实时监控按键状态即可。若要实现STC-ISP软件和用户触发软复位完全同步，则需要使用STC-ISP软件中所提供的“收到用户命令后复位到ISP监控程序区”这个功能。



实现不停电 ISP 下载的步骤如下:

1、编写用户代码,并在用户代码中添加串口命令监控程序

(参考代码如下,测试单片机型号为 STC8H8K64U)

```
#include "stc8h.h"

#define FOSC 11059200UL
#define BAUD (65536 - (FOSC/115200+2)/4) //加 2 操作是为了让 Keil 编译器
//自动实现四舍五入运算

char code *STCISPCMD = "@STCISP#"; //自定义下载命令
char index;

void uart_isr() interrupt 4
{
    char dat;

    if (TI)
    {
        TI = 0;
    }

    if (RI)
    {
        RI = 0;
        dat = SBUF; //接收串口数据

        if (dat == STCISPCMD[index]) //判断接收的数据和当前的命令字符是否匹配
        {
            index++; //若匹配则索引+1
            if (STCISPCMD[index] == '\0') //判断命令是否配完成
                IAP_CONTR = 0x60; //若匹配完成则软复位到 ISP
        }
        else
        {
            index = 0; //若不匹配,则需要从头开始
            if (dat == STCISPCMD[index])
                index++;
        }
    }
}

void main()
{
```

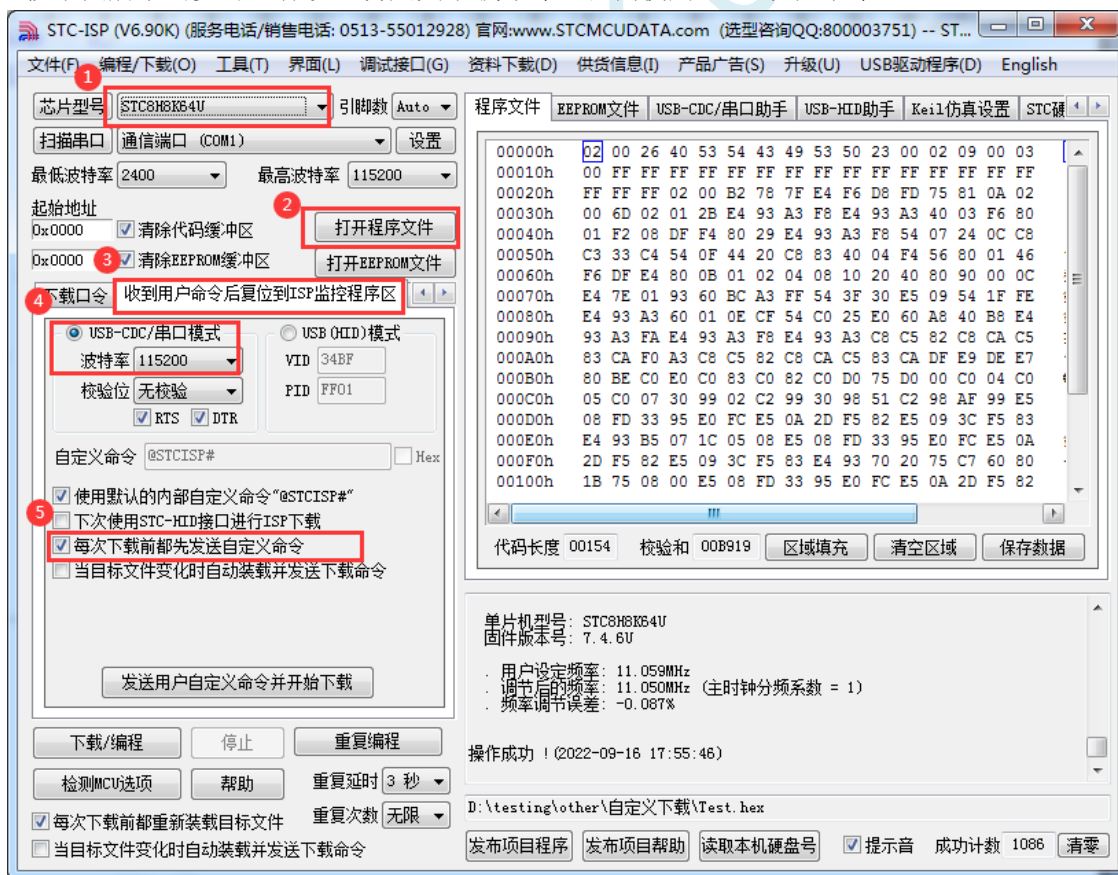
```
POM0 = 0x00; P0M1 = 0x00;  
P1M0 = 0x00; P1M1 = 0x00;  
P2M0 = 0x00; P2M1 = 0x00;  
P3M0 = 0x00; P3M1 = 0x00;
```

```
SCON = 0x50; //串口初始化  
AUXR = 0x40;  
TMOD = 0x00;  
TH1 = BAUD >> 8;  
TL1 = BAUD;  
TR1 = 1;  
ES = 1;  
EA = 1;
```

```
index = 0; //初始化命令
```

```
while (1);  
}
```

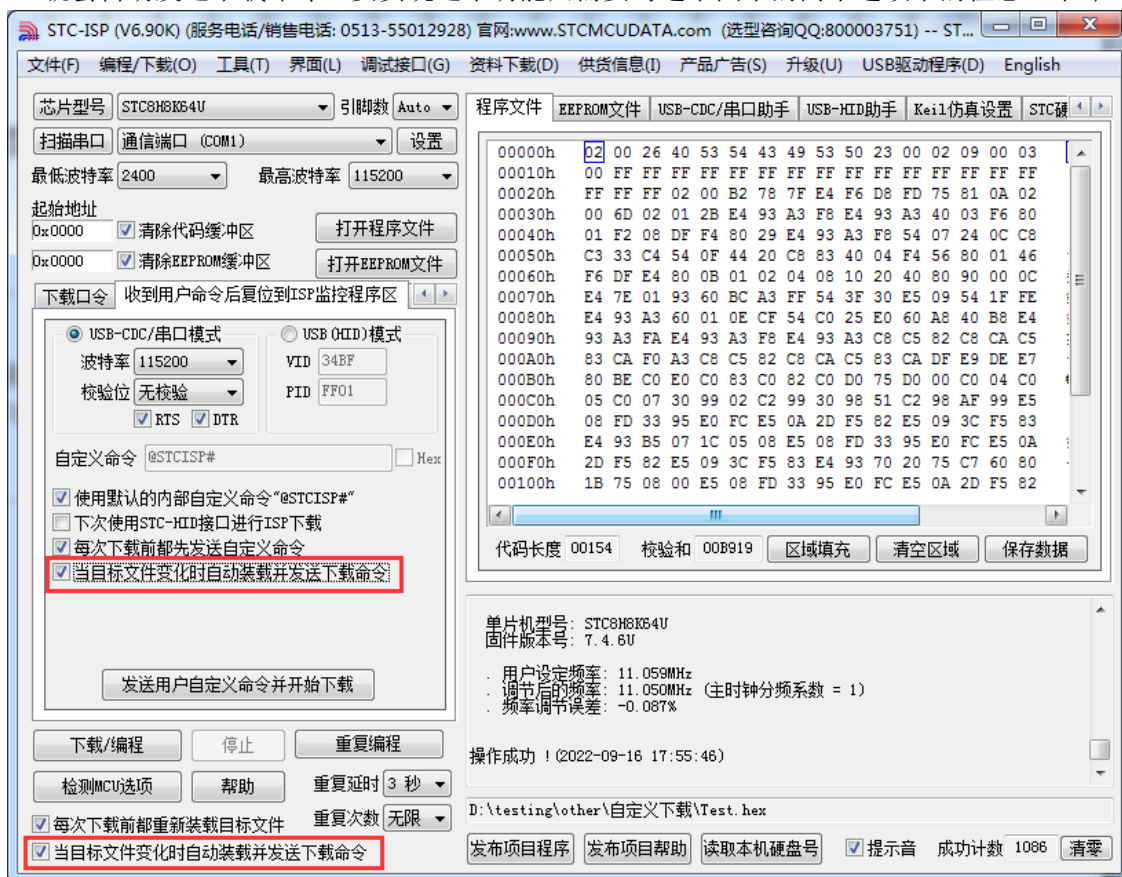
2、按下图所示的步骤进行设置自定义下载命令（范例使用 STC 默认命令 “@STCISP#”）



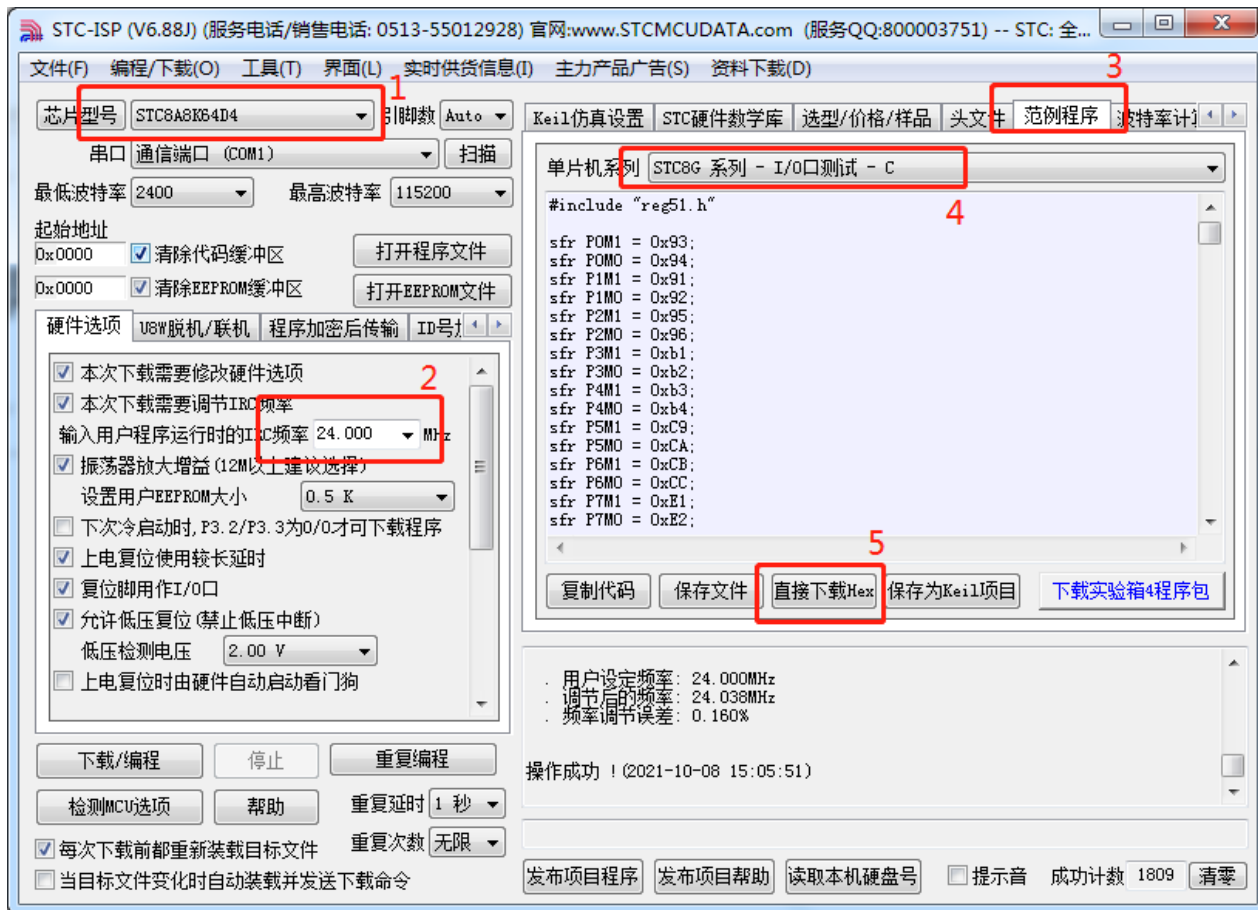
3、第一次下载时需要为目标单片机重新上电，之后的每次更新只需要点击下载软件中的“下载/编程”按钮，下载软件自动将下载命令发送给目标单片机，目标单片机接收到命令后自动复位到系统 ISP 区，即可实现不停电更新用户代码。



4、STC-ISP 还可实现项目开发阶段，完全自动下载功能，即当下载软件侦测到目标代码被更新了，就会自动发送下载命令。要实现这个功能只需要勾选下图中的两个选项中的任意一个即可



## 附录C 如何测试 I/O 口



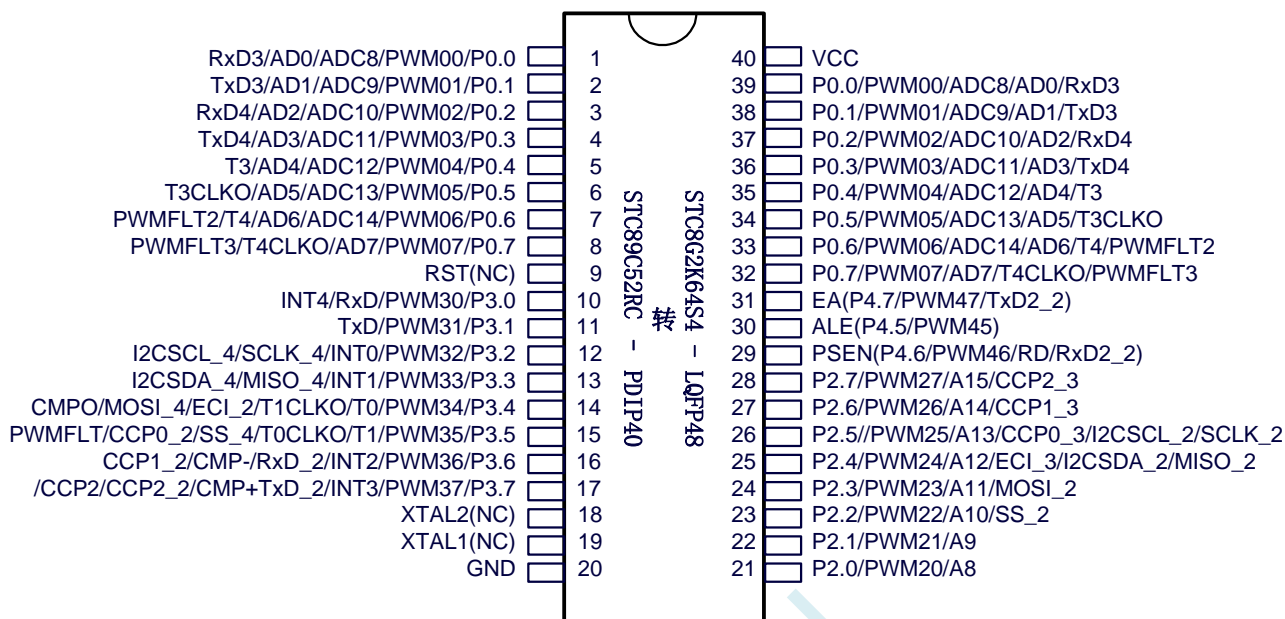
测试 I/O 口步骤:

- 1、选择单片机型号
- 2、设置测试程序的工作频率 (24MHz)
- 3、打开“范例程序”页
- 4、选择 STC8G 或者 STC8H 系列中的“I/O 口测试”程序
- 5、点击页面中的“直接下载 Hex”

下载完成后, 会对所有的 I/O 口执行流水灯程序, 此时可在 I/O 口接 LED 或者用示波器即可看到波形。



下图为转换板功能示意图



#### 注意:

- ✓ 由于内置高精度 R/C 时钟，因此不需要外部晶振，XTAL1 和 XTAL2 是空的
- ✓ WR 和 RD 是 (WR/P4.2 和 RD/P4.4)，而不是传统的 (WR/P3.6 和 RD/P3.7)。  
(转换板中，P4.2 与 P3.6 连接在一起，P4.4 与 P3.7 连接在一起。当用户需要用此转换板访问外部总线时，需要将 P3.6 和 P3.7 设置为高阻输入模式，从而使 P4.2 和 P4.4 正常输出总线读写信号；若不需要访问外部总线，则需将 P4.2 和 P4.4 设置高阻输入模式，3.6 和 P3.7 即为普通 I/O。)
- ✓ 由于 STC8G 系列 MCU 是低电平复位，与传统 8051 的高电平复位不兼容，因此 RST 管脚是悬空，而用转换板上的复位按键加复位电路取代

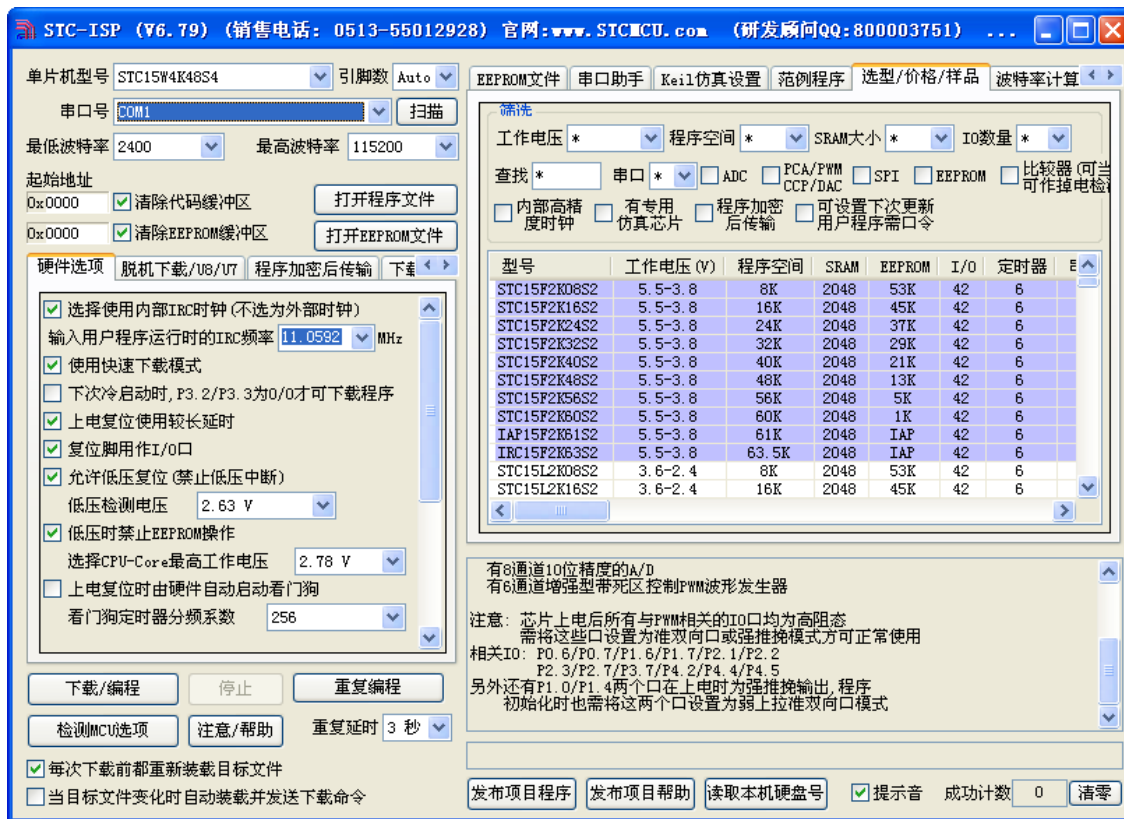
## 附录E STC 软件模拟硬件 USB 设备驱动程序安装说明

### 安装前必须先仔细阅读

- 1、如果使用 STC 的硬件 USB 进行 ISP 直接下载，则**不需要安装任何驱动**，可直接下载。目前支持硬件 USB 进行 ISP 直接下载的单片机有 **STC32G12K128** 系列、**STC8H8K64U** 系列。这些系列下载**不需要安装任何驱动**。
- 2、如果使用 STC 的软件模拟硬件 USB 进行 ISP 直接下载，请按照本小节中介绍的步骤安装 STC-ISP 软件包中“STC-USB Driver”目录里面的驱动。**(不建议使用)**
- 3、如果使用 STC-USB Link1 进行 ISP 下载，**不需要安装任何驱动**，可直接连接到目标芯片进行 ISP 下载。
- 4、如果使用 STC 的 U8W/U8W-Mini 工具进行在线下载，请安装 STC-ISP 软件包中“USB to UART Driver\CH340\_CH341\CH341SER.EXE”目录里面的驱动。
- 5、如果使用淘宝上购买的 USB 转串口工具进行 ISP 下载，则必须先正确安装驱动后才能正常使用，如果是 CH340，可安装 STC-ISP 软件包中“USB to UART Driver\CH340\_CH341\CH341SER.EXE”；如果是 PL2303，可安装软件包中“USB to UART Driver\PL2303\PL23XX\_v200.exe”；如果是其他型号的 USB 转串口，请自行下载驱动进行安装

## Windows XP 安装方法

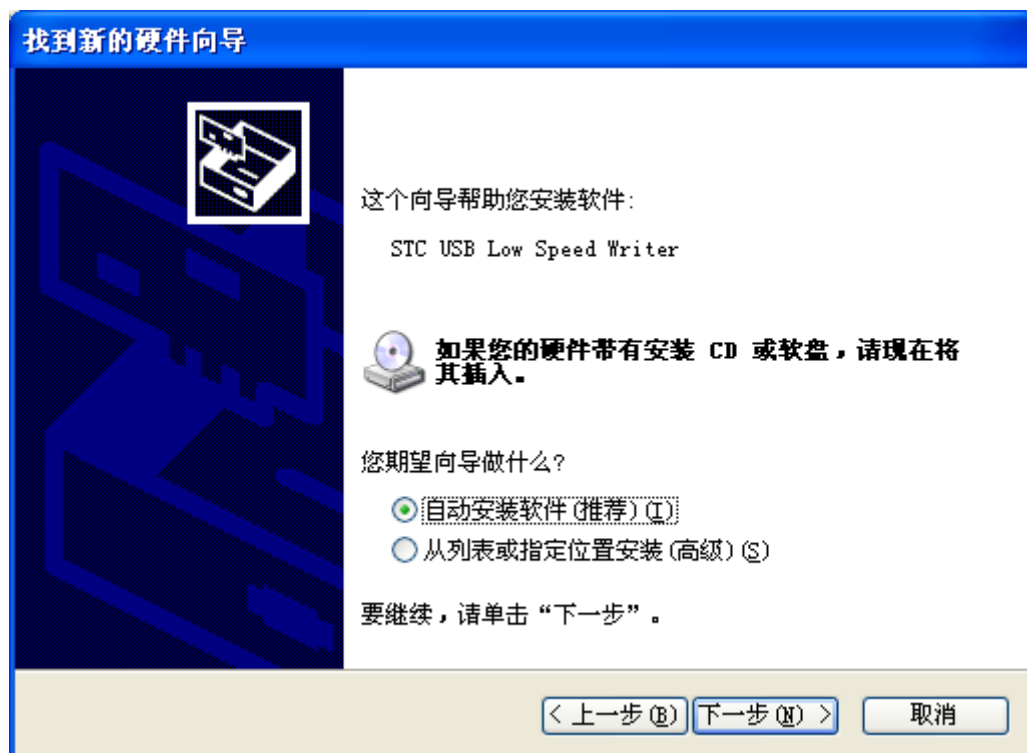
打开 V6.79 版（或者更新的版本）的 STC-ISP 下载软件，下载软件会自动将驱动文件复制到相关的系统目录



插入 USB 设备，系统找到设备后自动弹出如下对话框，选择其中的“否，暂时不”项

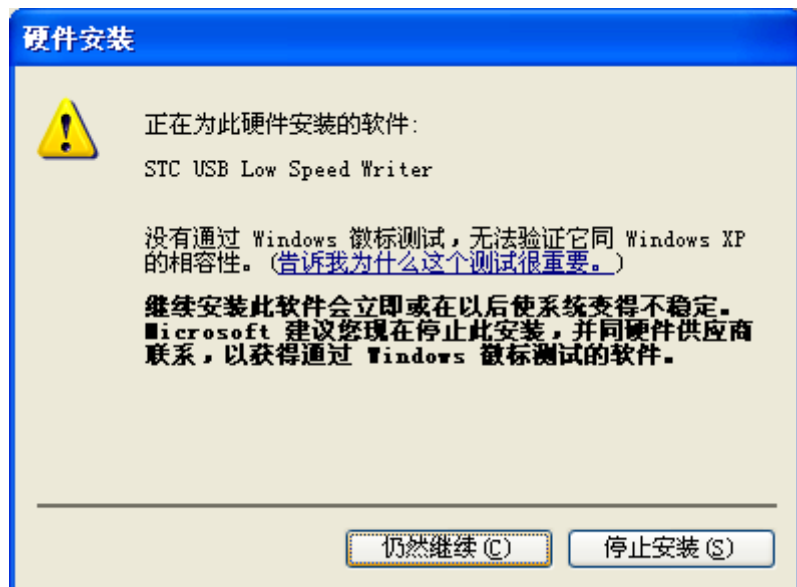


在下面的对话框中选择“自动安装软件(推荐)”项





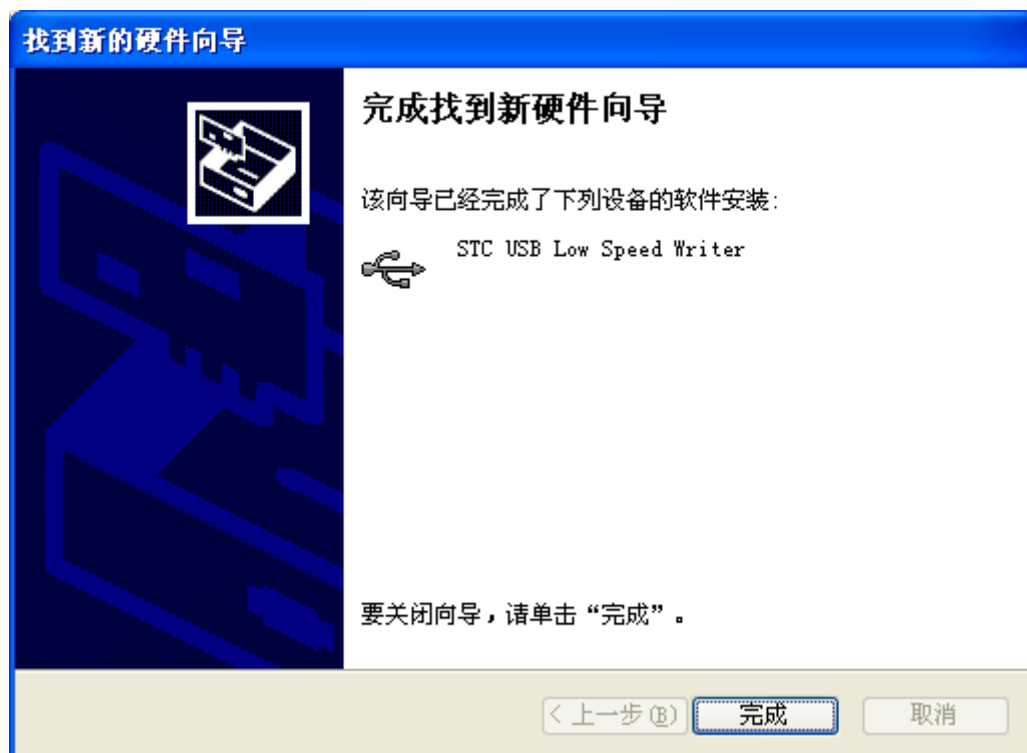
在弹出的下列对话框中, 选择“仍然继续”按钮



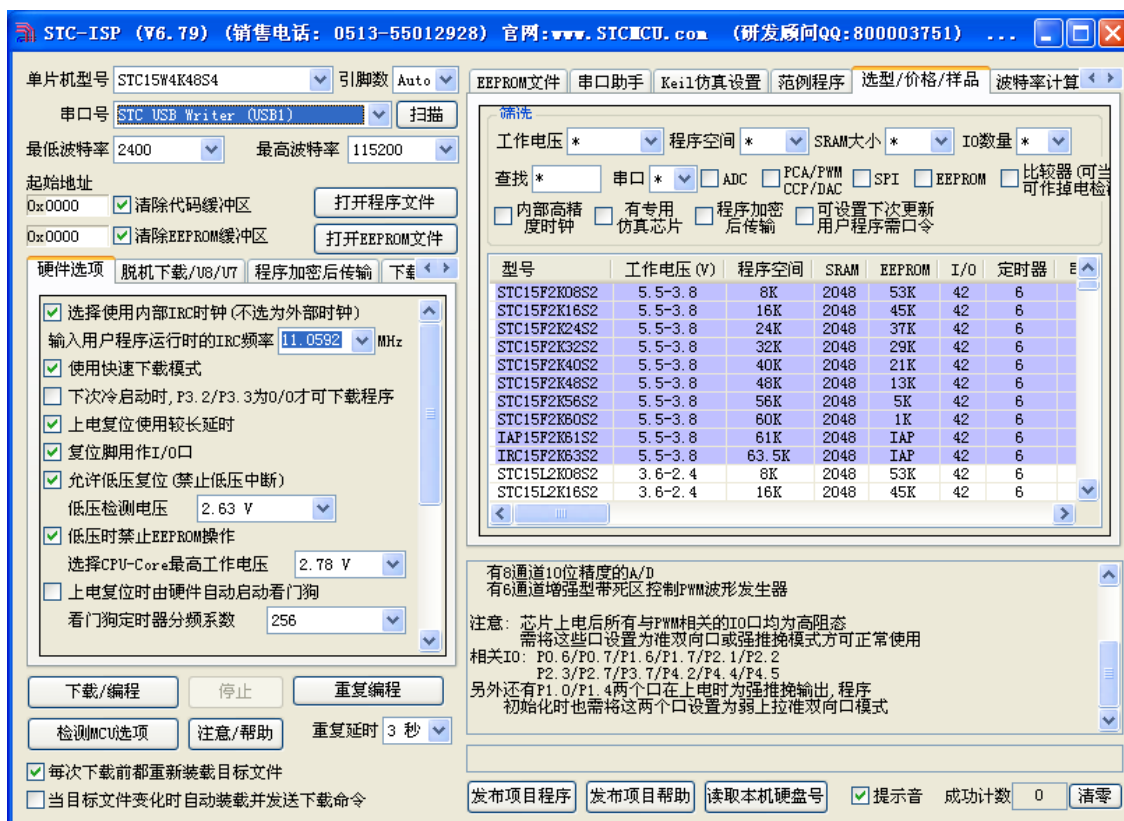
接下系统会自动安装驱动，如下图



出现下面的对话框表示驱动安装完成



此时, 之前打开的 STC-ISP 下载软件中的串口号列表会自动选择所插入的 USB 设备, 并显示设备名称为“STC USB Writer (USB1)”, 如下图:



## Windows 7 (32 位) 安装方法

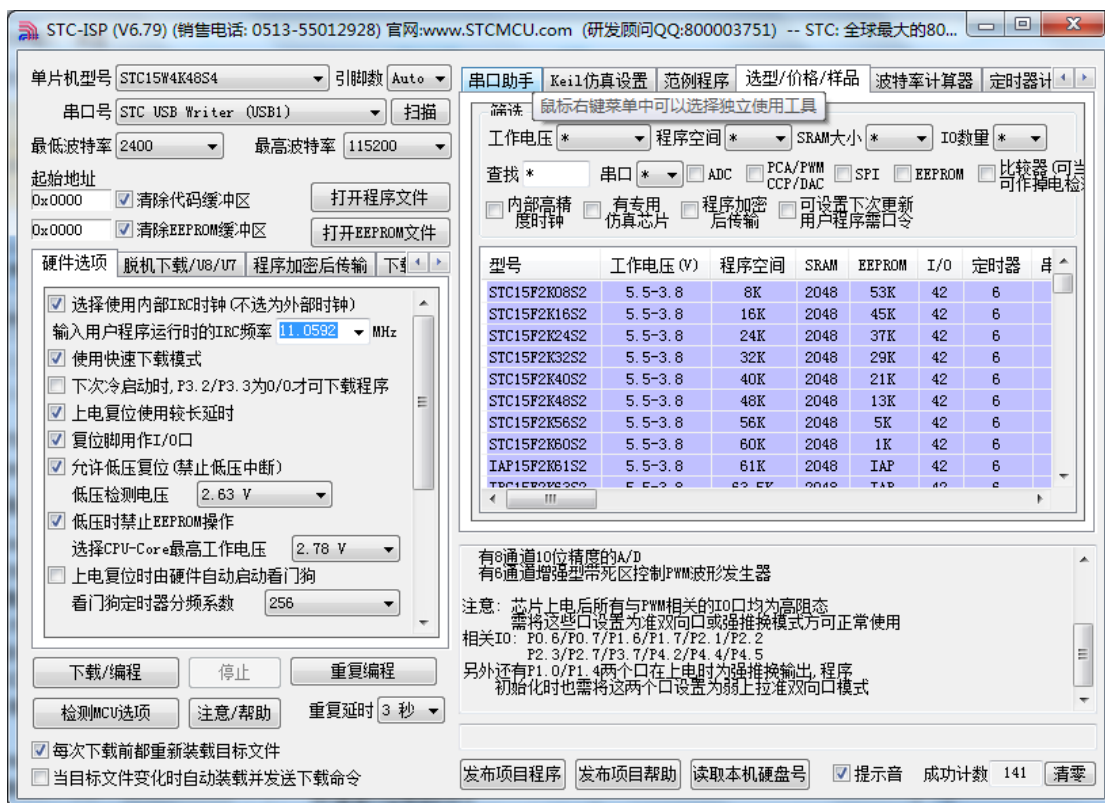
打开 V6.79 版（或者更新的版本）的 STC-ISP 下载软件，下载软件会自动将驱动文件复制到相关的系统目录



插入 USB 设备, 系统找到设备后会自动安装驱动。安装完成后会有如下的提示框。



此时, 之前打开的 STC-ISP 下载软件中的串口号列表会自动选择所插入的 USB 设备, 并显示设备名称为“STC USB Writer (USB1)”, 如下图:

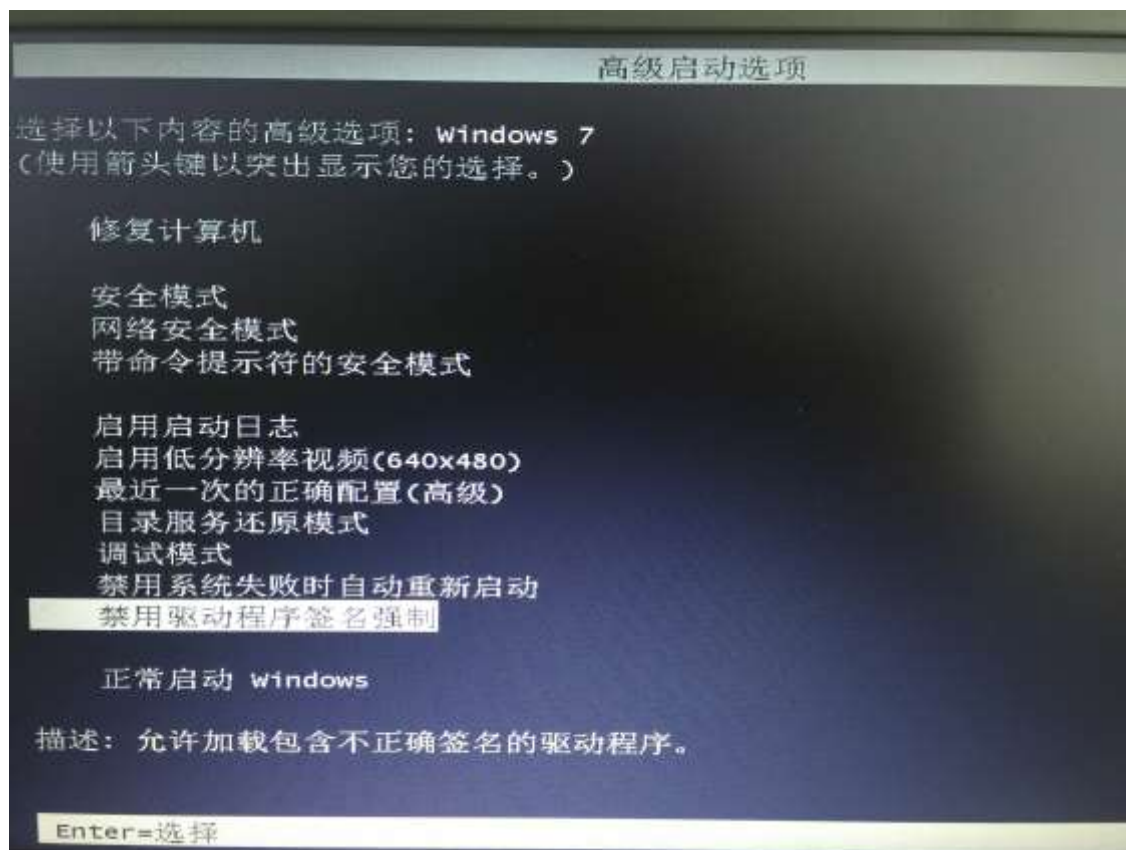


注: 若 Windows 7 下, 系统并没有自动安装驱动, 则驱动的安装方法请参考 [Windows 8 \(32 位\) 的安装方法](#)

## Windows 7 (64 位) 安装方法

由于 Windows7 64 位操作系统在默认状态下, 对于没有数字签名的驱动程序是不能安装成功的。所以在安装 STC-USB 驱动前, 需要按照如下步骤, 暂时跳过数字签名, 即可顺利安装成功。

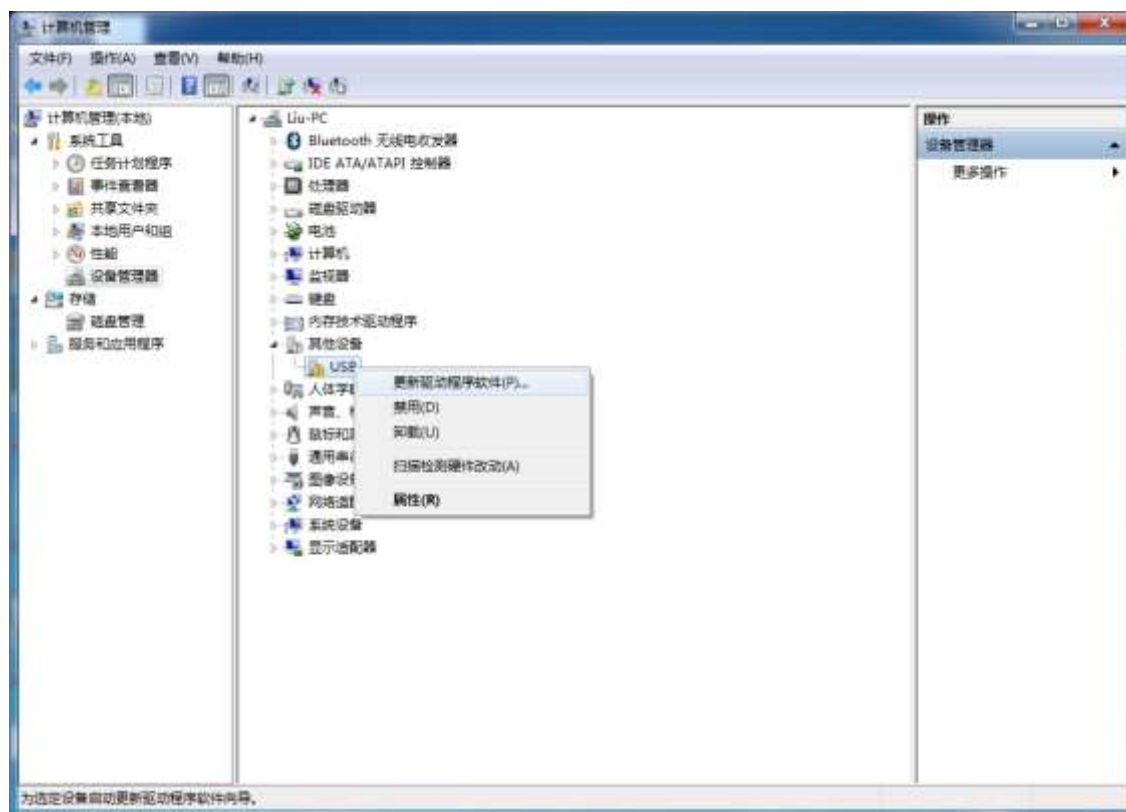
首先重启电脑, 并一直按住 F8, 直到出现下面启动画面



选择“禁用驱动程序签名强制”。启动后即可暂时关闭数字签名验证功能



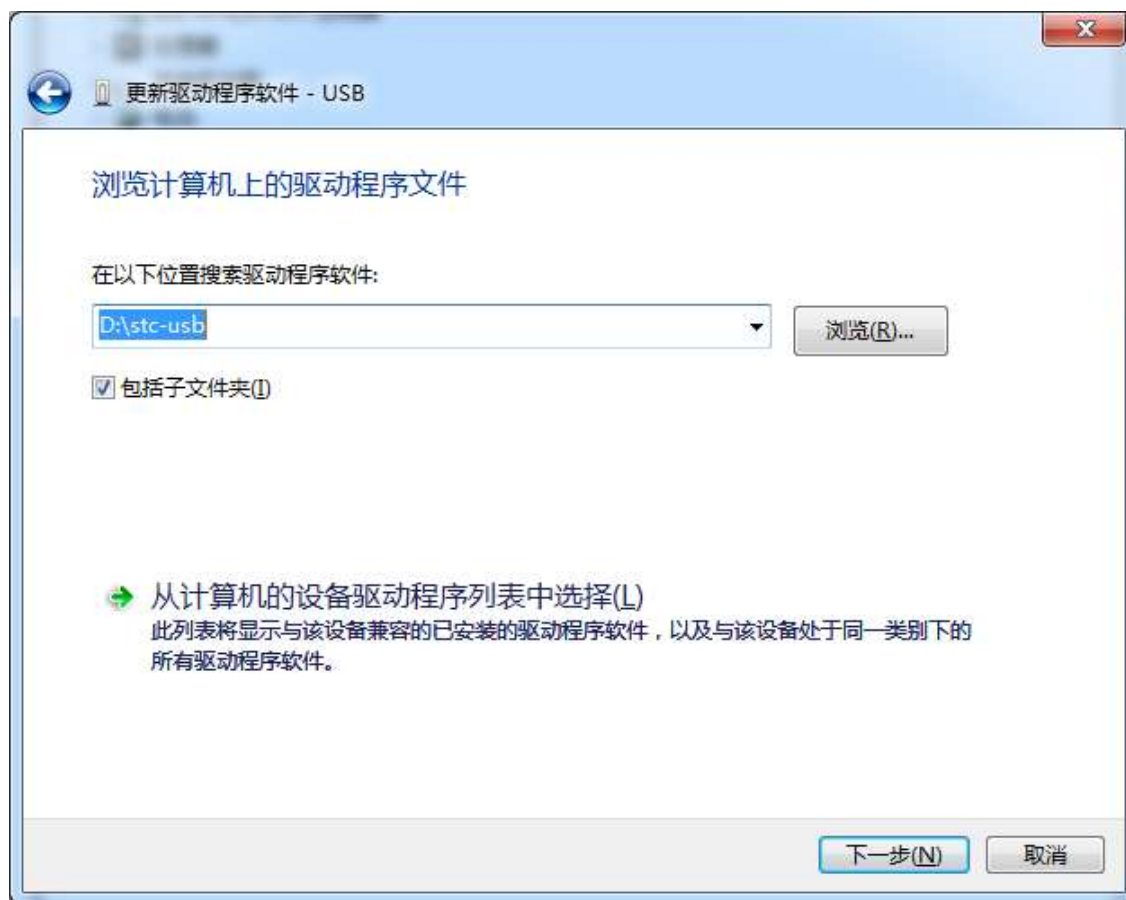
插入 USB 设备，并打开“设备管理器”。找到设备列表中带黄色感叹号的 USB 设备，在设备的右键菜单中，选择“更新驱动程序软件”



在下面的对话框中选择“浏览计算机以查找驱动程序软件”



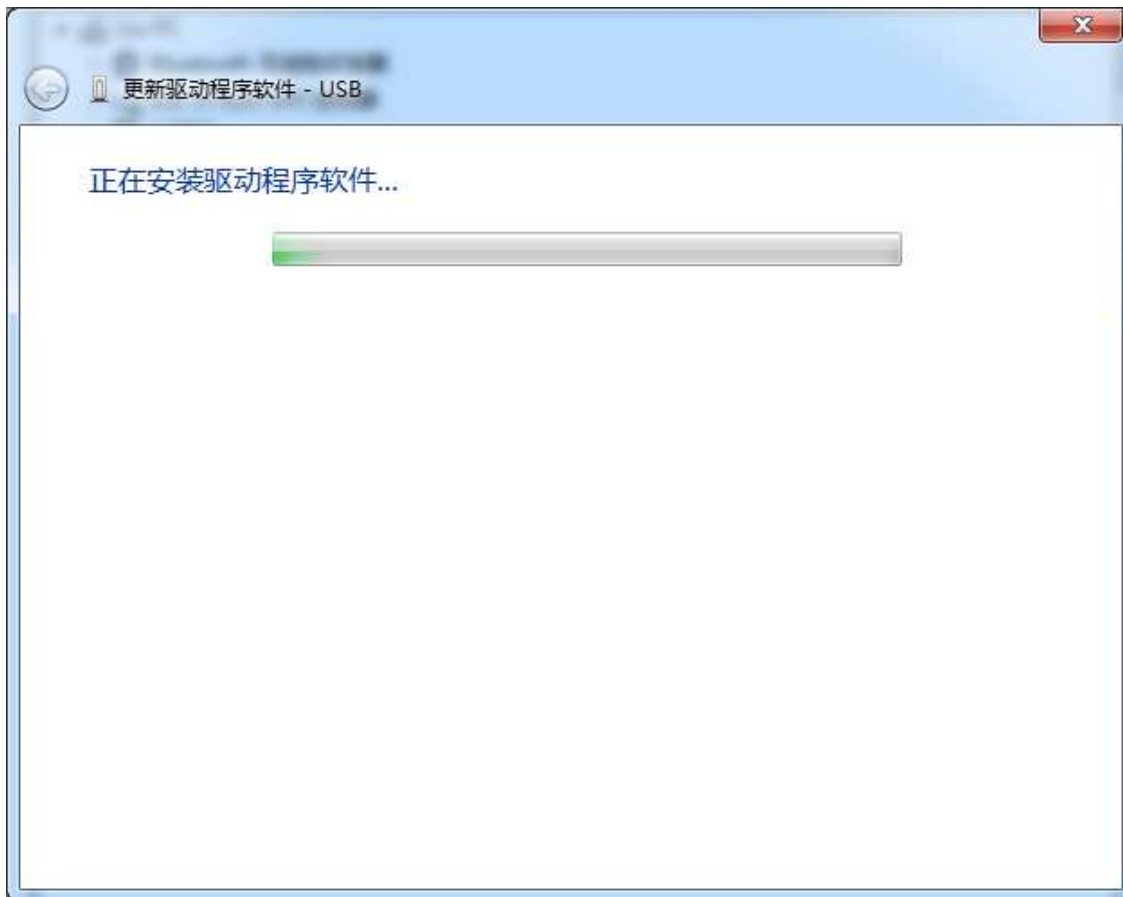
单击下面对话框中的“浏览”按钮，找到之前 STC-USB 驱动程序的存放目录（例如：之前的示例目录为 “ D:\STC-USB ”，用户将路径定位到实际的解压目录）



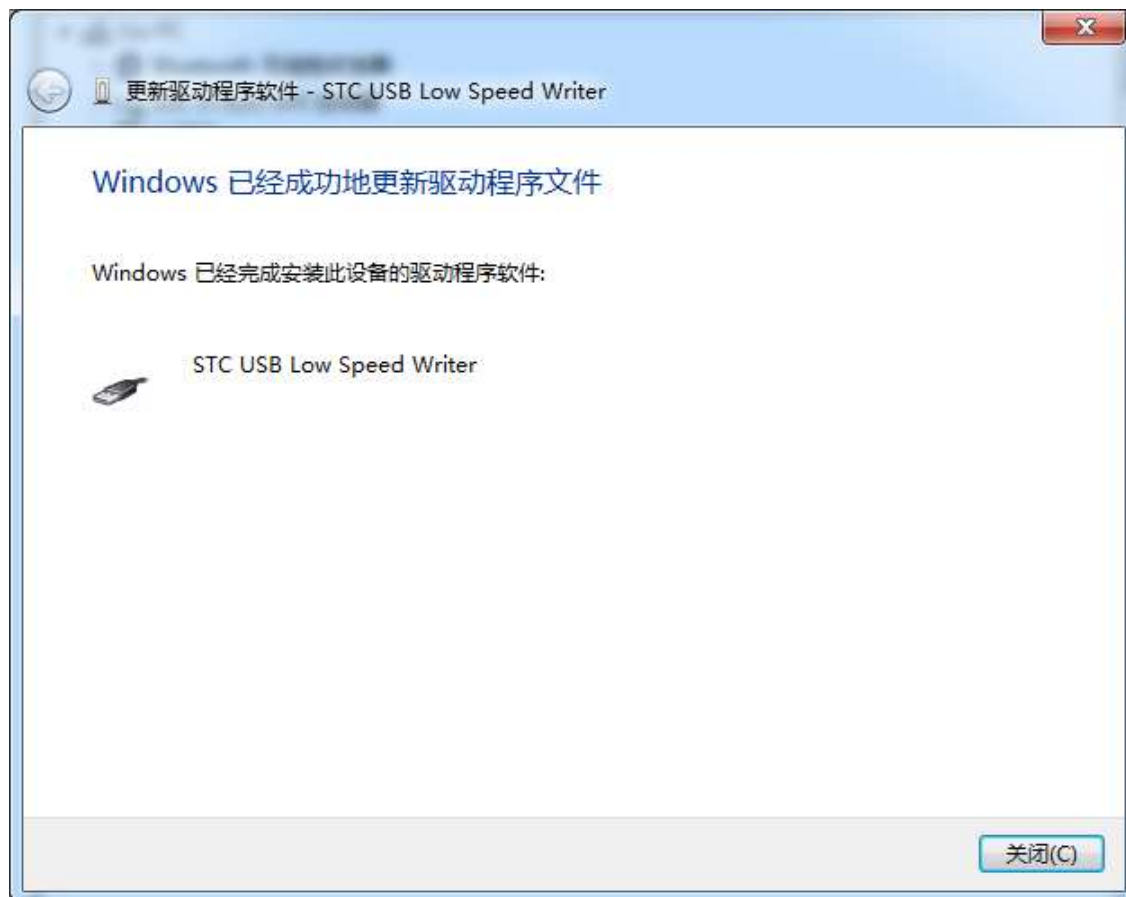
驱动程序开始安装时，会弹出如下对话框，选择“始终安装此驱动程序软件”



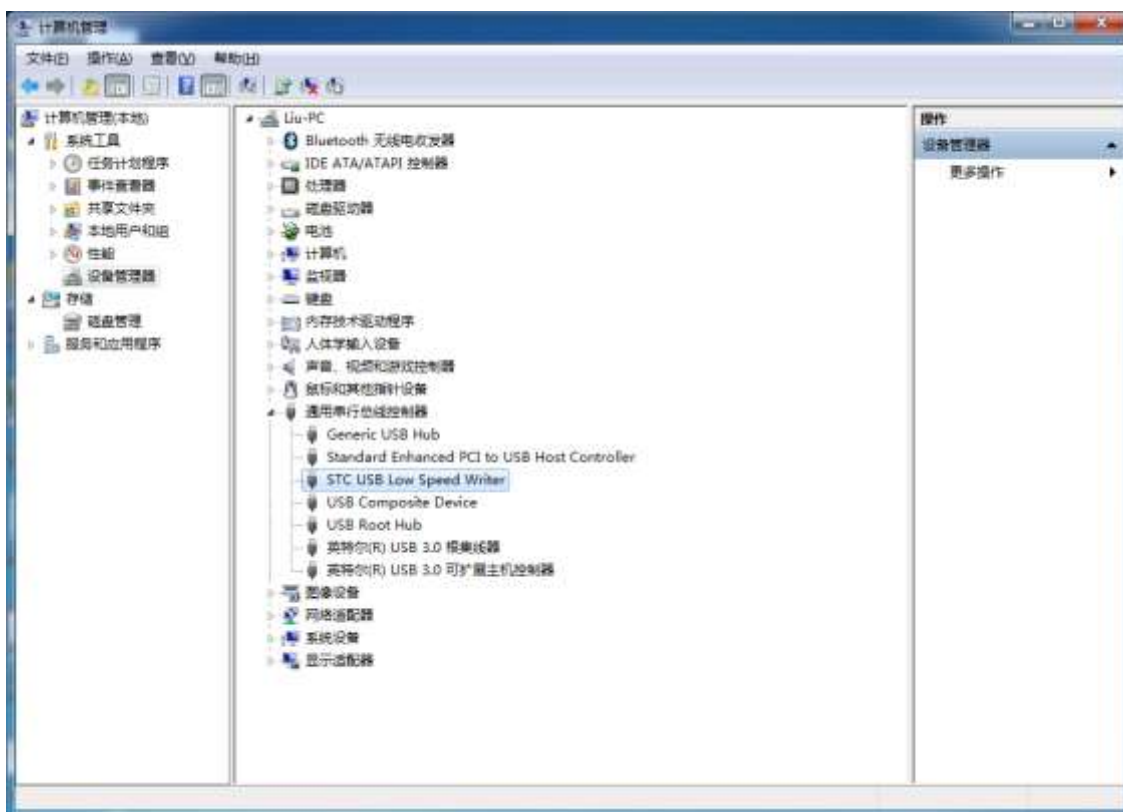
接下来，系统会自动安装驱动，如下图



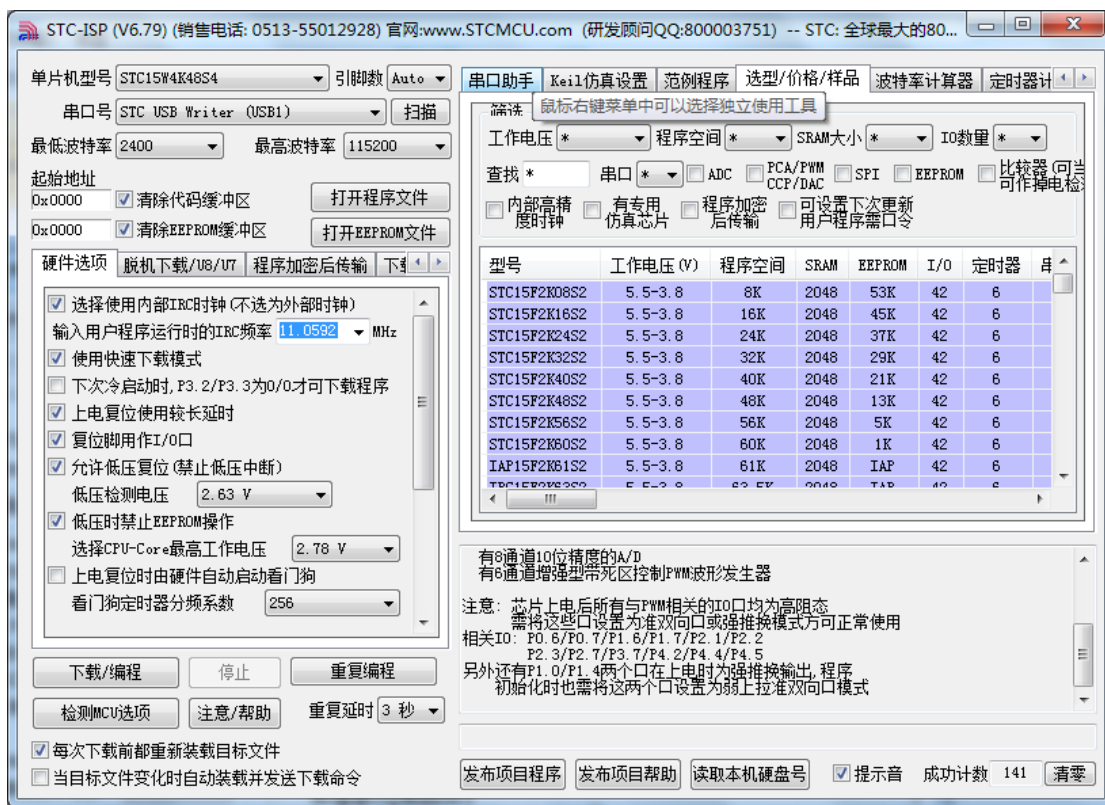
出现下面的对话框表示驱动安装完成



此时在设备管理器中，之前带有黄色感叹号的设备，此时会显示为“STC USB Low Speed Writer”的设备名



在之前打开的 STC-ISP 下载软件中的串口号列表会自动选择所插入的 USB 设备, 并显示设备名称为“STC USB Writer (USB1)”, 如下图:

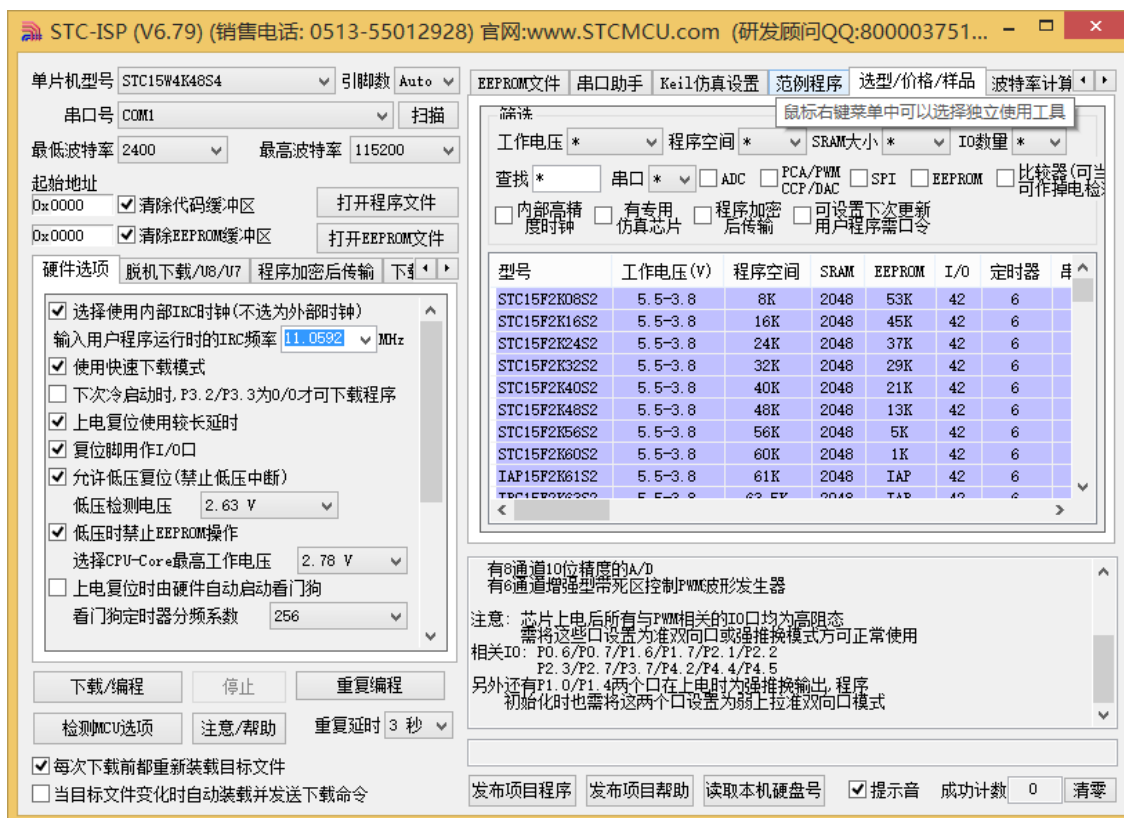




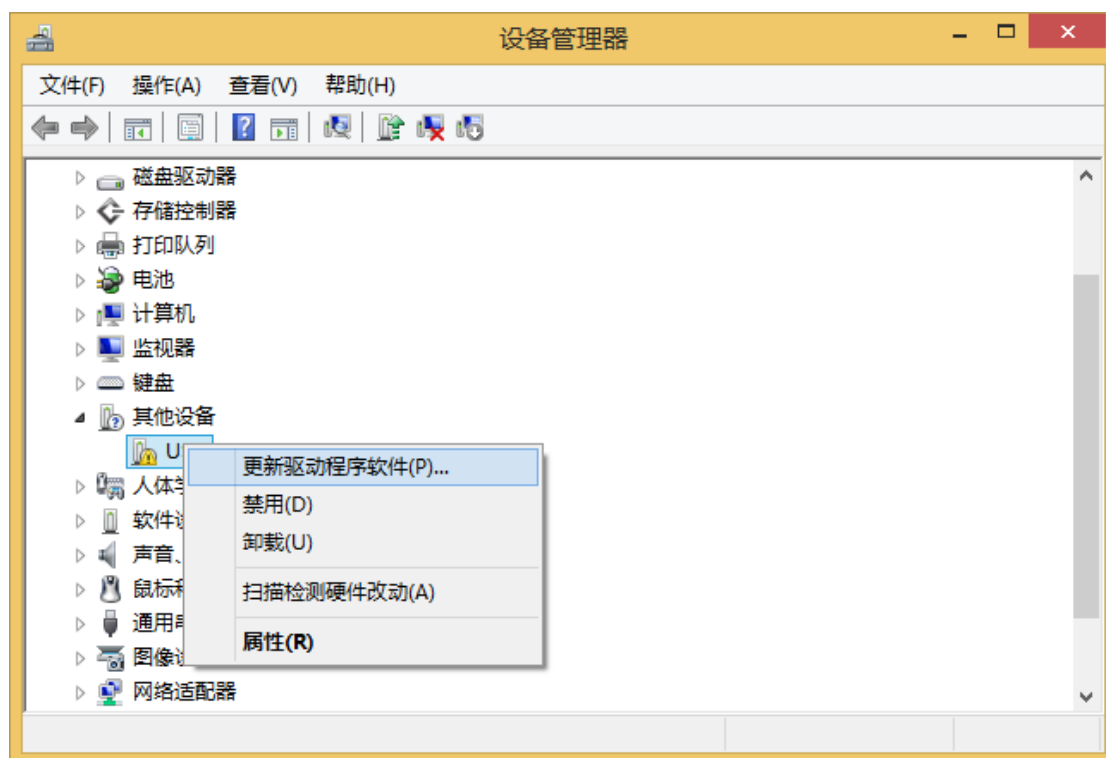
## Windows 8 (32 位) 安装方法

打开 V6.79 版 (或者更新的版本) 的 STC-ISP 下载软件 (由于权限的原因, 在 Windows 8 中下载软件不会将驱动文件复制到相关的系统目录, 需要用户手动安装。首先从 STC 官方网站下载

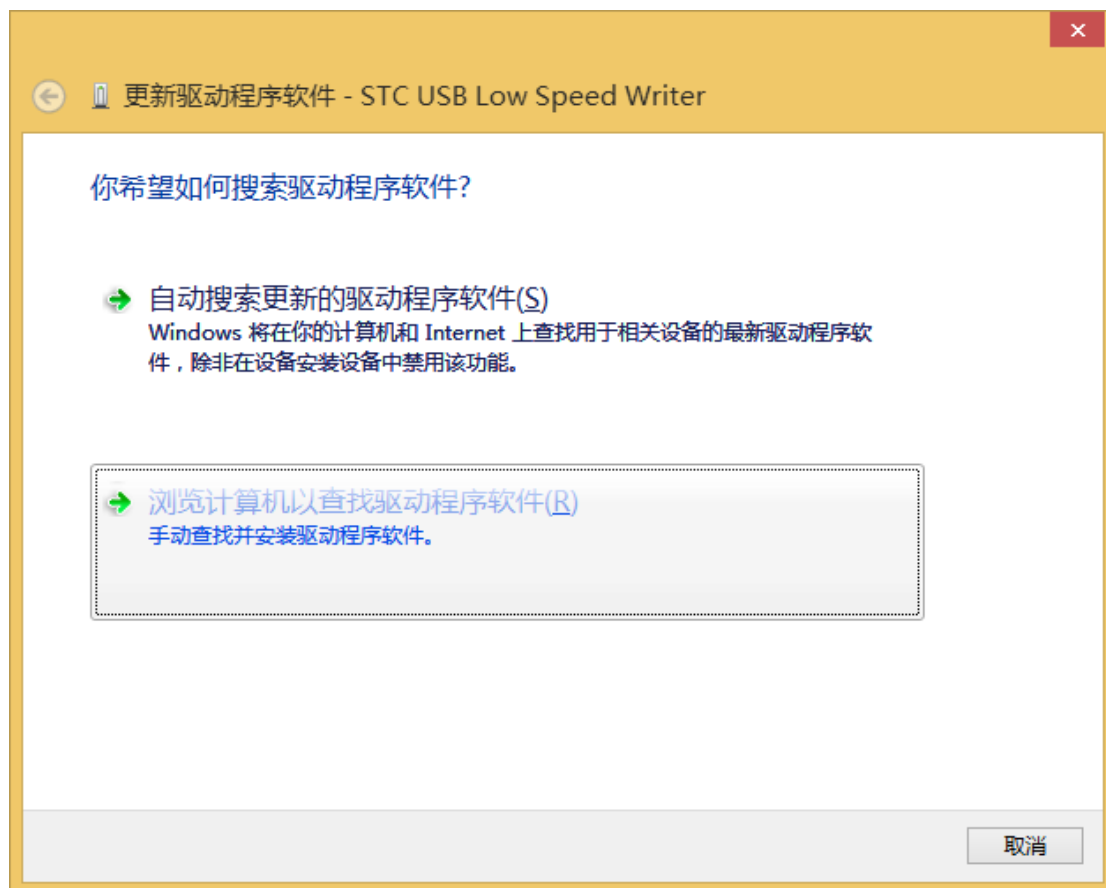
“stc-isp-15xx-v6.79.zip” (或更新版本), 下载后解压到本地磁盘, 则 STC-USB 的驱动文件也会被解压到当前解压目录中的“STC-USB Driver”中 (例如将下载的压缩文件“stc-isp-15xx-v6.79.zip”解压到“F:\”, 则 STC-USB 驱动程序在“F:\STC-USB Driver”目录中))



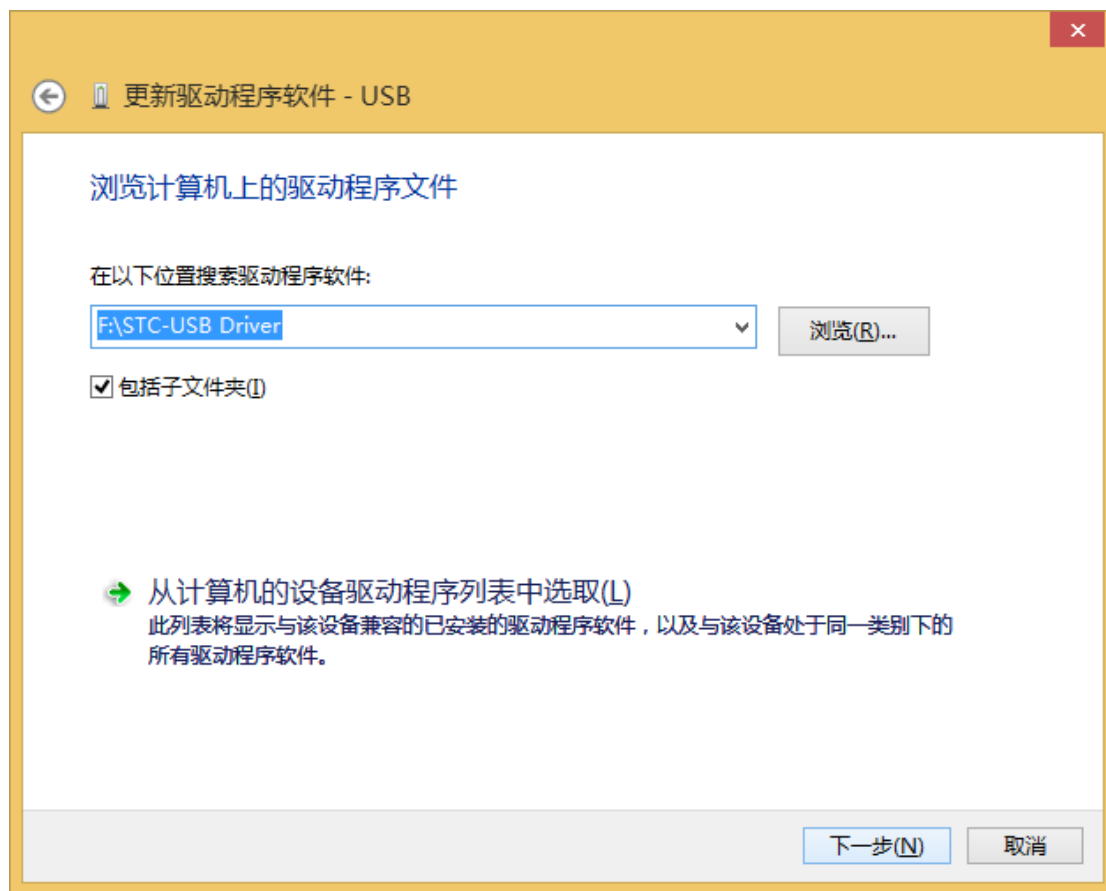
插入 USB 设备，并打开“设备管理器”。找到设备列表中带黄色感叹号的 USB 设备，在设备的右键菜单中，选择“更新驱动程序软件”



在下面的对话框中选择“浏览计算机以查找驱动程序软件”



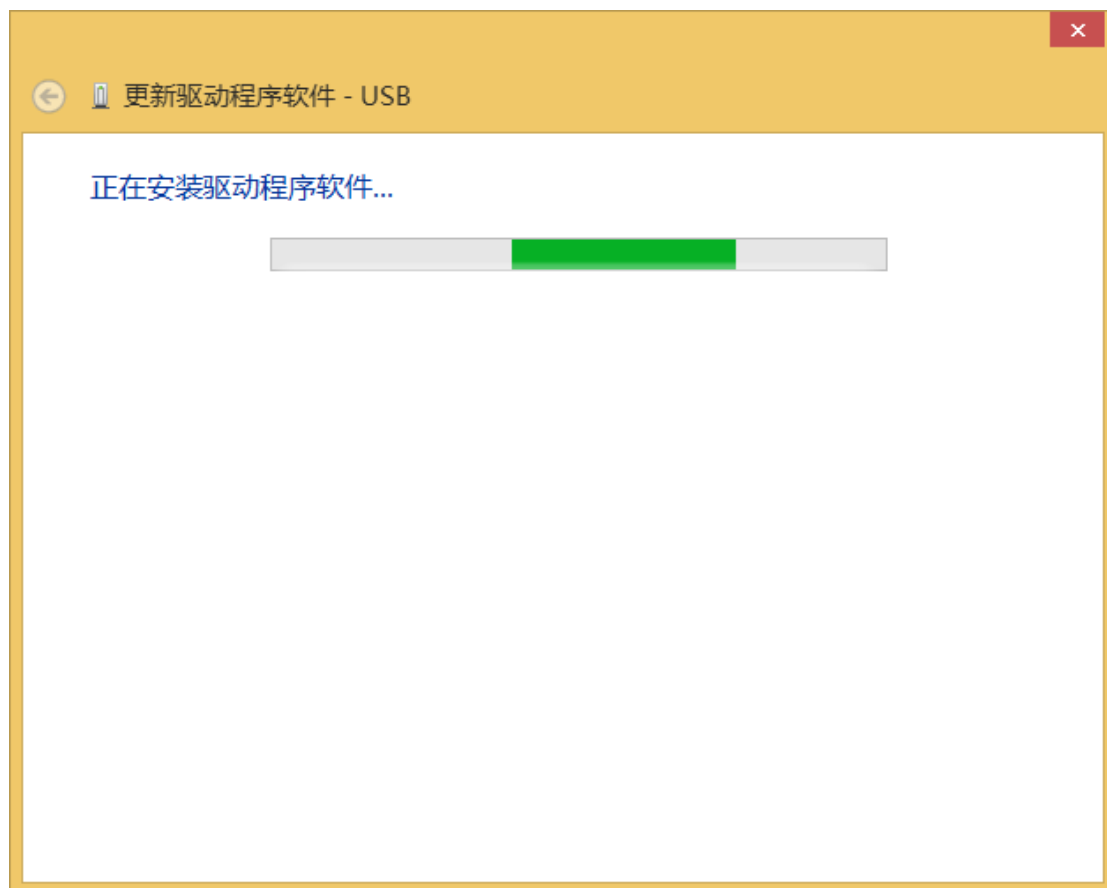
单击下面对话框中的“浏览”按钮，找到之前 STC-USB 驱动程序的存放目录（例如：之前的示例目录为“F:\STC-USB Driver”，用户将路径定位到实际的解压目录）



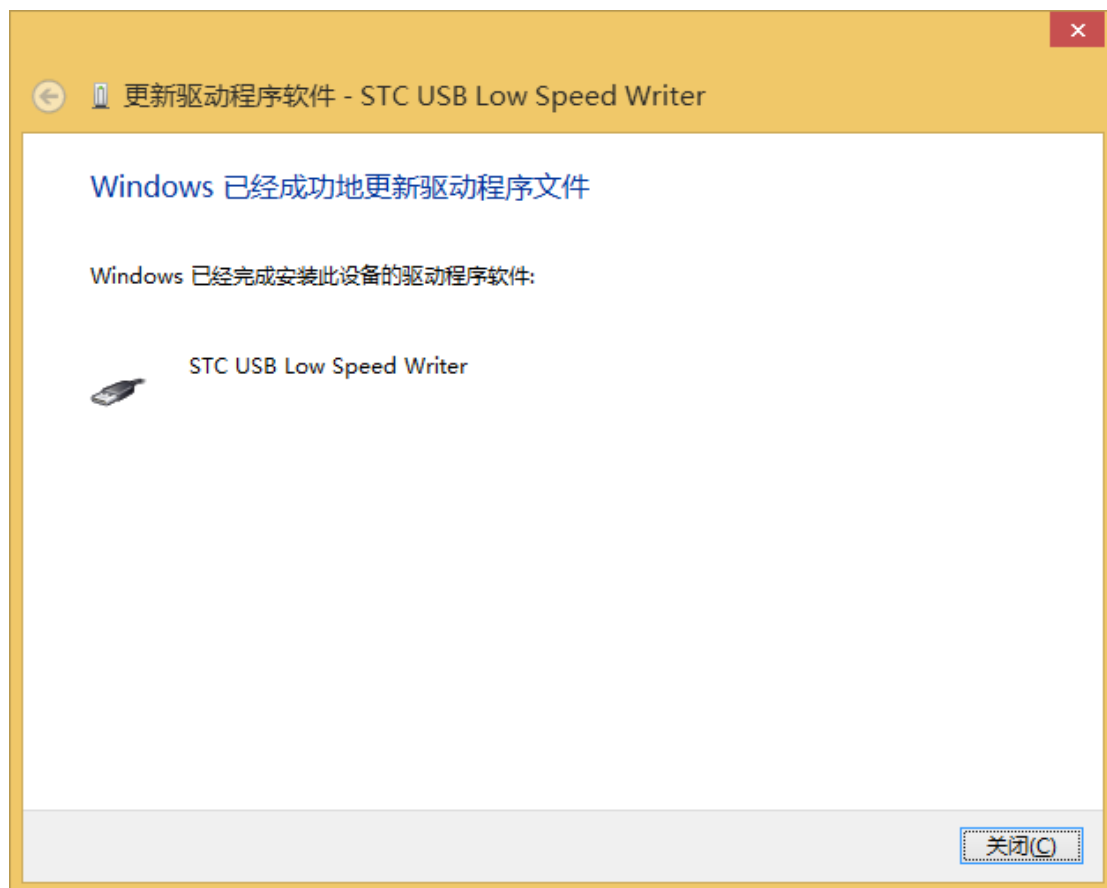
驱动程序开始安装时，会弹出如下对话框，选择“始终安装此驱动程序软件”



接下来，系统会自动安装驱动，如下图



出现下面的对话框表示驱动安装完成

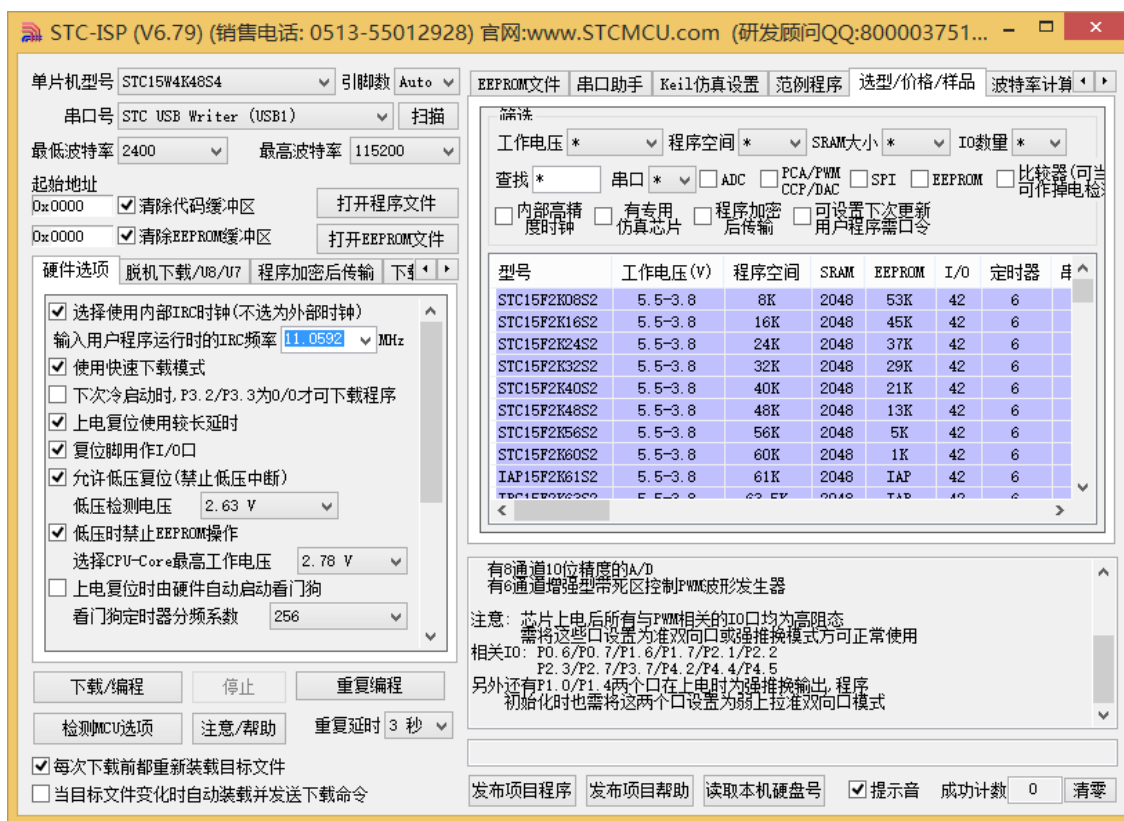


此时在设备管理器中，之前带有黄色感叹号的设备，此时会显示为“STC USB Low Speed Writer”的设备名





在之前打开的 STC-ISP 下载软件中的串口号列表会自动选择所插入的 USB 设备, 并显示设备名称为“STC USB Writer (USB1)”, 如下图:



## Windows 8 (64 位) 安装方法

由于 Windows8 64 位操作系统在默认状态下, 对于没有数字签名的驱动程序是不能安装成功的。所以在安装 STC-USB 驱动前, 需要按照如下步骤, 暂时跳过数字签名, 即可顺利安装成功。

首先将鼠标移动到屏幕的右下角, 选择其中的“设置”按钮



然后在设置界面中选择“更改电脑设置”项



在电脑设置中，选择“常规”属性页中“高级启动”项下面的“立即启动”按钮



在下面的界面中，选择“疑难解答”项



然后选择“疑难解答”中的“高级选项”



在下面的“高级选项”界面中，选择“启动设置”

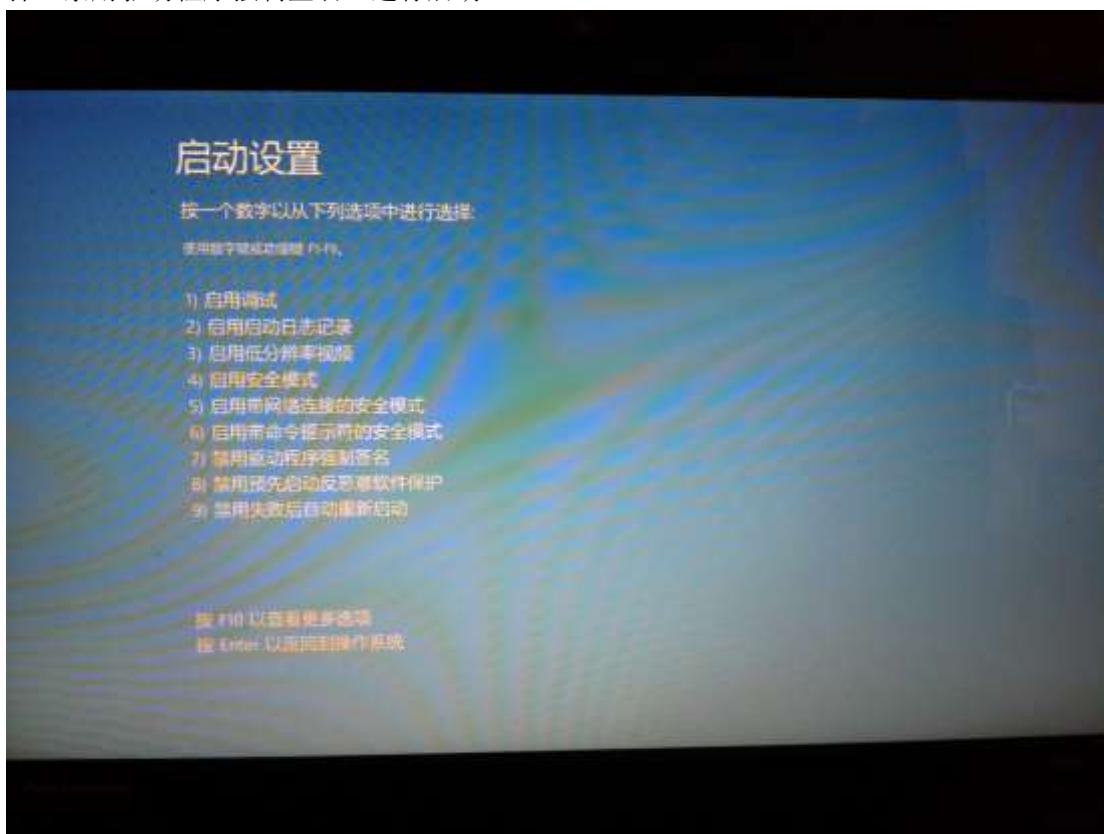


在下面的“启动设置”界面中，单击“重启”按钮对电脑进行重新启动





在电脑重新启动后会自动进入如下图所示的“启动设置”界面，按数字键“7”或者按功能键“F7”选择“禁用驱动程序强制签名”进行启动



启动到 Windows 8 后，按照 [Windows 8（32 位）的安装方法](#)即可完成驱动的安装

## Windows 8.1 (64 位) 安装方法

Windows 8.1 与 Windows 8 进入高级启动菜单的方法不一样,在此专门进行说明。

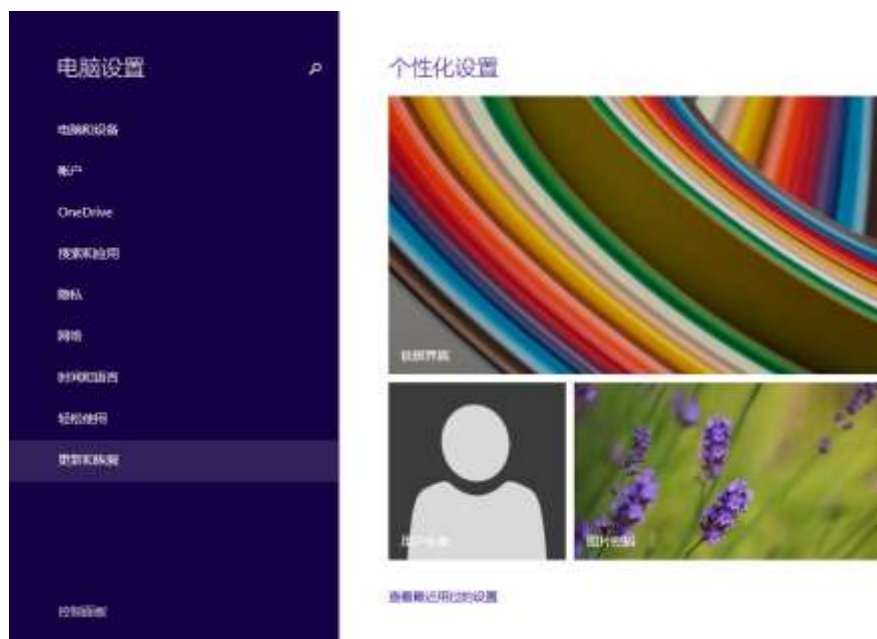
首先将鼠标移动到屏幕的右下角, 选择其中的“设置”按钮



然后在设置界面中选择“更改电脑设置”项



在电脑设置中, 选择“更新和恢复”(这里与 Windows 8 不一样, Windows 8 选择的是“常规”)



在更新和恢复页面中选择“恢复”属性页，单击“高级启动”项下面的“立即启动”按钮



接下来的操作与 Window 8 的步骤相同  
在下面的界面中, 选择“疑难解答”项



然后选择“疑难解答”中的“高级选项”



在下面的“高级选项”界面中，选择“启动设置”

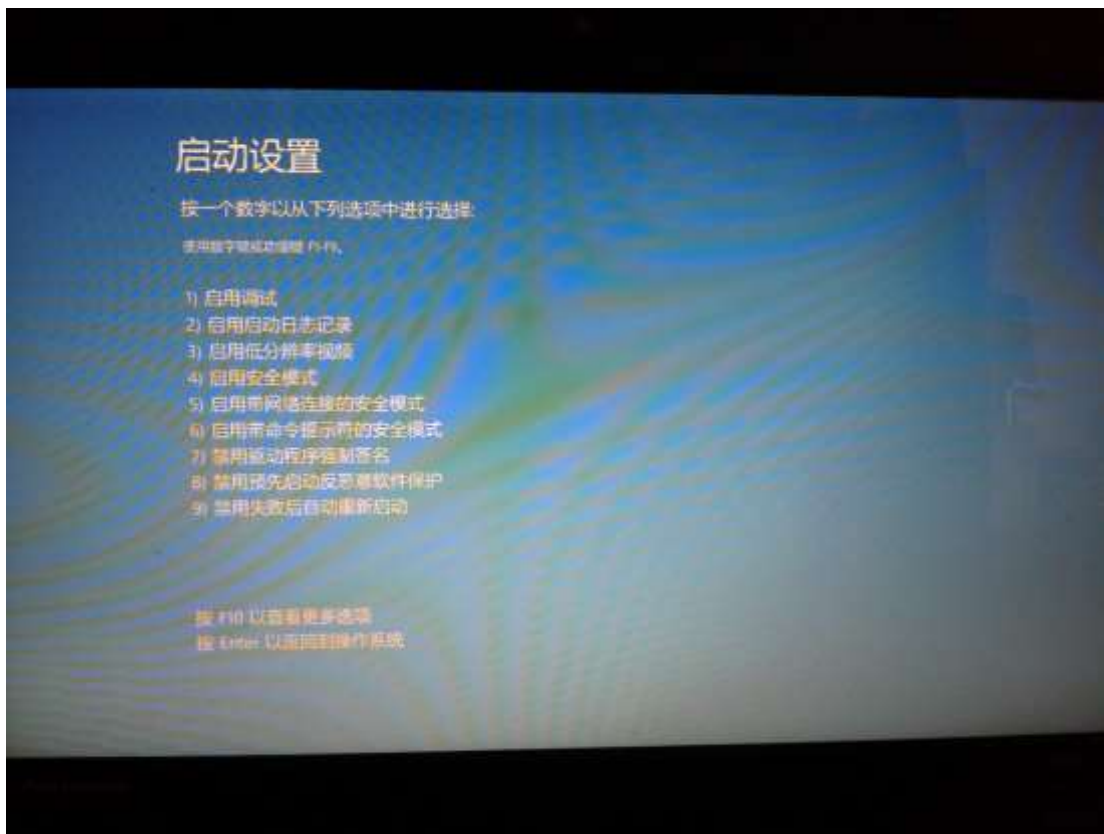




在下面的“启动设置”界面中，单击“重启”按钮对电脑进行重新启动



在电脑重新启动后会进入如下图所示的“启动设置”界面，按数字键“7”或者按功能键“F7”选择“禁用驱动程序强制签名”进行启动



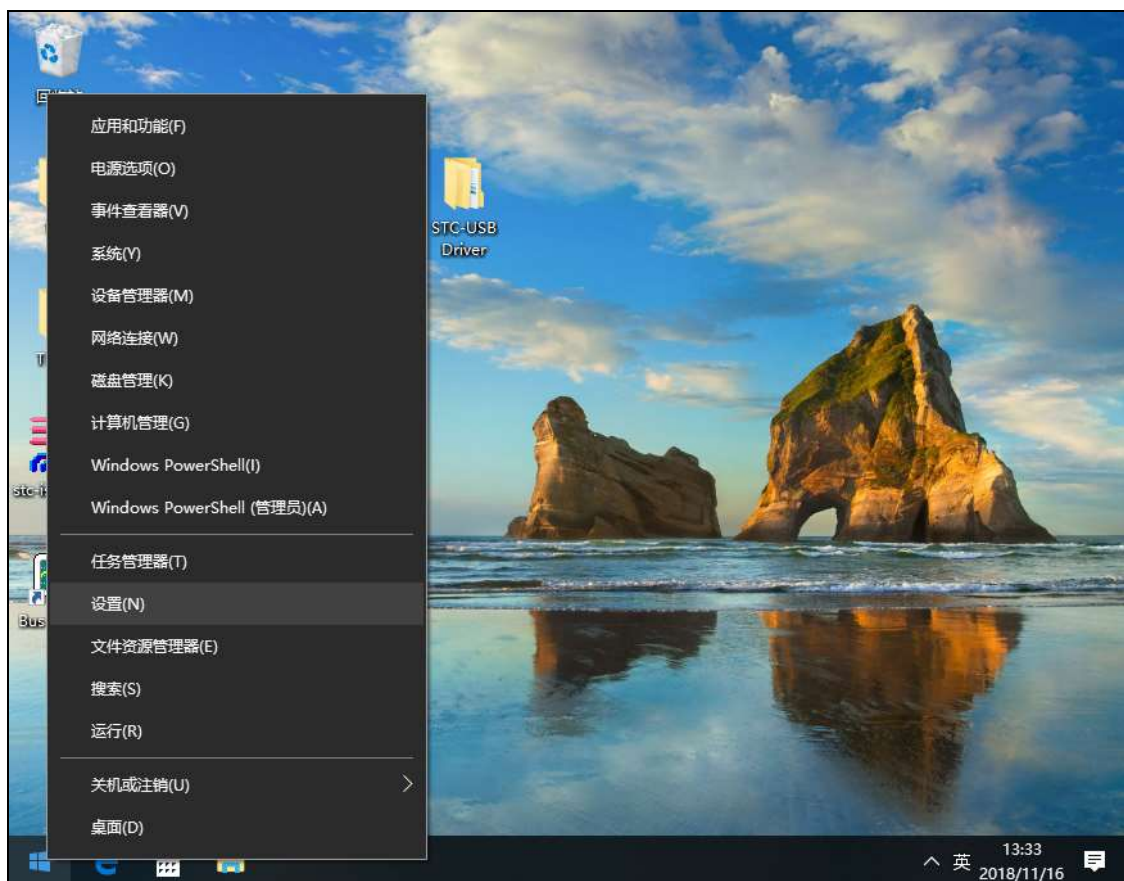
启动到 Windows 8 后，按照 **Windows 8（32 位）的安装方法**即可完成驱动的安装

## Windows10（64 位）安装方法

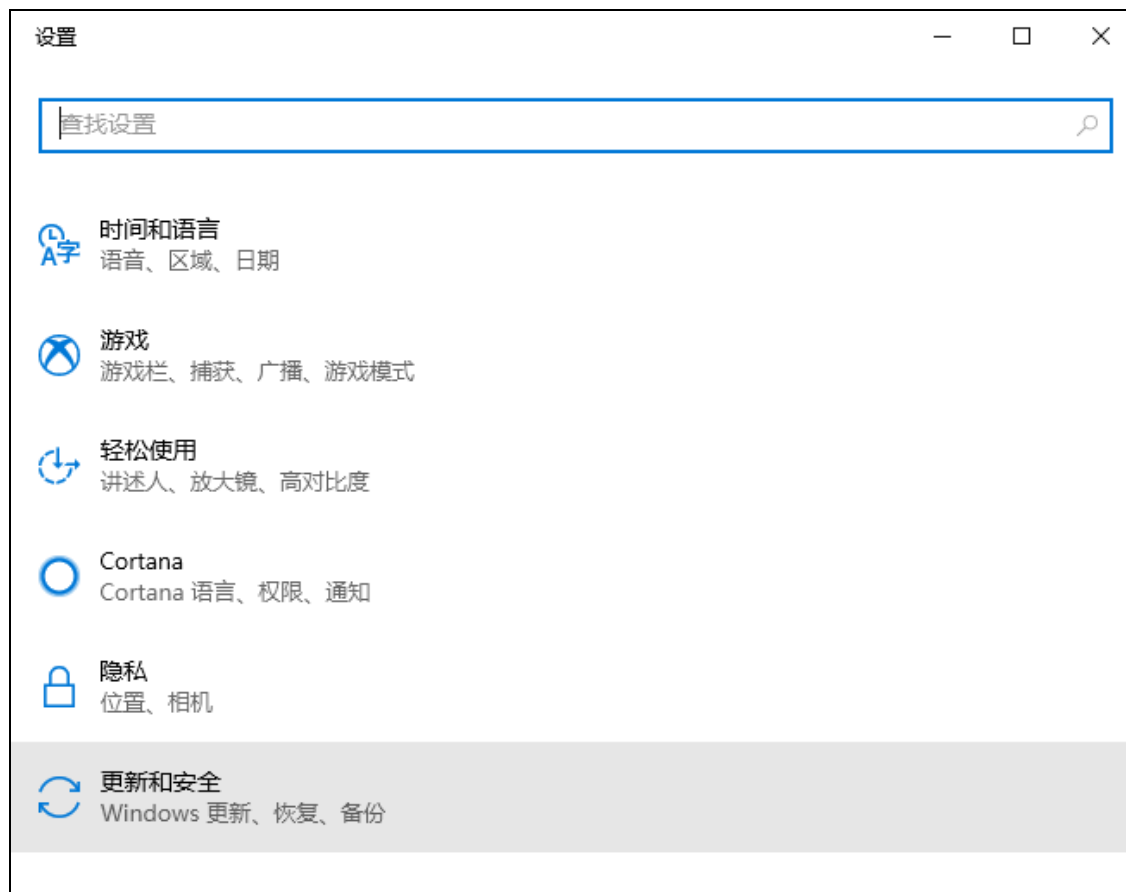
由于 Windows10 64 位操作系统在默认状态下，对于没有数字签名的驱动程序是不能安装成功的。所以在安装 STC-USB 驱动前，需要按照如下步骤，暂时跳过数字签名，即可顺利安装成功。

安装驱动前需要从 STC 官网下载的 STC-ISP 下载软件压缩包中将“STC-USB Driver”文件夹解压缩到硬盘中。将具有 USB 下载功能的芯片准备好，但先不要连接电脑

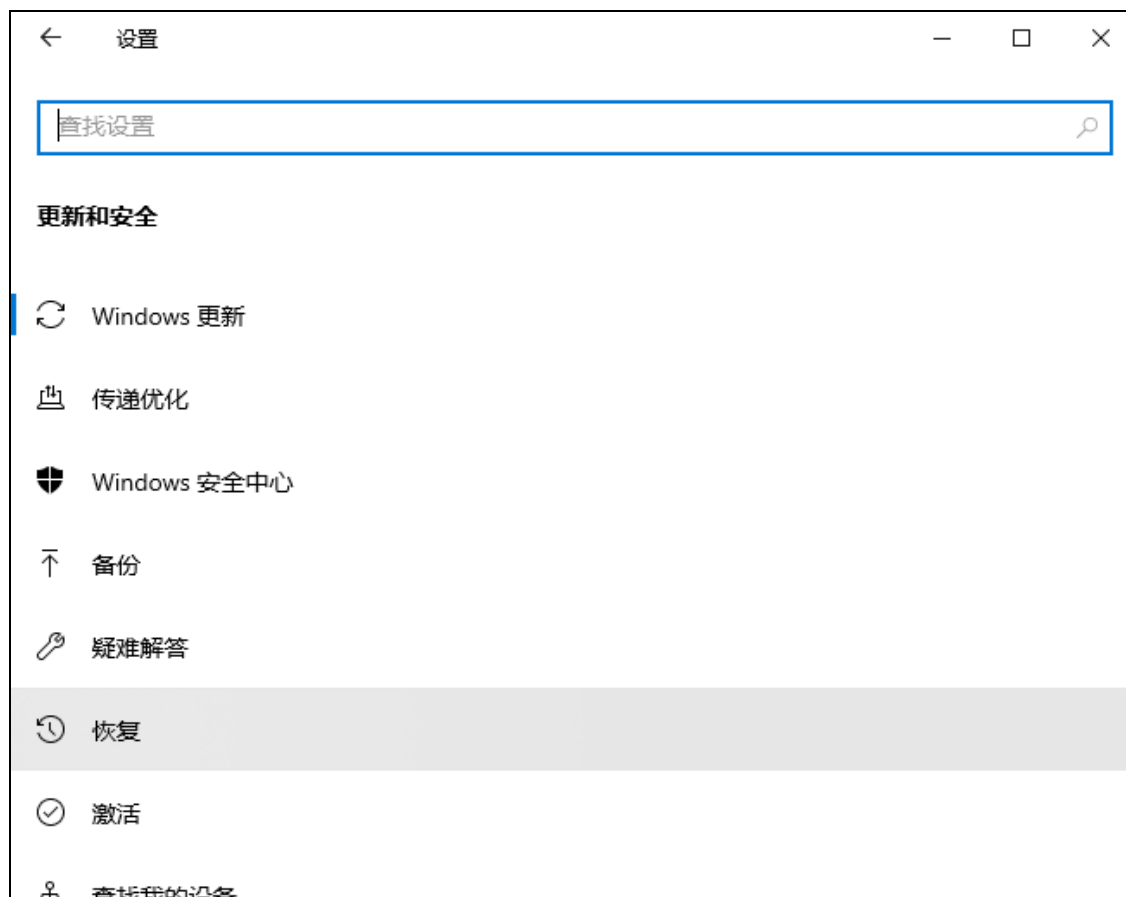
鼠标右键点击“开始”菜单，选择“设置”选项



然后在设置界面中选择“更新和安全”项



然后在设置界面中选择“恢复”项



在恢复界面中，点击“高级启动”项中的“立即重新启动”按钮



在电脑重启前，系统会先进入如下的启动菜单，选择“疑难解答”项



在疑难解答界面中选择“高级选项”





然后选择“查看更多恢复选项”



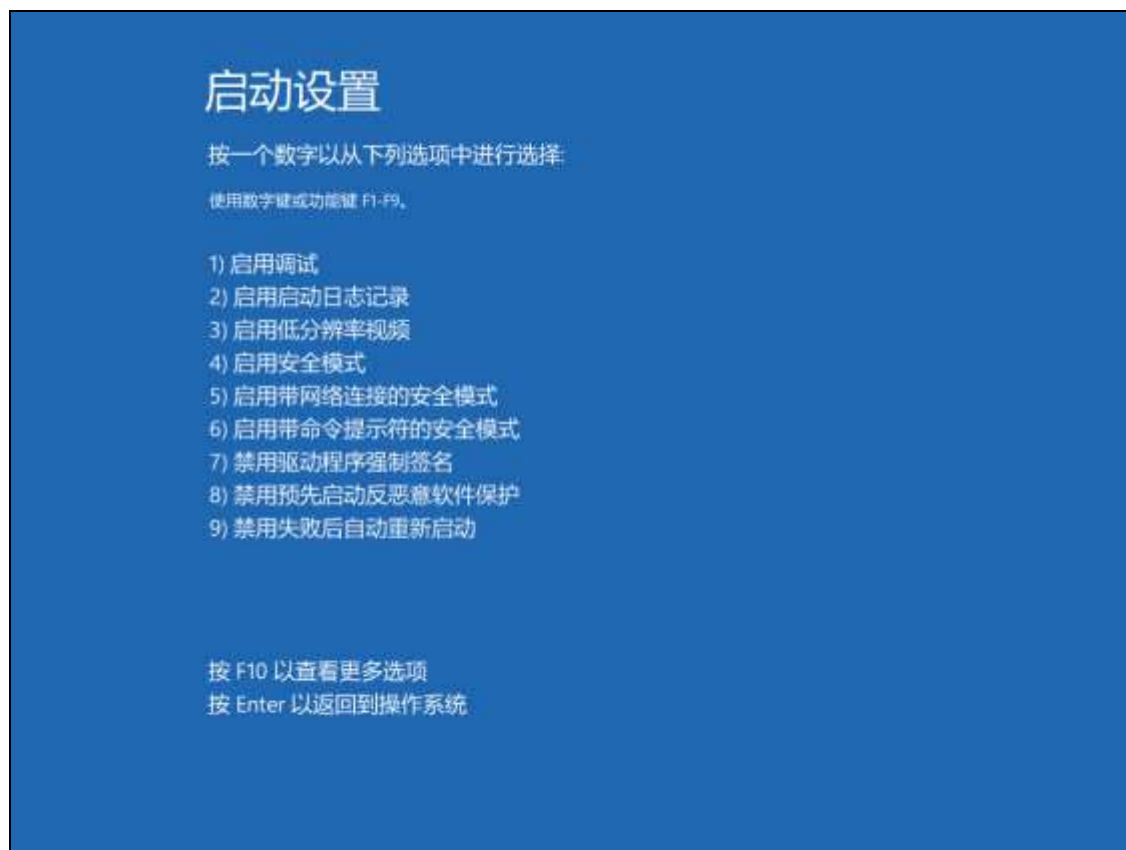
选择“启动设置”项



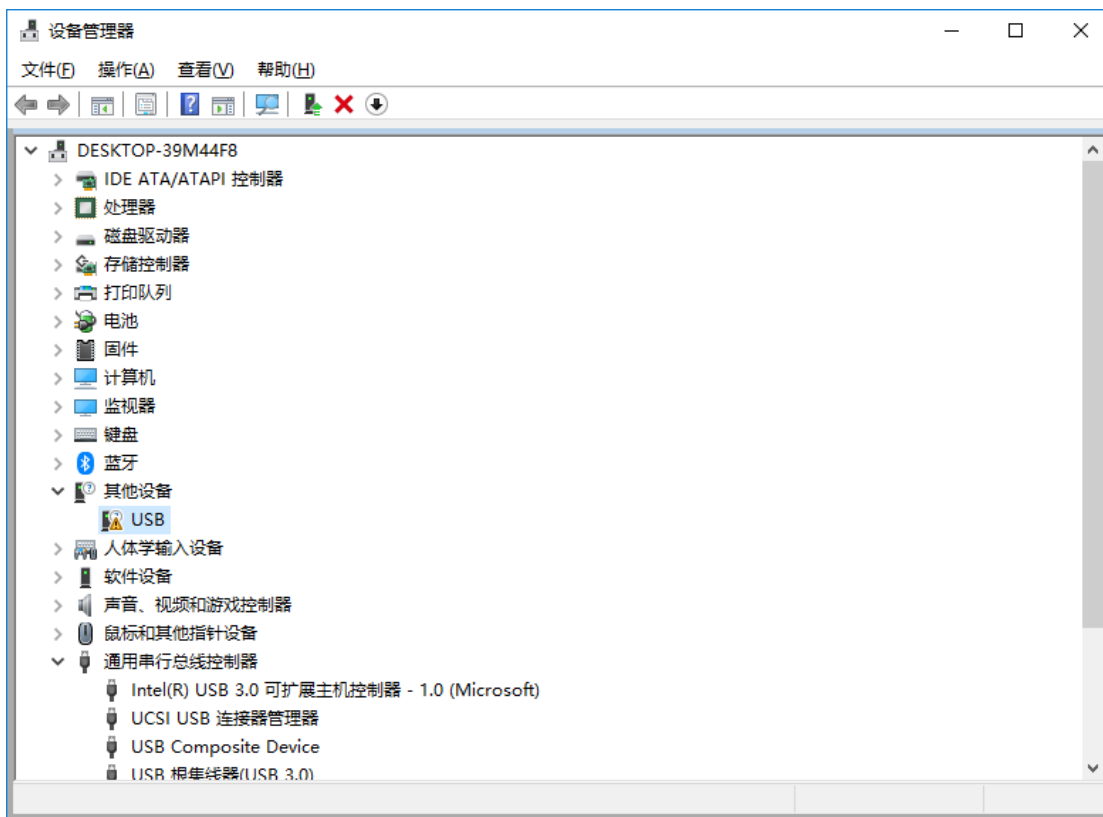
出现如下画面后，点击“重启”按钮重启电脑



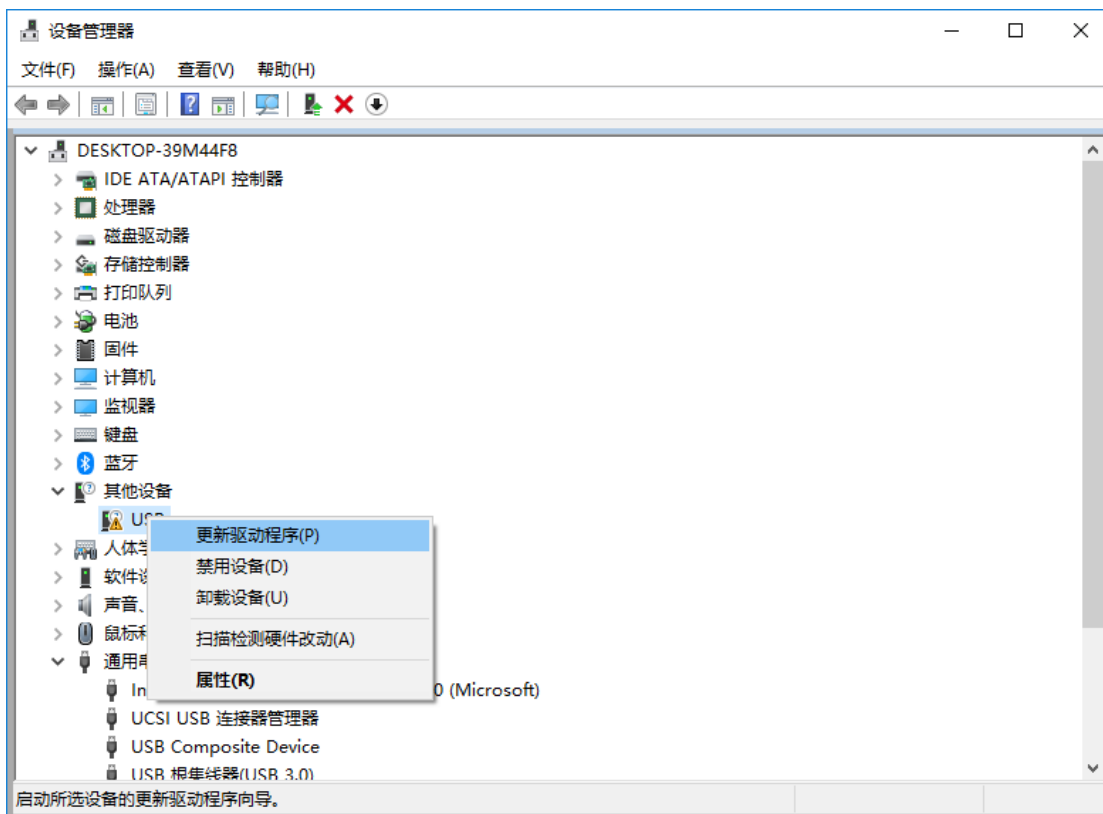
电脑重启后，会弹出“启动设置”界面，按“F7”按钮来选择“禁止驱动程序强制签名”项



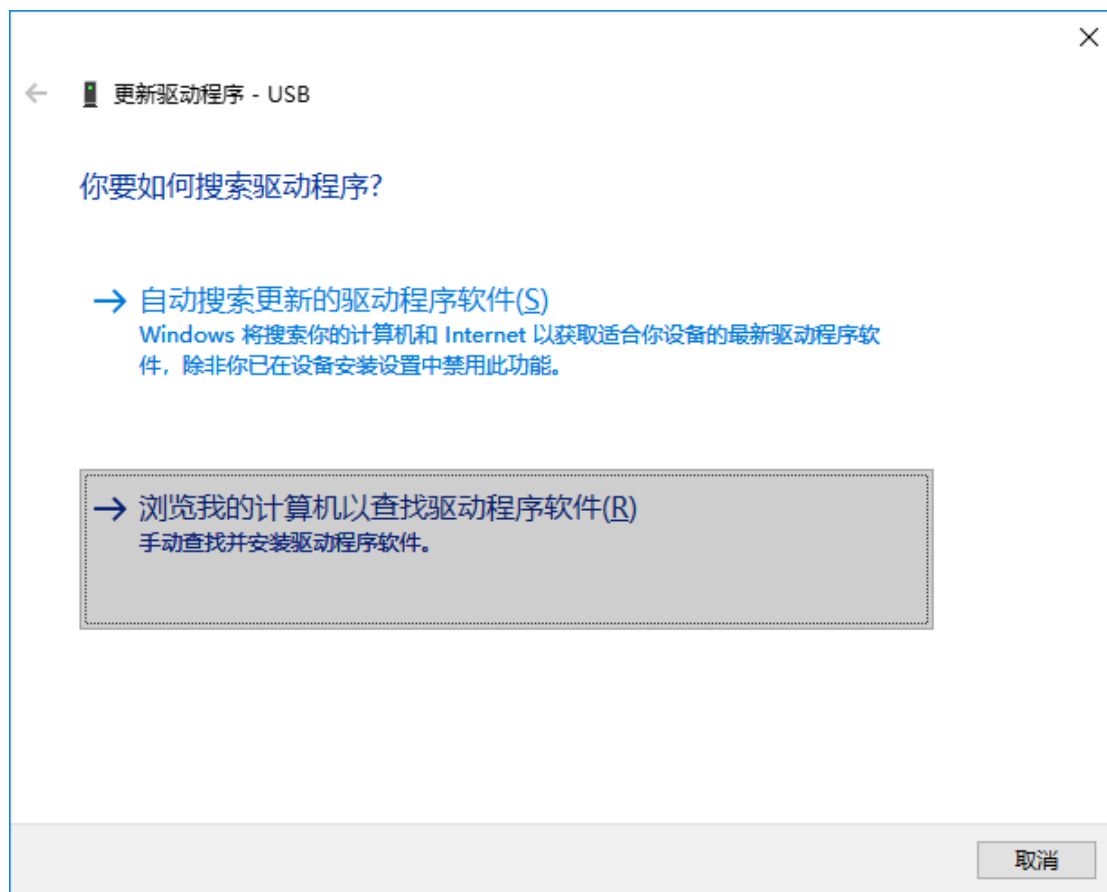
电脑启动完成后, 将准备好的芯片用 USB 线与电脑相连, 并打开“设备管理器”, 此时由于驱动还没有开始安装, 所以在设备管理器中会显示为一个带感叹号的未知设备



鼠标右键单击未知设备，选择右键菜单中的“更新驱动程序”



在弹出的驱动安装程序选择画面中, 选择“浏览我的计算机以查找驱动程序软件”项

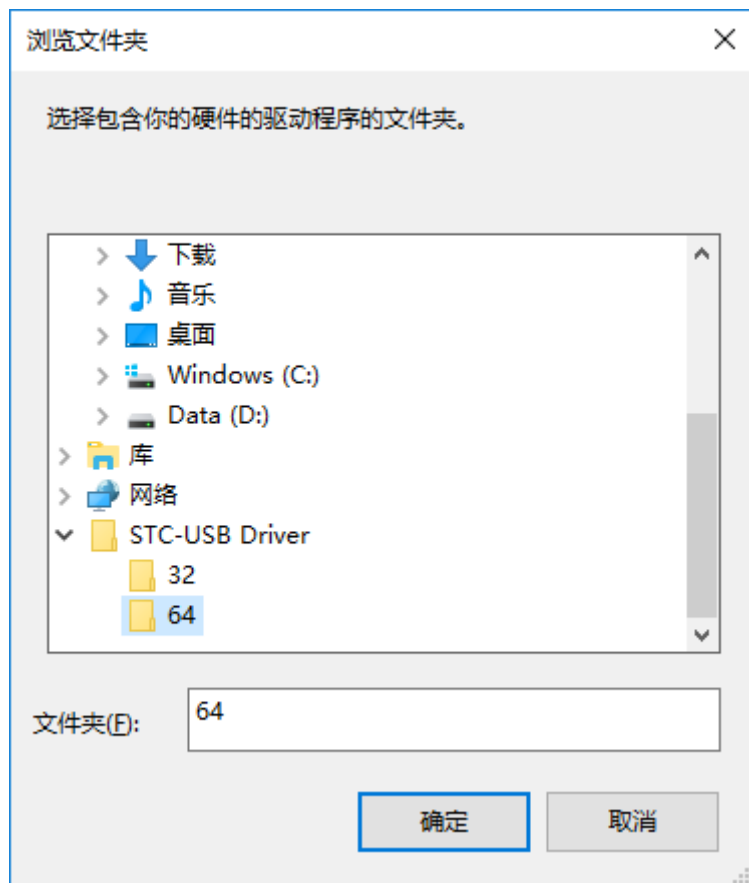


在如下界面中, 点击“浏览”按钮

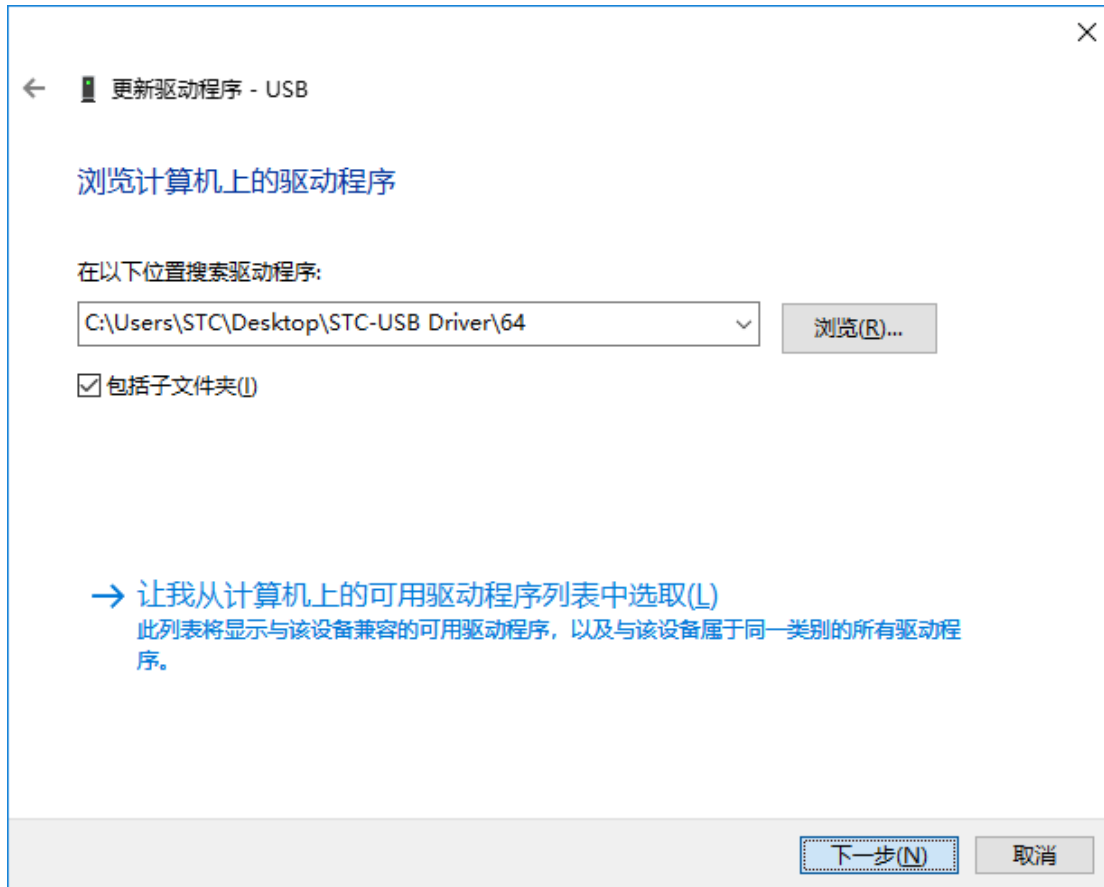




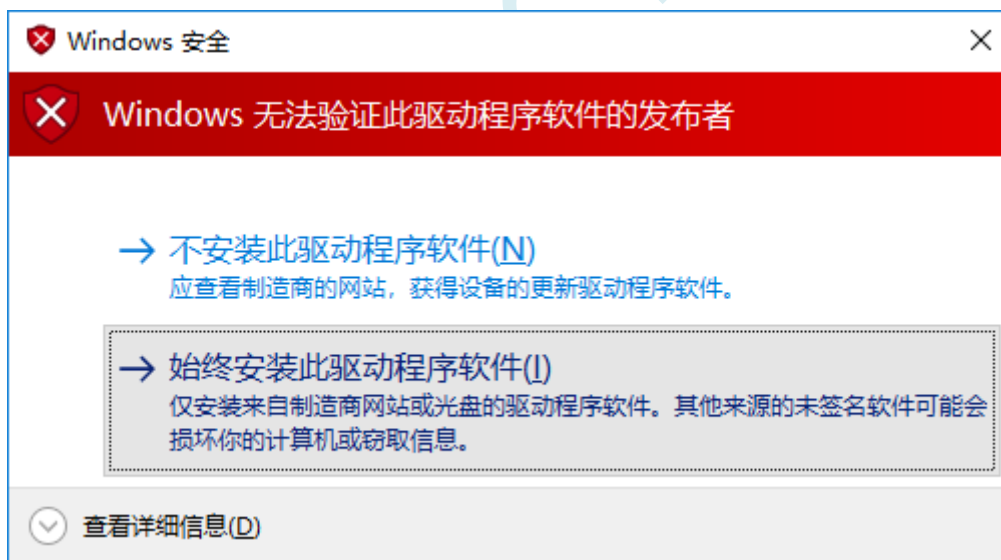
找到之前解压缩到硬盘中的“STC-USB Driver”目录，选择目录中的“64”目录，并确定



点击“下一步”开始安装驱动



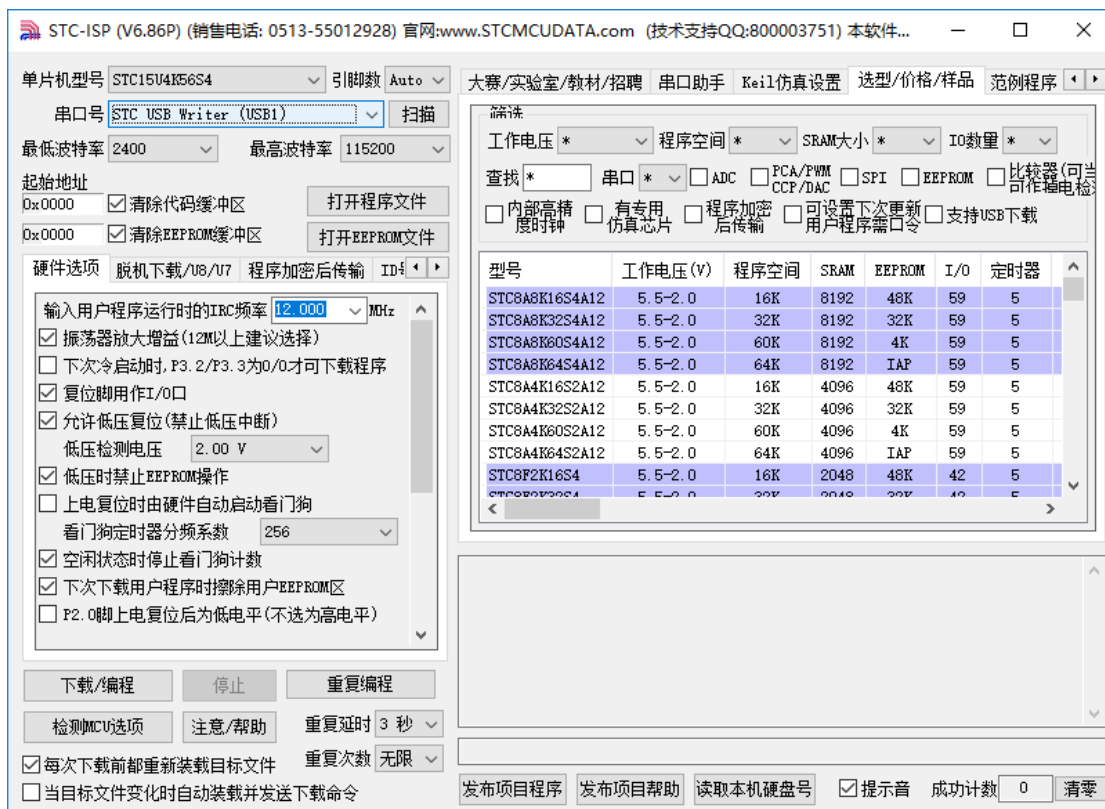
驱动安装的过程中, 会弹出如下的警告画面, 选择“始终安装此驱动程序软件”



出现下面的画面时，驱动程序就安装成功了

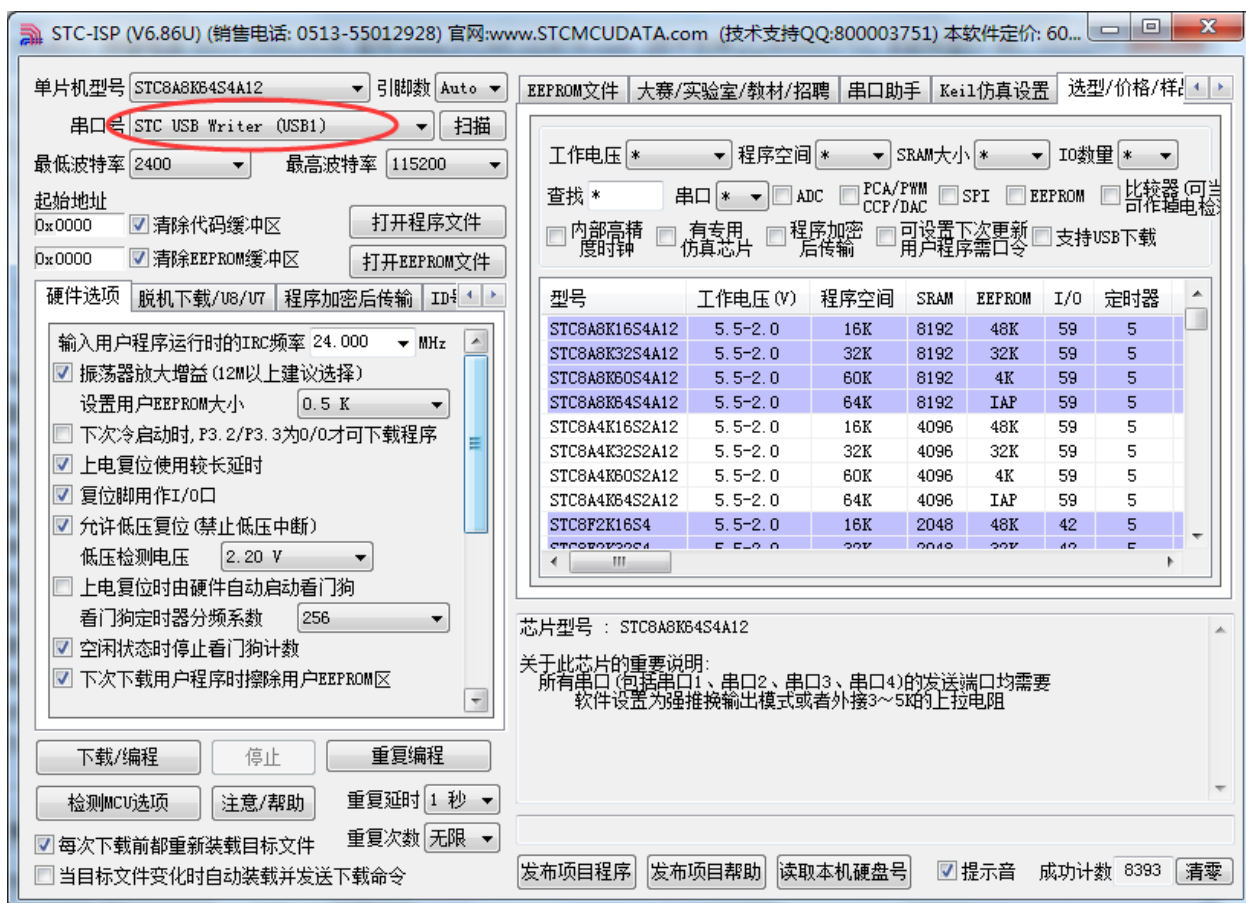


在回到 STC-ISP 的下载软件, 此时“串口号”的下拉列表中已自动选择了“STC USB Writer (USB1)”, 即可使用 USB 进行 ISP 下载了

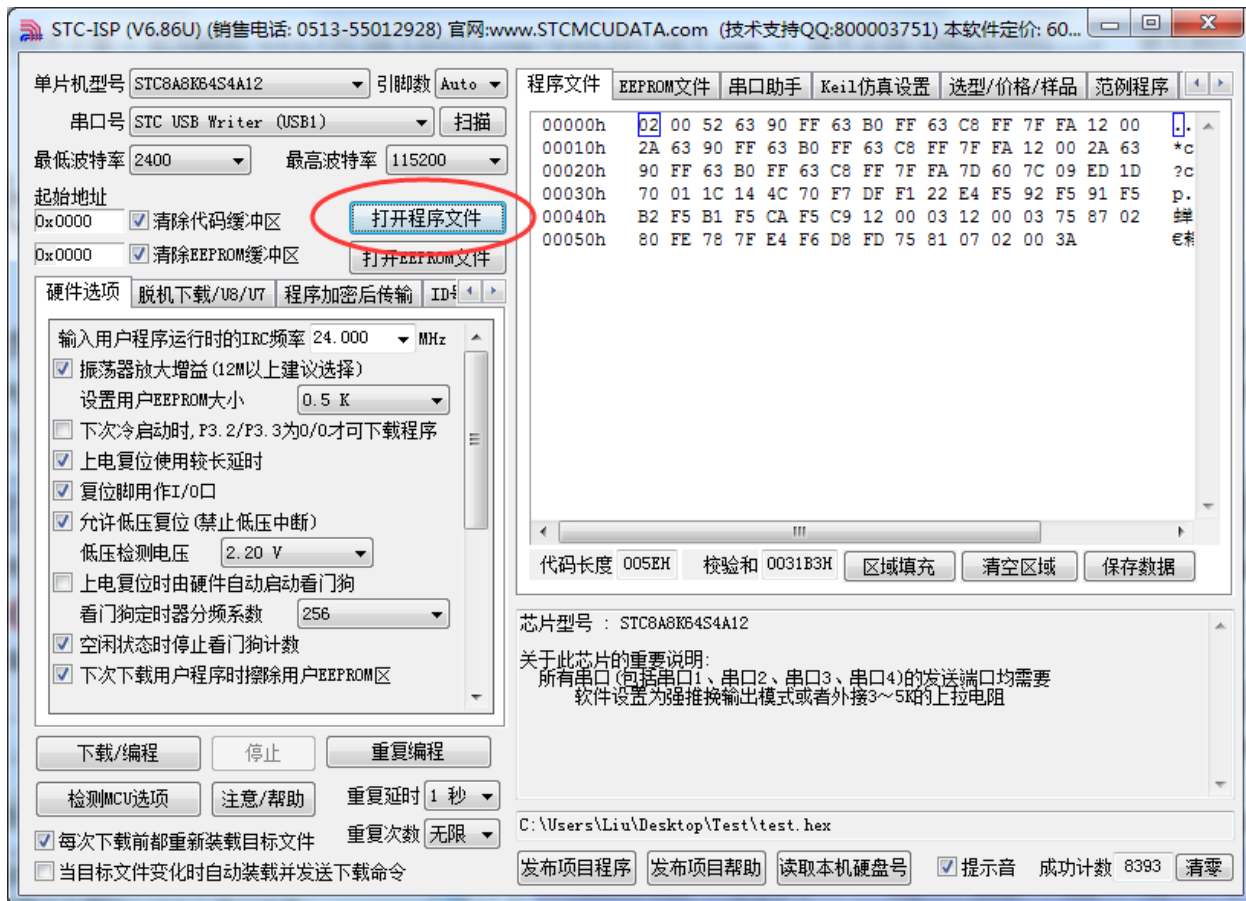


## 附录F USB 下载步骤演示

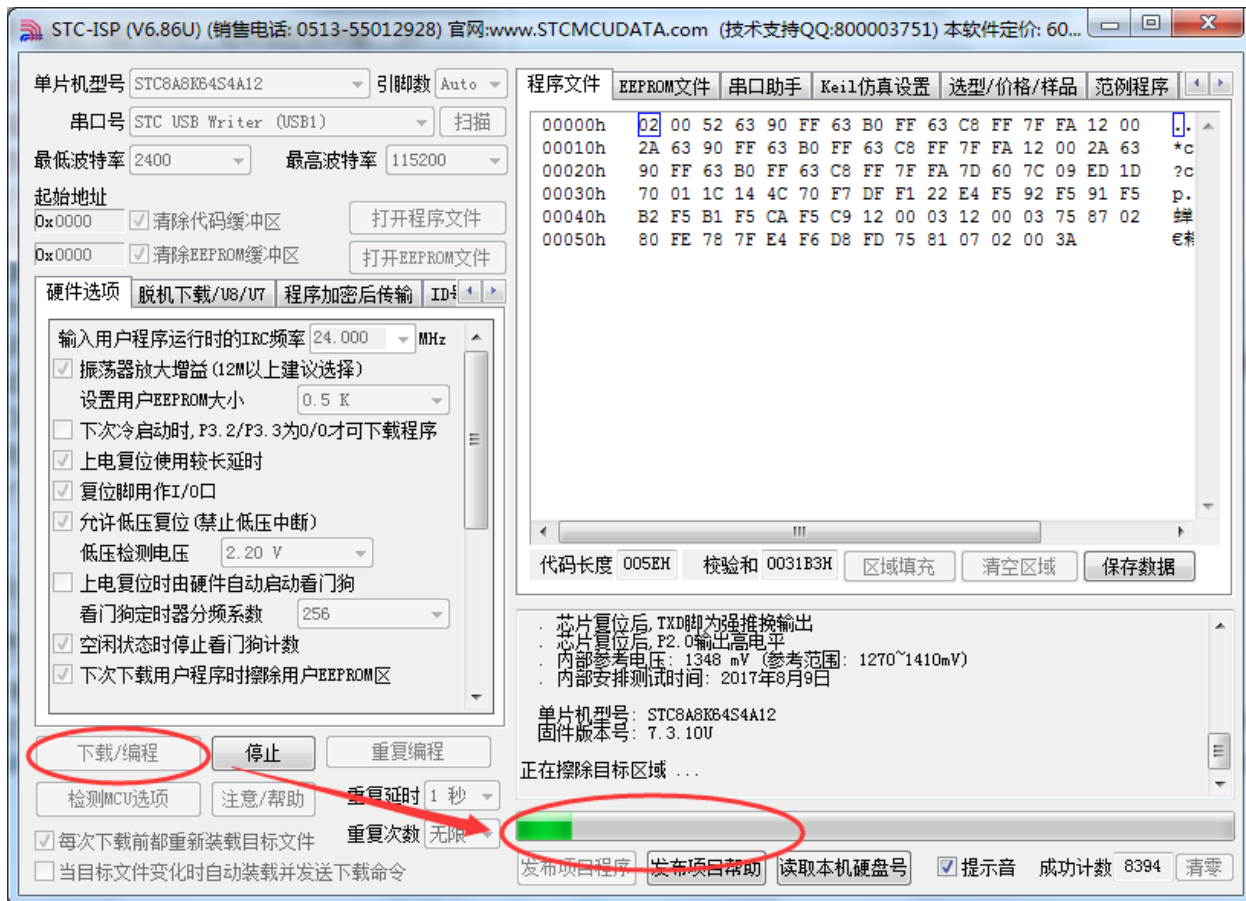
1、首先参考 P5.1.5 章的应用线路图连接好单片机，并将目标芯片的 P3.2 口连接到 Gnd，然后将系统连接到 PC 端的 USB 端口上。打开 ISP 下载软件，即可在下载软件的串口号中自动搜索到“STC USB Writer (USB1)”的 USB 设备



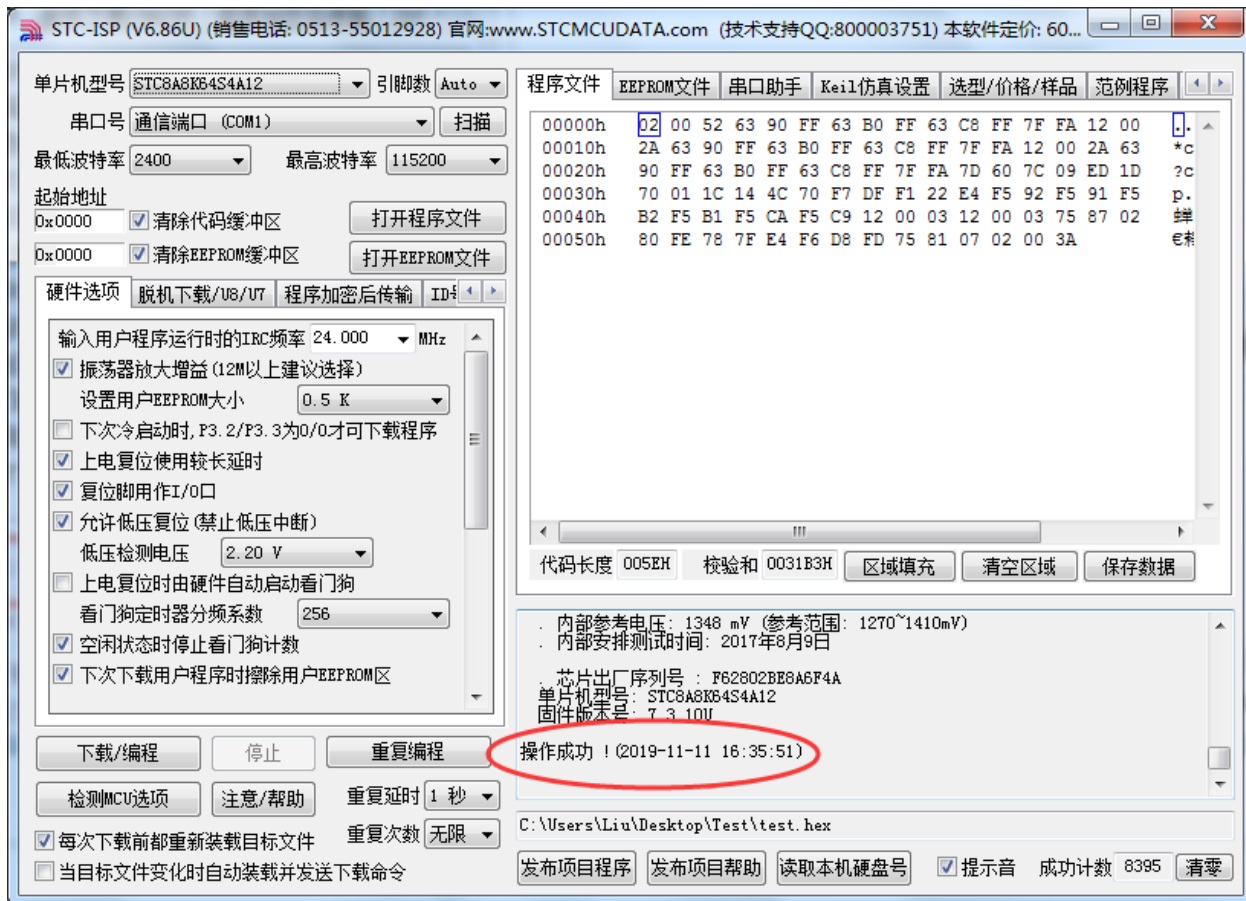
## 2、打开用户代码程序



## 3、点击“下载/编程”按钮开始下载用户代码



4、直到提示“操作成功”，表示程序代码下载完成。





## 附录G USB 仿真步骤演示

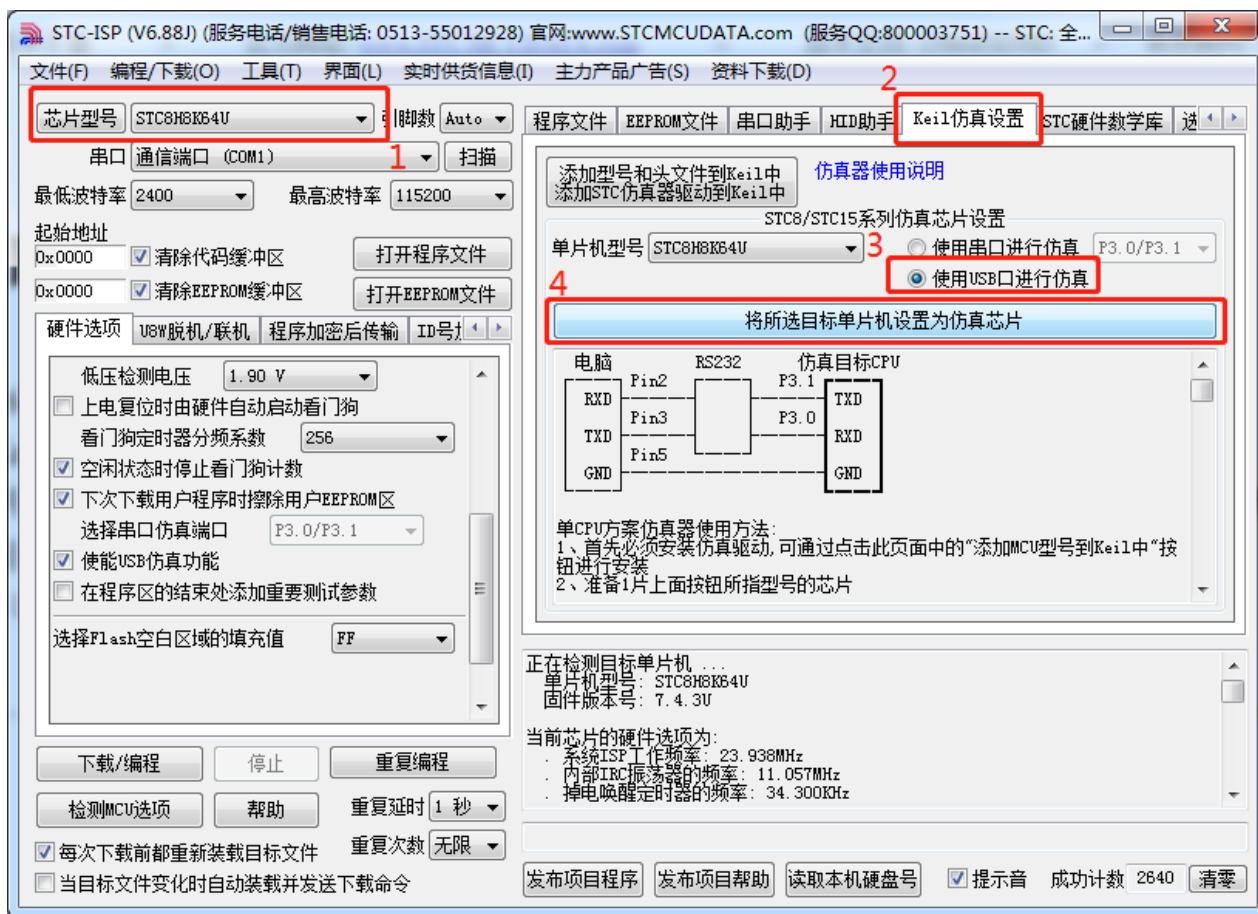
STC8H8K64U 系列的 B 版芯片以及后续版本均支持 USB 直接仿真功能。

具体操作步骤如下:

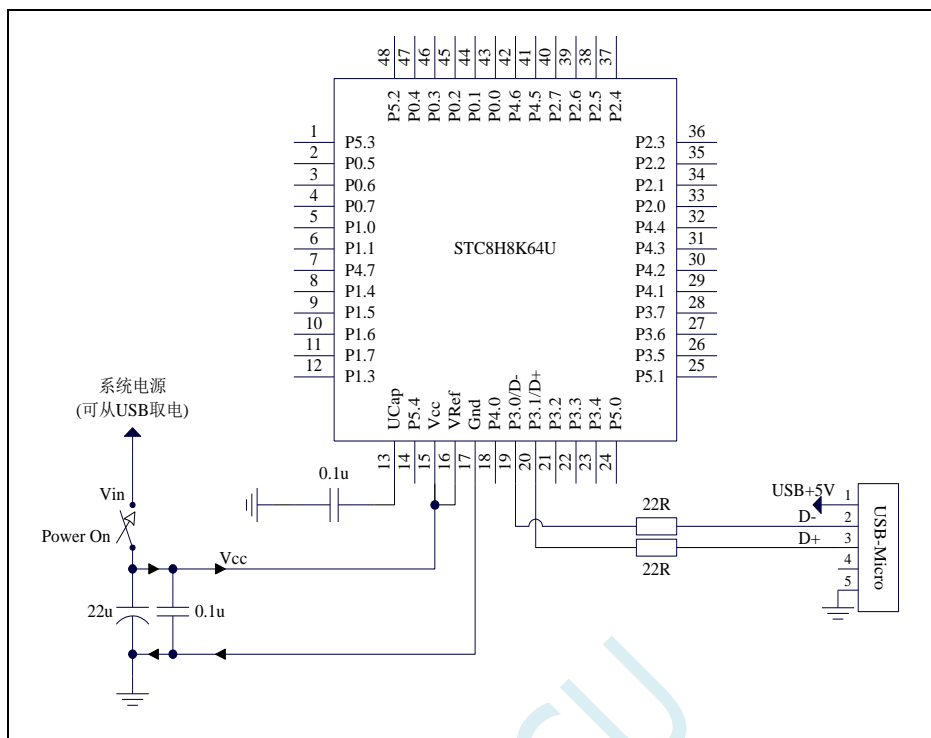
### 1、安装 Keil 环境下的最新的 STC 仿真驱动

从官网 ([www.STCMCUDATA.com](http://www.STCMCUDATA.com)) 下载最新的 STC-ISP 下载软件, 并按照附录 A 的步骤安装 STC 仿真驱动

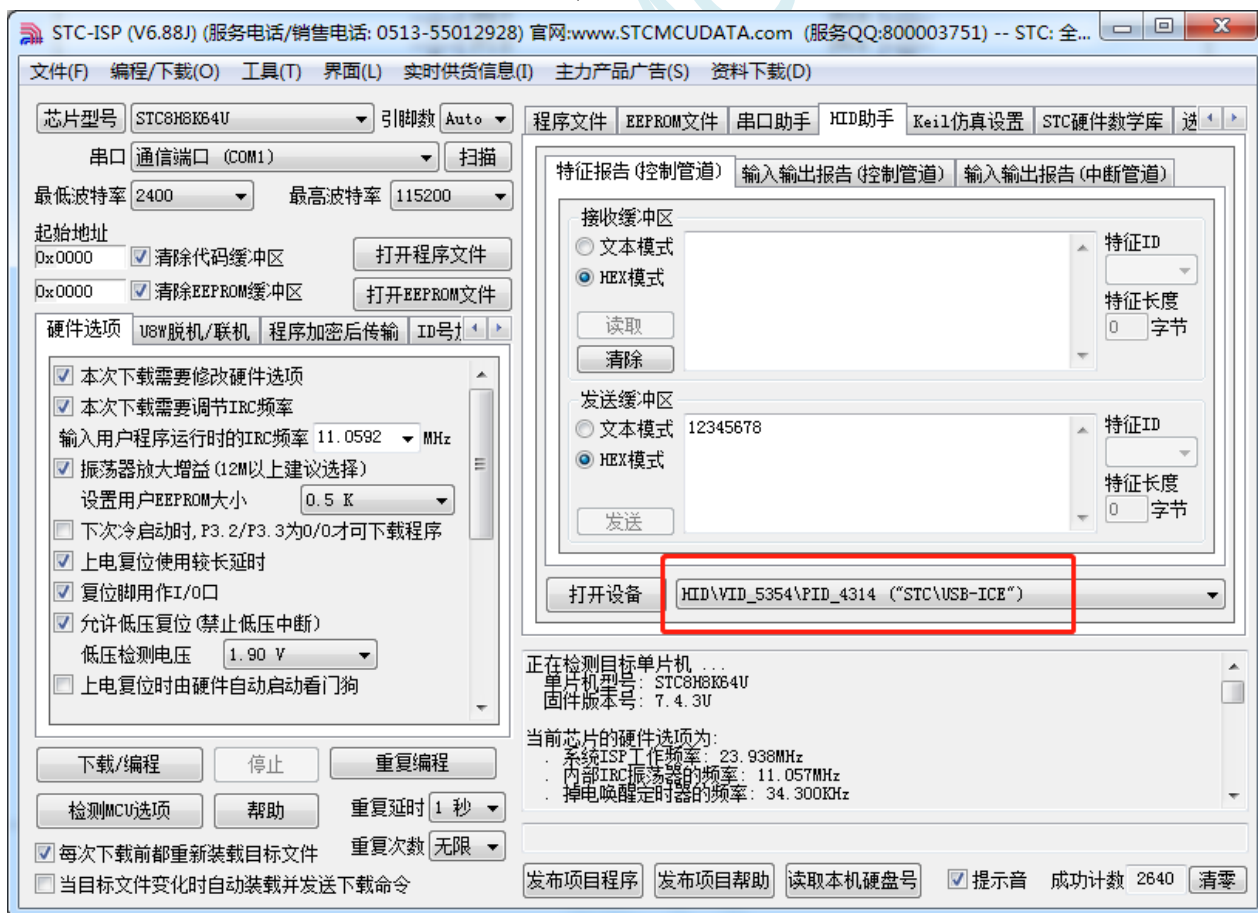
### 2、使用串口 ISP 或者 USB ISP 制作仿真芯片



3、参考下图中的连接方式，将制作完仿真芯片的目标单片机连接到电脑的 USB 口上

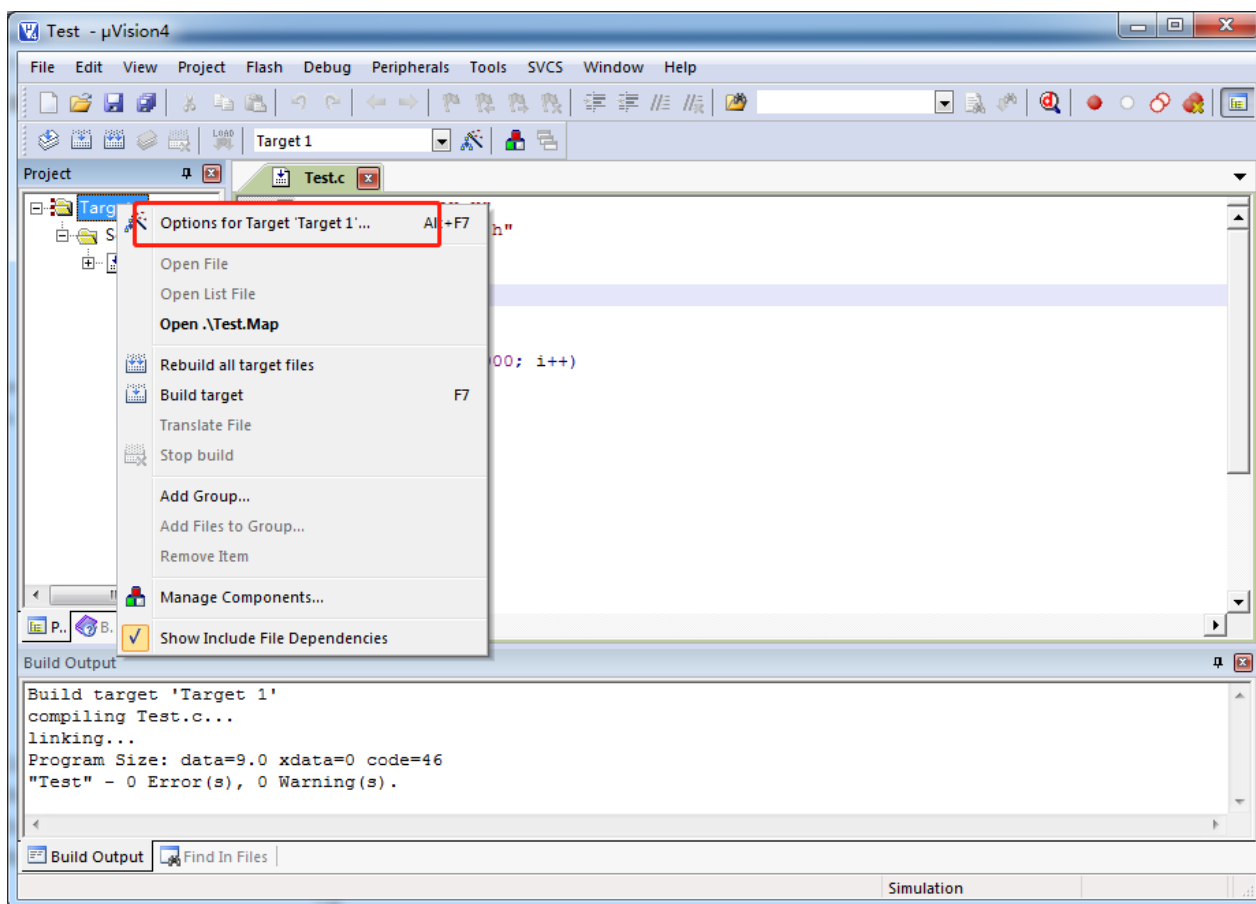


当在下载软件中 HID 助手中能正确显示“STC\USB-ICE”设备时，代表硬件已正确连接

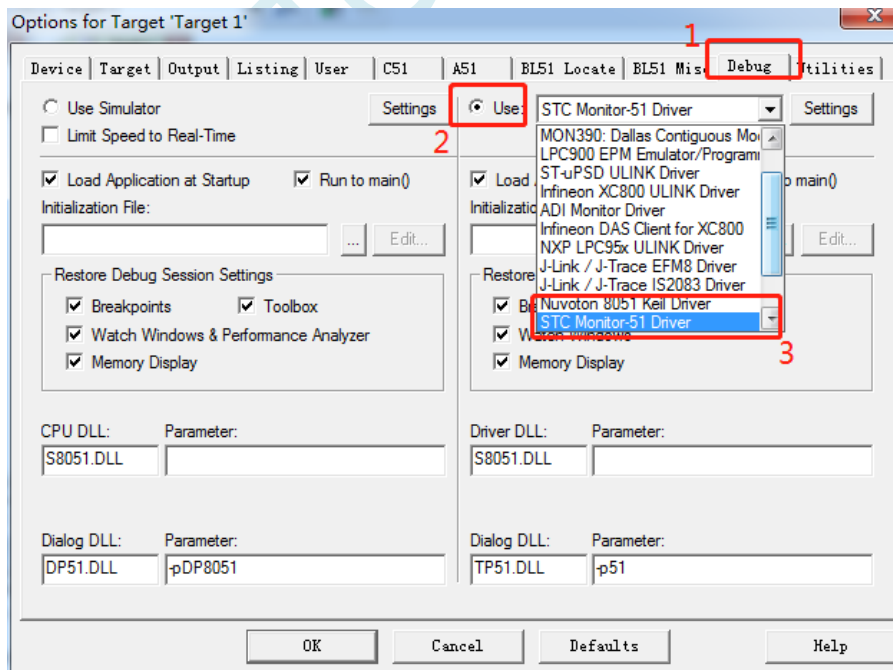


注意：当显示的设备名称为“STC\USB-ISP”时，表示目标芯片此时出于 USB-ISP 下载模式，请确保 P3.2 口为高电平，并重新连接到 USB 即可。

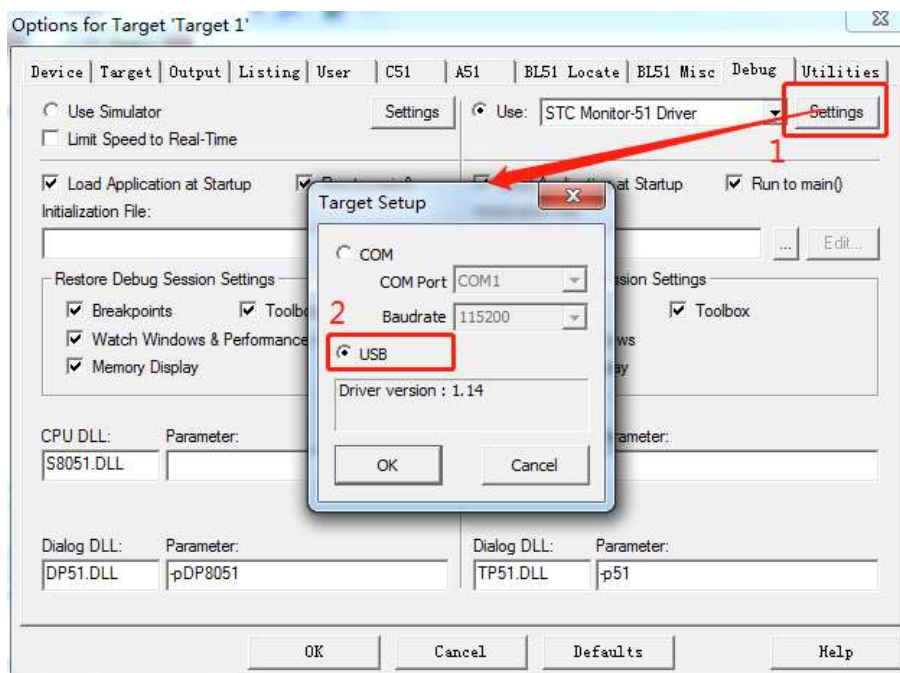
#### 4、在 Keil 软件进行项目设置



打开项目选项设置页面，按下图所示选择 STC 仿真驱动

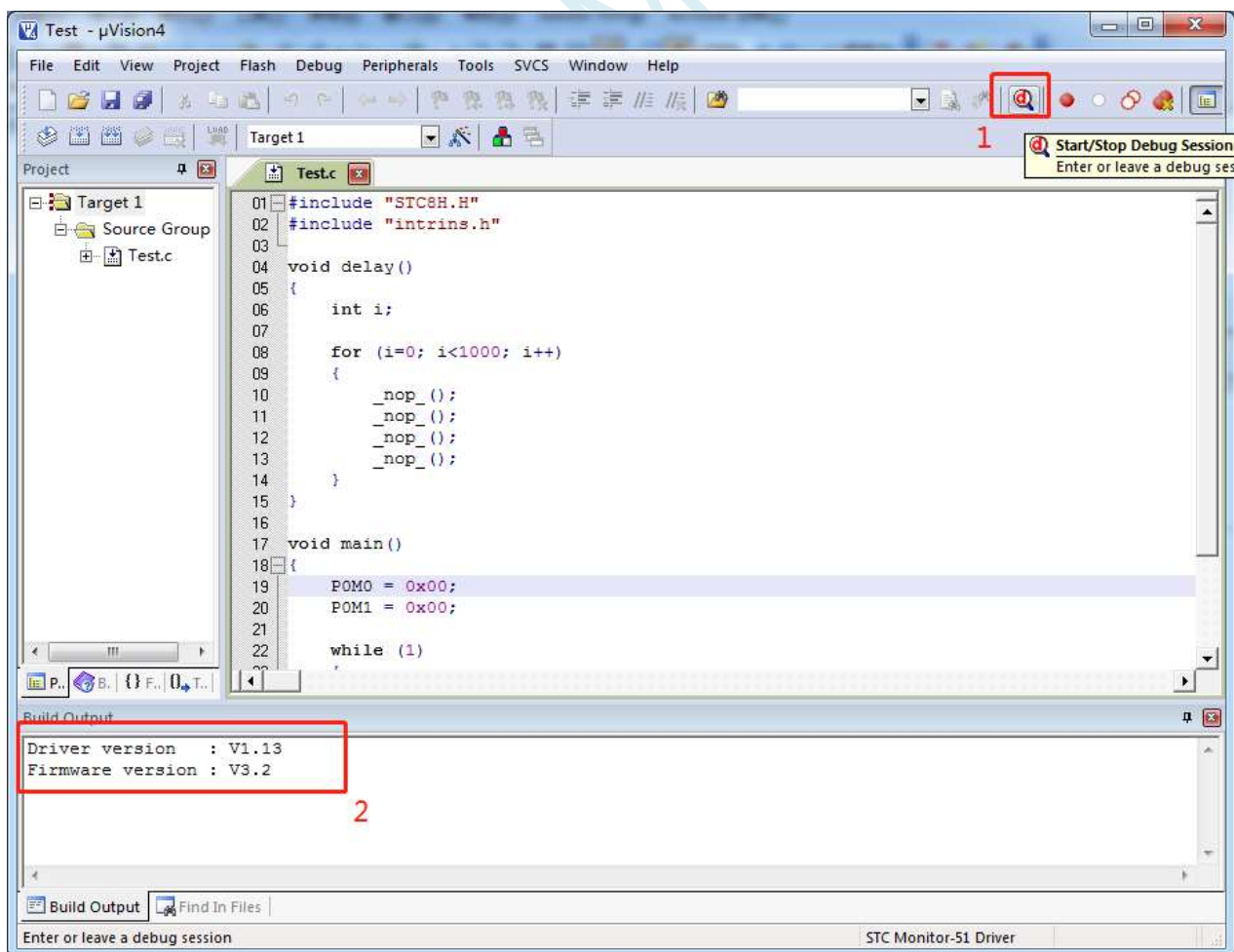


按下图所示, 选择 STC 的 USB 仿真接口

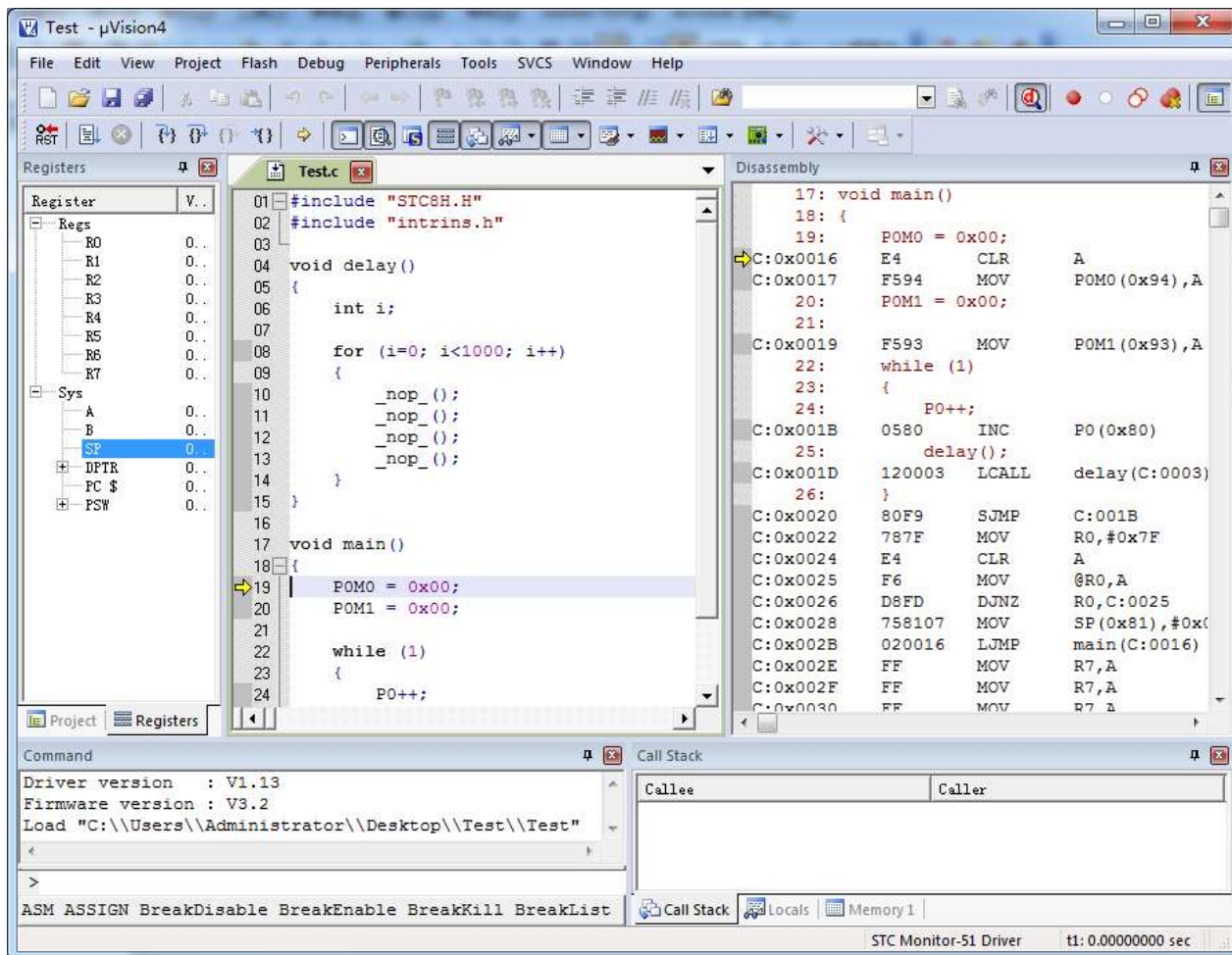


#### 5、设置完成后开始仿真

点击 Keil 软件中的开始仿真按钮, 如果在输出窗口能正确显示版本号, 表示连接和设置均配置正确



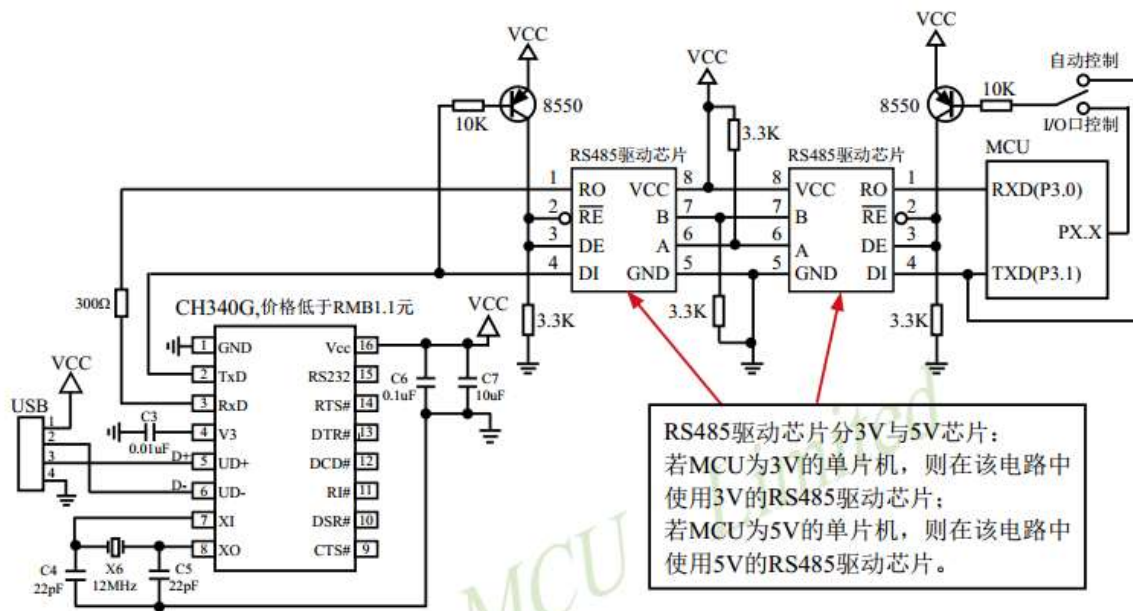
代码下载完成后, 就会进入如下调试画面



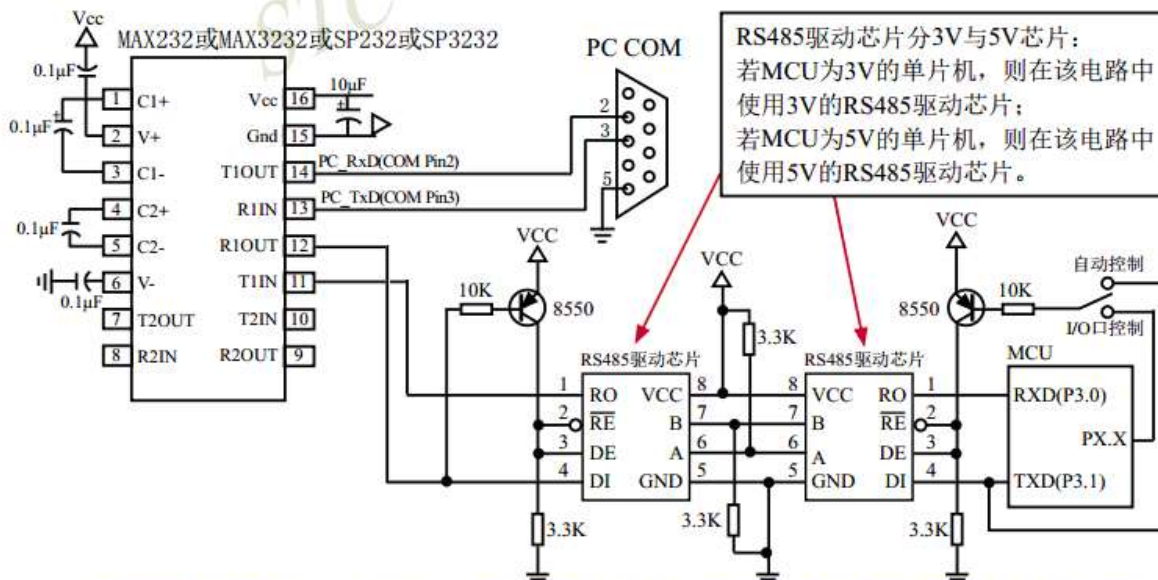


## 附录H RS485 自动控制或 I/O 口控制线路图

### 1、利用 USB 转串口连接电脑的 RS485 控制下载线路图(自动控制或 I/O 口控制)



### 2、利用 RS232 转串口连接电脑的 RS485 控制下载线路图(自动控制或 I/O 口控制)



注意：如果要设置单片机某个I/O口控制RS485发送或接收命令有效，则必须将单片机焊入电路板之前先用U8下载工具结合电脑ISP软件对该单片机进行“RS485控制”设置并烧录一下（如上节所述），否则将单片机实现不了RS485控制功能。

建议用户将本节所述“RS485控制下载线路图(自动控制或I/O口控制)”设计到您的用户板上

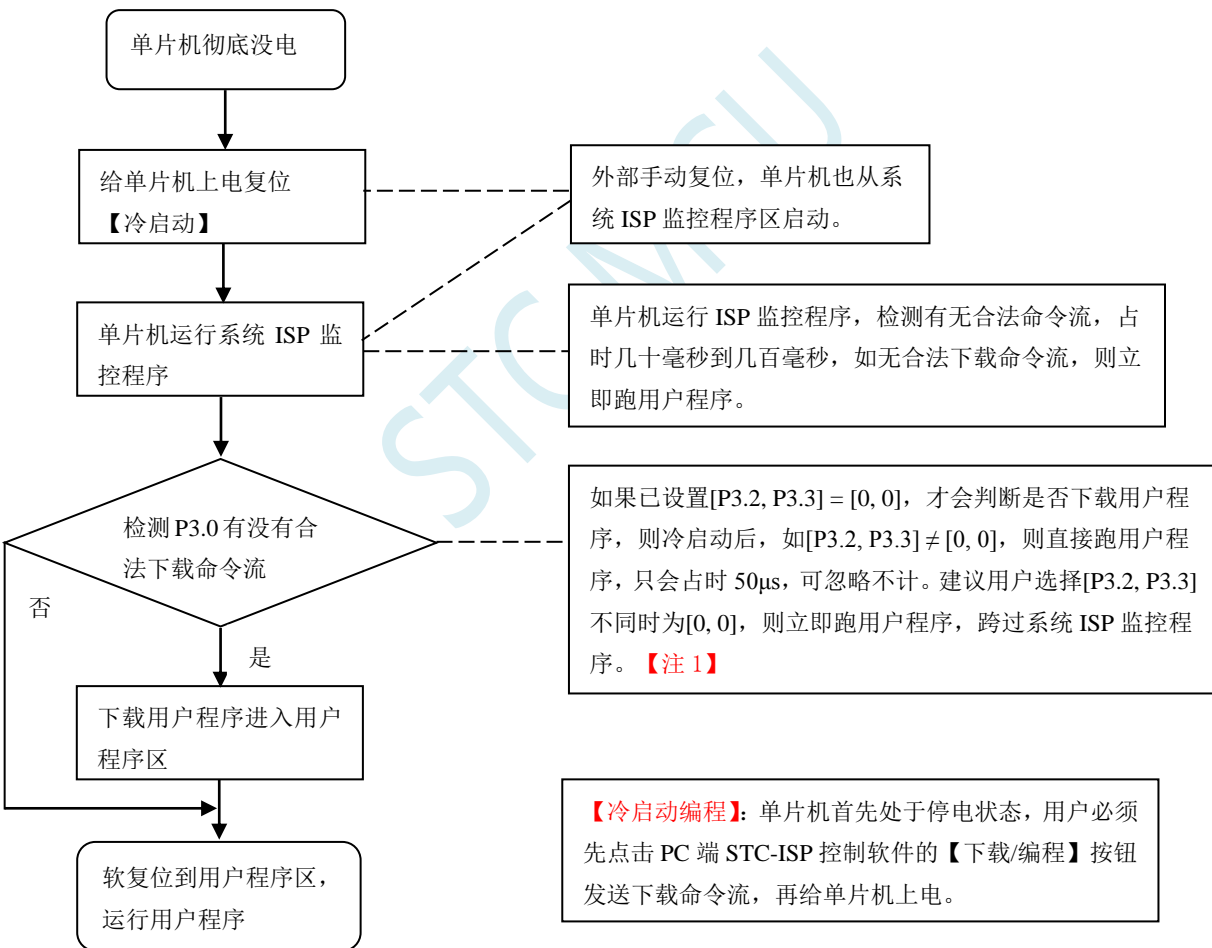
# 附录I STC 工具使用说明书

## I.1 概述

U8W/U8W-Mini 是一款集在线联机下载和脱机下载于一体的编程工具系列。STC 通用 USB 转串口工具则是支持在线下载与在线仿真的编程工具。

| 工具类型       | 在线下载 | 脱机下载 | 烧录座下载 | 在线仿真    | 价格(人民币) |
|------------|------|------|-------|---------|---------|
| U8W        | 支持   | 支持   | 支持    | 需设置直通模式 | 100 元   |
| U8W-Mini   | 支持   | 支持   | 不支持   | 需设置直通模式 | 50 元    |
| 通用 USB 转串口 | 支持   | 不支持  | 不支持   | 支持      | 30 元    |

## I.2 系统可编程（ISP）流程说明

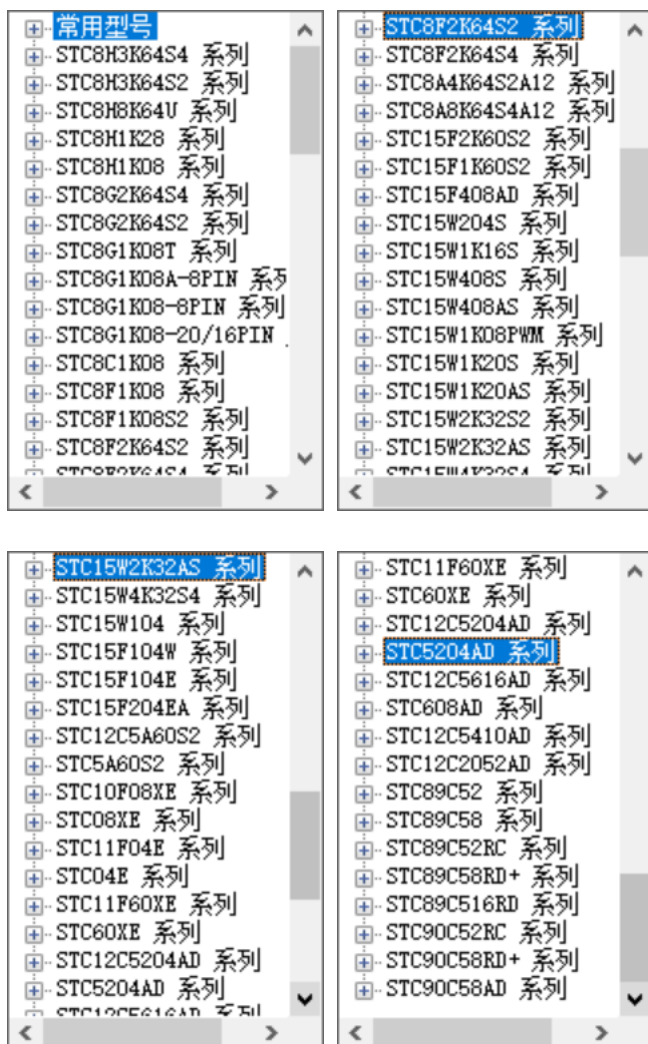


注意：因 [P3.0, P3.1] 作下载/仿真用(下载/仿真接口仅可用 [P3.0, P3.1])，故建议用户将串口 1 放在 P3.6/P3.7 或 P1.6/P1.7，若用户不想切换，坚持使用 P3.0/P3.1 工作或作为串口 1 进行通信，则务必在下载程序时，在软件上勾选“下次冷启动时，P3.2/P3.3 为 0/0 时才可以下载程序”。【注 1】

【注 1】：STC15，STC8 系列及以后新出的芯片的烧录保护引脚为 P3.2/P3.3，之前早期芯片的烧录保护引脚为 P1.0/P1.1。

## I.3 USB 型联机/脱机下载工具 U8W/U8W-Mini

U8W/U8W-Min 的应用范围可支持 STC 目前的全部系列的 MCU, Flash 程序空间和 EEPROM 数据空间不受限制。支持包括如下和即将推出的 STC 全系列芯片:



脱机下载工具可以在脱离电脑的情况下进行下载工作，可用于批量生产和远程升级。脱机下载板可支持自动增量、下载次数限制以及用户程序加密后传输等多种功能。

下图为 U8W 工具的正反面图以及 U8W-Mini 的正反面图：





另外还有如下的一些线材与工具相搭配使用, 如:

- (1) 两头公的 USB 连接线(如下图左所示) 及 USB-Micro 连接线(如下图右所示):



注意: 此 USB 线为我公司特别定制的 USB 加强线, 可确保直接用 USB 供电时能够下载成功。而市面上一些比较劣质的两头公的 USB 线, 内阻太大而导致压降很大(如 USB 空载时的电压为 5.0V 左右, 当使用劣质的 USB 线连接 U8W/U8W-Mini/U8/U8-Mini, 到我们的下载板上的电压可能降到 4.2V 或者更低, 从而导致芯片处于复位状态而无法成功下载)。

- (2) U8W/U8W-Mini 与用户系统连接的下载连接线(即 U8W/U8W-Mini 与用户板上的目标单片机的连接线), 如下图所示:



U8W/U8W-Mini 与  
用户系统各自独立  
供电的连接线

U8W/U8W-Mini 给  
用户系统供电的连  
接线

用户系统给  
U8W/U8W-Mini  
供电的连接线

### I.3.1 安装 U8W/U8W-Mini 驱动程序

U8W/U8W-Mini 下载板上使用了一颗 CH340 的 USB 转串口通用芯片。这样可以省去部分没有串口的电脑必须额外买一个 USB 转串口工具才可下载的麻烦。但 CH340 和其它 USB 转串口工具一样，在使用之前必须先安装驱动程序。

#### 通过下载 STC-ISP 软件包获取驱动程序

以下是 STC 官网 ([www.STCMCUDATA.com](http://www.STCMCUDATA.com)) 提供的 STC-ISP 软件包下载位置：

## ■ STC-ISP下载编程烧录软件

- ◆ STC-ISP软件V6.87K版

- ## ◆ STC开发/烧录工具说明

STC超强工具包,已含89系

使用该软件的Keil仿真设置向Keil中添加STC器件/头文件和仿真驱动


- ◆STC-ISP V6.87K请测试

- ### ◆ STC-ISP软件升级原因

- ◆ [STC-ISP V6.87K简化版](#)

下载后进行解压，CH340 的驱动安装包路径 stc-isp-15xx-v6.87K\USB to UART Driver\CH340\_CH341:

i > 下载 > stc-isp-15xx-v6.87K > USB to UART Driver > CH340\_CH341

| 名称   | 修改日期           |
|--|----------------|
|  ch341ser | 2020/5/9 15:03 |

通过 STC 的官方网站或在最新的 STC-ISP 下载软件中手动下载驱动程序

在 STC 的官方网站上或在最新的 STC-ISP 下载软件中手动下载驱动程序，驱动力的下载链接为：[U8 编程器 USB 转串口驱动](http://www.stcmcu.com/STCISP/CH341SER.exe)（<http://www.stcmcu.com/STCISP/CH341SER.exe>）。网站上及 STC-ISP 下载软件上的驱动地址如下图所示：

The screenshot shows the STC-ISP software interface with several key settings highlighted:

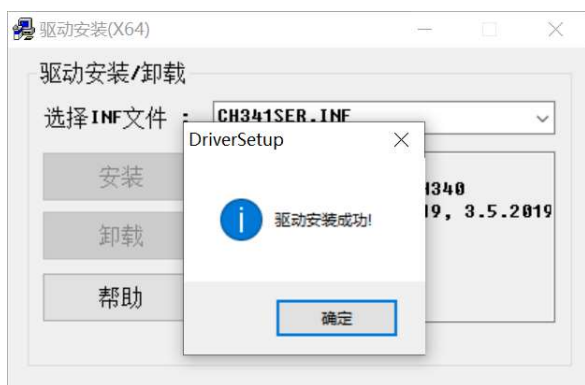
- 最低波特率** (Minimum Baud Rate) set to 2400.
- 最高波特率** (Maximum Baud Rate) set to 115200.
- 起始地址** (Start Address) set to 0x0000.
- 清除代码缓冲区** (Clear Code Buffer) checked.
- 清除EEPROM缓冲区** (Clear EEPROM Buffer) checked.
- 硬件选项** (Hardware Options) section includes:
  - USB脱机/联机** (USB Offline/Online).
  - 程序加密后传输** (Program Encryption After Transfer).
  - ID号** (ID Number).
- 将用户程序下载到U8/U7编程器以供脱机下载** (Download user program to U8/U7 programmer for offline download) selected.
- 清除U8/U7编程器中的用户脱机代码及数据** (Clear user offline code and data in U8/U7 programmer) selected.
- 将U8W/U8-SV/U8-3V设置为脱机下载主控芯片** (Set U8W/U8-SV/U8-3V as offline download master chip) selected.
- 设置U8/U7脱机下载时的最高波特率** (Set maximum baud rate during U8/U7 offline download) set to 115200.
- 脱机编程数量** (Offline programming quantity) set to 10000.
- 握手超时时间** (Handshake timeout time) set to 5.
- S-VCC输出电压** (S-VCC output voltage) set to 5.0 V.
- 上电延时时间** (Power-on delay time) set to 0.5 s.
- 使用U8/U7在线下载完成后持续给目标芯片供电** (Use U8/U7 online download to continue power supply to target chip after completion) unchecked.
- 使用U8脱机下载完成后持续给目标芯片供电** (Use U8 offline download to continue power supply to target chip after completion) unchecked.
- 使用U8脱机下载时关闭自动检测有无芯片功能** (Use U8 offline download to close automatic detection of whether there are chips) unchecked.
- 脱机下载器U8/U7工具使用说明** (Offline downloader U8/U7 tool usage instructions) link.
- 请下载U8/U7工具的USB转串口驱动，并安装** (Please download the USB-to-serial port driver for the U8/U7 tool and install it) link.
- 提供电流** (Supply current) set to 500mA.
- BIR保持时间** (BIR hold time) set to 40 ms.

## 安装 U8W/U8W-Mini 的驱动程序

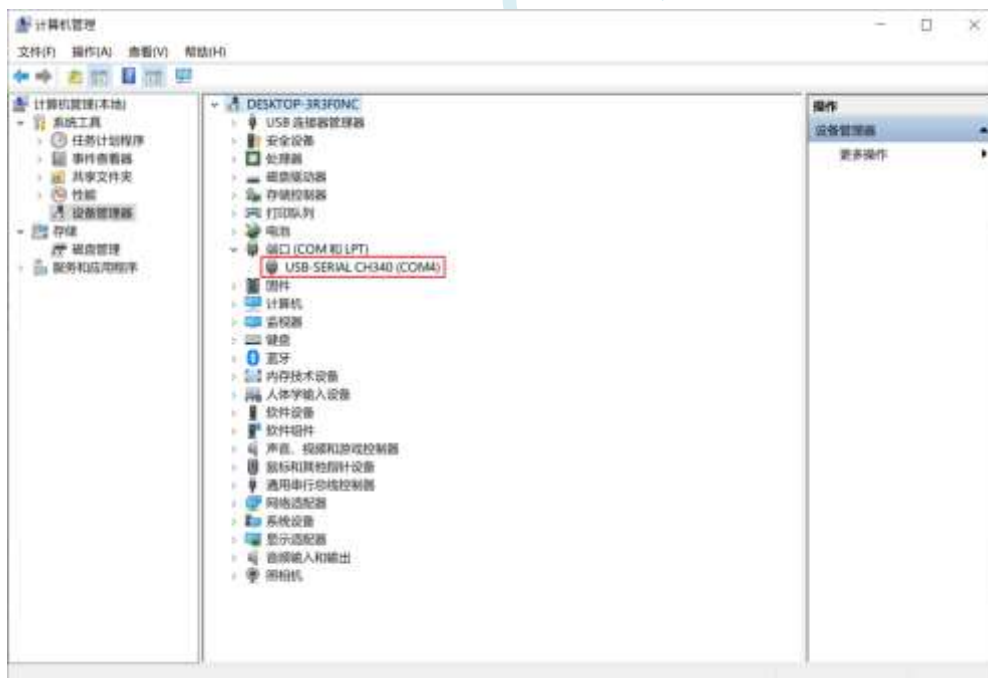
驱动程序下载到本机后，直接双击可执行程序并运行，出现下图所示的界面，点击“安装”按钮开始自动安装驱动：



然后弹出驱动安装成功对话框，点击“确定”按钮完成安装：



然后使用 STC 提供的 USB 连接线将 U8W/U8W-Mini 下载板连接到电脑，打开电脑的设备管理器，在端口设备类下面，如果有类似“USB-SERIAL CH340 (COMx)”的设备，就表示 U8W/U8W-Mini 可以正常使用了。如下图所示（不同的电脑，串口号可能会不同）：



注意：在后面使用 STC-ISP 下载软件时，选择的串口号必须选择与此相对应的串口号，如下图所示：

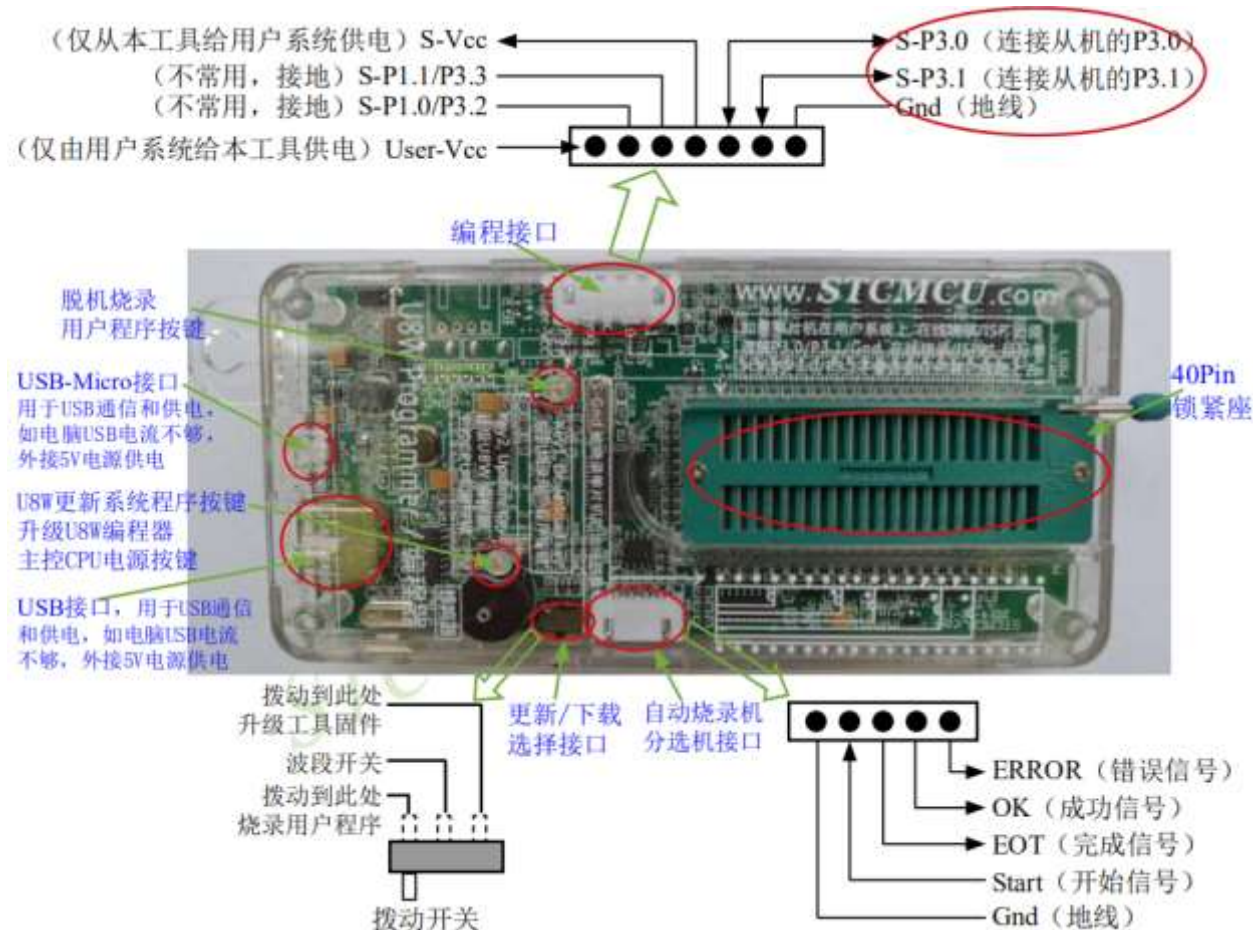


## I.3.2 U8W 的功能介绍

下面详细介绍 U8W 工具的各主要接口及功能:

如果单片机在用户系统上, 在线烧录/ISP 时必须连接 P3.0/P3.1/Gnd, 在线烧录/ISP 时, 目标单片机的 P3.0/P3.1 不要连到任何其他线路上去





**编程接口:** 根据不同的供电方式, 使用不同的下载连接线连接 U8W 下载板和用户系统。

**U8W 更新系统程序按键:** 用于更新 U8W 工具, 当有新版本的 U8W 固件时, 需要按下此按键对 U8W 的主控芯片进行更新 (**注意: 必须先将更新/下载选择接口上的拨动开关拨动到升级工具固件**)。

**脱机下载用户程序按钮:** 开始脱机下载按钮。首先 PC 将脱机代码下载到 U8W 板上, 然后使用下载连接线将用户系统连接到 U8W, 再按下此按钮即可开始脱机下载 (每次上电时也会立即开始下载用户代码)。

**更新/下载选择接口:** 当需要对 U8W 的底层固件进行升级时, 需将此拨动开关拨到升级工具固件处, 当需通过 U8W 对目标芯片进行烧录程序, 则需将拨动开关拨到烧录用户程序处。

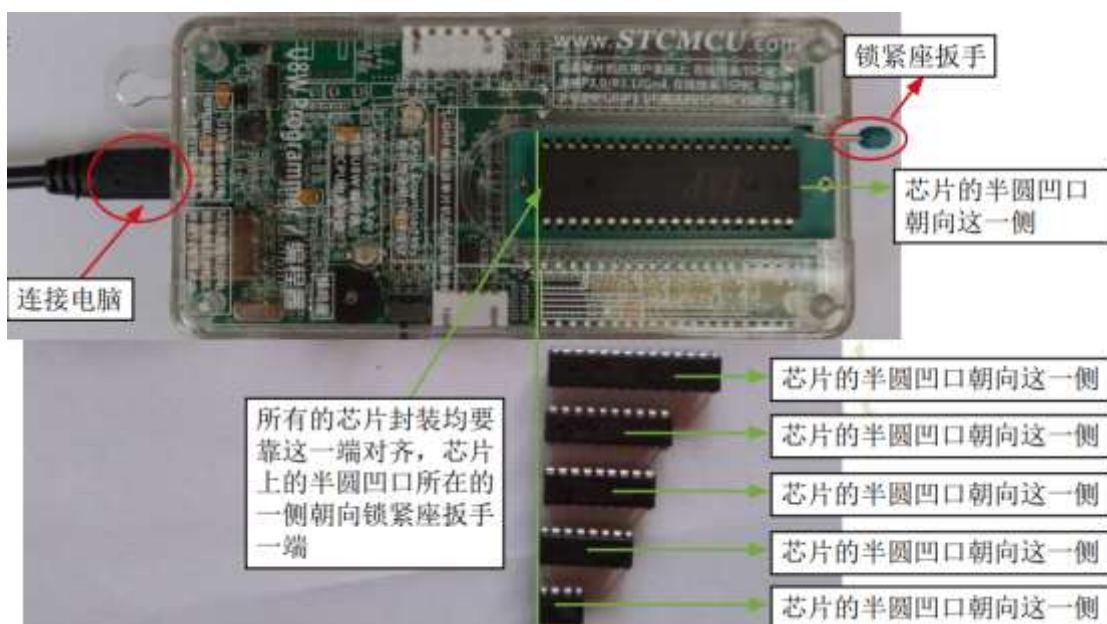
(拨动开关连接方式请参考上图)

**自动烧录机/分选机接口:** 是用于控制自动烧录机/分选机进行自动生产的控制接口。

### I.3.3 U8W 的在线联机下载使用说明

目标芯片安装于 U8W 锁紧座上并由 U8W 连接电脑进行在线联机下载

首先使用 STC 提供的 USB 连接线将 U8W 连接电脑, 再将目标单片机按如下图所示的方向安装在 U8W 上:



然后使用 STC-ISP 下载软件下载程序, 步骤如下:



- 1 选择单片机型号;
- 2 选择引脚数, 芯片直接安装于 U8W 上下载时, 一定要注意选择正确的引脚数, 否则将会下载失败;
- 3 选择 U8W 所对应的串口号;
- 4 打开目标文件 (HEX 格式或者 BIN 格式);
- 5 设置硬件选项;
- 6 点击“下载/编程”按钮开始烧录;
- 7 显示烧录过程的步骤信息, 烧录完成提示“操作成功!”。

当信息框中有输出下载板的版本号信息以及外挂 Flash 的相应信息时, 表示已正确检测到 U8W 下载工具。

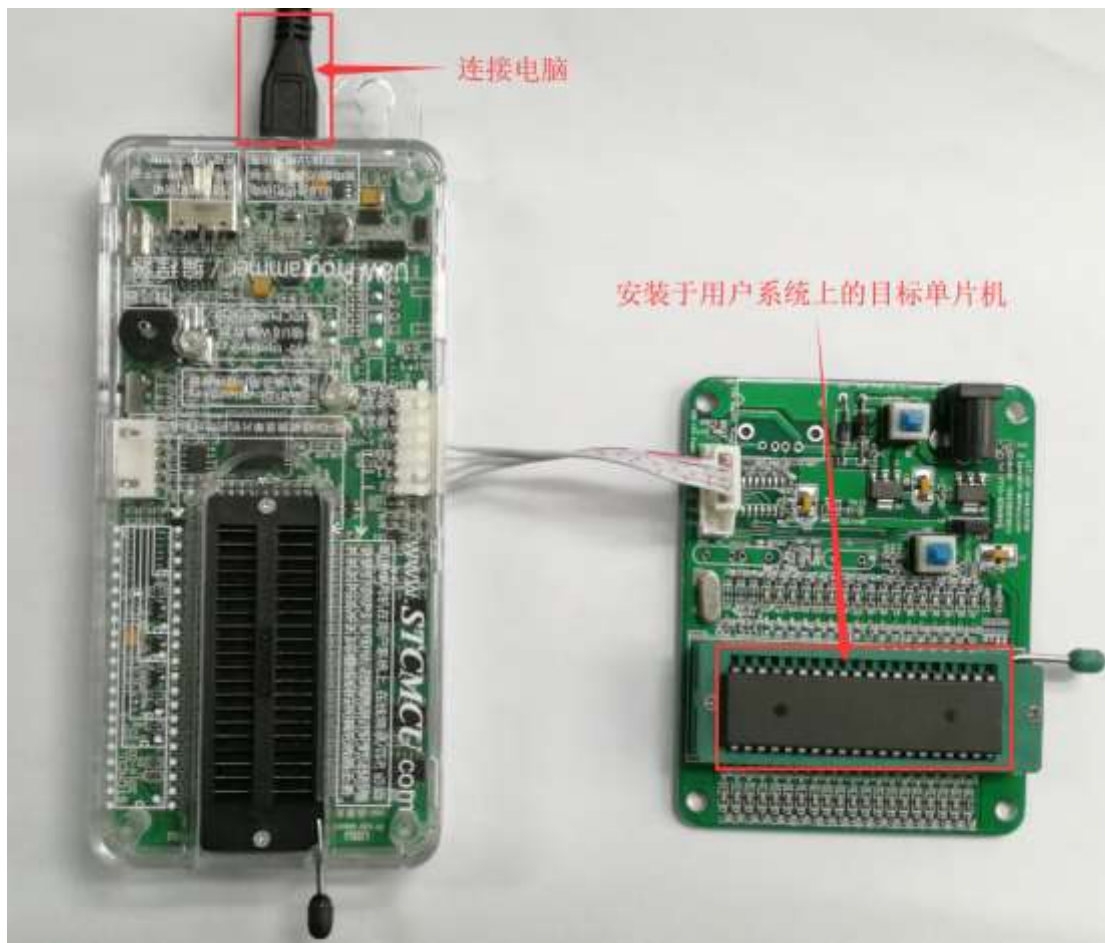


下载的过程中, U8W 下载工具上的 4 个 LED 会以跑马灯的模式显示。下载完成后, 若下载成功, 则 4 个 LED 会同时亮、同时灭; 若下载失败, 则 4 个 LED 全部不亮。

建议用户用最新版本的 STC-ISP 下载软件(请随时留意 STC 官方网站 <http://www.STCMCUDATA.com> 中 STC-ISP 下载软件的更新, 强烈建议用户在官方网站 <http://www.STCMCUDATA.com> 中下载最新版本的软件使用)。

## 目标芯片通过用户系统引线连接 U8W 并由 U8W 连接电脑进行在线联机下载

首先使用 STC 提供的 USB 连接线将 U8W 连接电脑, 再将 U8W 通过下载线与用户系统的目标单片机相连接, 连接方式如下图所示:



然后使用 STC-ISP 下载软件下载程序, 步骤如下:

1. 选择单片机型号;
2. 选择 U8W 所对应的串口号;
3. 打开目标文件 (HEX 格式或者 BIN 格式);
4. 设置硬件选项;
5. 点击“下载/编程”按钮开始烧录;
6. 显示烧录过程的步骤信息, 烧录完成提示“操作成功!”。





当信息框中有输出下载板的版本号信息以及外挂 Flash 的相应信息时,表示已正确检测到 U8W 下载工具。下载的过程中, U8W 下载工具上的 4 个 LED 会以跑马灯的模式显示。下载完成后,若下载成功,则 4 个 LED 会同时亮、同时灭;若下载失败,则 4 个 LED 全部不亮。

建议用户用最新版本的 STC-ISP 下载软件(请随时留意 STC 官方网站 <http://www.STCMCUDATA.com> 中 STC-ISP 下载软件的更新,强烈建议用户在官方网站 <http://www.STCMCUDATA.com> 中下载最新版本的软件使用)。

### I.3.4 U8W 的脱机下载使用说明

目标芯片安装于 U8W 座锁紧上并通过 USB 连接电脑给 U8W 供电进行脱机下载

使用 USB 给 U8W 供电从而进行脱机下载的步骤如下:

(1) 使用 STC 提供的 USB 连接线将 U8W 下载板连接到电脑,如下图:



(2) 在 STC-ISP 下载软件中按如下图所示的步骤进行设置:

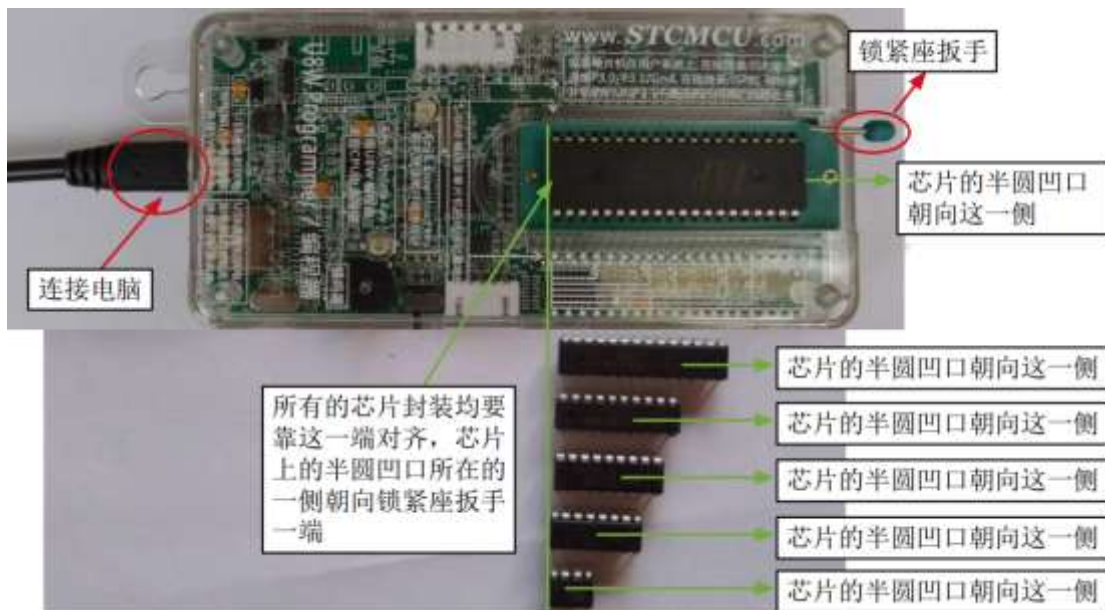


1. 选择单片机型号；
2. 选择引脚数，芯片直接安装于 U8W 上下载时，一定要注意选择正确的引脚数，否则将会下载失败；
3. 选择 U8W 所对应的串口号；
4. 打开目标文件（HEX 格式或者 BIN 格式）；
5. 设置硬件选项；
6. 选择“U8W 脱机/联机”标签，设置脱机编程选项，注意 S-VCC 输出电压与目标芯片工作电压匹配；点击“将用户程序下载到 U8/U7 编程器以供脱机下载”按钮；
7. 显示设置过程的步骤信息，设置完成提示“操作成功！”。

按照上图的步骤，操作完成后，若下载成功则表示用户代码和相关的设置选项都已下载到 U8W 下载工具中。

建议用户用最新版本的 **STC-ISP** 下载软件（请随时留意 **STC** 官方网站 <http://www.STCMCUDATA.com> 中 **STC-ISP** 下载软件的更新，强烈建议用户在官方网站 <http://www.STCMCUDATA.com> 中下载最新版本的软件使用）。

(3) 再将目标单片机如下图所示的方向放在 U8W 下载工具, 如下图所示:



(4) 然后按下如下图所示的按钮后松开，即可开始脱机下载：



下载的过程中，U8W 下载工具上的 4 个 LED 会以跑马灯的模式显示。下载完成后，若下载成功，则 4 个 LED 会同时亮、同时灭；若下载失败，则 4 个 LED 全部不亮。

#### 脱机下载即插即用烧录功能介绍：

1. 以上步骤完成 (1)、(2) 步之后 U8W 连接电脑上电时默认处于即插即用烧录状态；
2. 按照第 (3) 步指示将芯片放入烧录座，在锁紧座扳手的同时，U8W 会自动开始烧录；
3. 通过指示灯显示烧录过程跟烧录结果；
4. 烧录完成后松开座扳手，取出芯片；
5. 重复 2, 3, 4 步骤可进行连续烧录，省掉按按钮触发烧录的动作。

目标芯片由用户系统引线连接 U8W 并通过 USB 连接电脑给 U8W 供电进行脱机下载

使用 USB 给 U8W 供电从而进行脱机下载的步骤如下：

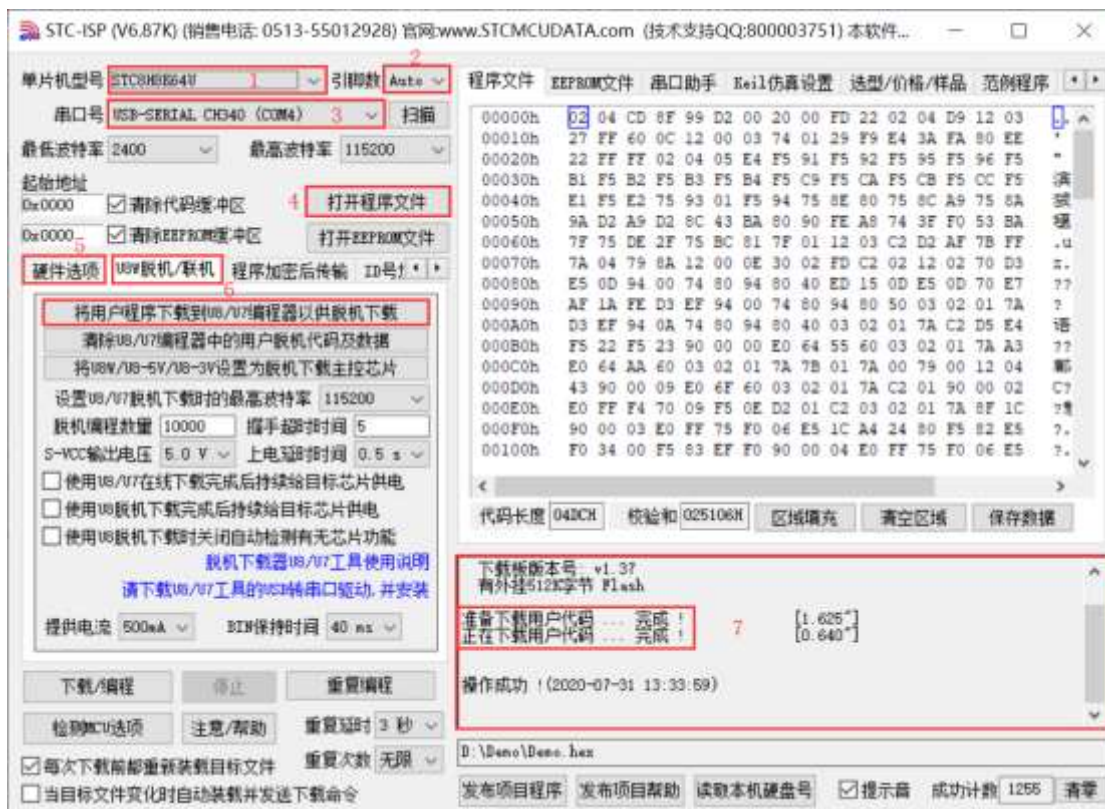
- (1) 使用 STC 提供的 USB 连接线将 U8W 下载板连接到电脑，如下图：





(2) 在 STC-ISP 下载软件中按如下图所示的步骤进行设置:

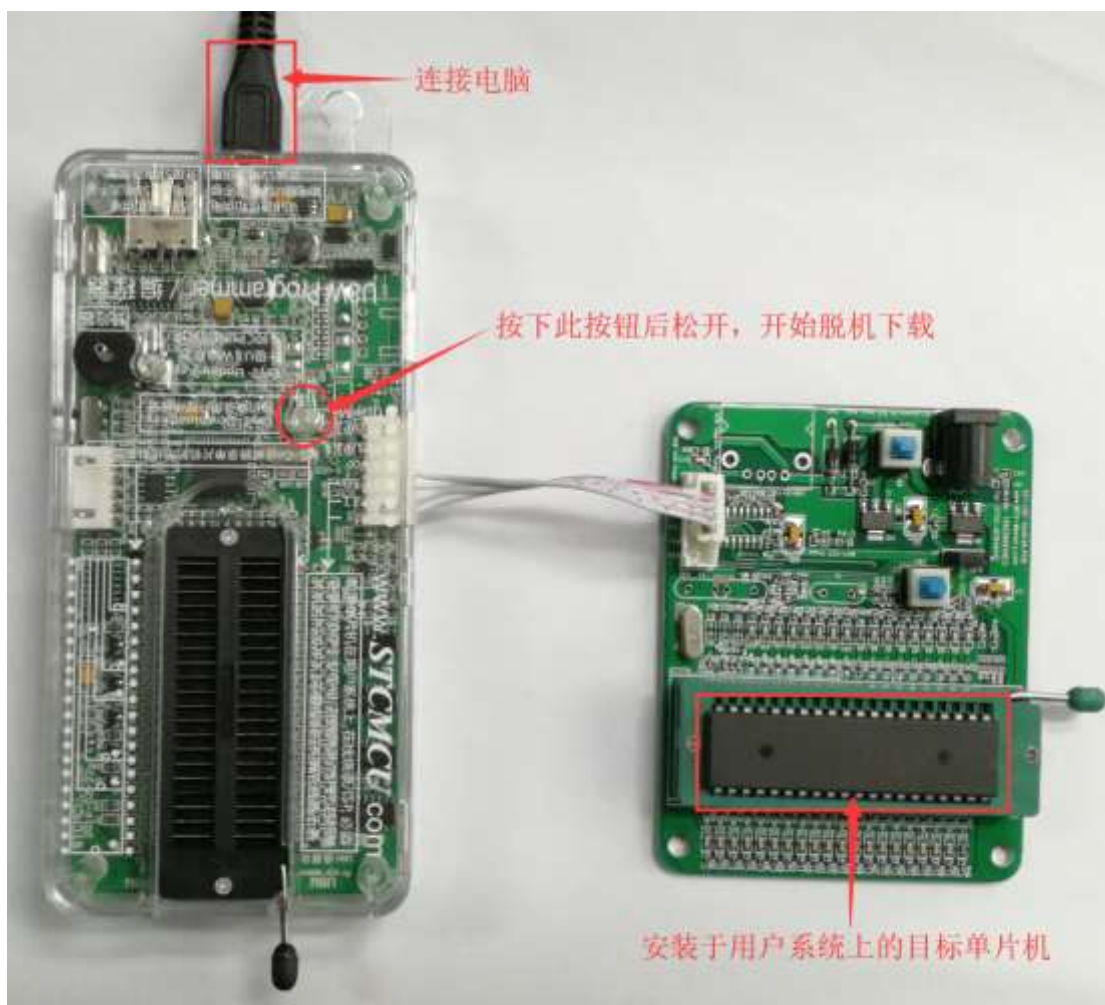
建议用户用最新版本的 STC-ISP 下载软件(请随时留意 STC 官方网站 <http://www.STCMCUDATA.com> 中 STC-ISP 下载软件的更新, 强烈建议用户在官方网站 <http://www.STCMCUDATA.com> 中下载最新版本的软件使用)。



1. 选择单片机型号;
2. 选择引脚数, 芯片直接安装于 U8W 上下载时, 一定要注意选择正确的引脚数, 否则将会下载失败;
3. 选择 U8W 所对应的串口号;
4. 打开目标文件 (HEX 格式或者 BIN 格式);
5. 设置硬件选项;
6. 选择“U8W 脱机/联机”标签, 设置脱机编程选项, 注意 S-VCC 输出电压与目标芯片工作电压匹配; 点击“将用户程序下载到 U8/U7 编程器以供脱机下载”按钮;
7. 显示设置过程的步骤信息, 设置完成提示“操作成功!”。

按照上图的步骤, 操作完成后, 若下载成功则表示用户代码和相关的设置选项都已下载到 U8W 下载工具中。

(3) 然后使用连接线连接电脑、将 U8W 下载工具以及用户系统 (目标单片机) 如下图所示的方式连接起来, 并按下图所示的按钮后松开, 即可开始脱机下载:



下载的过程中, U8W 下载工具上的 4 个 LED 会以跑马灯的模式显示。下载完成后, 若下载成功, 则 4 个 LED 会同时亮、同时灭; 若下载失败, 则 4 个 LED 全部不亮。

### 目标芯片由用户系统引线连接 U8W 并通过用户系统给 U8W 供电进行脱机下载

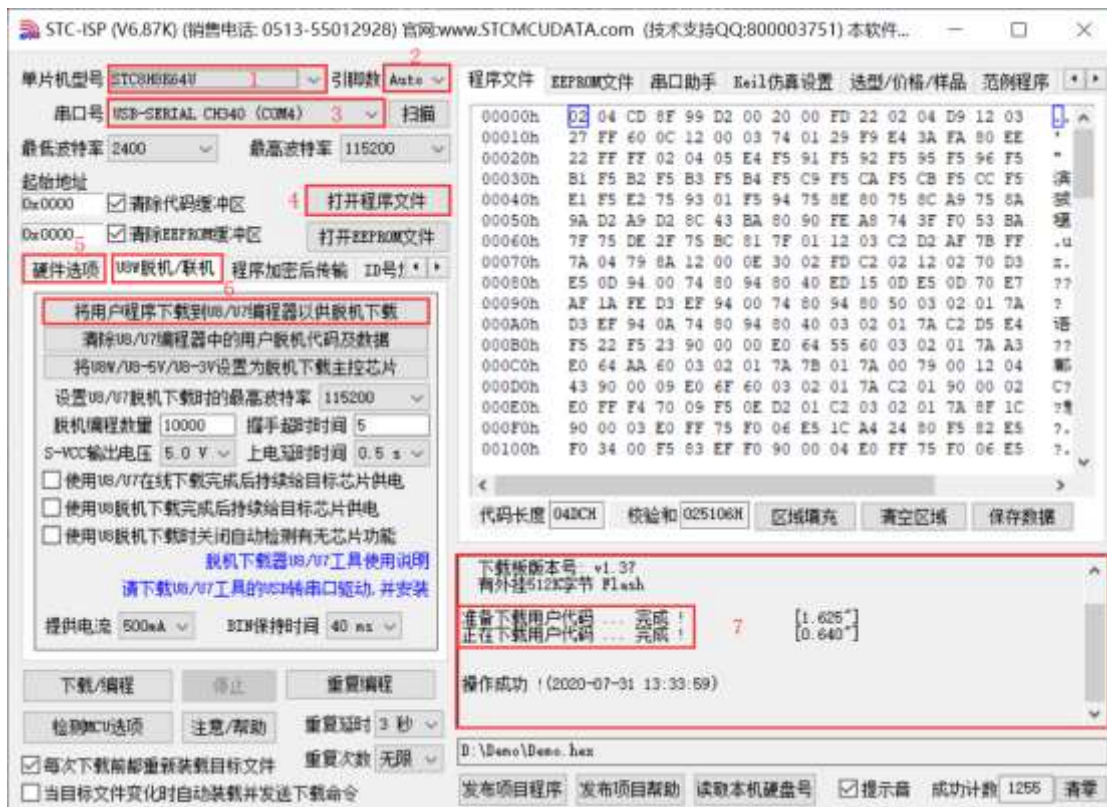
(1) 首先使用 STC 提供的 USB 连接线将 U8W 下载板连接到电脑, 如下图:



(2) 在 STC-ISP 下载软件中按如下图所示的步骤进行设置:

建议用户用最新版本的 STC-ISP 下载软件(请随时留意 STC 官方网站 <http://www.STCMCUDATA.com> 中 STC-ISP 下载软件的更新, 强烈建议用户在官方网站 <http://www.STCMCUDATA.com> 中下载最新版本的

软件使用)。

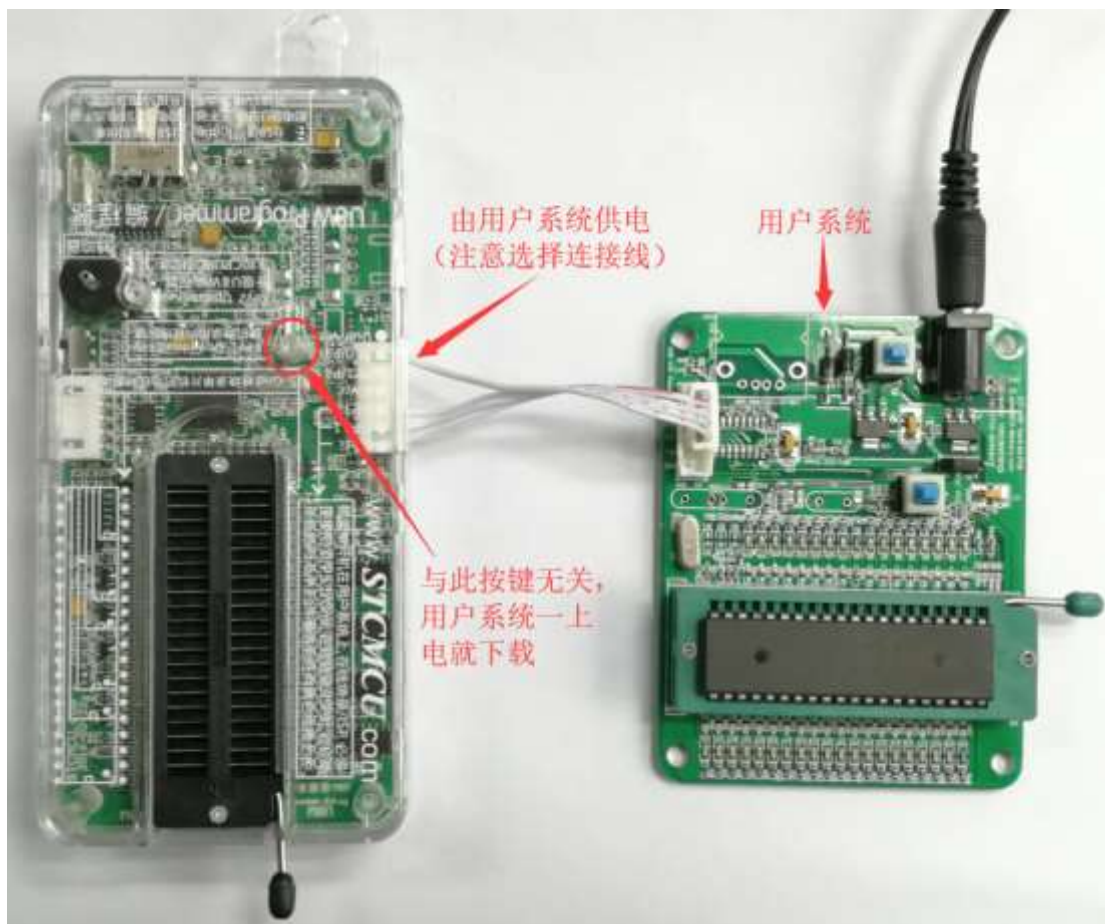


1. 选择单片机型号；
2. 选择引脚数，芯片直接安装于 U8W 上下载时，一定要注意选择正确的引脚数，否则将会下载失败；
3. 选择 U8W 所对应的串口号；
4. 打开目标文件（HEX 格式或者 BIN 格式）；
5. 设置硬件选项；
6. 选择“U8W 脱机/联机”标签，设置脱机编程选项，注意 S-VCC 输出电压与目标芯片工作电压匹配；点击“将用户程序下载到 U8/U7 编程器以供脱机下载”按钮；
7. 显示设置过程的步骤信息，设置完成提示“操作成功！”。

按照上图的步骤，操作完成后，若下载成功则表示用户代码和相关的设置选项都已下载到 U8W 下载工具中。

(3) 然后按下图所示的方式连接 U8W 与用户系统，给用户系统供电，即可开始脱机下载：





下载的过程中, U8W 下载工具上的 4 个 LED 会以跑马灯的模式显示。下载完成后, 若下载成功, 则 4 个 LED 会同时亮、同时灭; 若下载失败, 则 4 个 LED 全部不亮。

## 目标芯片由用户系统引线连接 U8W 且 U8W 与用户系统各自独立供电进行脱机下载

(1) 首先使用 STC 提供的 USB 连接线将 U8W 下载板连接到电脑, 如下图:



(2) 在 STC-ISP 下载软件中按如下图所示的步骤进行设置:

建议用户用最新版本的 STC-ISP 下载软件 (请随时留意 STC 官方网站 <http://www.STCMCUDATA.com> 中 STC-ISP 下载软件的更新, 强烈建议用户在官方网站 <http://www.STCMCUDATA.com> 中下载最新版本的

软件使用)。

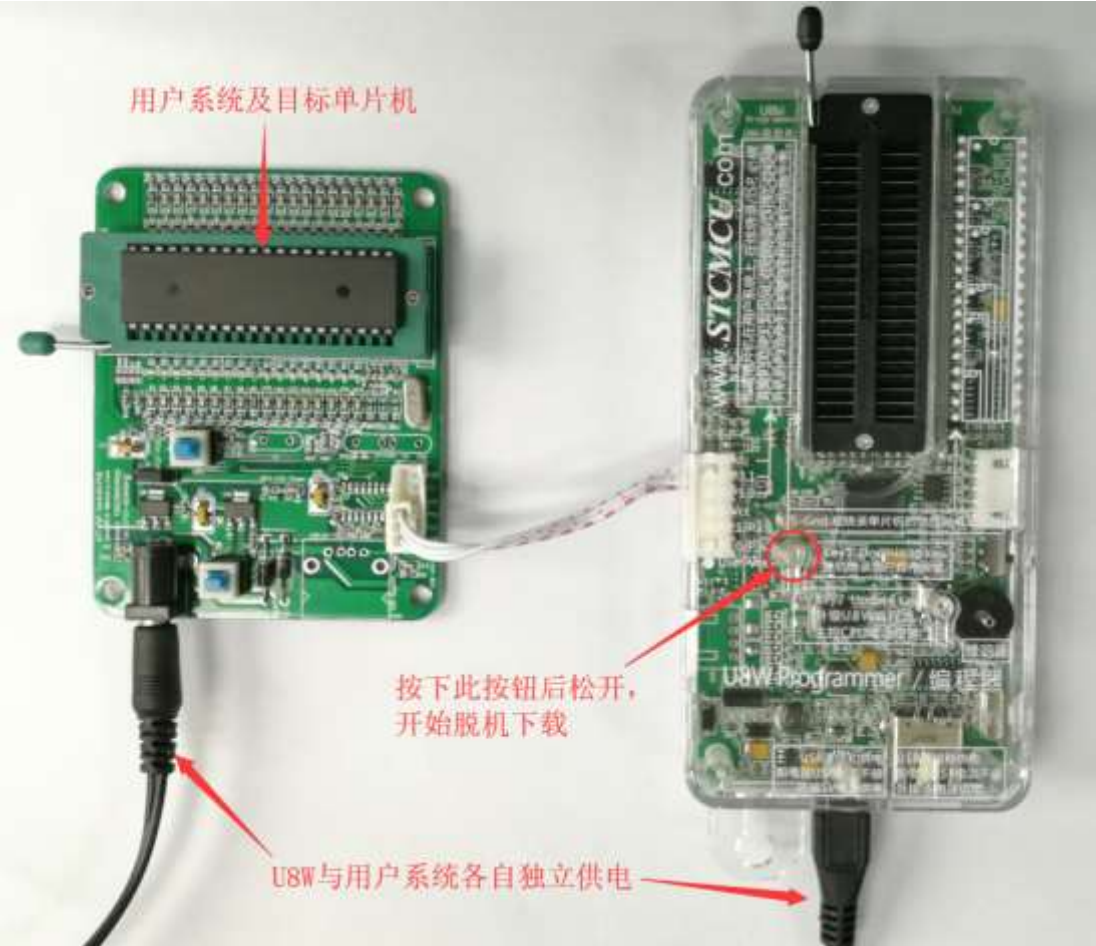


1. 选择单片机型号；
2. 选择引脚数，芯片直接安装于 U8W 上下载时，一定要注意选择正确的引脚数，否则将会下载失败；
3. 选择 U8W 所对应的串口号；
4. 打开目标文件（HEX 格式或者 BIN 格式）；
5. 设置硬件选项；
6. 选择“U8W 脱机/联机”标签，设置脱机编程选项，注意 S-VCC 输出电压与目标芯片工作电压匹配；点击“将用户程序下载到 U8/U7 编程器以供脱机下载”按钮；
7. 显示设置过程的步骤信息，设置完成提示“操作成功！”。

按照上图的步骤，操作完成后，若下载成功则表示用户代码和相关的设置选项都已下载到 U8W 下载工具中。

(3) 然后按下图所示的方式连接 U8W 与用户系统，并将图中所示按钮先按下后松开，准备开始脱机下载，最后给用户系统上电/开电源，下载用户程序正式开始：

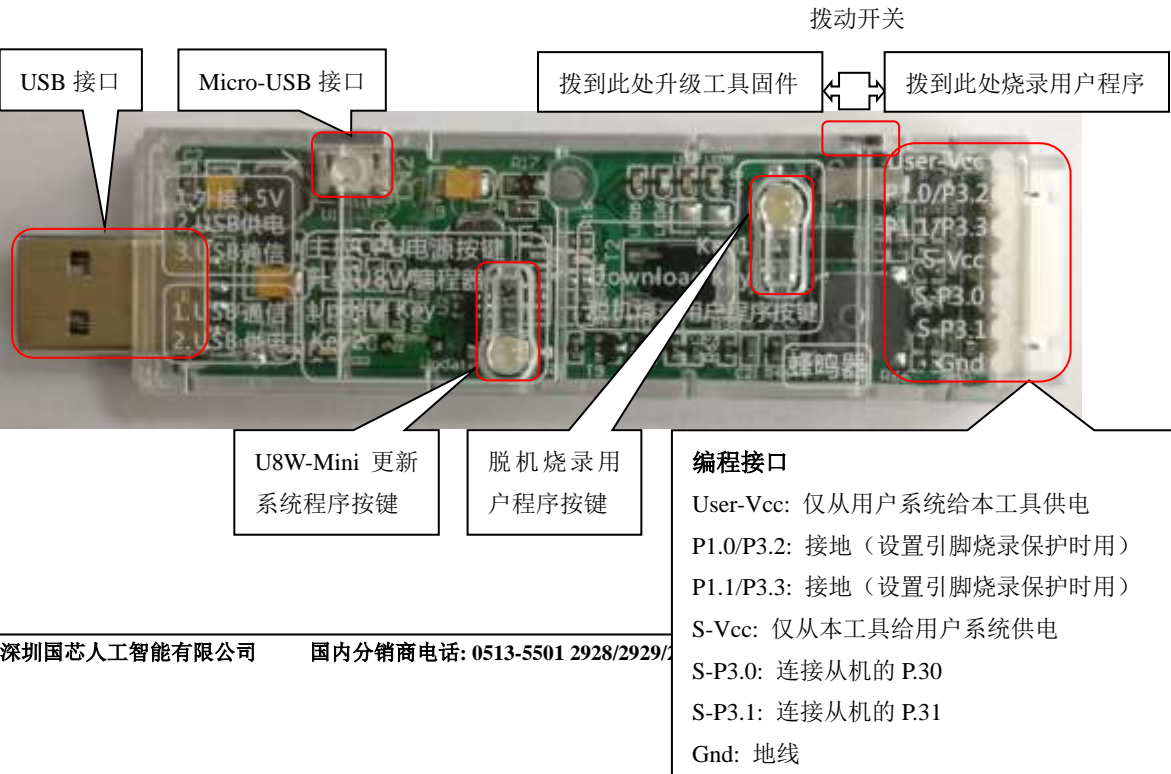




下载的过程中, U8W 下载工具上的 4 个 LED 会以跑马灯的模式显示。下载完成后, 若下载成功, 则 4 个 LED 会同时亮、同时灭; 若下载失败, 则 4 个 LED 全部不亮。

### I.3.5 U8W-Mini 的功能介绍

下面详细介绍 U8W-Mini 工具的各主要接口及功能:



**编程接口:** 根据不同的供电方式, 使用不同的下载连接线连接 U8W-Mini 下载板和用户系统。

**U8W-Mini 更新系统程序按键:** 用于更新 U8W-Mini 工具, 当有新版本的 U8W 固件时, 需要按下此按键对 U8W-Mini 的主控芯片进行更新 (**注意: 必须先将更新/下载选择接口上的拨动开关拨动到升级工具固件**)。

**脱机下载用户程序按钮:** 开始脱机下载按钮。首先 PC 将脱机代码下载到 U8W-Mini 上, 然后使用下载连接线将用户系统连接到 U8W-Mini, 再按下此按钮即可开始脱机下载 (每次上电时也会立即开始下载用户代码)。

**更新/下载选择接口:** 当需要对 U8W-Mini 的底层固件进行升级时, 需将此拨动开关拨动到升级工具固件处, 当需通过 U8W-Mini 对目标芯片进行烧录程序, 则需将拨动开关拨动到烧录用户程序处。(拨动开关连接方式请参考上图)

**USB 接口:** USB 接口与 Micro-USB 接口是相同的功能, 用户根据需要连接其中一个接口到电脑即可。

### I.3.6 U8W-Mini 的在线联机下载使用说明

目标芯片通过用户系统引线连接 U8W-Mini 并由 U8W-Mini 连接电脑进行在线联机下载

首先使用 STC 提供的 USB 连接线将 U8W-Mini 连接电脑, 再将 U8W-Mini 通过下载线与用户系统的目标单片机相连接, 连接方式如下图所示:



然后使用 STC-ISP 下载软件下载程序, 步骤如下:



1. 选择单片机型号;
2. 选择引脚数, 芯片直接安装于 U8W-Mini 上下载时, 一定要注意选择正确的引脚数, 否则将会下载失败;
3. 选择 U8W-Mini 所对应的串口号;
4. 打开目标文件 (HEX 格式或者 BIN 格式);
5. 设置硬件选项;
6. 点击“下载/编程”按钮开始烧录;
7. 显示烧录过程的步骤信息, 烧录完成提示“操作成功!”。

当信息框中有输出下载板的版本号信息以及外挂 Flash 的相应信息时, 表示已正确检测到 U8W-Mini 下载工具。

下载的过程中, U8W-Mini 下载工具上的 4 个 LED 会以跑马灯的模式显示。下载完成后, 若下载成功, 则 4 个 LED 会同时亮、同时灭; 若下载失败, 则 4 个 LED 全部不亮。

建议用户用最新版本的 STC-ISP 下载软件 (请随时留意 STC 官方网站 <http://www.STCMCUDATA.com> 中 STC-ISP 下载软件的更新, 强烈建议用户在官方网站 <http://www.STCMCUDATA.com> 中下载最新版本的软件使用)。

### I.3.7 U8W-Mini 的脱机下载使用说明

目标芯片由用户系统引线连接 U8W-Mini 并通过 USB 连接电脑给 U8W-Mini 供电进行脱机下载

使用 USB 给 U8W-Mini 供电从而进行脱机下载的步骤如下:

- (1) 使用 STC 提供的 USB 连接线将 U8W-Mini 下载板连接到电脑, 如下图:





(2) 在 STC-ISP 下载软件中按如下图所示的步骤进行设置:

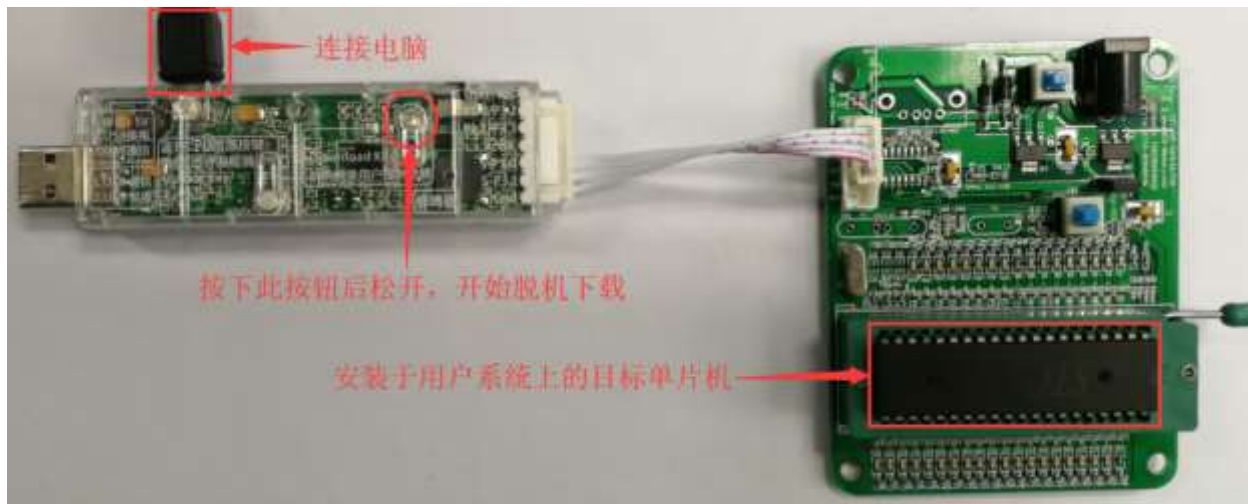


1. 选择单片机型号;
2. 选择引脚数, 芯片直接安装于 U8W-Mini 上下载时, 一定要注意选择正确的引脚数, 否则将会下载失败;
3. 选择 U8W-Mini 所对应的串口号;
4. 打开目标文件 (HEX 格式或者 BIN 格式);
5. 设置硬件选项;
6. 选择“U8W 脱机/联机”标签, 设置脱机编程选项, 注意 S-VCC 输出电压与目标芯片工作电压匹配; 点击“将用户程序下载到 U8/U7 编程器以供脱机下载”按钮;
7. 显示设置过程的步骤信息, 设置完成提示“操作成功!”。

按照上图的步骤, 操作完成后, 若下载成功则表示用户代码和相关的设置选项都已下载到 U8W-Mini 下载工具中。

建议用户用最新版本的 STC-ISP 下载软件(请随时留意 STC 官方网站 <http://www.STCMCUDATA.com> 中 STC-ISP 下载软件的更新, 强烈建议用户在官方网站 <http://www.STCMCUDATA.com> 中下载最新版本的软件使用)。

(3) 然后使用连接线连接电脑、将 U8W-Mini 下载工具以及用户系统(目标单片机)如下图所示的方式连接起来, 并按下图所示的按钮后松开, 即可开始脱机下载:



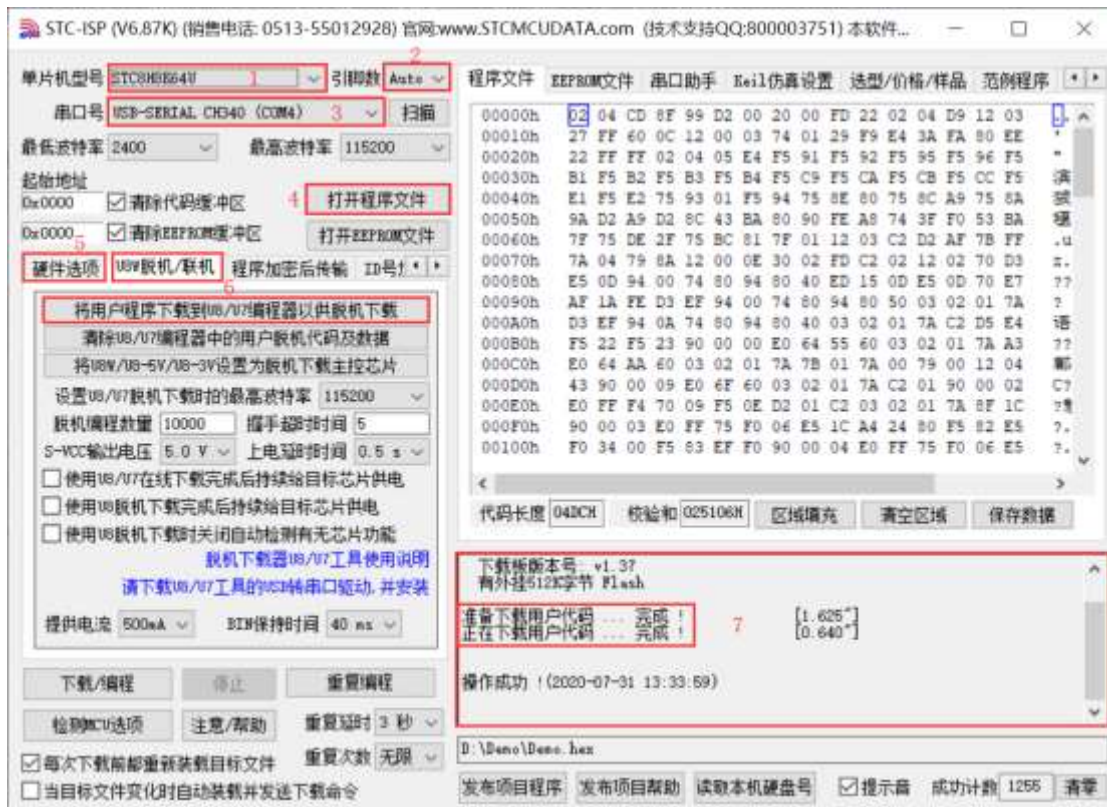
下载的过程中, U8W-Mini 下载工具上的 4 个 LED 会以跑马灯的模式显示。下载完成后, 若下载成功, 则 4 个 LED 会同时亮、同时灭; 若下载失败, 则 4 个 LED 全部不亮。

目标芯片由用户系统引线连接 U8W-Mini 并通过用户系统给 U8W-Mini 供电进行脱机下载

(1) 首先使用 STC 提供的 USB 连接线将 U8W-Mini 下载板连接到电脑, 如下图:



(2) 在 STC-ISP 下载软件中按如下图所示的步骤进行设置:



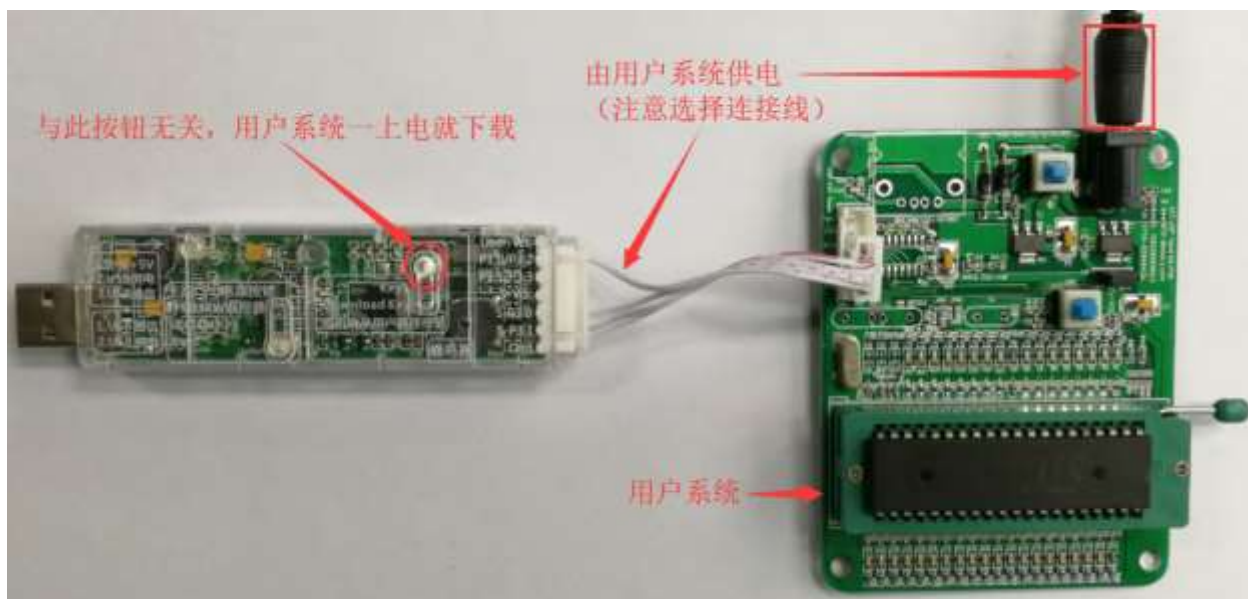
1. 选择单片机型号;
2. 选择引脚数, 芯片直接安装于 U8W-Mini 上下载时, 一定要注意选择正确的引脚数, 否则将会下载失败;
3. 选择 U8W-Mini 所对应的串口号;
4. 打开目标文件 (HEX 格式或者 BIN 格式);
5. 设置硬件选项;
6. 选择“U8W 脱机/联机”标签, 设置脱机编程选项, 注意 S-VCC 输出电压与目标芯片工作电压匹配; 点击“将用户程序下载到 U8/U7 编程器以供脱机下载”按钮;
7. 显示设置过程的步骤信息, 设置完成提示“操作成功!”。

按照上图的步骤, 操作完成后, 若下载成功则表示用户代码和相关的设置选项都已下载到 U8W-Mini 下载工具中。

建议用户用最新版本的 STC-ISP 下载软件 (请随时留意 STC 官方网站 <http://www.STCMCUDATA.com> 中 STC-ISP 下载软件的更新, 强烈建议用户在官方网站 <http://www.STCMCUDATA.com> 中下载最新版本的软件使用)。

(3) 然后按下图所示的方式连接 U8W-Mini 与用户系统, 用户系统一上电就开始脱机下载:





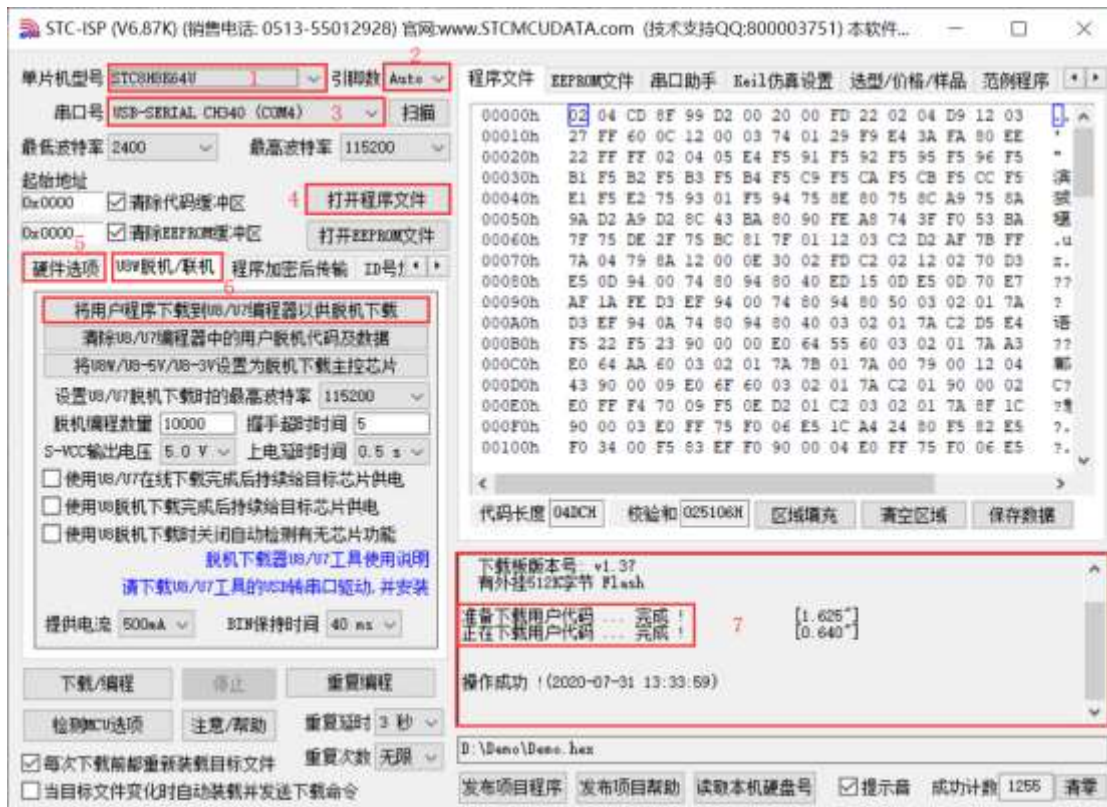
下载的过程中，U8W-Mini 下载工具上的 4 个 LED 会以跑马灯的模式显示。下载完成后，若下载成功，则 4 个 LED 会同时亮、同时灭；若下载失败，则 4 个 LED 全部不亮。

目标芯片由用户系统引线连接 U8W-Mini 且 U8W-Mini 与用户系统各自独立供电进行脱机下载

(1) 首先使用 STC 提供的 USB 连接线将 U8W-Mini 下载板连接到电脑，如下图：



(2) 在 STC-ISP 下载软件中按如下图所示的步骤进行设置：



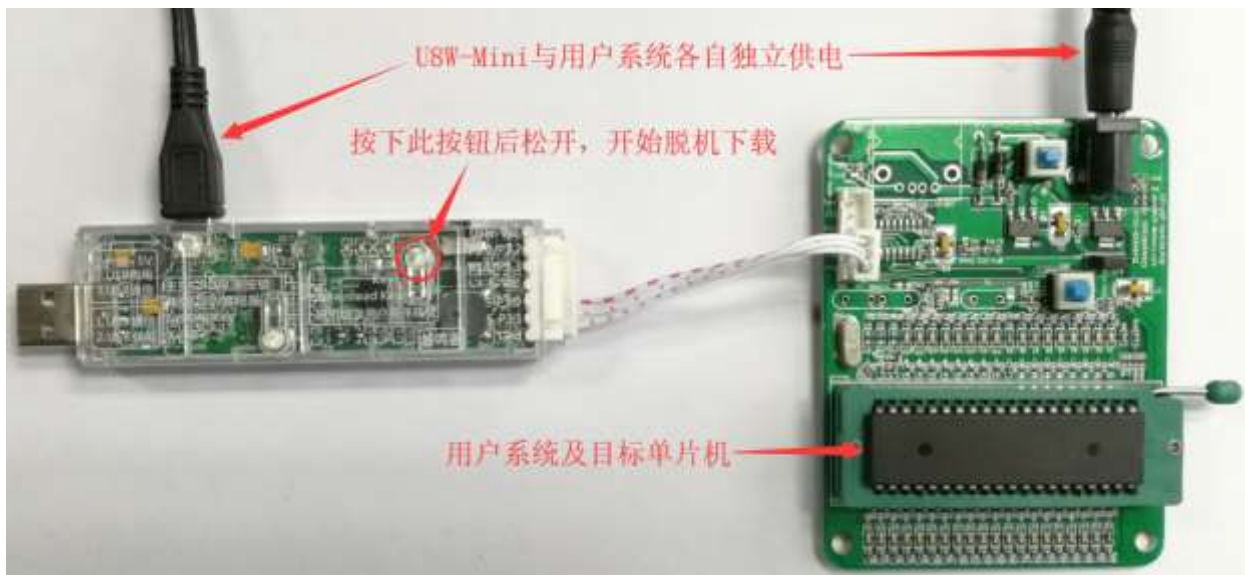
1. 选择单片机型号;
2. 选择引脚数, 芯片直接安装于 U8W-Mini 上下载时, 一定要注意选择正确的引脚数, 否则将会下载失败;
3. 选择 U8W-Mini 所对应的串口号;
4. 打开目标文件 (HEX 格式或者 BIN 格式);
5. 设置硬件选项;
6. 选择“U8W 脱机/联机”标签, 设置脱机编程选项, 注意 S-VCC 输出电压与目标芯片工作电压匹配; 点击“将用户程序下载到 U8/U7 编程器以供脱机下载”按钮;
7. 显示设置过程的步骤信息, 设置完成提示“操作成功!”。

按照上图的步骤, 操作完成后, 若下载成功则表示用户代码和相关的设置选项都已下载到 U8W-Mini 下载工具中。

建议用户用最新版本的 STC-ISP 下载软件 (请随时留意 STC 官方网站 <http://www.STCMCUDATA.com> 中 STC-ISP 下载软件的更新, 强烈建议用户在官方网站 <http://www.STCMCUDATA.com> 中下载最新版本的软件使用)。

(3) 然后按下图所示的方式连接 U8W-Mini 与用户系统, 并将图中所示按钮先按下后松开, 准备开始脱机下载, 最后给用户系统上电/开电源, 下载用户程序正式开始:

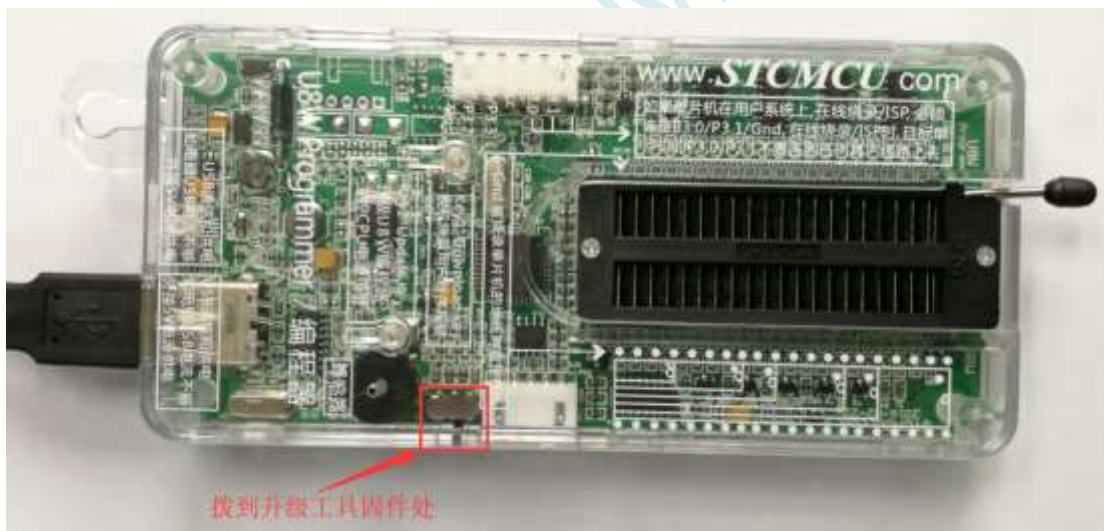




下载的过程中, U8W-Mini 下载工具上的 4 个 LED 会以跑马灯的模式显示。下载完成后, 若下载成功, 则 4 个 LED 会同时亮、同时灭; 若下载失败, 则 4 个 LED 全部不亮。

### I.3.8 制作/更新 U8W/U8W-Mini

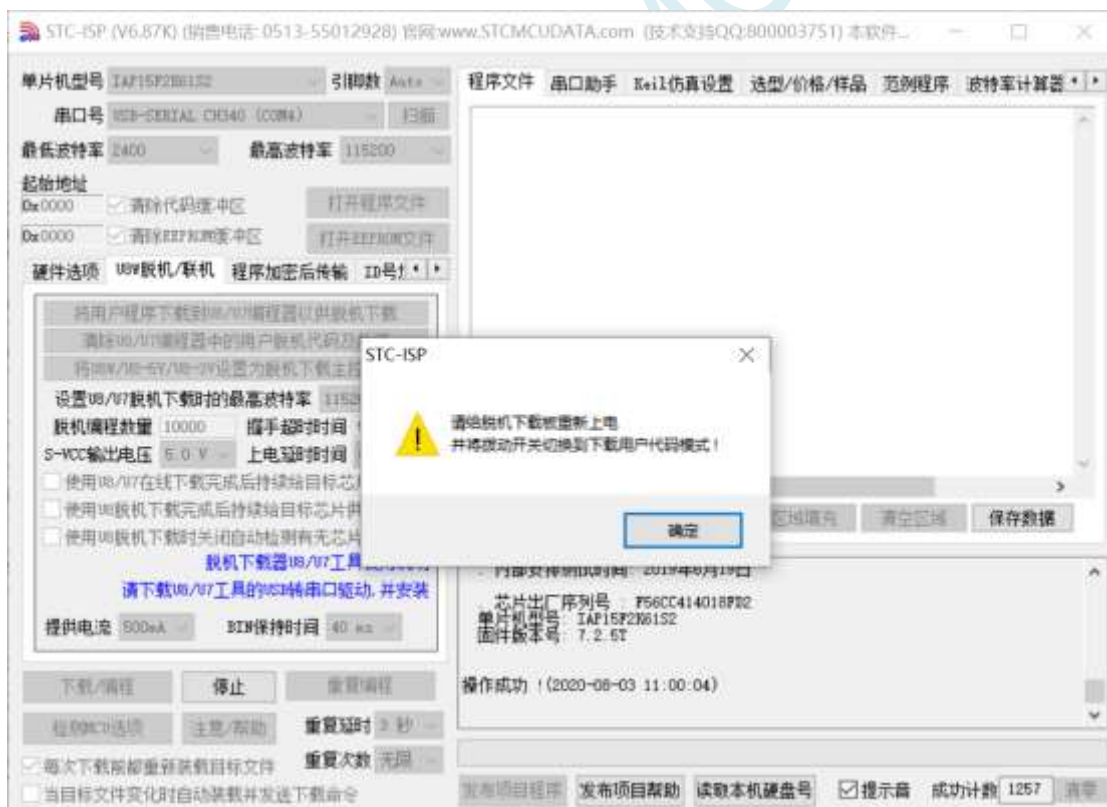
制作 U8W/U8W-Mini 下载母片的过程类似, 为节约篇幅, 下文以 U8W 为例, 详述如何制作 U8W 下载母片。在制作 U8W 下载母片之前需要将 U8W 下载板的“更新/下载选择接口”拨到“升级工具固件”, 如下图所示:



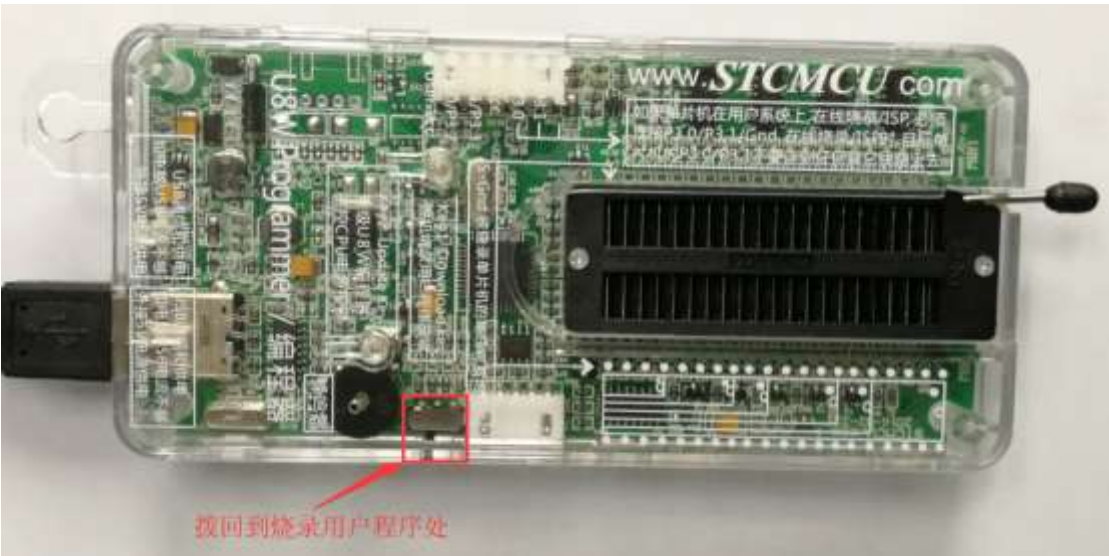
然后在 STC-ISP 下载程序中的“U8W 脱机/联机”页面中点击“将 U8W/U8-5V/U8-3V 设置为脱机下载主控芯片”按钮, 如下图: (注意: 一定要选择 U8W 所对应的串口)



在出现如下画面表示 U8W 控制芯片制作完成:



制作完成后, 一定不要忘记将 U8W 的“更新/下载选择接口”拨回到“烧录用户程序”模式, 并将 U8W 下载工具重新上电, 如下图所示: (否则将不能正常进行烧录)



### I.3.9 U8W/U8W-Mini 设置直通模式（可用于仿真）

若要使用 U8W/U8-Mini 进行仿真，首先必须将 U8W/U8-Mini 设置为直通模式。U8W/U8W-Mini 实现 USB 转串口直通模式的方法如下：

1. 首先 U8W/U8W-Mini 固件必须升级到 v1.37 及以上版本；
2. U8W/U8W-Mini 上电后为正常下载模式，此时按住工具上的 Key1（下载）按键不要松开，再按一下 Key2（电源）按键，然后放开 Key2（电源）按键后，再松开 Key1（下载）按键，U8W/U8W-Mini 会进入 USB 转串口直通模式。（按下 Key1 → 按下 Key2 → 松开 Key2 → 松开 Key1）；
3. 进入直通模式的 U8W/U8W-Mini 工具只是简单的 USB 转串口不具备脱机下载功能，若需要恢复 U8W/U8W-Mini 的原有功能，只需要再次单独按一下 Key2（电源）按键即可。

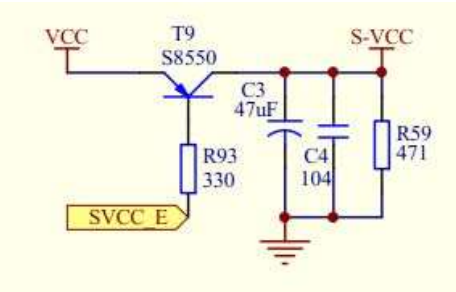
### I.3.10 U8W/U8W-Mini 的参考电路

USB 型联机/脱机下载板 U8W/U8W-Mini 为用户提供了如下的常用控制接口：

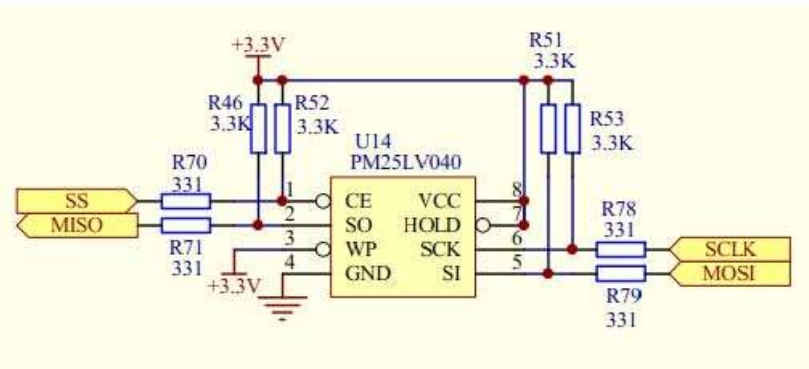
| 脚位功能           | 端口   | 功能描述                     |
|----------------|------|--------------------------|
| 电源控制脚          | P2.6 | 低位有效                     |
| 下载通讯脚          | P1.0 | 串口 RXD，连接目标芯片的 TXD（P3.1） |
|                | P1.1 | 串口 TXD，连接目标芯片的 RXD（P3.0） |
| 编程按键           | P3.6 | 低有效                      |
| 显示             | P3.2 | LED1                     |
|                | P3.3 | LED2                     |
|                | P3.4 | LED3                     |
|                | P5.5 | LED4                     |
| 外挂串行 Flash 控制脚 | P2.4 | Flash 的 CE 脚             |
|                | P2.2 | Flash 的 SO 脚             |
|                | P2.3 | Flash 的 SI 脚             |
|                | P2.1 | Flash 的 SCLK 脚           |

|                  |      |                  |
|------------------|------|------------------|
| 全自动烧录工具<br>分选机信号 | P3.6 | 起始信号             |
|                  | P1.5 | 完成信号             |
|                  | P5.4 | OK 信号 (良品信号)     |
|                  | P3.7 | ERROR 信号 (不良品信号) |
| 蜂鸣器 (BEEP) 控制    | P2.5 | 高有效 (高电平发出声音)    |

电源控制部分参考电路图:

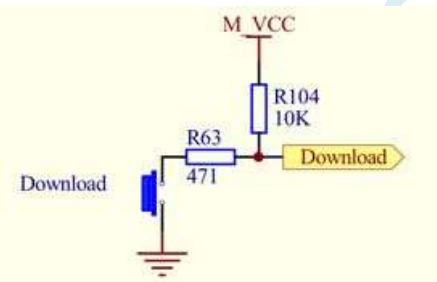


Flash 控制部分参考电路图:

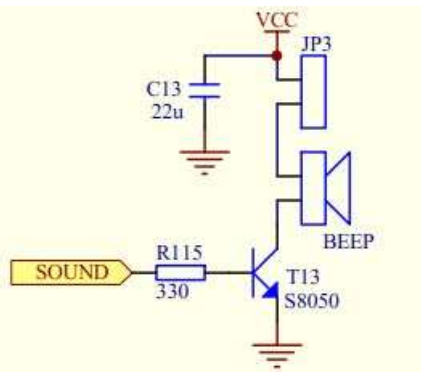


用户程序大于 41K 时需要此 Flash 存储器

按键部分参考电路图:

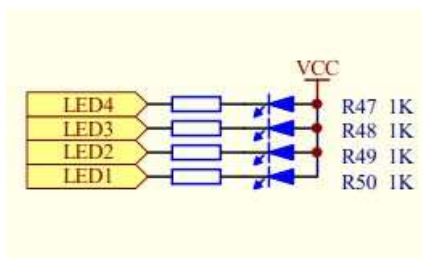


蜂鸣器部分参考电路图:

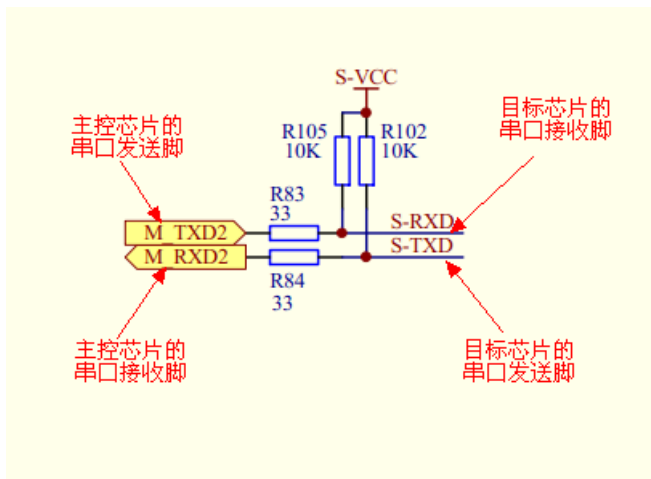


LED 显示部分参考电路图:





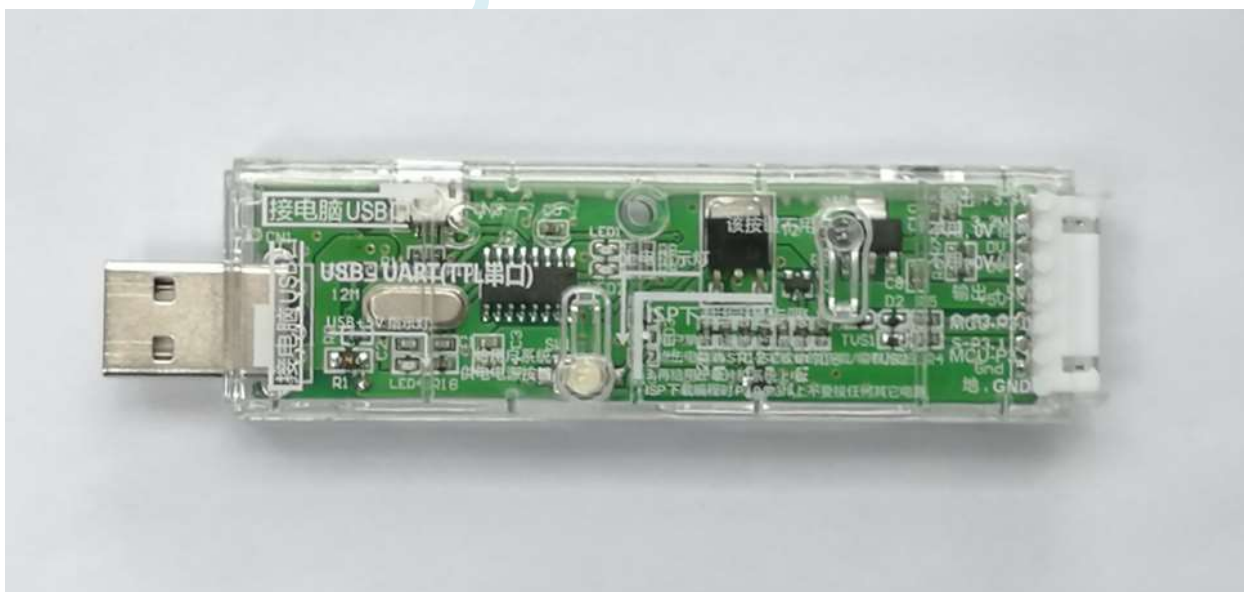
串口通讯脚连接部分参考电路图:



## I.4 STC 通用 USB 转串口工具

### I.4.1 STC 通用 USB 转串口工具外观图

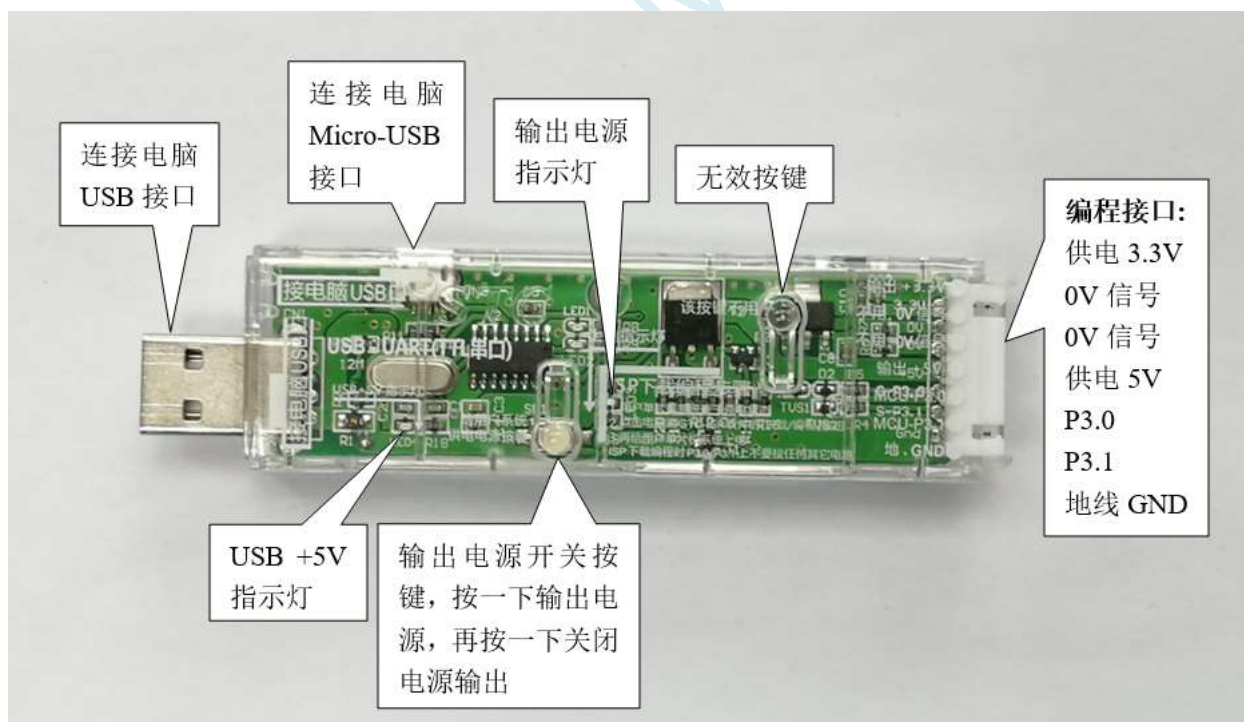
正面:



背面:



## I.4.2 STC 通用 USB 转串口工具布局图



在此，需要对“电源开关”进行说明：

此按钮的作用与自锁开关相同，在开关按钮第一次按时，开关接通电源并保持，即自锁，在开关按钮第二次按时，开关断开电源。鉴于自锁开关使用过程中容易损坏的特点，我们设计了一套利用轻触开关替代自锁开关功能的电路，提高工具的使用寿命。

而对于 STC 的单片机，要想进行 ISP 下载，则必须是在上电复位时接收到串口命令才会开始执行 ISP

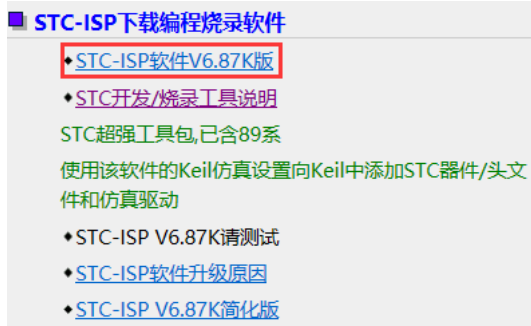
程序, 所以使用 STC 通用 USB 转串口工具下载程序到 MCU 的正确步骤为:

1. 使用 STC 通用 USB 转串口工具将待烧录 MCU 与电脑进行连接;
  2. 打开 STC 的 ISP 下载软件;
  3. 选择单片机型号;
  4. 选择 STC 通用 USB 转串口工具所对应的串口;
  5. 打开目标文件 (HEX 格式或者 BIN 格式);
  6. 点击 ISP 下载软件中的“下载/编程”按钮;
  7. 按一下 STC 通用 USB 转串口工具上的“电源开关”给 MCU 供电, 即可开始下载。
- 【冷启动烧录】

此外, USB 接口与 Micro-USB 接口是相同的功能, 用户根据需要连接其中一个接口到电脑即可。  
编程接口的 0V 信号脚内部有 470 欧姆电阻接地, 如果设置了 P1.0/P1.1=0/0 或者 P3.2/P3.3=0/0 时才能下载, 可将 P1.0, P1.1 或者 P3.2, P3.3 接到 0V 信号脚。

### I.4.3 STC 通用 USB 转串口工具驱动安装

STC 通用 USB 转串口工具采用 CH340 USB 转串口芯片(可以外挂晶振, 更精准), 只要下载通用的 CH340 串口驱动程序进行安装即可, 以下是 STC 官网 ([www.STCMCUDATA.com](http://www.STCMCUDATA.com)) 提供的 CH341SER 串口驱动下载位置:



下载后进行解压, CH340 的驱动安装包路径 stc-isp-15xx-v6.87K\USB to UART Driver\CH340\_CH341:

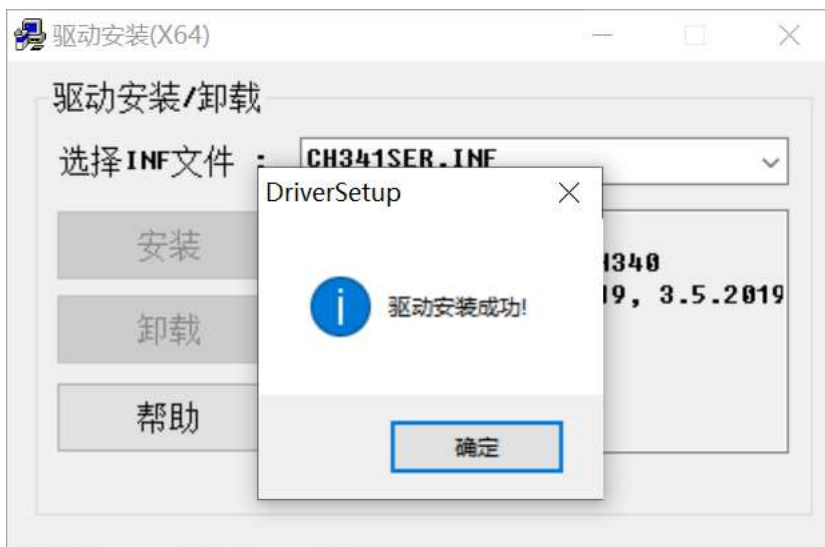
i > 下载 > stc-isp-15xx-v6.87K > USB to UART Driver > CH340\_CH341

| 名称   | 修改日期           |
|--|----------------|
|  ch341ser | 2020/5/9 15:03 |

以 STC 官网提供的 CH341SER 串口驱动为例, 双击“CH341SER.exe”安装包, 在弹出的主界面点击“安装”按钮开始安装驱动程序:



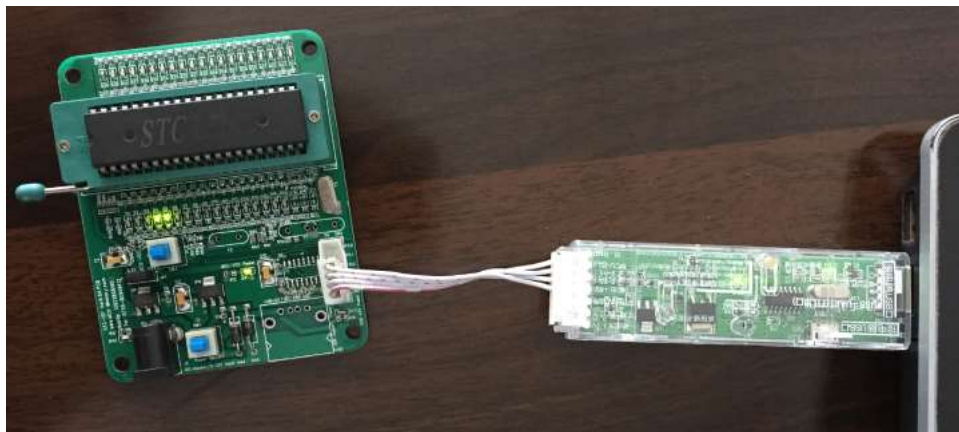
然后弹出驱动安装成功对话框，点击“确定”按钮完成安装：



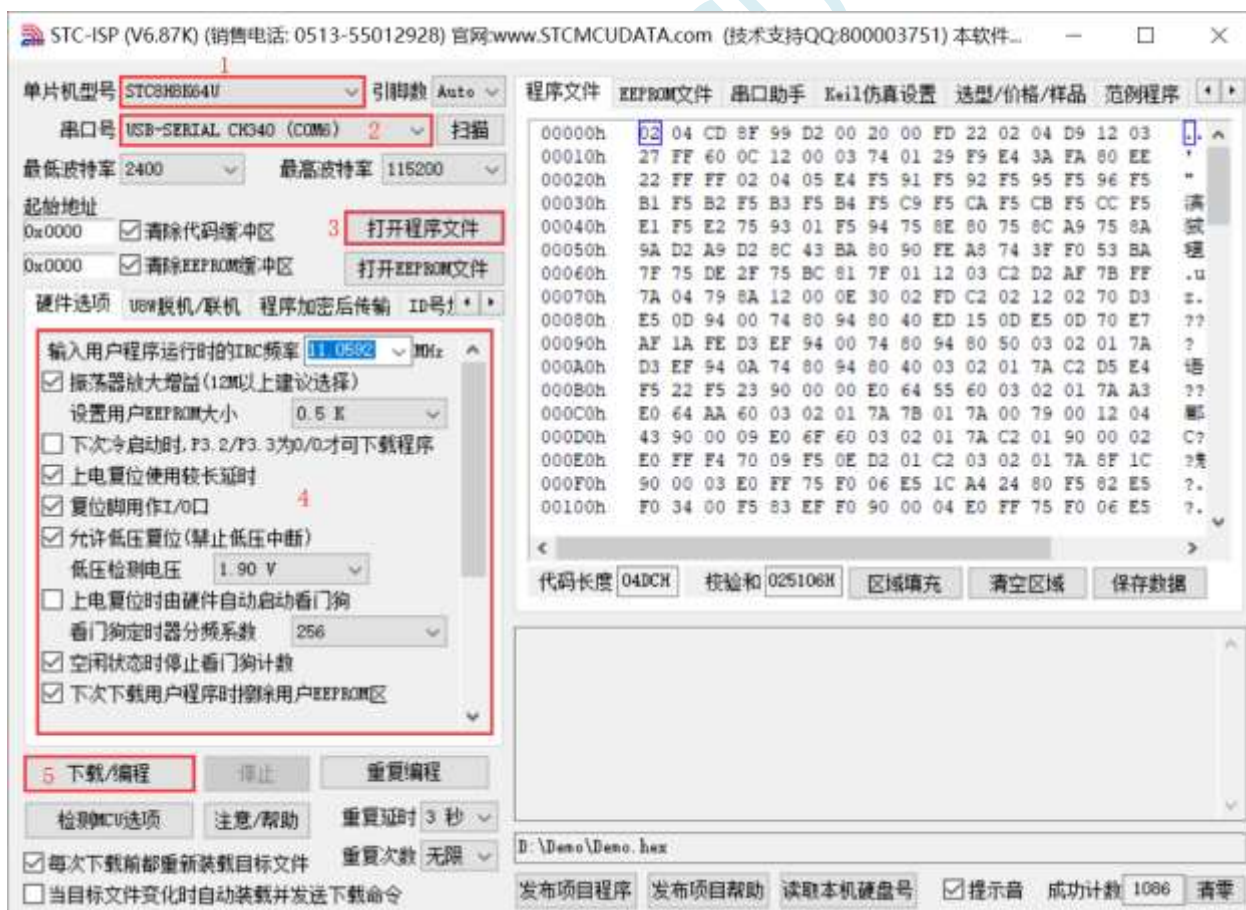
## I.4.4 使用 STC 通用 USB 转串口工具下载程序到 MCU

1. 使用 STC 通用 USB 转串口工具将待烧录 MCU 与电脑进行连接：

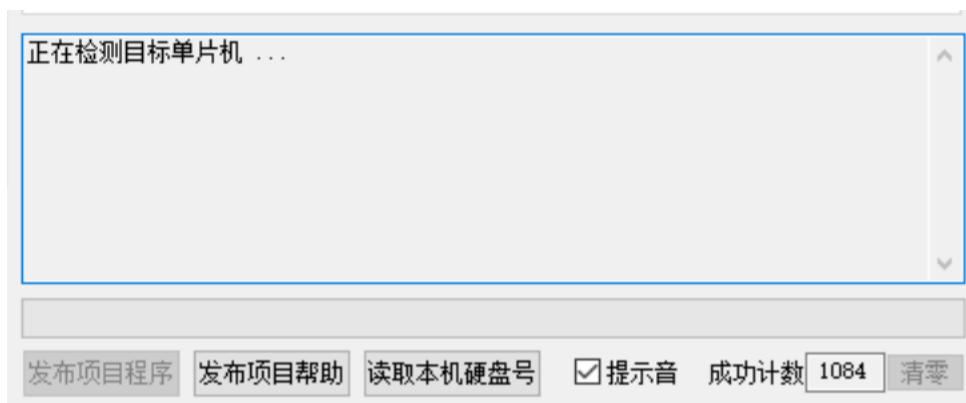




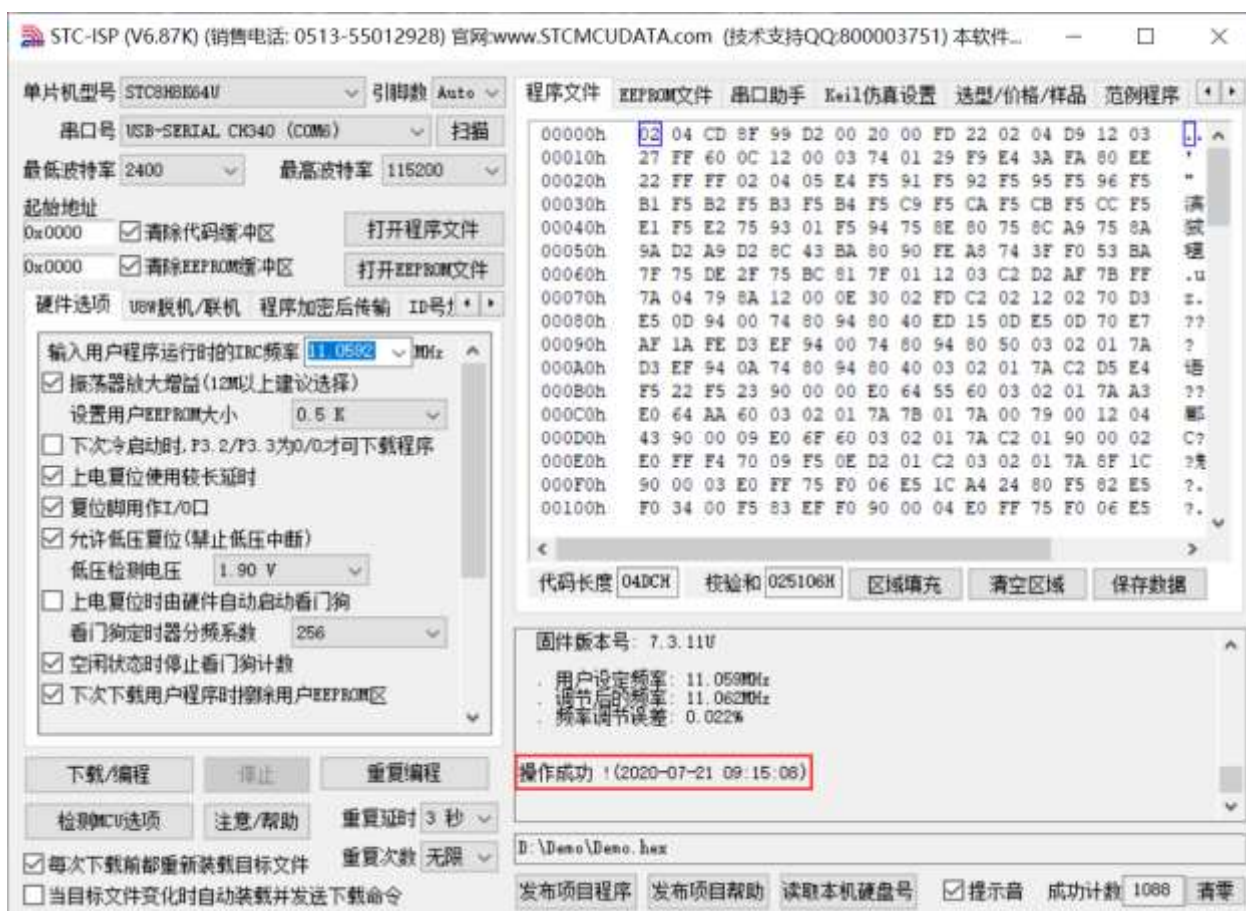
2. 打开 STC-ISP 软件;
3. 选择烧录芯片对应的型号;
4. 选择 STC 通用 USB 转串口工具所识别的串口号 (当 STC 通用 USB 转串口工具与电脑正确连接后, 软件会自动扫描并识别名称为 “USB-SERIAL CH340 (COMx)” 串口, 具体的 COM 编号会因电脑不同而不同)。当有多个 USB 转串口线与电脑相连时, 则必须手动选择;
5. 加载烧录程序;
6. 设置烧录选项;
7. 点击 “下载/编程” 按钮;



8. 右下角提示框显示 “正在检测目标单片机 ...” 时按一下 STC 通用 USB 转串口工具上的 “电源开关” 给 MCU 供电, 即可开始下载【冷启动烧录】;



9. 等待下载结束，若下载成功，右下角提示框会显示“操作成功!”。



## I.4.5 使用 STC 通用 USB 转串口工具仿真用户代码

目前 STC 的仿真都是基于 Keil 环境的，所以若需要使用 STC 通用 USB 转串口工具仿真用户代码，则必须要安装 Keil 软件。

Keil 软件安装完成后，还需要安装 STC 的仿真驱动。STC 的仿真驱动的安装步骤如下：

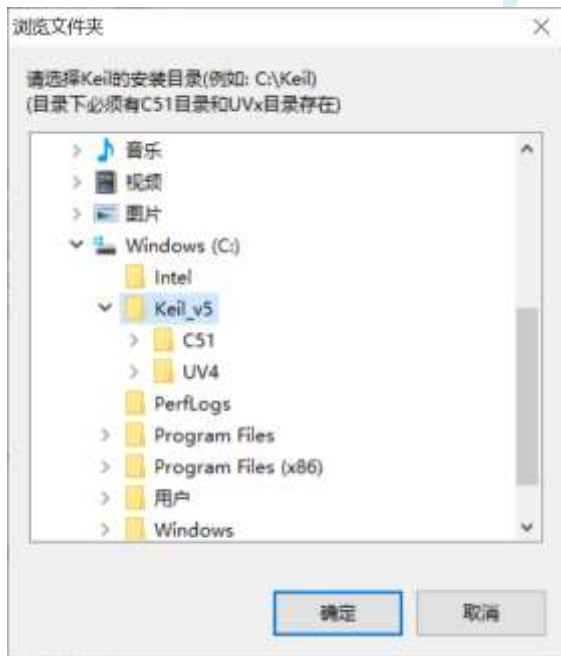
首先开 STC-ISP 下载软件；

然后在软件右边功能区的“Keil 仿真设置”页面中点击“添加型号和头文件到 Keil 中 添加 STC 仿真器

驱动到 Keil 中”按钮:

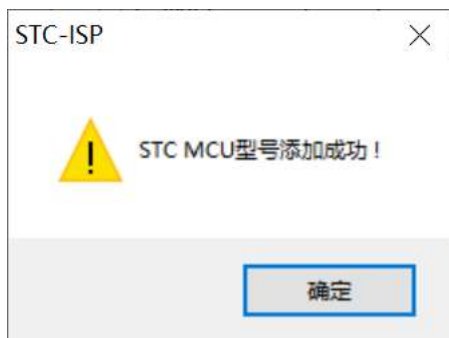


按下后会出现如下画面:

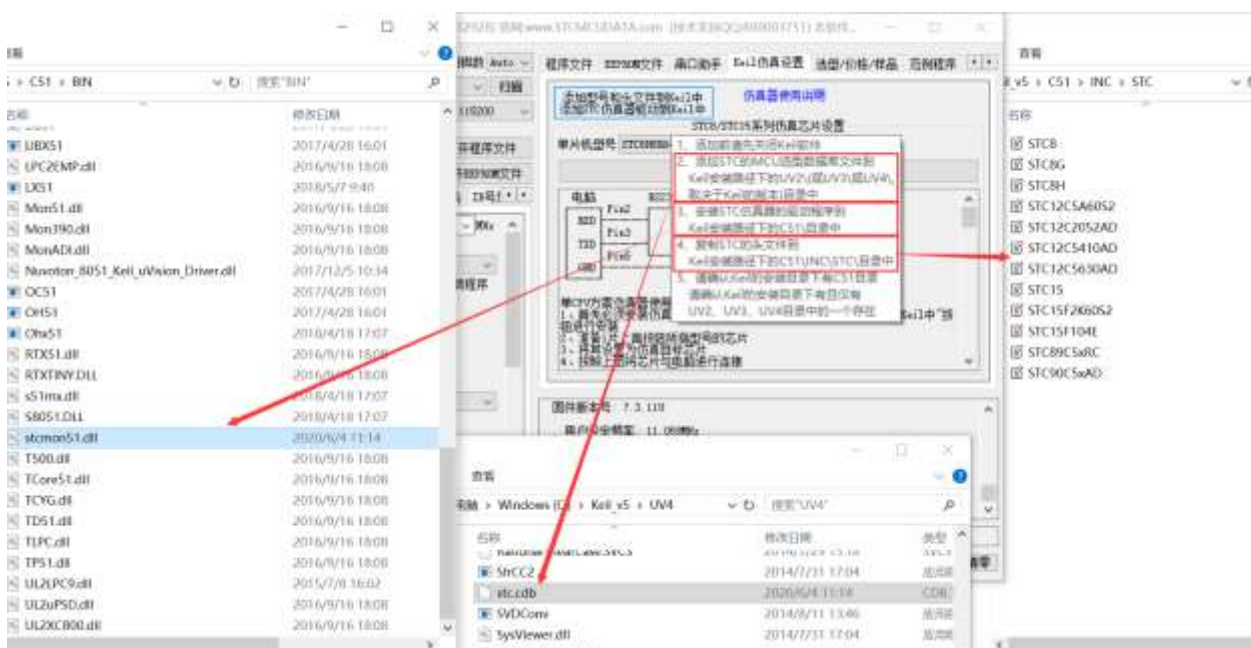


将目录定位到 Keil 软件的安装目录, 然后确定。  
安装成功后会弹出如下的提示框:





在 Keil 的相关目录中可以看到如下的文件，即表示驱动正确安装了。



由于在默认状态下，STC 的主控芯片并不是一颗仿真芯片，不具有仿真功能，所以若需要进行仿真，则还需要将 STC 的主控芯片设置为仿真芯片。

制作仿真芯片步骤如下：

首先使用 STC 通用 USB 转串口工具将 MCU 与电脑进行连接；

然后打开 STC 的 ISP 下载软件，并在串口号的下拉列表中选择串口工具所对应的串口号；

选择 MCU 单片机型号；

选择用户程序运行的 IRC 频率，制作仿真芯片时选择的频率与所仿真的用户程序所设置的频率一致，才能达到真实的运行效果。



然后在软件右边功能区的“Keil 仿真设置”页面中点击“将所选目标单片机设置为仿真芯片”按钮，按下后会出现如下画面：



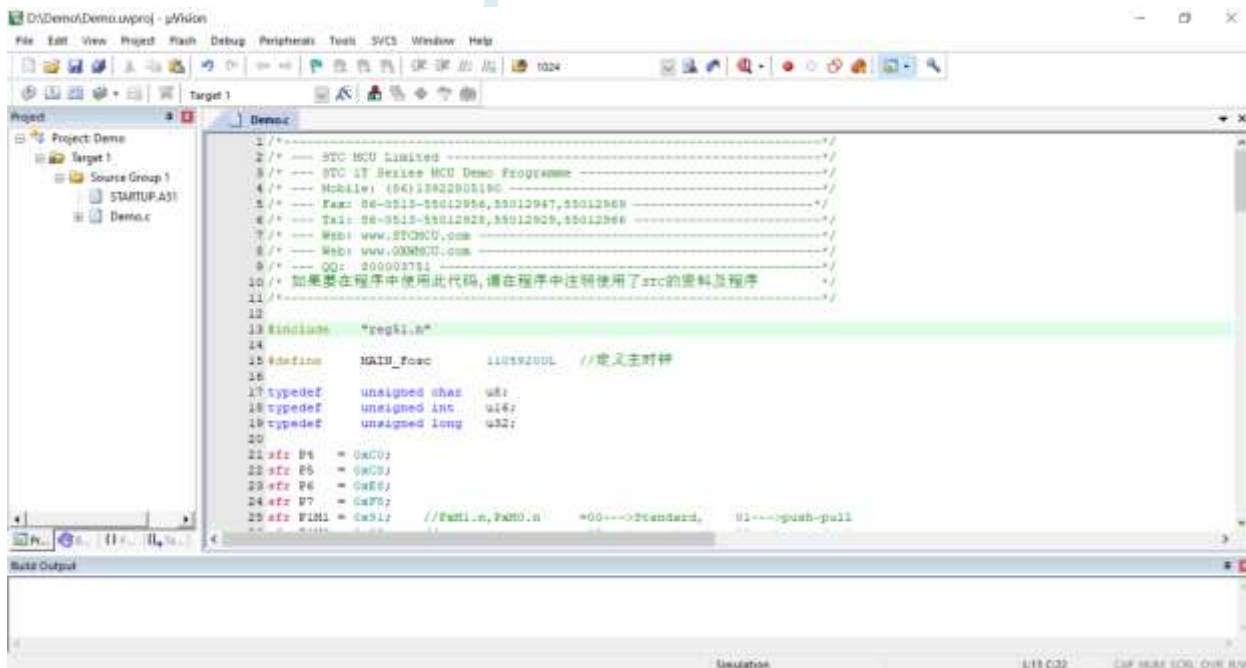
接下来需要按一下 STC 通用 USB 转串口工具上的“电源开关”给 MCU 供电【冷启动】，即可开始制作仿真芯片。

若设置成功，会出现如下的画面：



到此，仿真芯片便制作成功了。

接下来我们打开一个项目进行仿真：

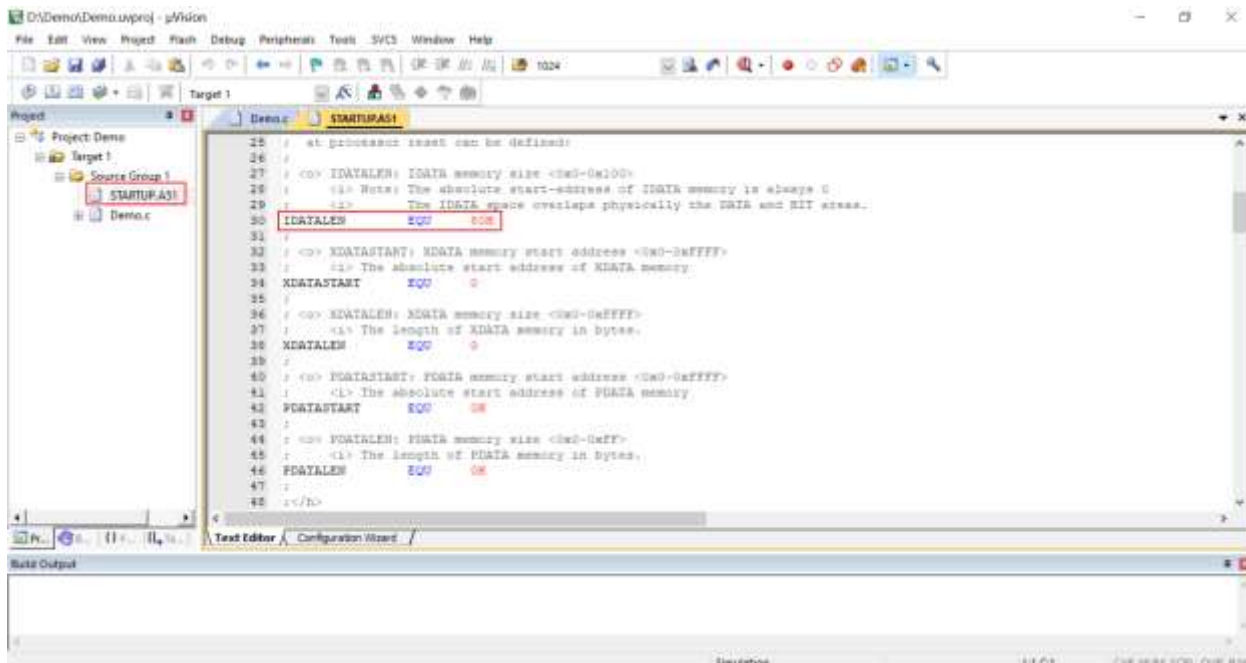


然后进行下面的项目设置：



附加说明一点:

当创建的是 C 语言项目, 且有将启动文件 “STARTUP.A51” 添加到项目中时, 里面有一个命名为 “IDATALEN” 的宏定义, 它是用来定义 IDATA 大小的一个宏, 默认值是 128, 即十六进制的 80H, 同时它也是启动文件中需要初始化为 0 的 IDATA 的大小。所以当 IDATA 定义为 80H, 那么 STARTUP.A51 里面的代码则会将 IDATA 的 00-7F 的 RAM 初始化为 0; 同样若将 IDATA 定义为 0FFH, 则会将 IDATA 的 00-FF 的 RAM 初始化为 0。



我们所选的 STC8H 系列的单片机的 IDATA 大小为 256 字节 (00-7F 的 DATA 和 80H-FFH 的 IDATA), 但由于在 RAM 的最后 17 个字节有写入 ID 号以及相关的测试参数, 若用户在程序中需要使用这一部分数据, 则一定不要将 IDATALEN 定义为 256。

按下快捷键 “Alt+F7” 或者选择菜单 “Project” 中的 “Option for Target ‘Target1’”

在 “Option for Target ‘Target1’” 对话框中对项目进行配置:

第 1 步、进入到项目的设置页面, 选择 “Debug” 设置页;

第 2 步、选择右侧的硬件仿真 “Use ...”;

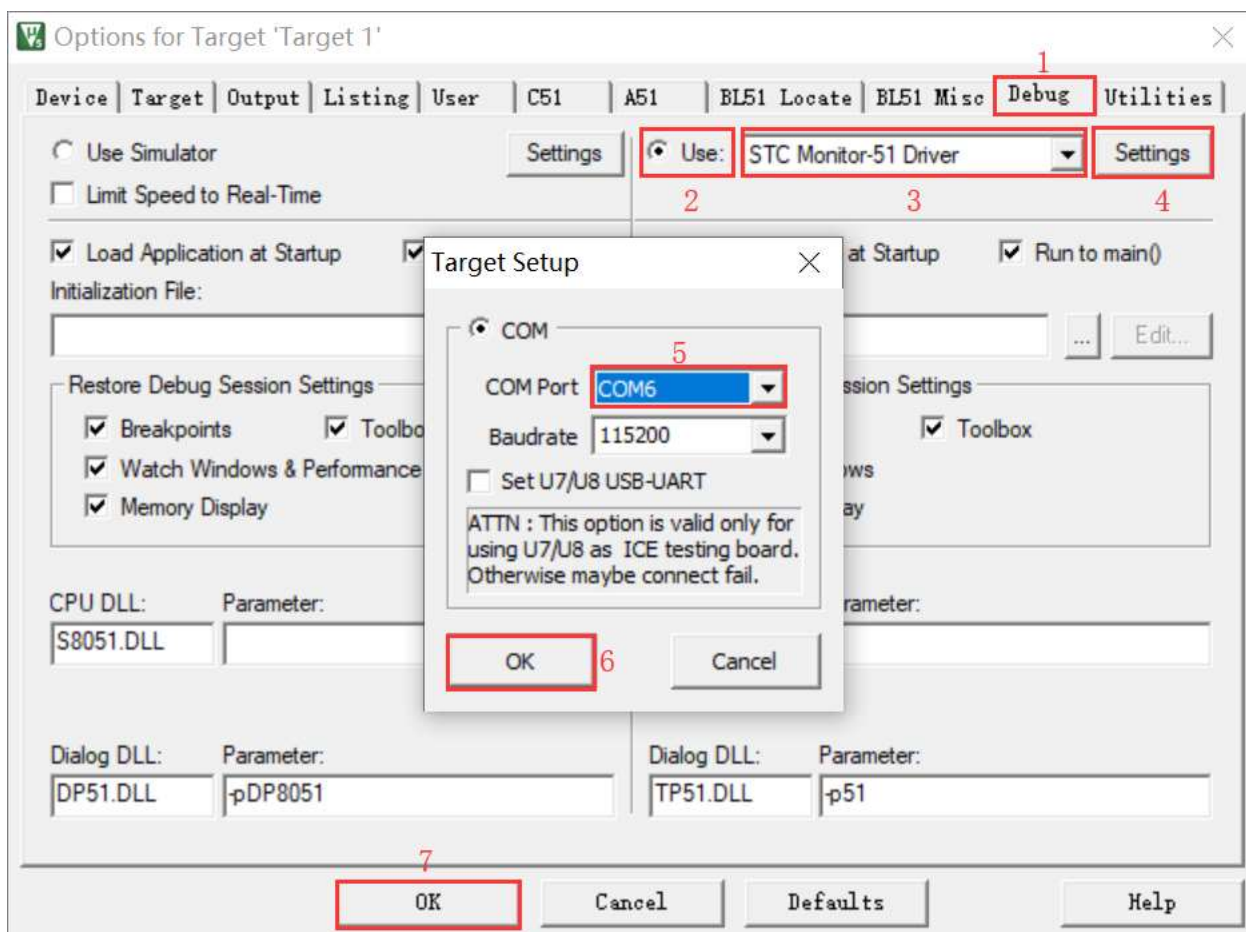
第 3 步、在仿真驱动下拉列表中选择 “STC Monitor-51 Driver” 项;

第 4 步、点击 “Settings” 按钮, 进入串口的设置画面;

第 5 步、对串口的端口号和波特率进行设置, 串口号要选择 STC 通用 USB 转串口工具所对应的串口, 波特率一般选择 115200 或者 57600。

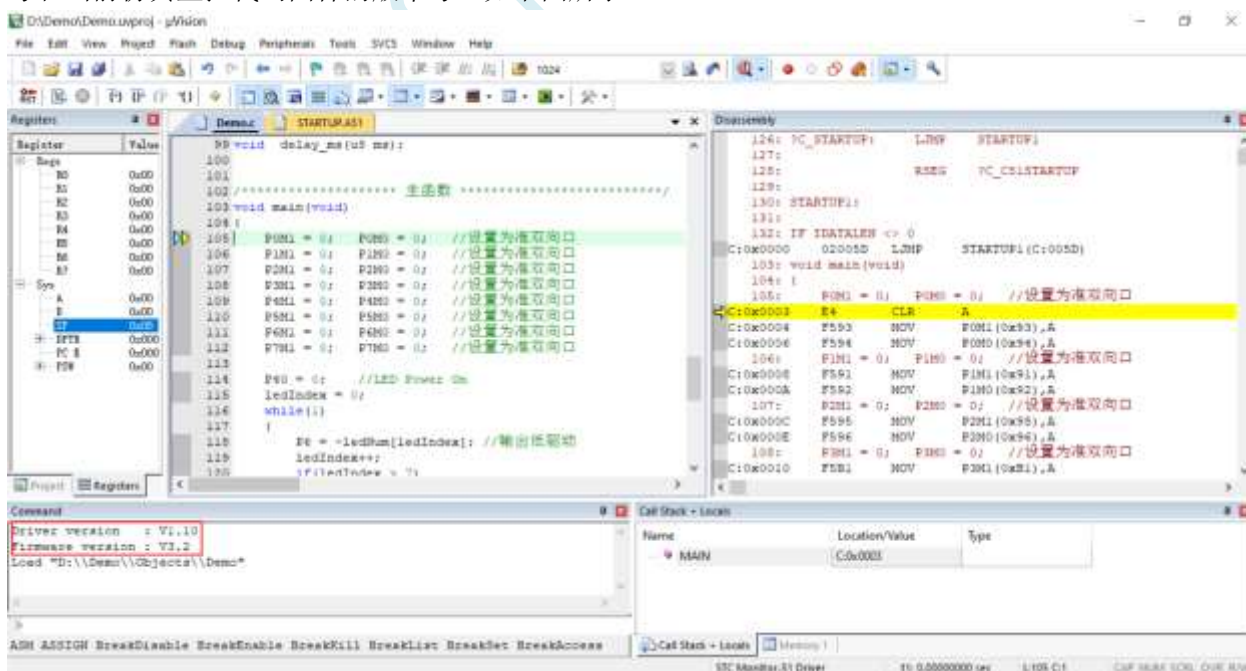
确定完成仿真设置。

详细步骤如下图所示:



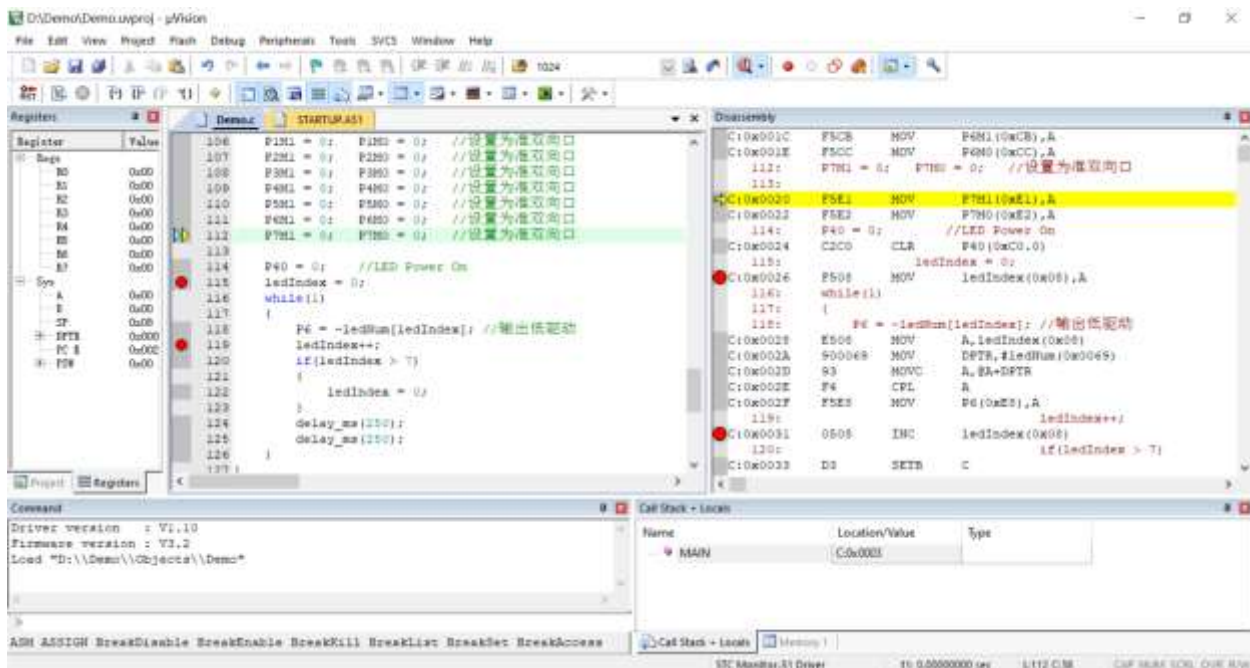
完成了上面所有的工作后，即可在 Keil 软件中按“Ctrl+F5”开始仿真调试。

若硬件连接无误的话，将会进入到类似于下面的调试界面，并在命令输出窗口显示当前的仿真驱动版本号 and 当前仿真监控代码固件的版本号，如下图所示：



仿真调试过程中，可执行复位、全速运行、单步运行、设置断点等多中操作。

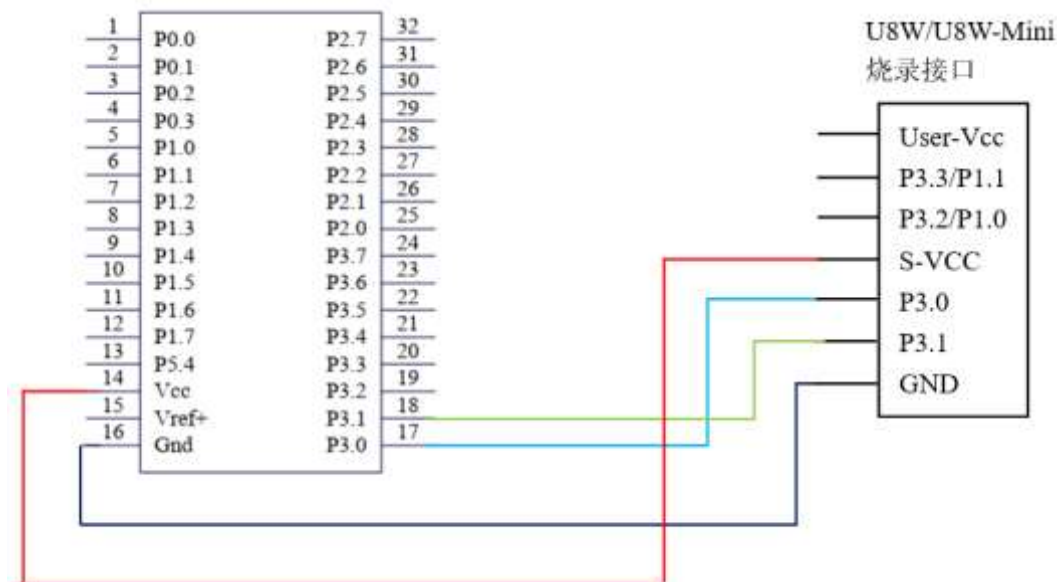




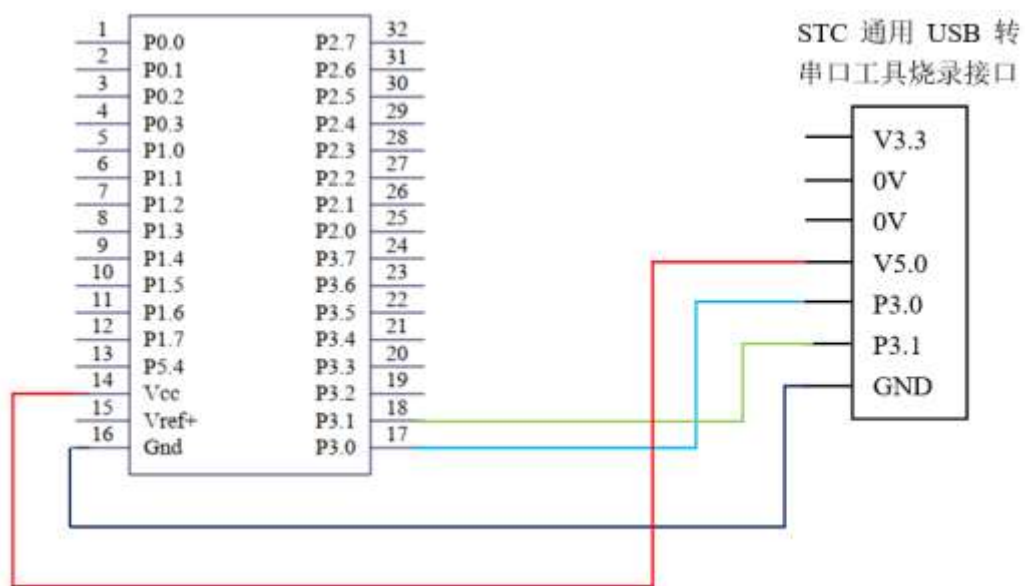
如上图所示，可在程序中设置多个断点，断点设置的个数目前最大允许 20 个（理论上可设置任意个，但是断点设置得过多会影响调试的速度）。

## I.5 应用线路图

### I.5.1 U8W 工具应用参考线路图



### I.5.2 STC 通用 USB 转串口工具应用参考线路图



## 附录J STC 仿真使用说明书

### J.1 概述

STC8G/8H 系列单片机均支持在线仿真。支持包括下载用户代码、芯片复位、全速运行、单步运行、设置断点（理论断点个数为无限个，但为了提高仿真效率，目前限制为最多 20 个断点）、查看变量等基本的仿真操作，方便用户调试代码，查找代码中的逻辑错误，进而缩短项目开发周期。

仿真接口可为 USB 或者串口，单片机本身就是仿真器，不需要额外的仿真器即可实现全部的仿真功能。相应的 USB 口或者串口本为仿真专用端口，但当关闭仿真功能后，用户将可随意将仿真接口当作 GPIO、USB 或者串口进行使用。

目前所有单片机的仿真模式均为软件监控仿真模式，会占用系统部分资源，各系列单片机仿真时所占用的资源如下表所示：

| 单片机系列                     | 仿真接口 | 占用资源       |                     |       |
|---------------------------|------|------------|---------------------|-------|
|                           |      | 端口         | 数据存储器（XDATA）        | 程序存储器 |
| STC8H8K64U 系列<br>B/C/D 版本 | USB  | D+, D-     | 768 字节（1D00H-1FFFH） | 0 字节  |
|                           | 串口   | P3.0, P3.1 | 768 字节（1D00H-1FFFH） | 0 字节  |
|                           |      | P3.6, P3.7 |                     |       |
|                           |      | P1.6, P1.7 |                     |       |
| STC8H8K64U 系列<br>A 版本     | 串口   | P4.3, P4.4 | 768 字节（1D00H-1FFFH） | 0 字节  |
|                           |      | P3.0, P3.1 |                     |       |
|                           |      | P3.6, P3.7 |                     |       |
|                           |      | P1.6, P1.7 |                     |       |
| STC8H4K64T 系列             | 串口   | P4.3, P4.4 | 768 字节（1D00H-1FFFH） | 0 字节  |
| STC8H4K64T 系列             | 串口   | P3.0, P3.1 | 768 字节（0D00H-0FFFH） | 0 字节  |
| STC8H3K64S4 系列            | 串口   | P3.0, P3.1 | 768 字节（0900H-0BFFH） | 0 字节  |
| STC8H1K16 系列              | 串口   | P3.0, P3.1 | 768 字节（0100H-03FFH） | 0 字节  |
| STC8H1K08 系列              | 串口   | P3.0, P3.1 | 768 字节（0100H-03FFH） | 0 字节  |
| STC8G2K64S4 系列            | 串口   | P3.0, P3.1 | 768 字节（0500H-07FFH） | 0 字节  |
| STC8G1K08 系列              | 串口   | P3.0, P3.1 | 768 字节（0100H-03FFH） | 0 字节  |
| STC8C2K64S4 系列            | 串口   | P3.0, P3.1 | 768 字节（0500H-07FFH） | 0 字节  |
| STC8A8K64D4 系列            | 串口   | P3.0, P3.1 | 768 字节（1D00H-1FFFH） | 0 字节  |
| STC8A8K64S4A12 系列         | 串口   | P3.0, P3.1 | 768 字节（1D00H-1FFFH） | 0 字节  |
| STC8F2K64S4 系列            | 串口   | P3.0, P3.1 | 768 字节（0500H-07FFH） | 0 字节  |
| STC8F1K08S2 系列            | 串口   | P3.0, P3.1 | 768 字节（0100H-03FFH） | 0 字节  |
| IAP15W4K58S4              | 串口   | P3.0, P3.1 | 768 字节（0D00H-0FFFH） | 0 字节  |
| IAP15F2K61S2              | 串口   | P3.0, P3.1 | 768 字节（0500H-07FFH） | 0 字节  |

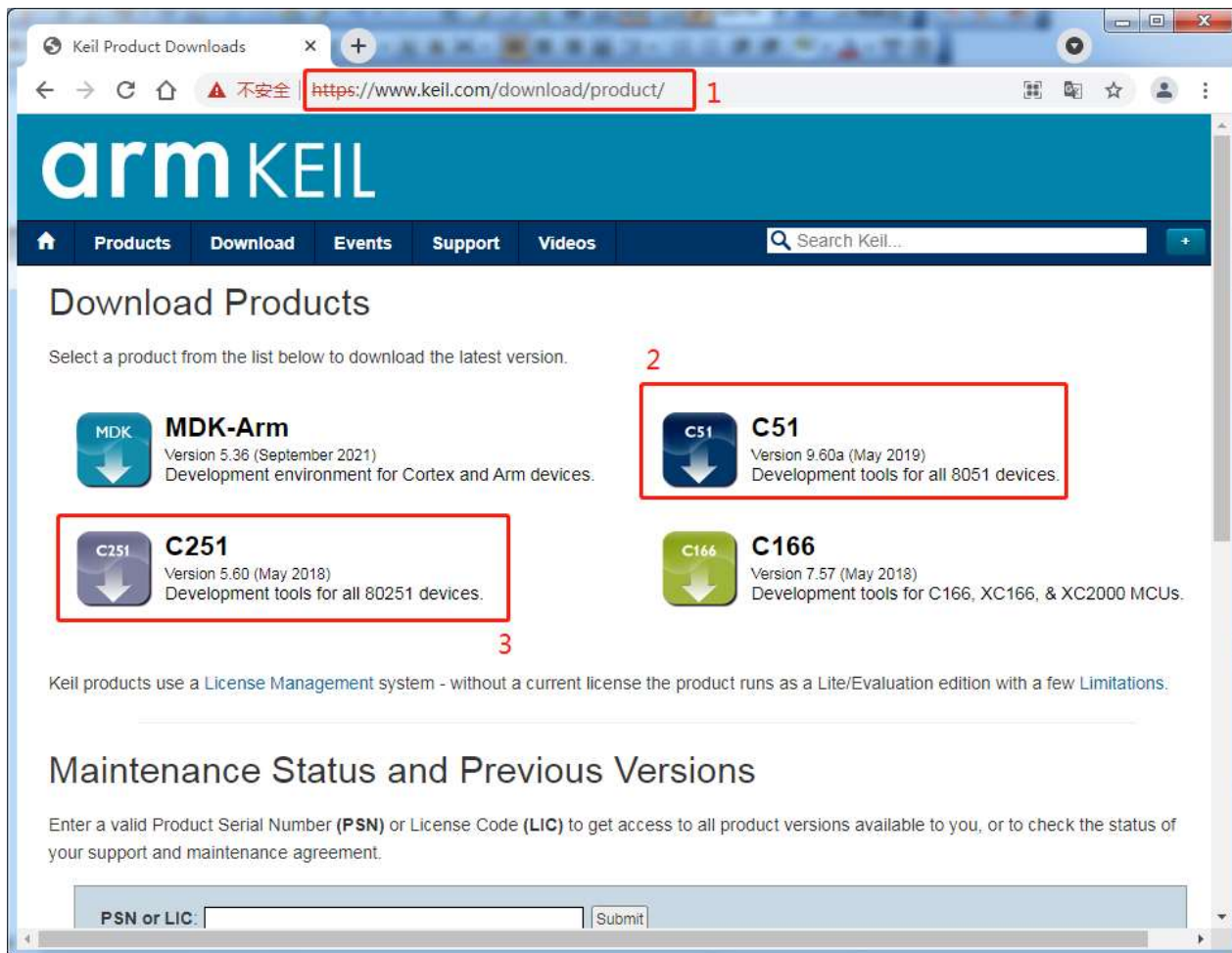
注意：仿真所使用的 768 字节 XDATA，在用户程序中不能使用。若创建的是 C 语言项目，且有将启动文件“STARTUP.A51”添加到项目中时，里面有一个命名为 XDATALEN 宏，是用来定义 XDATA 大小的，默认值是 0，STARTUP.A51 里面的代码会根据用户设置的 XDATALEN 长度来自动初始化 0~(XDATALEN-1)区域的 XDATA 为 0。一般保持 XDATALEN 为默认值 0 即可。若用户需要修改，请根据实际的仿真的 XDATA 大小并保留 768 字节来设置此项。比如 STC8H8K64U 的 XDATA 为 8K，则需

要将 XDATALEN 设置为 8192-768

## J.2 安装 Keil 软件

STC 单片机的仿真基于 Keil 开发环境，所以在进行仿真前，必须先安装 Keil 软件。

可在下图所示的地址下载 C51 和 C251 开发包



注意：最新的 Keil-UV5 软件默认是不包含 8051 和 80251 的工具包的，必须手动下载并安装。

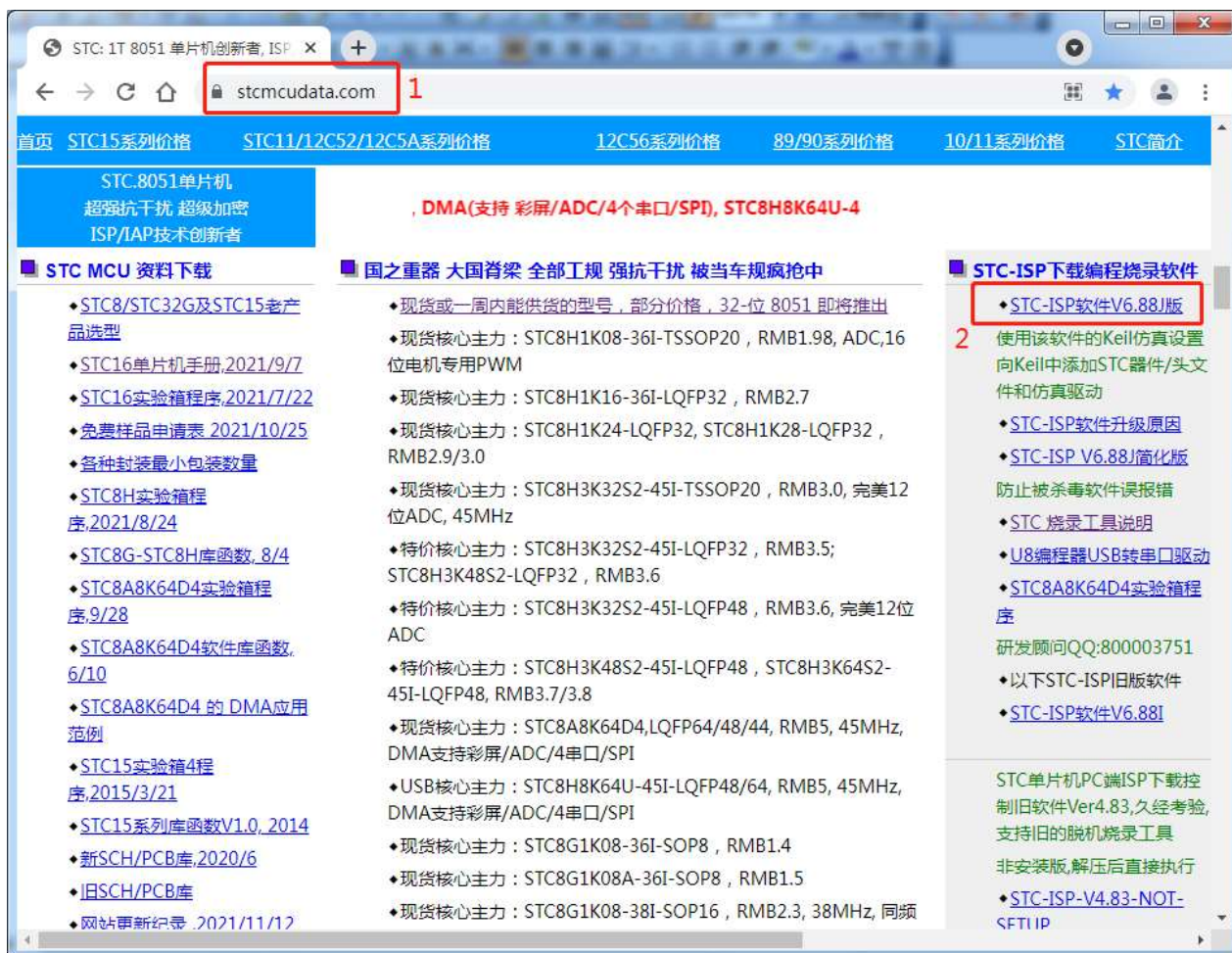


## J.3 安装仿真驱动

安装完成 Keil 开发环境后, 还需要安装 STC 专用仿真驱动程序。

步骤如下:

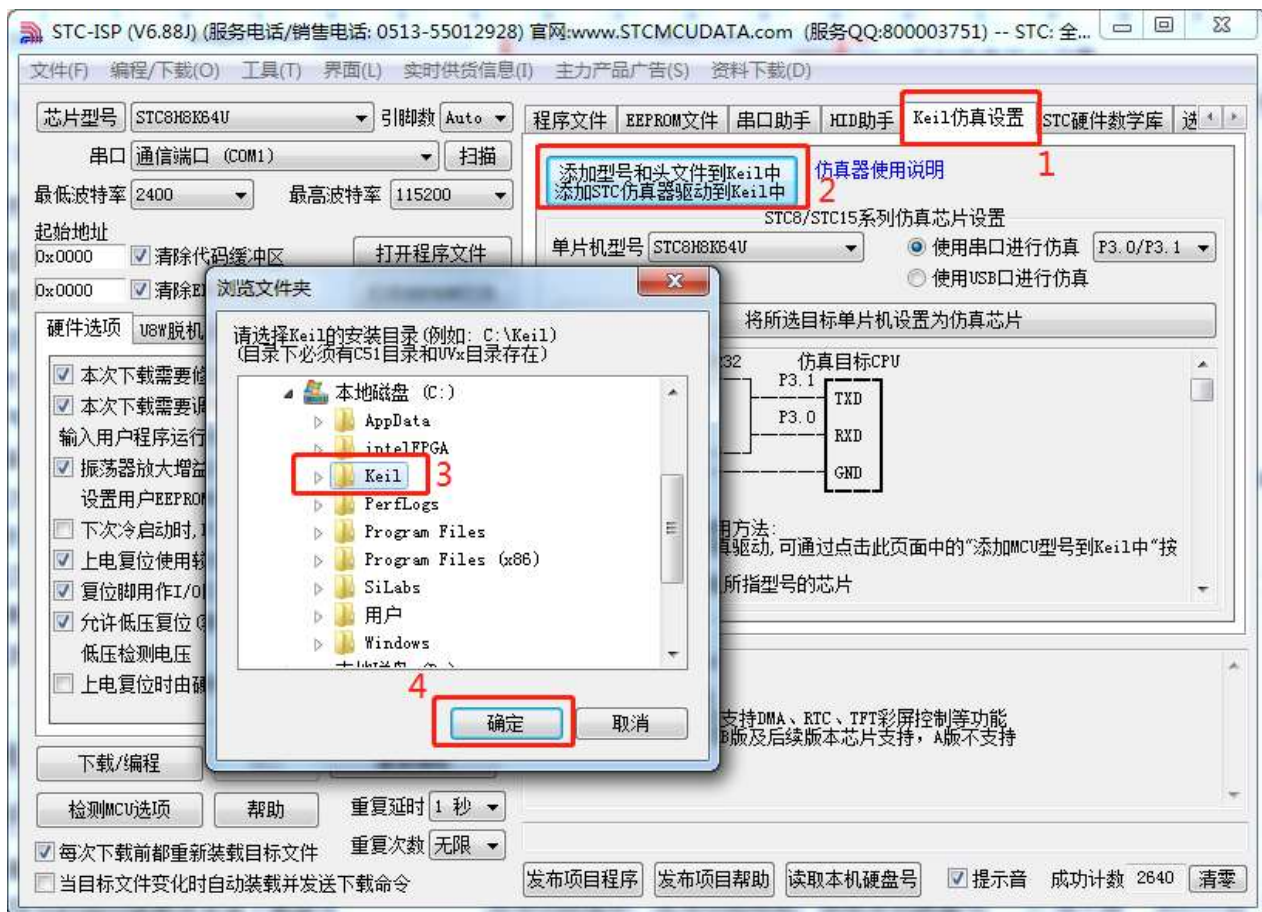
首先从 STC 官网下载最新的 STC-ISP 下载软件



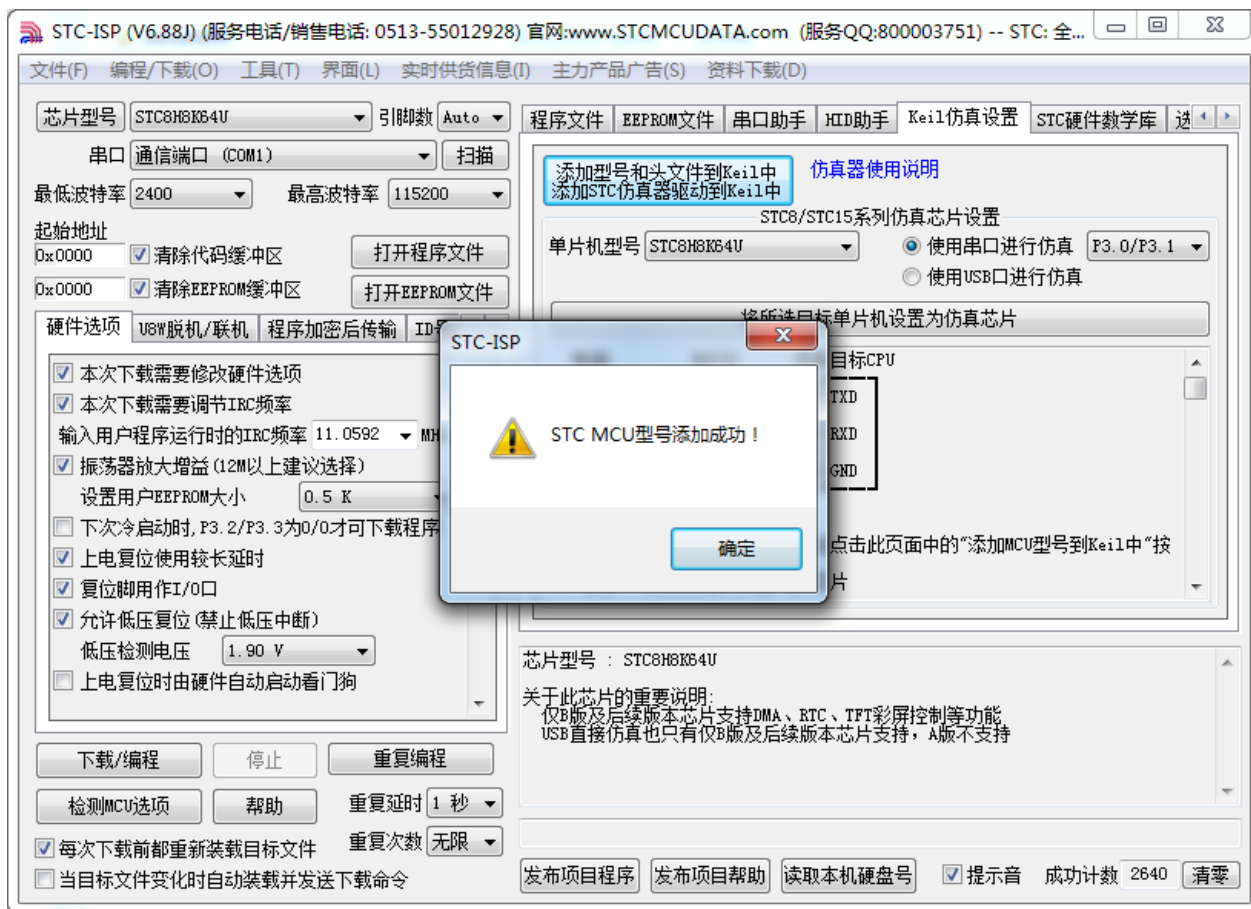
下载并解压完成后, 打开软件包中的“stc-isp-vxx.exe”可执行文件

| 名称                        | 修改日期             | 类型                 | 大小       |
|---------------------------|------------------|--------------------|----------|
| STC-USB Driver            | 2014/8/29 18:17  | 文件夹                |          |
| USB to UART Driver        | 2014/10/9 11:54  | 文件夹                |          |
| readme.txt                | 2020/6/9 14:43   | 文本文档               | 1 KB     |
| <b>stc-isp-v6.88J.exe</b> | 2021/10/20 17:07 | 应用程序               | 2,114 KB |
| STC-USB 驱动安装说明.pdf        | 2020/6/9 14:27   | Foxit Reader PD... | 3,585 KB |

点击下载软件“Keil 仿真设置”页面中的“添加型号和头文件...”按钮(如下图“2”)



在弹出的“浏览文件夹”窗口中,选中 Keil 的安装目录(一般 Keil 的安装目录为“c:\keil”),点击确定后,若弹出“STC MCU 型号添加成功”则表示驱动已安装完成。



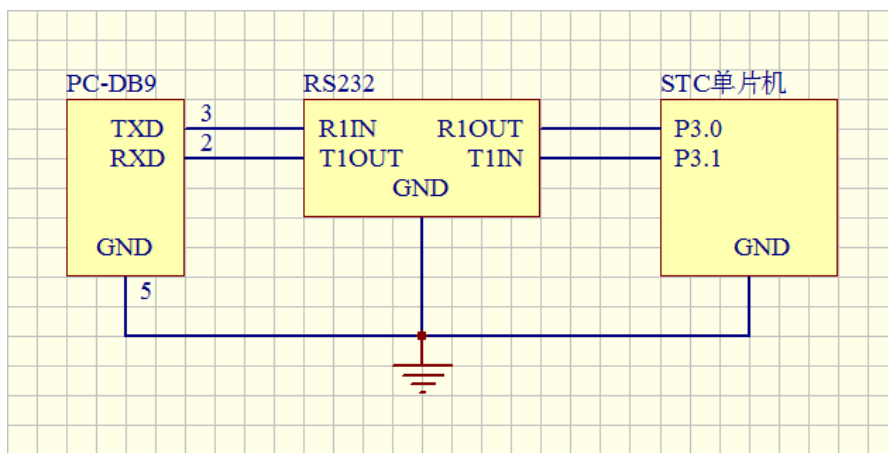
## J.4 串口直接仿真

### J.4.1 制作串口仿真芯片

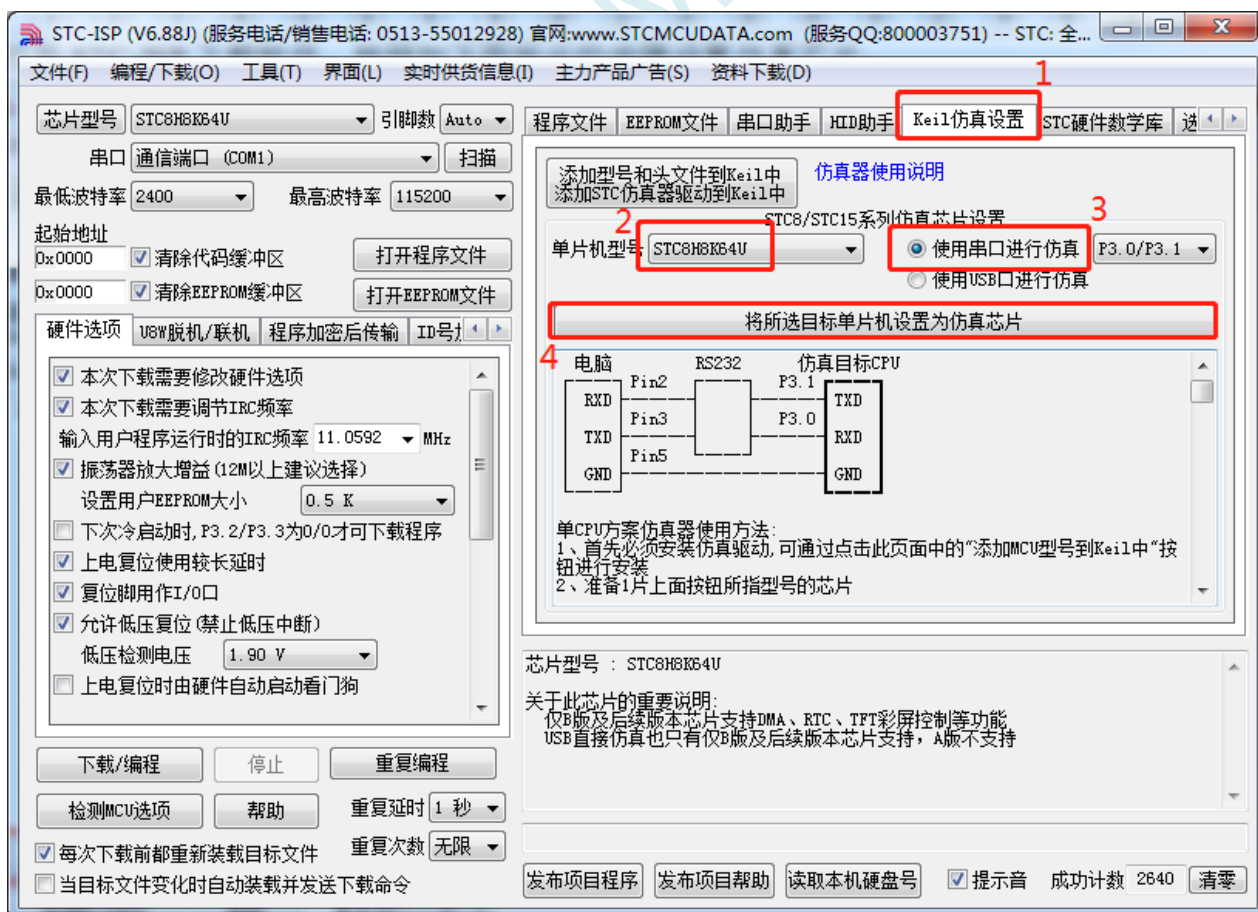
STC 单片机出厂时, 仿真功能默认是关闭的, 若要使用仿真功能, 则需使用 STC-ISP 下载软件将目标单片机设置为仿真芯片。

设置步骤如下:

首先将目标芯片如下图所示的方式和电脑的串口连接在一起, 并将单片机断电



打开 STC-ISP 下载软件, 按照如下图所示的步骤设置仿真芯片





当出现如下画面时, 再给单片机上电

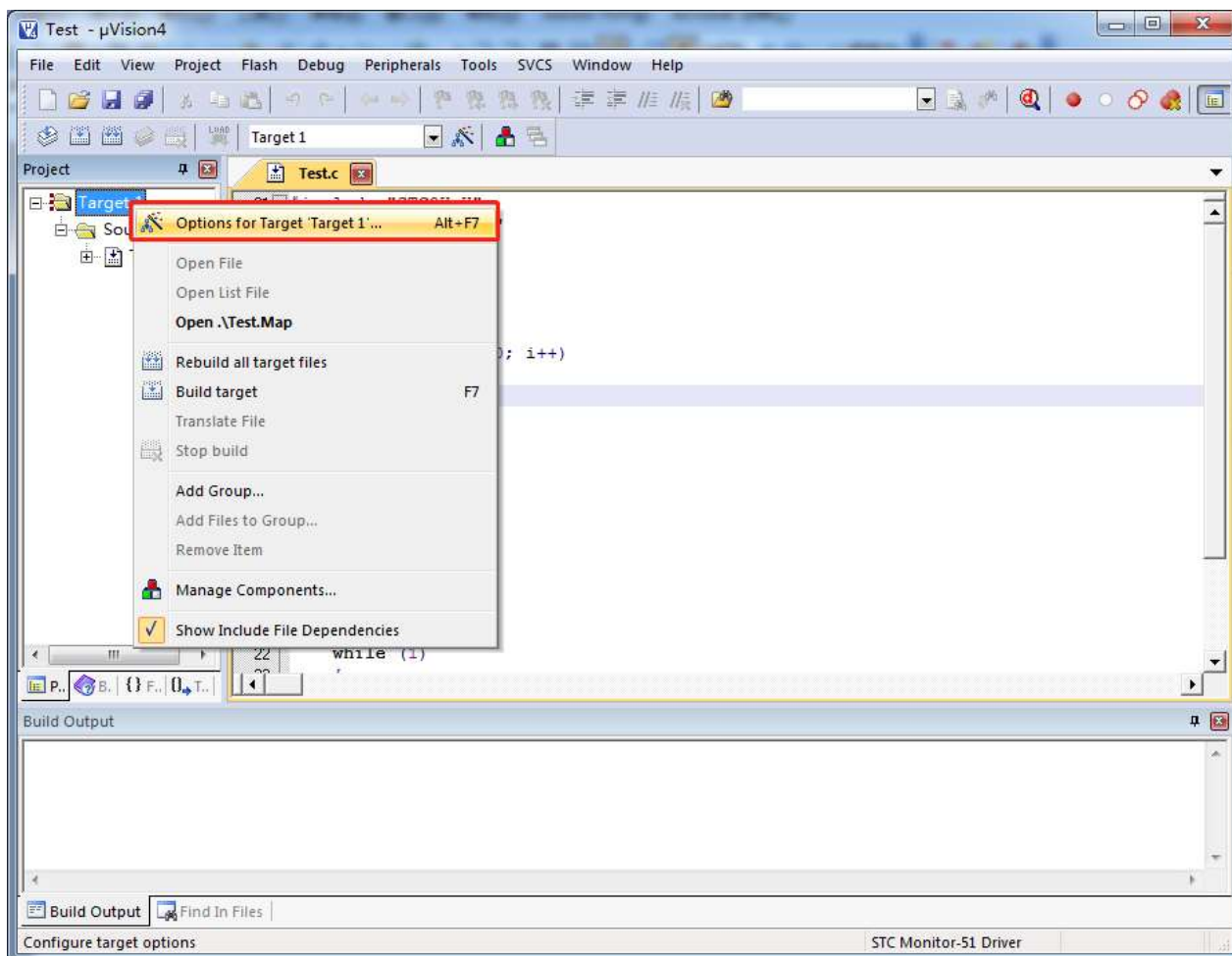


下载完成后, 仿真芯片即制作完成

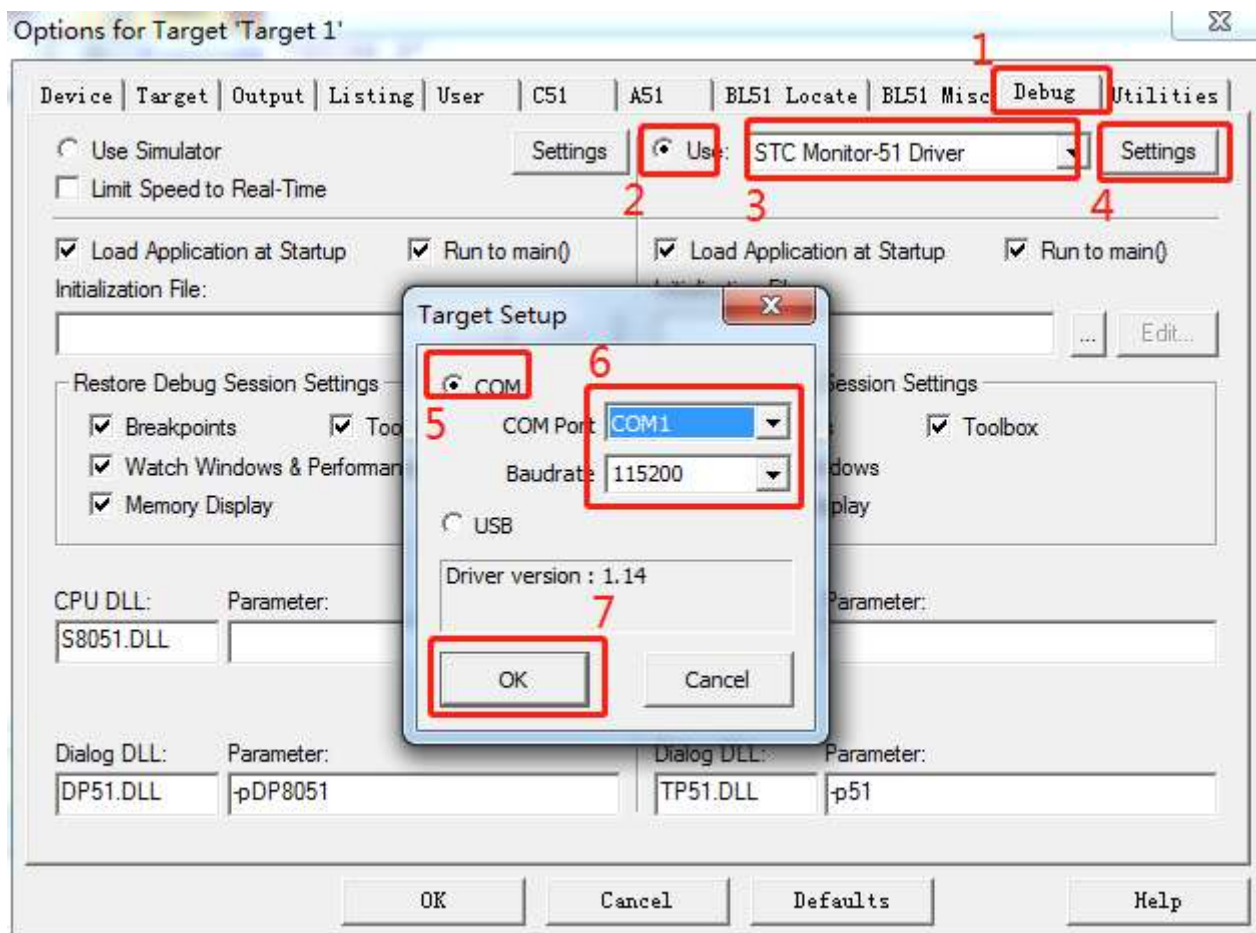


## J.4.2 在 Keil 软件中进行串口仿真设置

在 Keil 软件中打开项目文件，并在下图所示的右键菜单中点击“Options for ...”



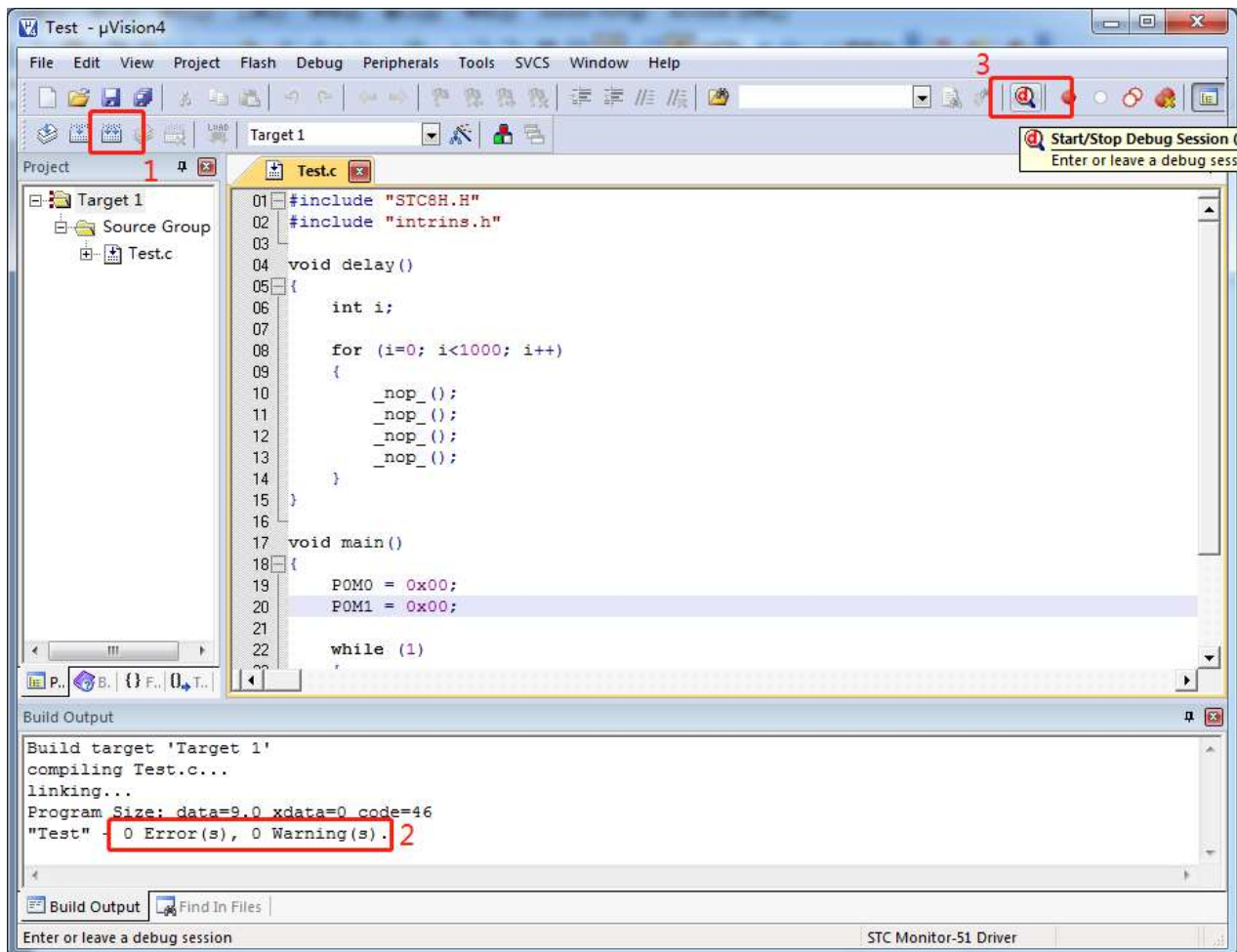
在项目选项中, 按如下图所示的步骤进行串口仿真设置

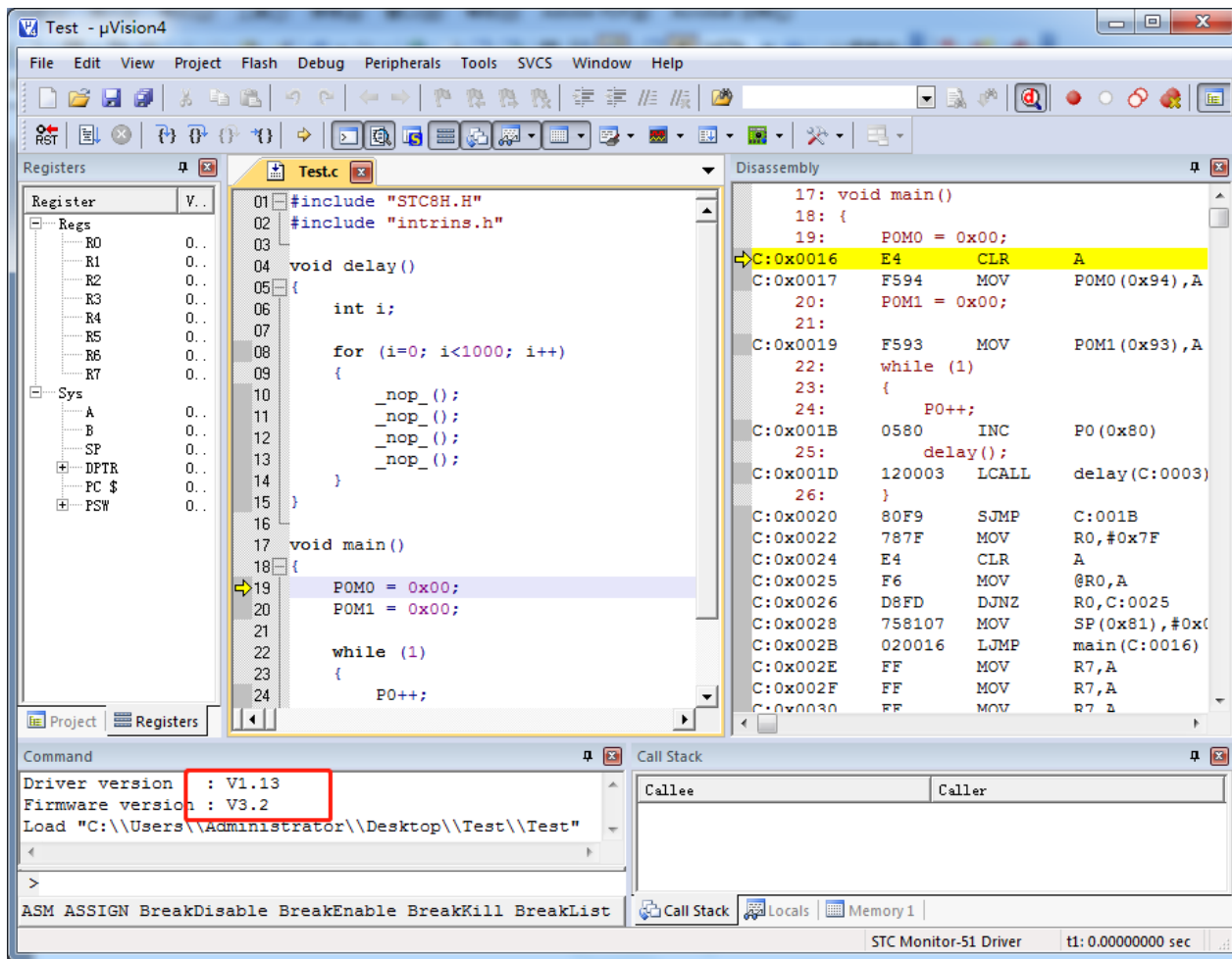


注意: 串口请根据实际的连接进行选择, 波特率一般选择 115200

### J.4.3 在 Keil 软件中使用串口进行仿真

在 Keil 环境下, 编辑完成源代码, 并编译无误后, 即可开始仿真





若芯片制作和连接均无误，则会如上图所示显示仿真驱动版本，并可正确下载用户代码到单片机，接下来便可进行运行、单步、断点等调试功能了。

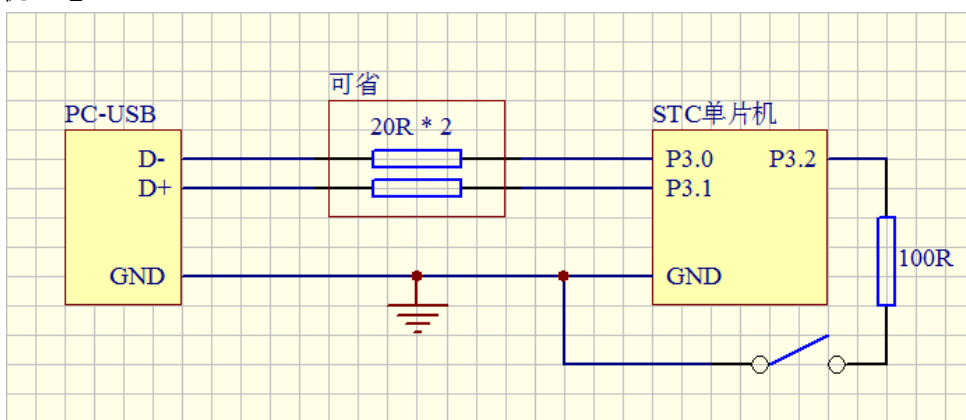
## J.5 USB 直接仿真（目前只有 STC8H8K64U-B/C/D 版本芯片支持）

### J.5.1 制作 USB 仿真芯片

制作 USB 仿真芯片，可按照 4.1 小节的步骤，使用串口 ISP 制作，也可以使用 USB-ISP 的方法制作，本小节将介绍如何使用 USB-ISP 制作。

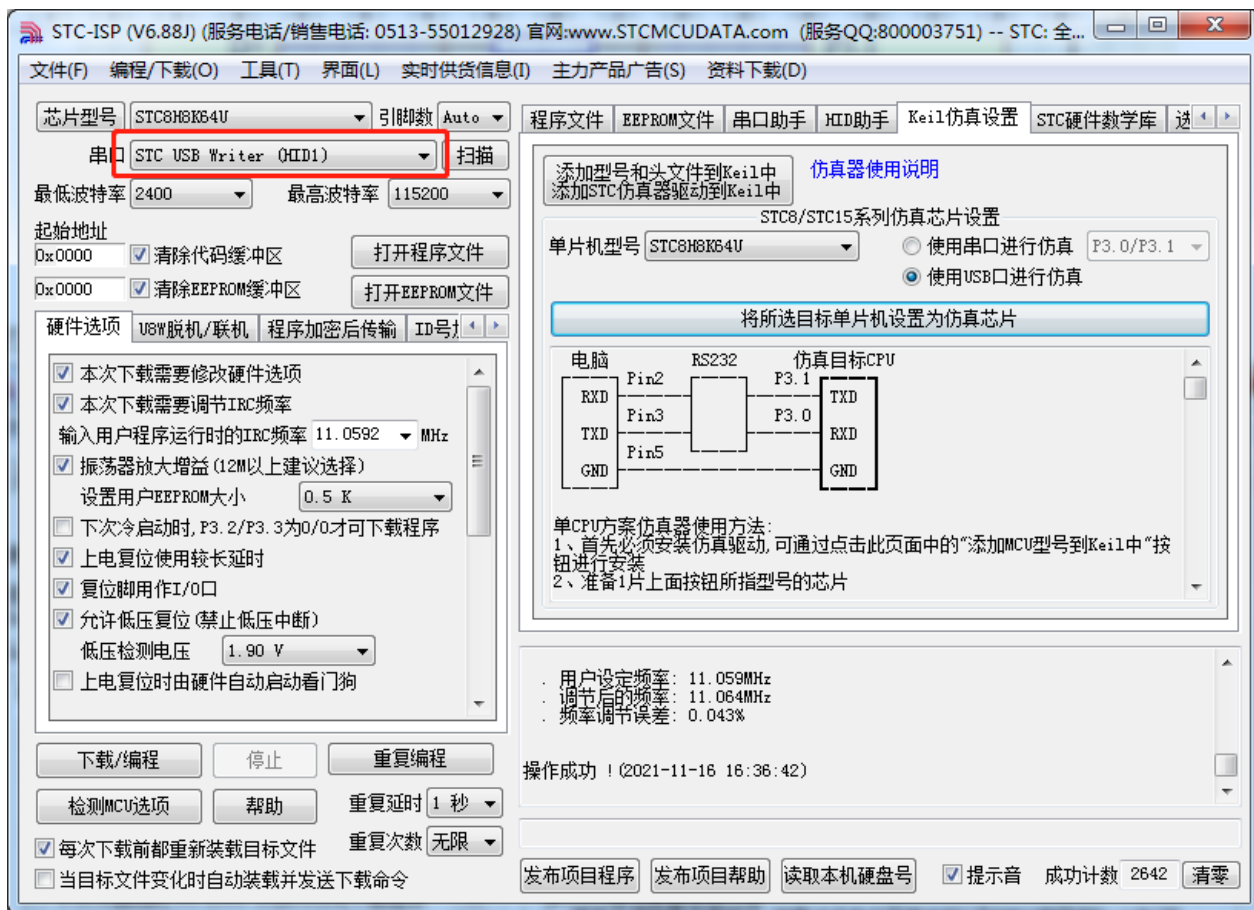
设置步骤如下：

首先将目标芯片如下图所示的方式和电脑的串口连接在一起，并将 P3.2 短路通过开关连接到 GND，然后给单片机上电

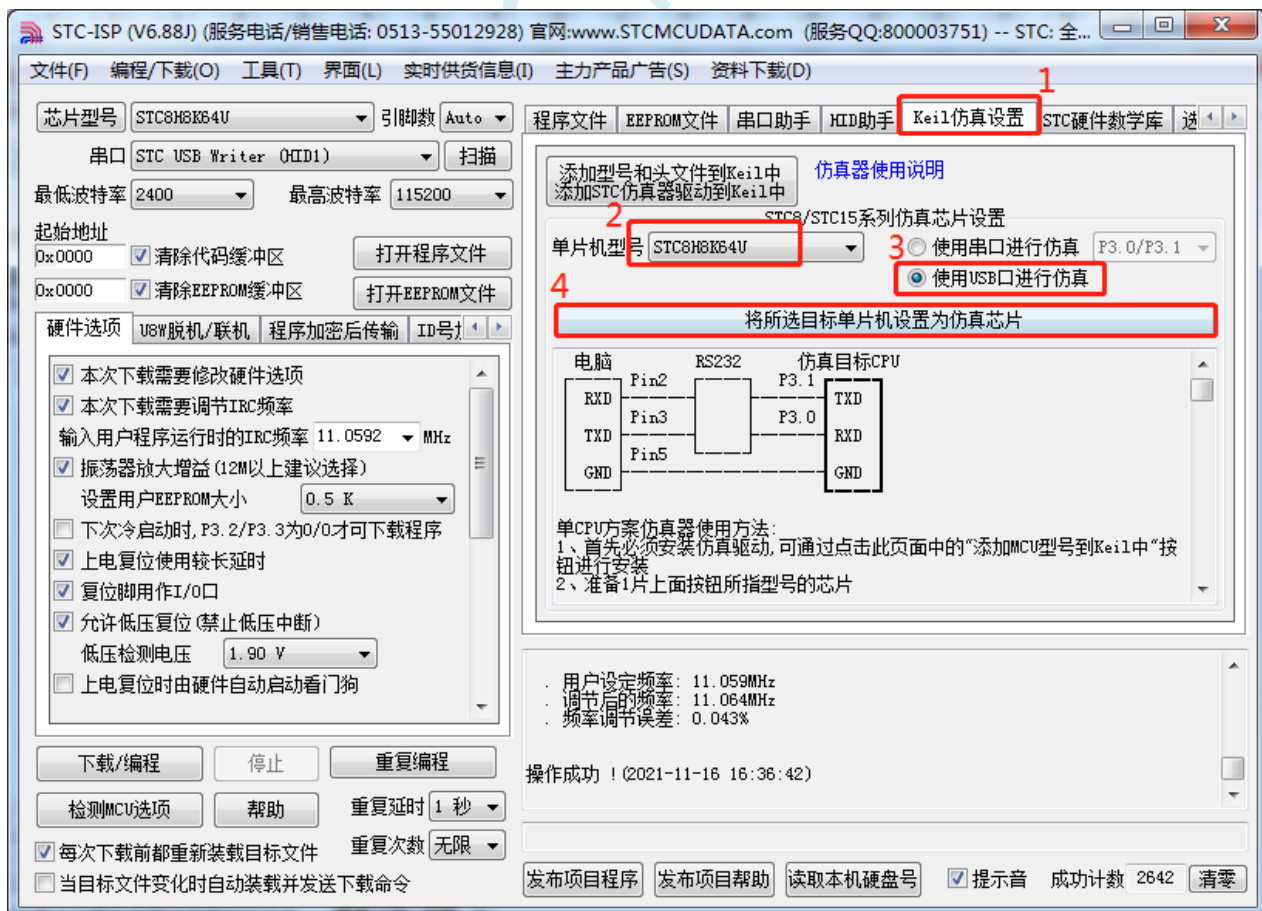


若在 ISP 软件中能自动扫描到“STC USB Writer (HID1)”表示连接正确





接下来在 STC-ISP 下载软件中, 按照如下图所示的步骤设置仿真芯片



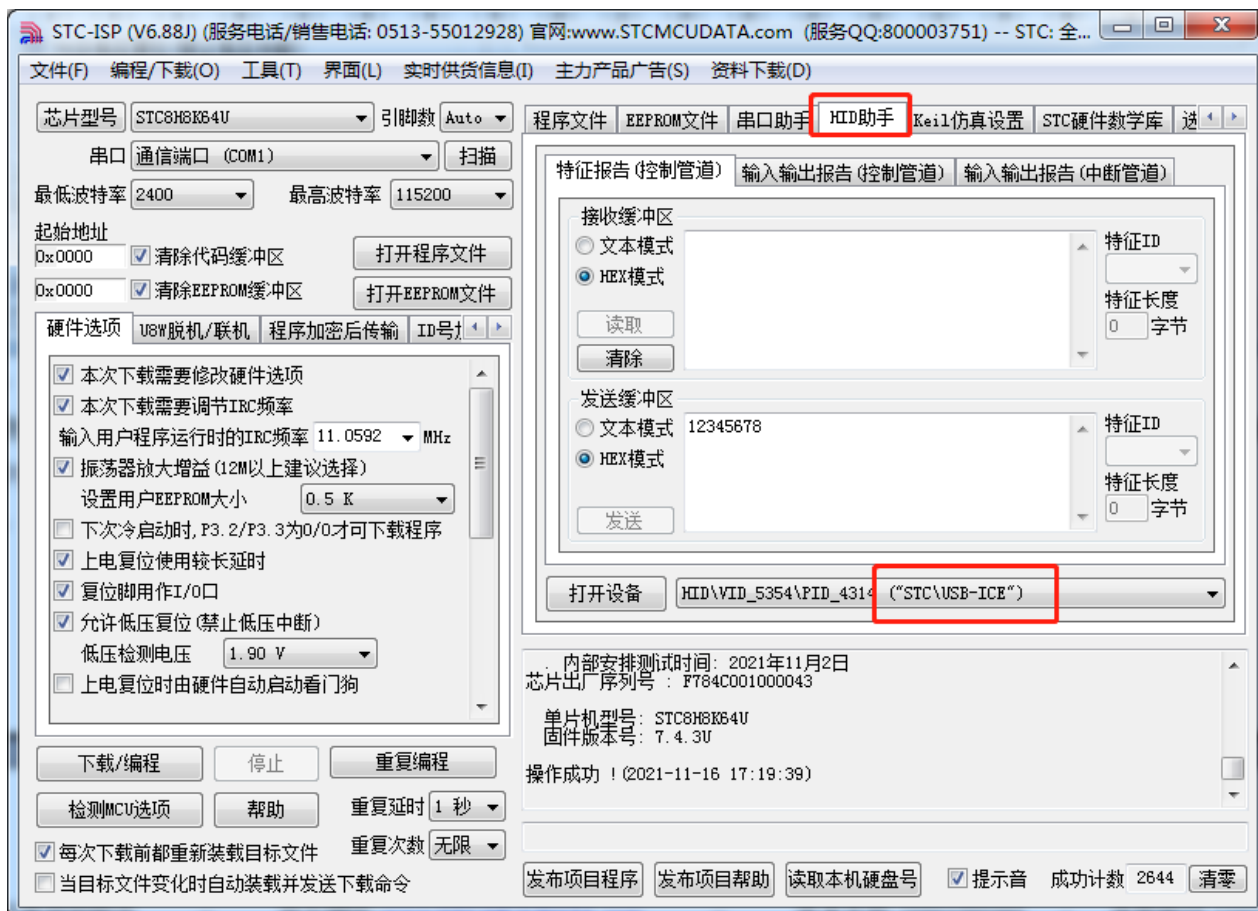


STC MCU

下载完成后如下图所示

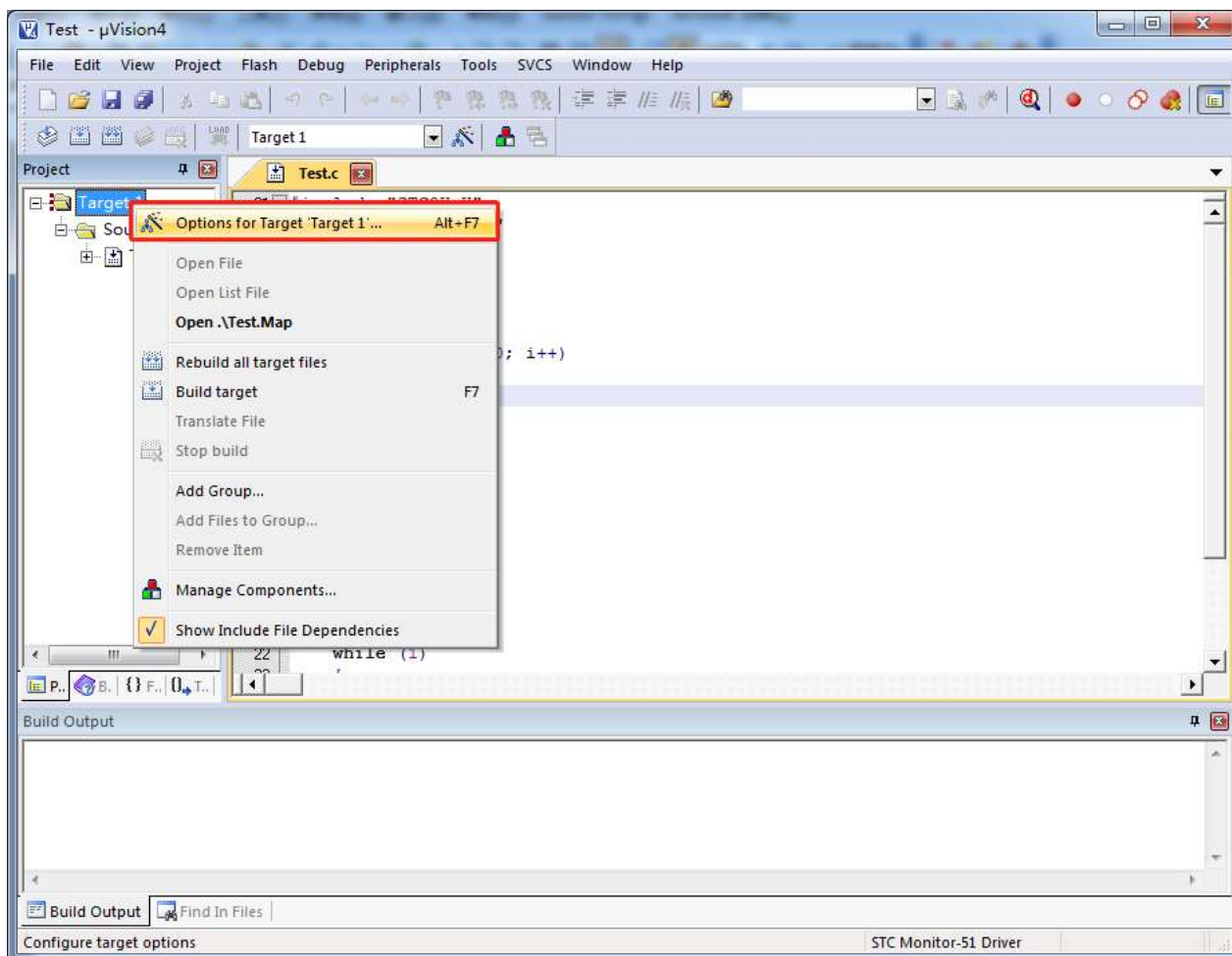


制作完成后, 需要将 P3.2 口的接地开关断开, 并重新对单片机上电, 若在下载软件的中“HID 助手”中能检测到“STC\USB-ICE”设备, 则表示 USB 仿真芯片制作成功

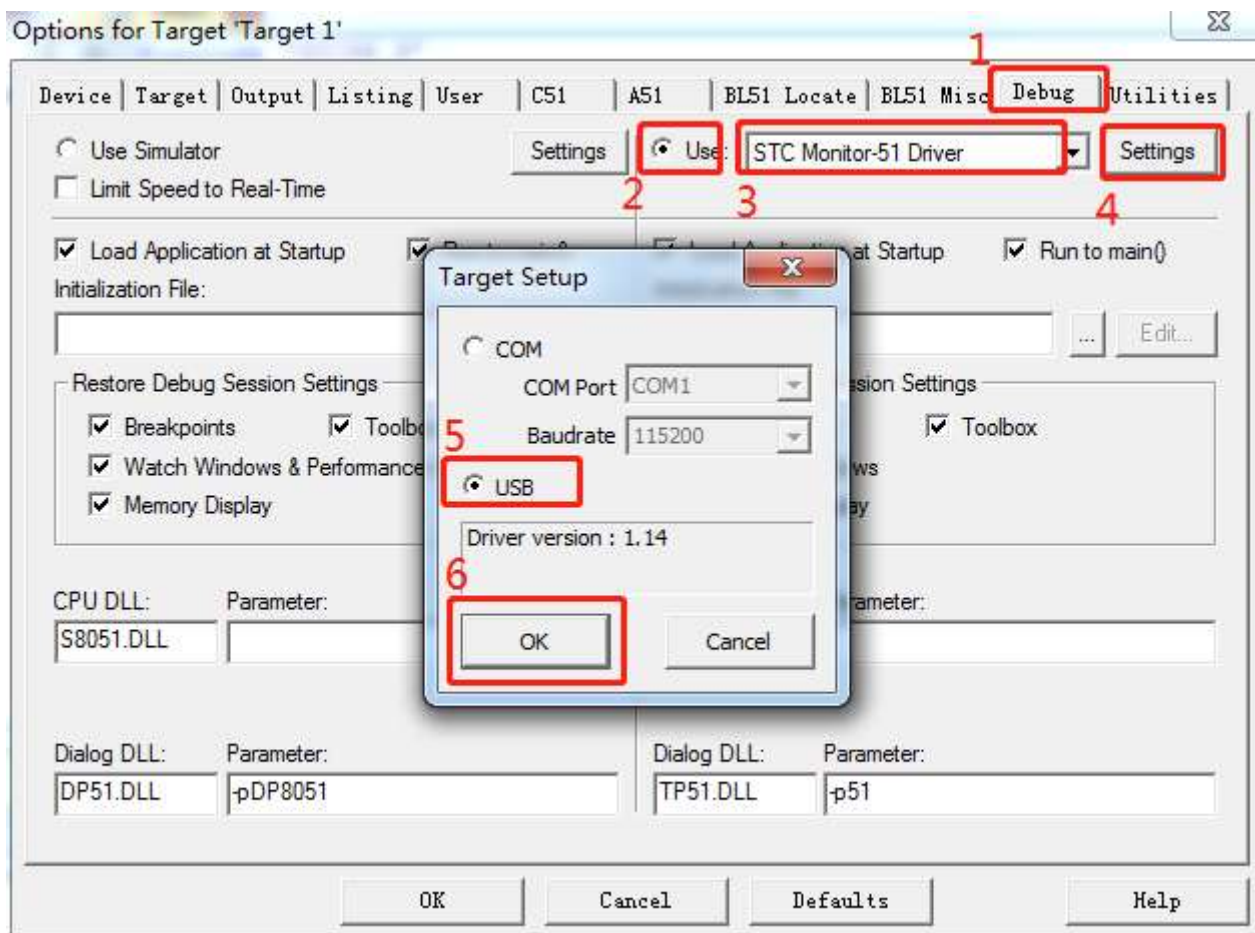


## J.5.2 在 Keil 软件中进行 USB 仿真设置

在 Keil 软件中打开项目文件，并在下图所示的右键菜单中点击“Options for ...”

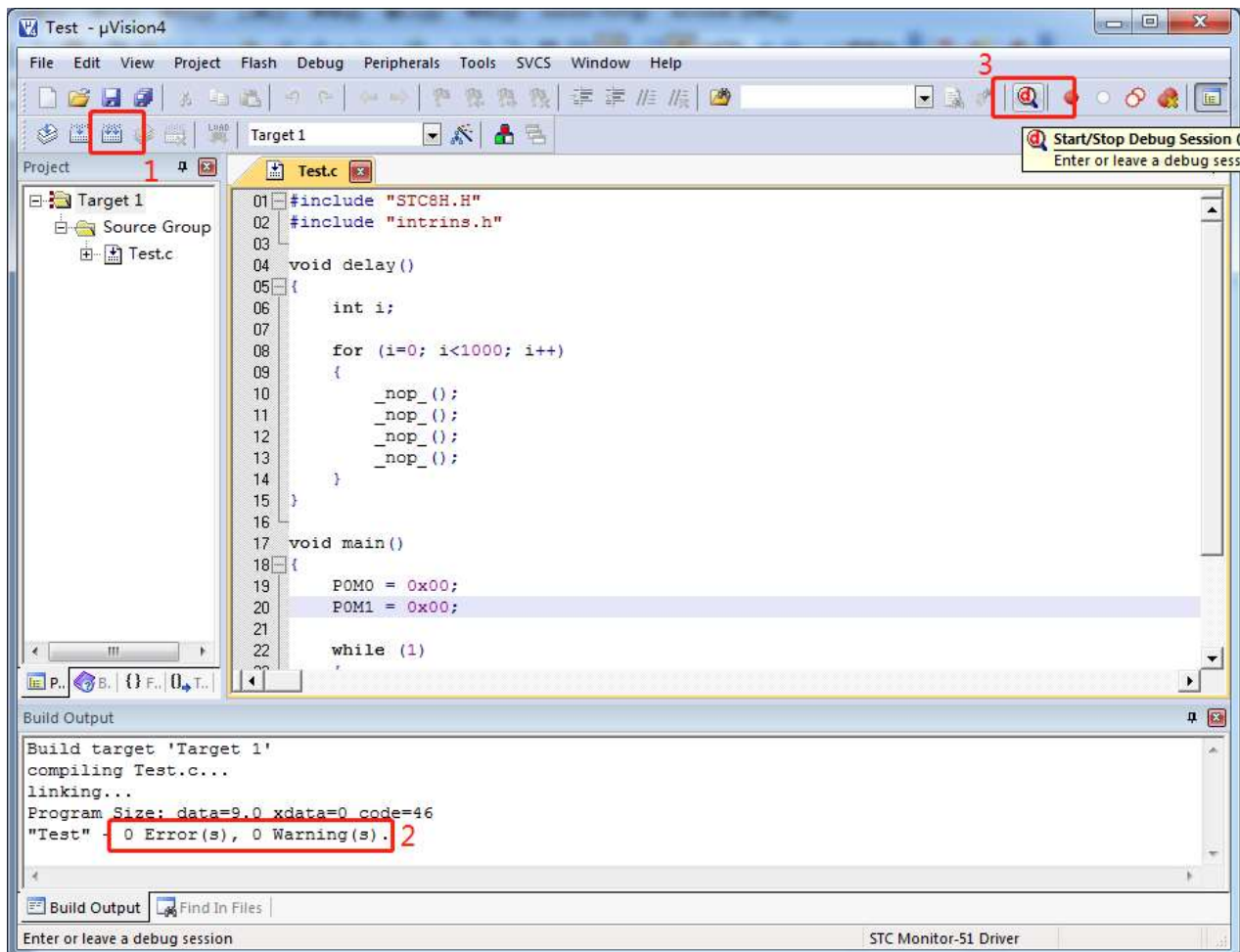


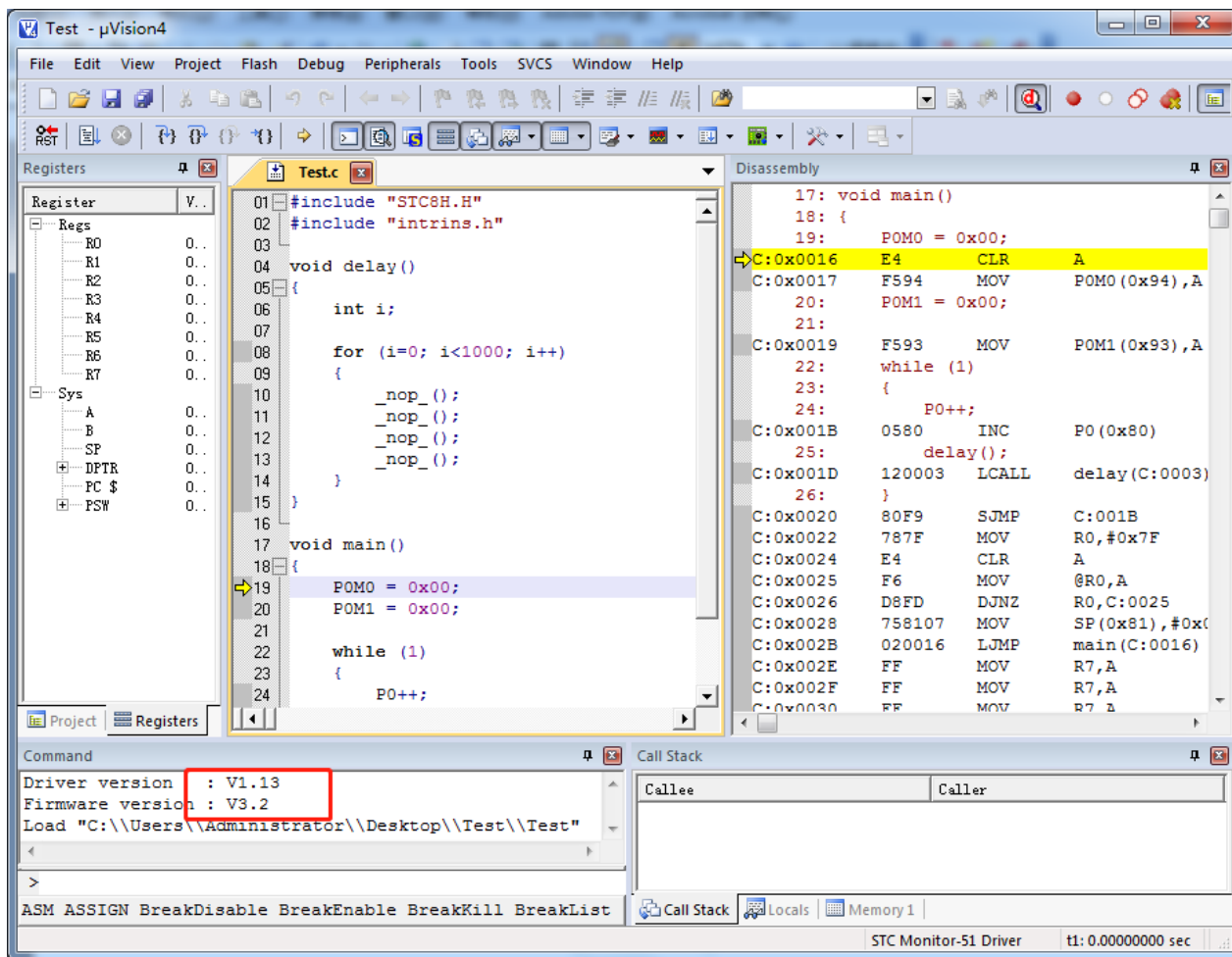
在项目选项中, 按如下图所示的步骤进行 USB 仿真设置



### J.5.3 在 Keil 软件中使用 USB 进行仿真

在 Keil 环境下, 编辑完成源代码, 并编译无误后, 即可开始仿真

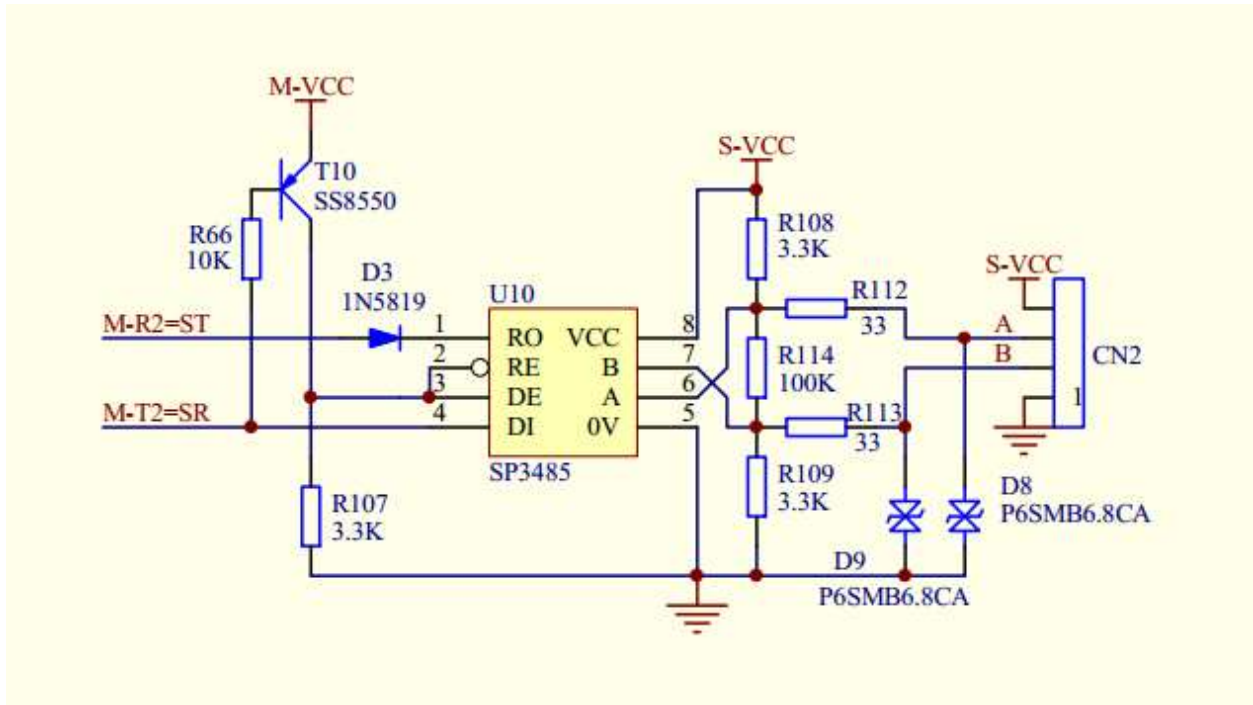




若芯片制作和连接均无误，则会如上图所示显示仿真驱动版本，并可正确下载用户代码到单片机，接下来便可进行运行、单步、断点等调试功能了。



# 附录K U8W 下载工具中 RS485 部分线路图



BOM 清单:

| 标号   | 型号         | 封装       | 备注      |
|------|------------|----------|---------|
| U10  | SP3485EN   | SOP8     | RS485芯片 |
| R66  | 10K        | 0603     | 电阻      |
| R107 | 3.3K       | 0603     | 电阻      |
| R108 | 3.3K       | 0603     | 电阻      |
| R109 | 3.3K       | 0603     | 电阻      |
| R112 | 33R        | 0603     | 电阻      |
| R113 | 33R        | 0603     | 电阻      |
| R114 | 100K       | 0603     | 电阻      |
| T10  | SS8550     | SOT-23   | PNP三极管  |
| D3   | 1N5819     | 0603     | 肖特基二极管  |
| D8   | P6SMB6.8CA | DO-214AA | TVS二极管  |
| D9   | P6SMB6.8CA | DO-214AA | TVS二极管  |
| CN2  |            | SIP4     | 通信接口    |



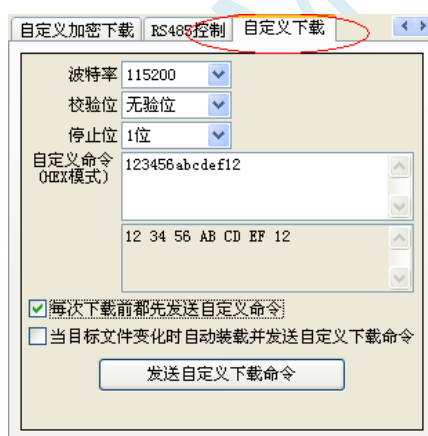
## 附录L 运行用户程序时收到用户命令后自动启动 ISP 下载(不停电)

“用户自定义下载”与“用户自定义加密下载”是两种完全不同功能。相对用户自定义加密下载的功能而言，用户自定义下载的功能要简单一些。

具体的功能为：电脑或脱机下载板在开始发送真正的 ISP 下载编程握手命令前，先发送用户自定义的一串命令（关于这一串串口命令，用户可以根据自己在应用程序中的串口设置来设置波特率、校验位以及停止位），然后再立即发送 ISP 下载编程握手命令。

“用户自定义下载”这一功能主要是在项目的早期开发阶段，实现不断电（不用给目标芯片重新上电）即可下载用户代码。具体的实现方法是：用户需要自己的程序中加入一段检测自定义命令的代码，当检测到后，执行一句“MOV IAP\_CONTR,#60H”的汇编代码或者“IAP\_CONTR = 0x60;”的 C 语言代码，MCU 就会自动复位到 ISP 区域执行 ISP 代码。

如下图所示，将自定义命令设置为波特率为 115200、无校验位、一位停止位的命令序列：0x12、0x34、0x56、0xAB、0xCD、0xEF、0x12。当勾选上“每次下载前都先发送自定义命令”的选项后，即可实现自定义下载功能



点击“发送自定义下载命令”或者点击界面左下角的“下载/编程”按钮，应用程序便会发送如下所示的串口数据



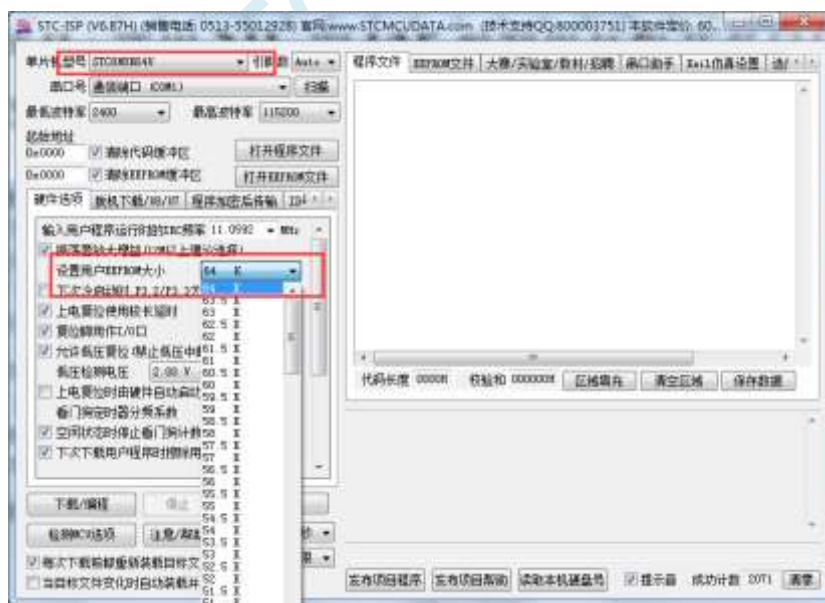
## 附录M 使用 STC 的 IAP 系列单片机开发自己的 ISP 程序

随着 IAP (In-Application-Programming) 技术在单片机领域的不断发展, 给应用系统程序代码升级带来了极大的方便。STC 的串口 ISP (In-System-Programming) 程序就是使用 IAP 功能来对用户的程序进行在线升级的, 但是出于对用户代码的安全着想, 底层代码和上层应用程序都没有开源, 为此 STC 推出了 IAP 系列单片机, 即整颗 MCU 的 Flash 空间, 用户均可在自己的程序中进行改写, 从而使得有用户需要开发自己的 ISP 程序的想法得以实现。

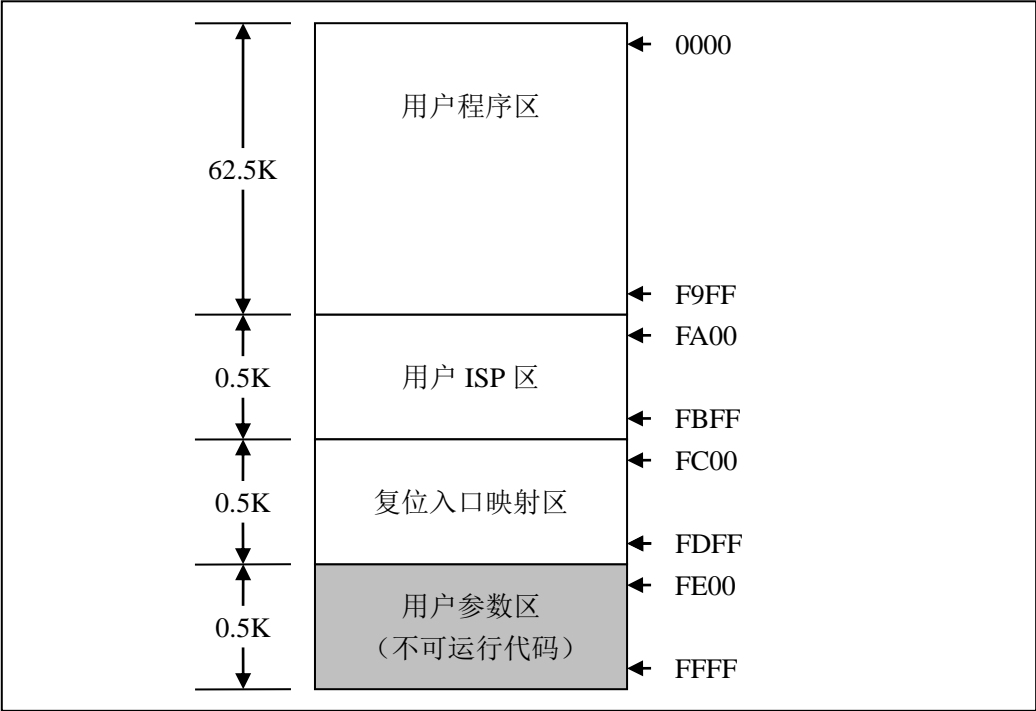
STC8H 系列单片机中的所有可以在 ISP 下载时用户自定义 EEPROM 大小的型号均为 IAP 系列单片机。目前 STC8H 系列有如下型号的单片为 IAP 系列: STC8H1K12、STC8H1K17、STC8H1K28、STC8H1K33、STC8H3K64S2、STC8H3K64S4、STC8H8K64U。本文以 STC8H8K64U 为例, 详细说明使用 STC 的 IAP 单片机开发用户自己的 ISP 程序的方法, 并给出了基于 Keil 环境的汇编和 C 源码。

### 第一步: 内部 FLASH 规划

由于 STC8H 系列的 IAP 型号单片机的 EEPROM 是在 ISP 下载时用户自己设置的, 所以若用户需要实现自己的 ISP, 则在下载用户自己的 ISP 程序时, 需要按照下图是方式, 将全部的 64K 都设置为 EEPROM, 让用户程序空间和 EEPROM 空间完全重合, 这样才能实现用户对自己程序空间进行修改和更新。

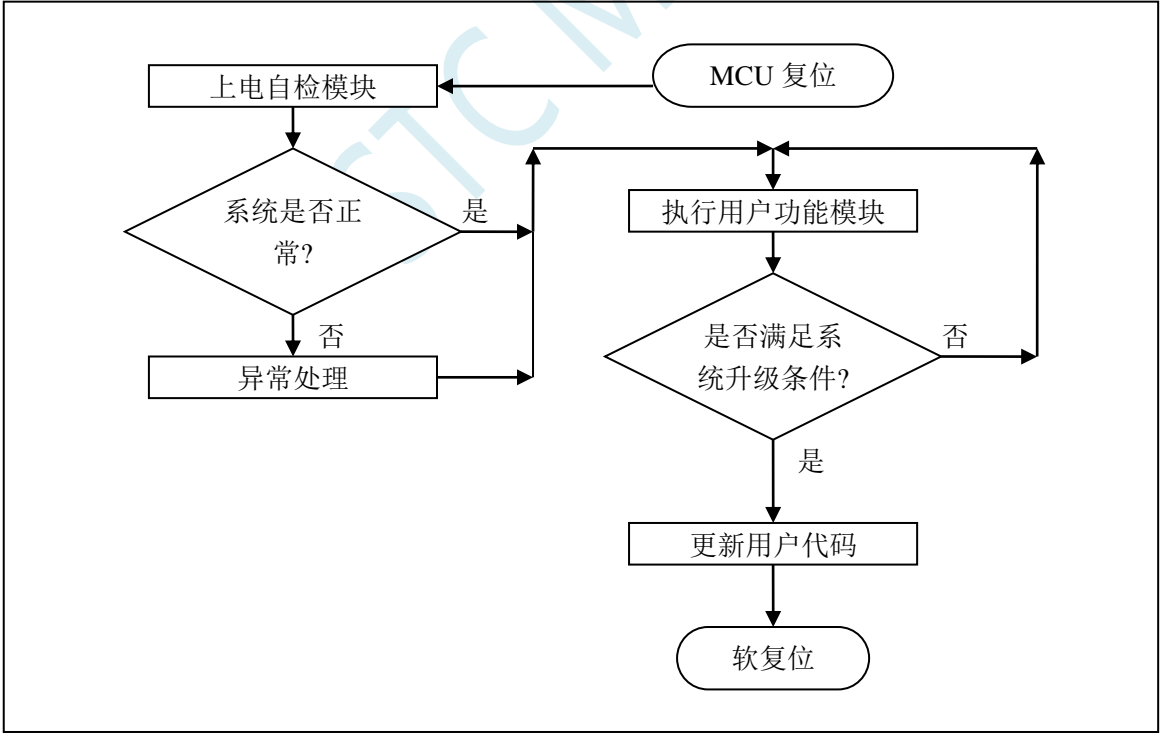


下面假设用户已将整个的 64K 的程序空间已全部设置为 EEPROM, 现将整个 64K 程序空间作如下划分:



FLASH 空间中，从地址 0000H 开始的连续 62.5K 字节的空间为用户程序区。当满足特定的下载条件时，需要用户将 PC 跳转到用户 ISP 程序区，此时可对用户程序区进行擦除和改写，以达到更新用户程序的目的。

第二步、程序的基本框架



第三步、下位机固件程序说明

下位机固件程序包括两部分：ISP（ISP 代码）和 AP（用户代码）

ISP 代码（汇编代码）

;测试工作频率为 11.0592MHz

```

UARTBAUD EQU 0FFE8H ;定义串口波特率 (65536-11059200/4/115200)

AUXR DATA 08EH ;附加功能控制寄存器
WDT_CONTR DATA 0C1H ;看门狗控制寄存器
IAP_DATA DATA 0C2H ;IAP 数据寄存器
IAP_ADDRH DATA 0C3H ;IAP 高地址寄存器
IAP_ADDRL DATA 0C4H ;IAP 低地址寄存器
IAP_CMD DATA 0C5H ;IAP 命令寄存器
IAP_TRIG DATA 0C6H ;IAP 命令触发寄存器
IAP_CONTR DATA 0C7H ;IAP 控制寄存器
IAP_TPS DATA 0F5H ;IAP 等待时间控制寄存器

ISPCODE EQU 0FA00H ;ISP 模块入口地址(1 页),同时也是外部接口地址
APENTRY EQU 0FC00H ;应用程序入口地址数据(1 页)

ORG 0000H

LJMP ISP_ENTRY ;系统复位入口

RESET:

MOV SCON,#50H ;设置串口模式(8 位数据位,无校验位)
MOV AUXR,#40H ;定时器1 为1T 模式
MOV TMOD,#00H ;定时器1 工作于模式0(16 位重载)
MOV TH1,#HIGH UARTBAUD ;设置重载值
MOV TL1,#LOW UARTBAUD
SETB TR1 ;启动定时器1

NEXT1:

MOV R0,#16

NEXT2:

JNB RI,$ ;等待串口数据
CLR RI
MOV A,SBUF
CJNE A,#7FH,NEXT1 ;判断是否为7F
DJNZ R0,NEXT2
LJMP ISP_DOWNLOAD ;跳转到下载界面

ORG ISPCODE

ISP_DOWNLOAD:

CLR A
MOV PSW,A ;ISP 模块使用第0 组寄存器
MOV IE,A ;关闭所有中断
CLR RI ;清除串口接收标志
SETB TI ;置串口发送标志
CLR TR0
MOV SP,#5FH ;设置堆栈指针

MOV A,#5AH ;返回 5A 55 到PC,表示ISP 擦除模块已准备就绪
LCALL ISP_SENDUART
MOV A,#055H
LCALL ISP_SENDUART
LCALL ISP_RECVACK ;接收应答数据

MOV IAP_ADDRL,#0 ;首先在第2 页起始地址写 "LJMP ISP_ENTRY" 指令
MOV IAP_ADDRH,#02H
LCALL ISP_ERASEIAP
MOV A,#02H
LCALL ISP_PROGRAMIAP ;编程用户代码复位向量代码

```

|                         |                                |                                 |
|-------------------------|--------------------------------|---------------------------------|
| <i>MOV</i>              | <i>A,#HIGH</i>                 | <i>ISP_ENTRY</i>                |
| <i>LCALL</i>            | <i>ISP_PROGRAMIAP</i>          | ;编程用户代码复位向量代码                   |
| <i>MOV</i>              | <i>A,#LOW ISP_ENTRY</i>        |                                 |
| <i>LCALL</i>            | <i>ISP_PROGRAMIAP</i>          | ;编程用户代码复位向量代码                   |
| <i>MOV</i>              | <i>IAP_ADDRL,#0</i>            | ;用户代码地址从 0 开始                   |
| <i>MOV</i>              | <i>IAP_ADDRH,#0</i>            |                                 |
| <i>LCALL</i>            | <i>ISP_ERASEIAP</i>            |                                 |
| <i>MOV</i>              | <i>A,#02H</i>                  |                                 |
| <i>LCALL</i>            | <i>ISP_PROGRAMIAP</i>          | ;编程用户代码复位向量代码                   |
| <i>MOV</i>              | <i>A,#HIGH</i>                 | <i>ISP_ENTRY</i>                |
| <i>LCALL</i>            | <i>ISP_PROGRAMIAP</i>          | ;编程用户代码复位向量代码                   |
| <i>MOV</i>              | <i>A,#LOW ISP_ENTRY</i>        |                                 |
| <i>LCALL</i>            | <i>ISP_PROGRAMIAP</i>          | ;编程用户代码复位向量代码                   |
| <i>MOV</i>              | <i>IAP_ADDRL,#0</i>            | ;新代码缓冲区地址                       |
| <i>MOV</i>              | <i>IAP_ADDRH,#02H</i>          |                                 |
| <i>MOV</i>              | <i>R7,#124</i>                 | ;擦除 62.5K 字节                    |
| <i>ISP_ERASEAP:</i>     |                                |                                 |
| <i>LCALL</i>            | <i>ISP_ERASEIAP</i>            |                                 |
| <i>INC</i>              | <i>IAP_ADDRH</i>               | ;目标地址+512                       |
| <i>INC</i>              | <i>IAP_ADDRH</i>               |                                 |
| <i>DJNZ</i>             | <i>R7,ISP_ERASEAP</i>          | ;判断是否擦除完成                       |
| <i>MOV</i>              | <i>IAP_ADDRL,#LOW APENTRY</i>  |                                 |
| <i>MOV</i>              | <i>IAP_ADDRH,#HIGH APENTRY</i> |                                 |
| <i>LCALL</i>            | <i>ISP_ERASEIAP</i>            |                                 |
| <i>MOV</i>              | <i>A,#5AH</i>                  | ;返回 5A A5 到 PC,表示 ISP 编程模块已准备就绪 |
| <i>LCALL</i>            | <i>ISP_SENDUART</i>            |                                 |
| <i>MOV</i>              | <i>A,#0A5H</i>                 |                                 |
| <i>LCALL</i>            | <i>ISP_SENDUART</i>            |                                 |
| <i>LCALL</i>            | <i>ISP_RECVACK</i>             | ;接收应答数据                         |
| <i>LCALL</i>            | <i>ISP_RECVUART</i>            | ;接收长度高字节                        |
| <i>MOV</i>              | <i>R0,A</i>                    |                                 |
| <i>LCALL</i>            | <i>ISP_RECVUART</i>            | ;接收长度低字节                        |
| <i>MOV</i>              | <i>R1,A</i>                    |                                 |
| <i>CLR</i>              | <i>C</i>                       | ;将总长度-3                         |
| <i>MOV</i>              | <i>A,#03H</i>                  |                                 |
| <i>SUBB</i>             | <i>A,R1</i>                    |                                 |
| <i>MOV</i>              | <i>DPL,A</i>                   |                                 |
| <i>CLR</i>              | <i>A</i>                       |                                 |
| <i>SUBB</i>             | <i>A,R0</i>                    |                                 |
| <i>MOV</i>              | <i>DPH,A</i>                   | ;总长度补码存入 DPTR                   |
| <i>LCALL</i>            | <i>ISP_RECVUART</i>            | ;映射用户代码复位入口代码到映射区               |
| <i>LCALL</i>            | <i>ISP_PROGRAMIAP</i>          | ;0000                           |
| <i>LCALL</i>            | <i>ISP_RECVUART</i>            |                                 |
| <i>LCALL</i>            | <i>ISP_PROGRAMIAP</i>          | ;0001                           |
| <i>LCALL</i>            | <i>ISP_RECVUART</i>            |                                 |
| <i>LCALL</i>            | <i>ISP_PROGRAMIAP</i>          | ;0002                           |
| <i>MOV</i>              | <i>IAP_ADDRL,#03H</i>          | ;用户代码起始地址                       |
| <i>MOV</i>              | <i>IAP_ADDRH,#00H</i>          |                                 |
| <i>ISP_PROGRAMNEXT:</i> |                                |                                 |
| <i>LCALL</i>            | <i>ISP_RECVUART</i>            | ;接收代码数据                         |
| <i>LCALL</i>            | <i>ISP_PROGRAMIAP</i>          | ;编程到用户代码区                       |
| <i>INC</i>              | <i>DPTR</i>                    |                                 |

```

MOV    A,DPL
ORL    A,DPH
JNZ    ISP_PROGRAMNEXT    ;长度检测

ISP_SOFTRESET:
MOV    IAP_CONTR,#20H    ;软件复位系统
SJMP   $

ISP_ENTRY:
MOV    WDT_CONTR,#17H    ;清看门狗
MOV    IAP_CONTR,#80H    ;使能 IAP 功能
MOV    IAP_TPS,#11    ;设置 IAP 等待时间参数
MOV    IAP_ADDRL,#LOW ISP_DOWNLOAD
MOV    IAP_ADDRH,#HIGH ISP_DOWNLOAD
MOV    IAP_DATA,#00H    ;测试数据 1
MOV    IAP_CMD,#1    ;读命令
MOV    IAP_TRIG,#5AH    ;触发 ISP 命令
MOV    IAP_TRIG,#0A5H
MOV    A,IAP_DATA
CJNE   A,#0E4H,ISP_ENTRY    ;若无法读出数据则需要等待电压稳定
INC    IAP_ADDRL    ;测试地址 FC01H
MOV    IAP_DATA,#45H    ;测试数据 2
MOV    IAP_CMD,#1    ;读命令
MOV    IAP_TRIG,#5AH    ;触发 ISP 命令
MOV    IAP_TRIG,#0A5H
MOV    A,IAP_DATA
CJNE   A,#0F5H,ISP_ENTRY    ;若无法读出数据则需要等待电压稳定

MOV    SCON,#50H    ;设置串口模式(8 位数据位,无校验位)
MOV    AUXR,#40H    ;定时器 1 为 1T 模式
MOV    TMOD,#00H    ;定时器 1 工作于模式 0(16 位重装载)
MOV    TH1,#HIGH UARTBAUD    ;设置重载值
MOV    TL1,#LOW UARTBAUD
SETB   TRI    ;启动定时器 1
SETB   TR0

LCALL  ISP_RECVUART    ;检测是否有串口数据
JC     GOTOAP
MOV    R0,#16

ISP_CHECKNEXT:
LCALL  ISP_RECVUART    ;接收同步数据
JC     GOTOAP
CJNE   A,#7FH,GOTOAP    ;判断是否为 7F
DJNZ   R0,ISP_CHECKNEXT
MOV    A,#5AH    ;返回 5A 69 到 PC,表示 ISP 模块已准备就绪
LCALL  ISP_SENDUART
MOV    A,#69H
LCALL  ISP_SENDUART
LCALL  ISP_RECVACK    ;接收应答数据
LJMP   ISP_DOWNLOAD    ;跳转到下载界面

GOTOAP:
CLR    A    ;将 SFR 恢复为复位值
MOV    TCON,A
MOV    TMOD,A
MOV    TL0,A
MOV    TH0,A
MOV    TL1,A
MOV    TH1,A

```



```

MOV     SCON,A
MOV     AUXR,A
LJMP    APENTRY           ;正常运行用户程序

```

**ISP\_RECVACK:**

```

LCALL   ISP_RECVUART
JC      GOTOAP
XRL     A,#7FH
JZ      ISP_RECVACK       ;跳过同步数据
CJNE    A,#25H,GOTOAP     ;应答数据1 检测
LCALL   ISP_RECVUART
JC      GOTOAP
CJNE    A,#69H,GOTOAP     ;应答数据2 检测
RET

```

**ISP\_RECVUART:**

```

CLR     A
MOV     TL0,A             ;初始化超时定时器
MOV     TH0,A
CLR     TF0
MOV     WDT_CONTR,#17H   ;清看门狗

```

**ISP\_RECVWAIT:**

```

JBC     TF0,ISP_RECVTIMEOUT ;超时检测
JNB     RI,ISP_RECVWAIT    ;等待接收完成
MOV     A,SBUF             ;读取串口数据
CLR     RI                 ;清除标志
CLR     C                  ;正确接收串口数据
RET

```

**ISP\_RECVTIMEOUT:**

```

SETB    C                  ;超时退出
RET

```

**ISP\_SENDUART:**

```

MOV     WDT_CONTR,#17H   ;清看门狗
JNB     TI,ISP_SENDUART  ;等待前一个数据发送完成
CLR     TI               ;清除标志
MOV     SBUF,A           ;发送当前数据
RET

```

**ISP\_ERASEIAP:**

```

MOV     WDT_CONTR,#17H   ;清看门狗
MOV     IAP_CMD,#3       ;擦除命令
MOV     IAP_TRIG,#5AH    ;触发ISP 命令
MOV     IAP_TRIG,#0A5H
NOP
NOP
NOP
NOP
RET

```

**ISP\_PROGRAMIAP:**

```

MOV     WDT_CONTR,#17H   ;清看门狗
MOV     IAP_CMD,#2       ;编程命令
MOV     IAP_DATA,A       ;将当前数据送 IAP 数据寄存器
MOV     IAP_TRIG,#5AH    ;触发ISP 命令
MOV     IAP_TRIG,#0A5H
NOP
NOP
NOP

```



```

NOP
MOV     A,IAP_ADDRL           ;IAP 地址+1
ADD     A,#01H
MOV     IAP_ADDRL,A
MOV     A,IAP_ADDRH
ADDC    A,#00H
MOV     IAP_ADDRH,A
RET

ORG     APENTRY
LJMP    RESET

END

```

ISP 代码包括如下外部接口模块:

ISP\_DOWNLOAD: 程序下载入口地址, 绝对地址 **FA00H**

ISP\_ENTRY: 上电系统自检程序 (系统自动调用)

对于用户程序而言, 用户只需要在满足下载条件时, 将 PC 值跳转到 ISPPROGRAM (即 FA00H 的绝对地址), 即可实现代码更新。

## 用户代码 (C 语言代码)

//测试工作频率为 11.0592MHz

```

#include "stc8h.h"

#define FOSC      11059200L      //系统时钟频率
#define BAUD      (65536 - (FOSC/115200+2)/4) //定义串口波特率
                                           //加2 操作是为了让 Keil 编译器
                                           //自动实现四舍五入运算
#define ISPPROGRAM 0xfa00      //ISP 下载程序入口地址

void (*IspProgram)() = ISPPROGRAM; //定义指针函数
char cnt7f;                        //Isp_Check 内部使用的变量

void uart() interrupt 4           //串口中断服务程序
{
    if (TI) TI = 0;              //发送完成中断
    if (RI)                      //接收完成中断
    {
        if (SBUF == 0x7f)
        {
            cnt7f++;
            if (cnt7f >= 16)
            {
                IspProgram();    //调用下载模块(**** 重要语句****)
            }
        }
        else
        {
            cnt7f = 0;
        }
        RI = 0;                 //清接收完成标志
    }
}

```

```

    }
}

void main()
{
    P_SW2 /= 0x80;                //使能访问 XFR

    SCON = 0x50;                  //定义串口模式为 8bit 可变,无校验位
    AUXR = 0x40;
    TH1 = BAUD >> 8;
    TL1 = BAUD;
    TR1 = 1;
    ES = 1;                        //使能串口中断
    EA = 1;                        //打开全局中断开关

    PIM0 = 0;
    PIM1 = 0;

    while (1)
    {
        PI++;
    }
}
```

用户代码（汇编代码）

;测试工作频率为 11.0592MHz

```

UARTBAUD    EQU    0FFE8H        ;定义串口波特率    (65536-11059200/4/115200)
ISPPROGRAM  EQU    0FA00H        ;ISP 下载程序入口地址

AUXR        DATA    08EH        ;附件功能控制寄存器

CNT7F       DATA    60H        ;接收 7F 的计数器

            ORG    0000H
            LJMP   START        ;系统复位入口

            ORG    0023H
            LJMP   UART_ISR     ;串口中断入口

UART_ISR:
    PUSH    ACC
    PUSH    PSW
    JNB     TI,CHECKRI          ;检测发送中断
    CLR     TI                  ;清除标志

CHECKRI:
    JNB     RI,UARTISR_EXIT     ;检测接收中断
    CLR     RI                  ;清除标志
    MOV     A,SBUF
    CJNE    A,#7FH,ISNOT7F
    INC     CNT7F
    MOV     A,CNT7F
    CJNE    A,#16,UARTISR_EXIT
    LJMP    ISPPROGRAM          ;调用下载模块(**** 重要语句****)

ISNOT7F:
    MOV     CNT7F,#0

UARTISR_EXIT:
    POP     PSW
```

*POP*  
*RETI*

*ACC*

*START:*

|             |                           |                           |
|-------------|---------------------------|---------------------------|
| <i>MOV</i>  | <i>R0,#7FH</i>            | <i>;清RAM</i>              |
| <i>CLR</i>  | <i>A</i>                  |                           |
| <i>MOV</i>  | <i>@R0,A</i>              |                           |
| <i>DJNZ</i> | <i>R0,\$-1</i>            |                           |
| <i>MOV</i>  | <i>SP,#7FH</i>            | <i>;初始化SP</i>             |
| <i>MOV</i>  | <i>SCON,#50H</i>          | <i>;设置串口模式(8位可变,无校验位)</i> |
| <i>MOV</i>  | <i>AUXR,#15H</i>          | <i>;BRT工作于1T模式,启动BRT</i>  |
| <i>MOV</i>  | <i>TMOD,#00H</i>          | <i>;定时器1工作于模式0(16位重载)</i> |
| <i>MOV</i>  | <i>TH1,#HIGH UARTBAUD</i> | <i>;设置重载值</i>             |
| <i>MOV</i>  | <i>TL1,#LOW UARTBAUD</i>  |                           |
| <i>SETB</i> | <i>TR1</i>                | <i>;启动定时器1</i>            |
| <i>SETB</i> | <i>ES</i>                 | <i>;使能串口中断</i>            |
| <i>SETB</i> | <i>EA</i>                 | <i>;开中断总开关</i>            |

*MAIN:*

|             |             |
|-------------|-------------|
| <i>INC</i>  | <i>P0</i>   |
| <i>SJMP</i> | <i>MAIN</i> |

*END*

用户代码可以使用 C 或者汇编语言编写, 但对于汇编代码需要注意一点: 位于 0000H 的复位入口地址处的指令必须是一个长跳转语句 (类似 LJMP START)。在用户代码中, 需要设置好串口, 并在满足下载条件时, 将 PC 值跳转到 ISPPROGRAM (即 FA00H 的绝对地址), 以实现代码更新。对于汇编代码, 我们可以使用 “LJMP 0FA00H” 指令进行调用, 如下图

|            |      |        |                                    |
|------------|------|--------|------------------------------------|
| UARTBAUD   | EQU  | 0FFE8H | ;定义串口波特率 (65536-11059200/4/115200) |
| ISPPROGRAM | EQU  | 0FA00H | ;ISP下载程序入口地址                       |
| AUXR       | DATA | 08EH   | ;附件功能控制寄存器                         |

```

18      CLR      TI                ;清除标志
19 CHECKRI:
20      JNB      RI, UARTISR_EXIT  ;检测接收中断
21      CLR      RI                ;清除标志
22      MOV      A, SBUF
23      CJNE     A, #7FH, ISNOT7F
24      INC      CNT7F
25      MOV      A, CNT7F
26      CJNE     A, #16, UARTISR_EXIT
27      LJMP     ISPPROGRAM        ;调用下载模块 (****重要语句****)
28 ISNOT7F:
29      MOV      CNT7F, #0
30 UARTISR_EXIT:
31      POP      PSW
32      POP      ACC
33      RETI
34
35 START:

```

在 C 代码中, 必须定义一个函数指针变量, 并将此变量赋值为 0xFA00, 然后再调用, 如下图

```

#include "reg51.h"

#define FOSC      11059200L        //系统时钟频率
#define BAUD      (65536 - FOSC/4/115200) //定义串口波特率
#define ISPPROGRAM 0xfa00         //ISP下载程序入口地址

sfr AUXR      = 0x8e;             //波特率发生器控制寄存器
sfr P1M0      = 0x92;
sfr P1M1      = 0x91;

void (*IspProgram)() = ISPPROGRAM; //定义指针函数
char cnt7f;                        //Isp_Check内部使用的变量

void uart() interrupt 4            //串口中断服务程序
{
    if (TI) TI = 0;                //发送完成中断
    if (RI)                        //接收完成中断
    {
        if (SBUF == 0x7f)
        {
            cnt7f++;
            if (cnt7f >= 16)
            {
                IspProgram();        //调用下载模块 (****重要语句****)
            }
        }
        else
        {
            cnt7f = 0;
        }
    }
    RI = 0;                        //清接收完成标志
}

```

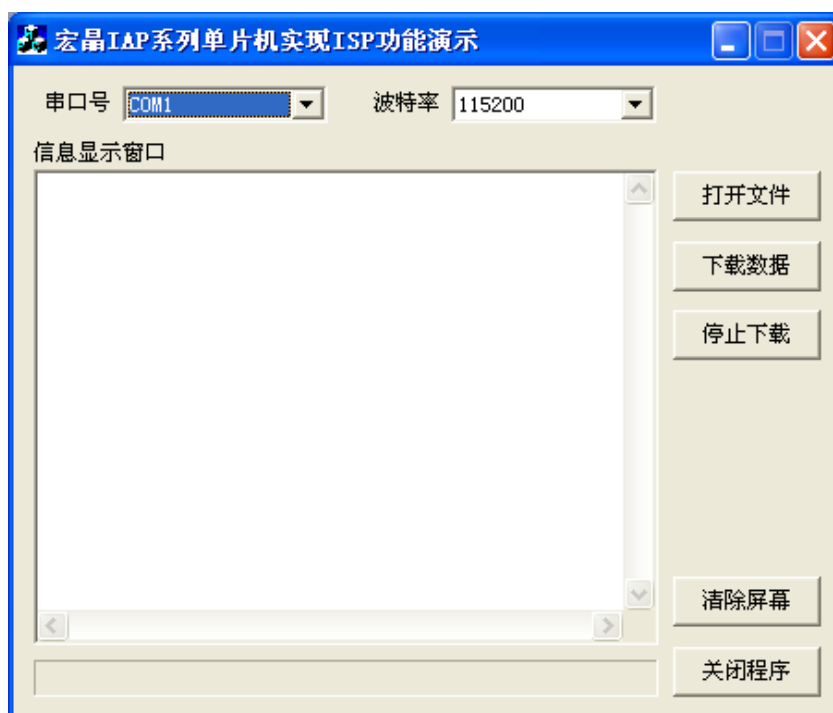
## 第四步、上位机应用程序说明

上位机的程序是基于 MFC 的对话框项目，对于串口的访问是直接调用 Windows 的 API 函数，而没有使用串口控件，从而省去的控件的注册以及系统版本不兼容的诸多问题。界面较简单，只是为这一功能的实现提供了一个框架，其他的功能及要求均还可以往上面添加。

上位机程序的核心模块是基于类 CISPDIg 的一个友元函数 “UINT Download(LPVOID pParam);”，此函数负责与下位机通讯，发送各种通讯命令来完成对用户程序的更新。用户可以根据各自不同的需求增加命令。

## 第五步、上位机应用程序的使用方法

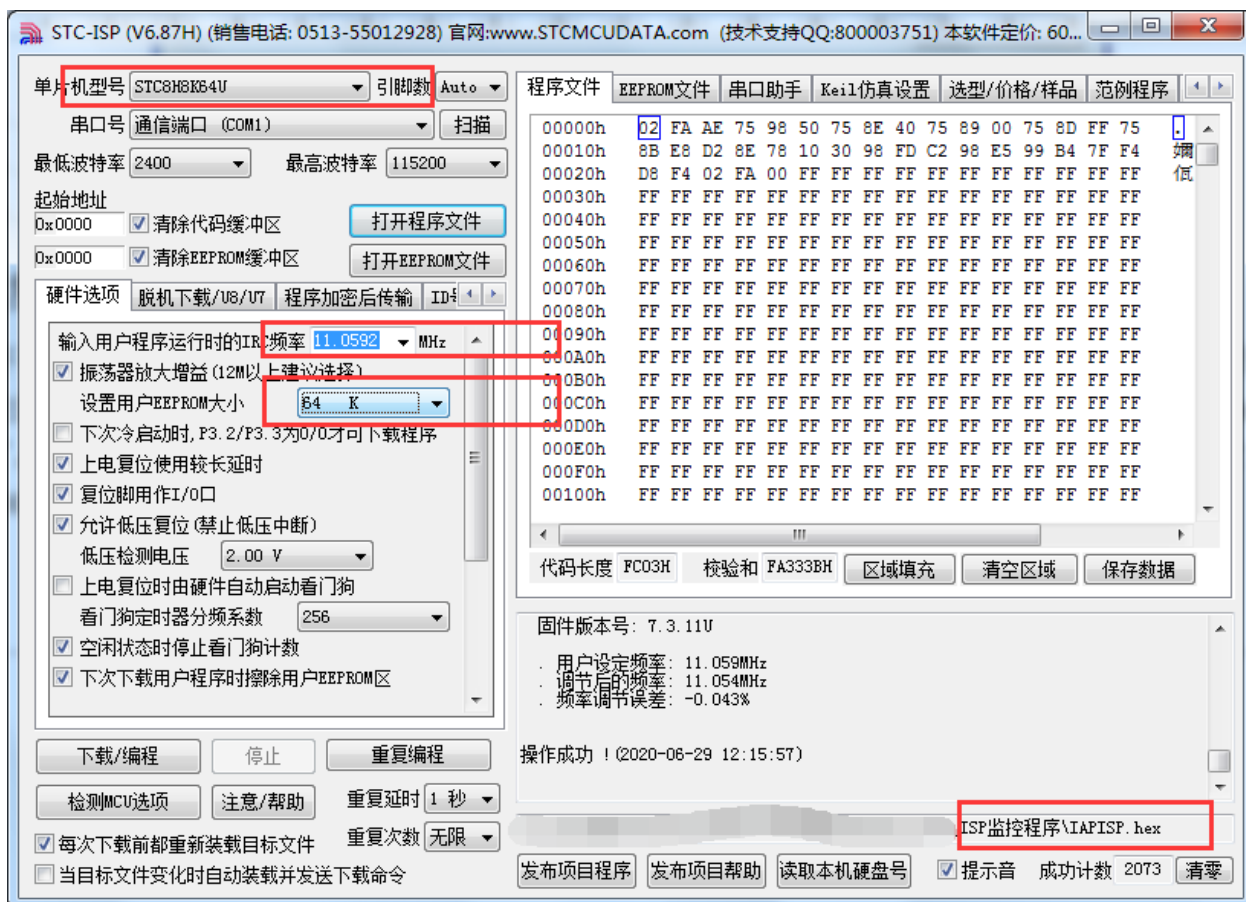
- 打开上位机界面，如下图



- 选择串口号，设置与下位机相同的串口波特率
- 打开要下载的源数据文件，Bin 或者 Intel hex 格式均可以
- 点击“下载数据”按钮即可开始下载数据

## 第六步、下位机固件程序的使用方法

下位机的目标文件有两个“IAPISP.hex”和“AP.hex”，对于一块新的单片机，第一次必须使用 ISP 下载工具将“IAPISP.hex”写入到芯片内，如下图所示。之后再更新便不再需要写“IAPISP.hex”这个文件了，附件中的“AP.hex”只是一个用户程序的模板，当满足下载条件时，用户只需要将 PC 值跳转到 FA00H 的地址，即可实现代码更新。



## 附录N 用户程序复位到系统区进行 ISP 下载的方法（不停电）

当项目处于开发阶段时，需要反复的下载用户代码到目标芯片中进行代码验证，而 STC 的单片机进行正常的 ISP 下载都需要对目标芯片进行重新上电，从而会使得项目在开发阶段比较繁琐。为此 STC 单片机增加了一个特殊功能寄存器 IAP\_CONTR，当用户向此寄存器写入 0x60，即可实现软件复位到系统区，进而实现不停电就可进行 ISP 下载。

但是用户如何判断是否正在进行 ISP 下载？何时向寄存器 IAP\_CONTR 写 0x60 触发软复位？就这两个问题，下面分别介绍四种判断方法：

### 使用 P3.0 口检测串口起始信号

STC 单片机的串口 ISP 固定使用 P3.0 和 P3.1 两个端口，当 ISP 下载软件开始下载时，会发送握手命令到单片机的 P3.0 口。若用户的 P3.0 和 P3.1 只是专门用于 ISP 下载，则可使用 P3.0 口检测串口的起始信号来判断 ISP 下载。

#### C 语言代码

---

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void main()
```

```
{
```

```
    P_SW2 /= 0x80;
```

```
//使能访问 XFR
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P30 = 1;
```

```
    while (1)
```

```
    {
```

```
        if (!P30) IAP_CONTR = 0x60;
```

```
//P3.0 的低电平即为串口起始信号
```

```
//软件复位到系统区
```

```
        ...
```

```
//用户代码
```

```
    }
```

```
}
```

---

### 使用 P3.0/INT4 口的下降沿中断，检测串口起始信号

方法 B 与方法 A 类似, 不同在于方法 A 使用的是查询方式, 方法 B 使用中断方式。因为 STC 单片机的 P3.0 口为 INT4 的中断口。

## C 语言代码

---

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
void Int4Isr() interrupt 16           //INT4 中断服务程序
{
    IAP_CONTR = 0x60;                //串口起始信号触发 INT4 中断
                                      //软件复位到系统区
}

void main()
{
    P_SW2 /= 0x80;                    //使能访问 XFR

    P3M0 = 0x00;
    P3M1 = 0x00;

    INTCLKO /= 0x40;                  //使能 INT4 中断
    EA = 1;

    while (1)
    {
        ...                           //用户代码
    }
}
```

---

## 使用 P3.0/RxD 口的串口接收, 检测 ISP 下载软件发送的 7F

方法 A 与方法 B 都非常简单, 但容易受干扰, 如果 P3.0 口有任何一个干扰信号, 都会触发软件复位, 所以方法 C 是对串口数据进行校验。

STC 的 ISP 下载软件进行 ISP 下载时, 首先都会使用最低波特率 (一般是 2400) + 偶校验 9+1 位停止位连续发送握手命令 7F, 因此用户可以在程序中, 将串口设置为 9 位数据位+2400 波特率, 然后持续检测 7F, 比如连续检测到 8 个 7F 表示可确定需要进行 ISP 下载, 此时再触发软件复位。

## C 语言代码

---

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BR2400    (65536 - FOSC / 4 / 2400)
```

```
char cnt7f;
```



```
void UartIsr() interrupt 4 //串口中断服务程序
{
    if (TI)
    {
        TI = 0;
    }

    if (RI)
    {
        RI = 0;
        if ((SBUF == 0x7f) && (RB8 == 1)) //ISP 下载软件发送的握手命令 7F
            //7F 的偶校验位为 1
        {
            if (++cnt7f == 8) //当连续检测到 8 个 7F 后
                IAP_CONTR = 0x60; //复位到系统区
        }
        else
        {
            cnt7f = 0;
        }
    }
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P3M0 = 0x00;
    P3M1 = 0x00;

    SCON = 0xd0; //设置串口为 9 位数据位
    TMOD = 0x00;
    AUXR = 0x40;
    TH1 = BR2400 >> 8; //设置串口波特率为 2400
    TL1 = BR2400;
    TR1 = 1;
    ES = 1;
    EA = 1;

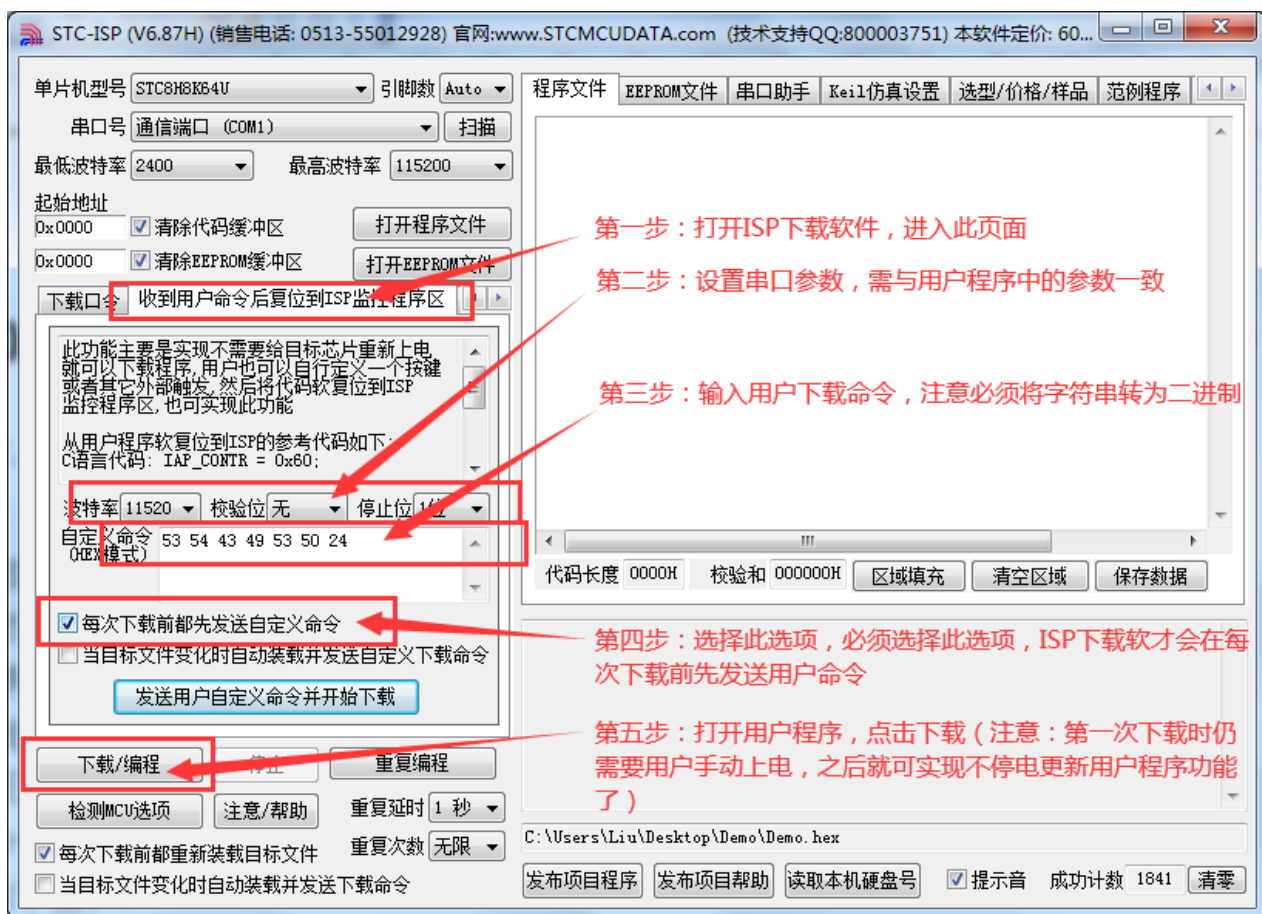
    cnt7f = 0;

    while (1)
    {
        ... //用户代码
    }
}
```

## 使用 P3.0/RxD 串口接收，检测 ISP 下载软件发送的用户下载命令

如果用户代码中需要使用串口进行通信，则上面的 3 中方法可能都不太适用，此时可以使用 STC 的 ISP 下载软件提供的接口，定制一组专用的用户下载命令（可指定波特率、校验位和停止位），若使能此功能，ISP 下载软件在进行 ISP 下载前，会使用用户指定的波特率、校验位和停止位发送用户下载命令，然后再发送握手命令。用户只需要在自己的代码中监控串口命令序列，当检测到有正确的用户下载命令时，软件复位到系统区即可实现不停电进行 ISP 功能。

下面假设用户下载命令为字符串“STCISP\$”，串口设置为波特率 115200，无校验位和 1 位停止位。ISP 下载软件中的设置如下图：



用户示例代码如下：

### C 语言代码

//测试工作频率为 11.0592MHz

```
#include "stc8h.h"
#include "intrins.h"
```

```
#define FOSC 11059200UL
#define BR115200 (65536 - (FOSC / 115200 + 2) / 4)
```

//加 2 操作是为了让 Keil 编译器  
//自动实现四舍五入运算

```
char stage;
```

```
void UartIsr() interrupt 4
{
```

//串口中断服务程序

```
    char dat;
```

```
    if (TI)
    {
        TI = 0;
    }
```

```
    if (RI)
    {
        RI = 0;
    }
```

```

    dat = SBUF;
    switch (stage)
    {
    case 0:
    default:
L_Check1st:
        if (dat == 'S') stage = 1;
        else stage = 0;
        break;
    case 1:
        if (dat == 'T') stage = 2;
        else goto L_Check1st;
        break;
    case 2:
        if (dat == 'C') stage = 3;
        else goto L_Check1st;
        break;
    case 3:
        if (dat == 'I') stage = 4;
        else goto L_Check1st;
        break;
    case 4:
        if (dat == 'S') stage = 5;
        else goto L_Check1st;
        break;
    case 5:
        if (dat == 'P') stage = 6;
        else goto L_Check1st;
        break;
    case 6:
        if (dat == '$')
            IAP_CONTR = 0x60; //当检测到正确的用户下载命令时
            //复位到系统区
        else goto L_Check1st;
        break;
    }
}

void main()
{
    P_SW2 /= 0x80; //使能访问 XFR

    P3M0 = 0x00;
    P3M1 = 0x00;

    SCON = 0x50; //设置用户串口模式为 8 位数据位
    TMOD = 0x00;
    AUXR = 0x40;
    TH1 = BR2400 >> 8; //设置串口波特率为 115200
    TL1 = BR2400;
    TR1 = 1;
    ES = 1;
    EA = 1;

    stage = 0;

    while (1)
    {

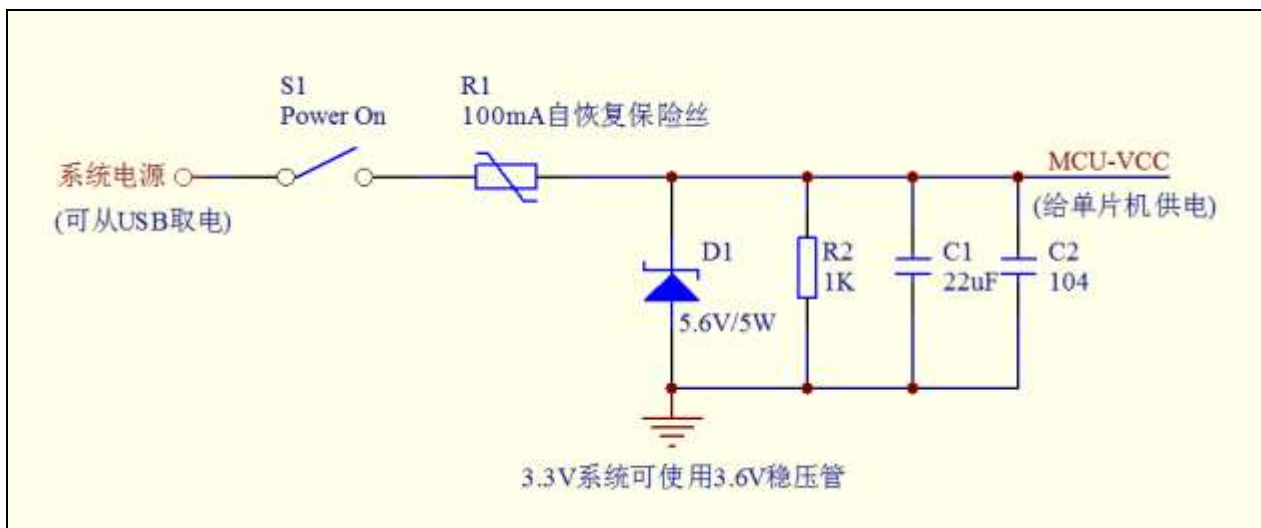
```

```
} ...  
}  
}
```

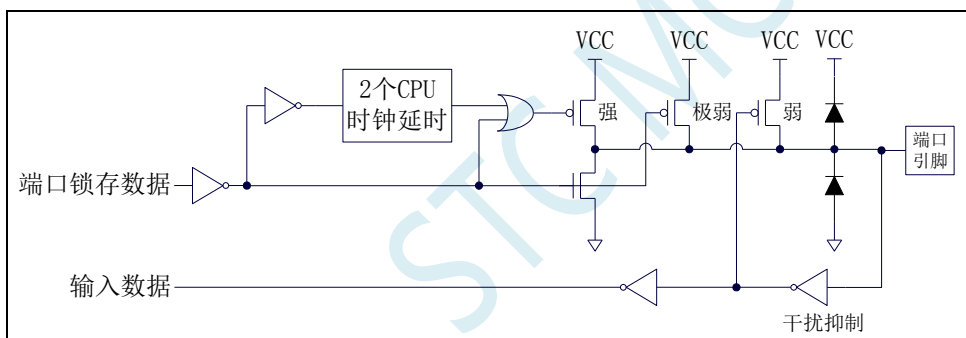
//用户代码

STC MCU

## 附录O 单机电源系统最简易自我保护电路



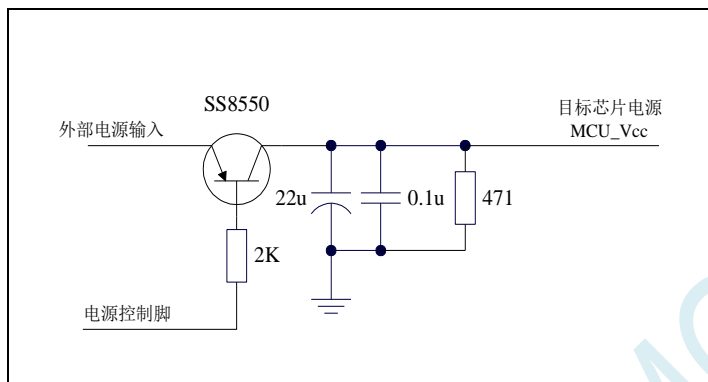
I/O 的内部结构图



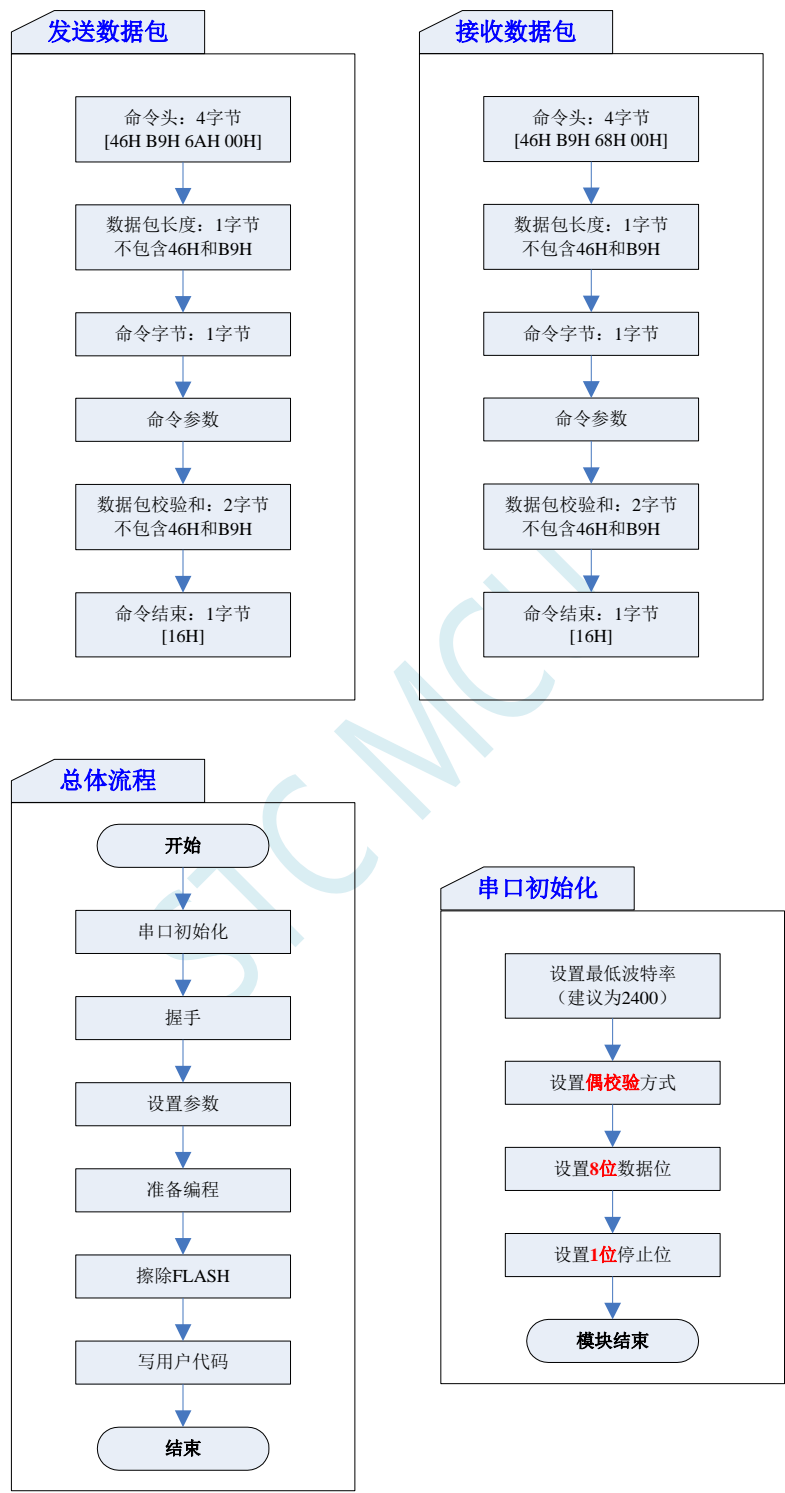
## 附录P 使用第三方 MCU 对 STC8H 系列单片机进行 ISP 下载范例程序

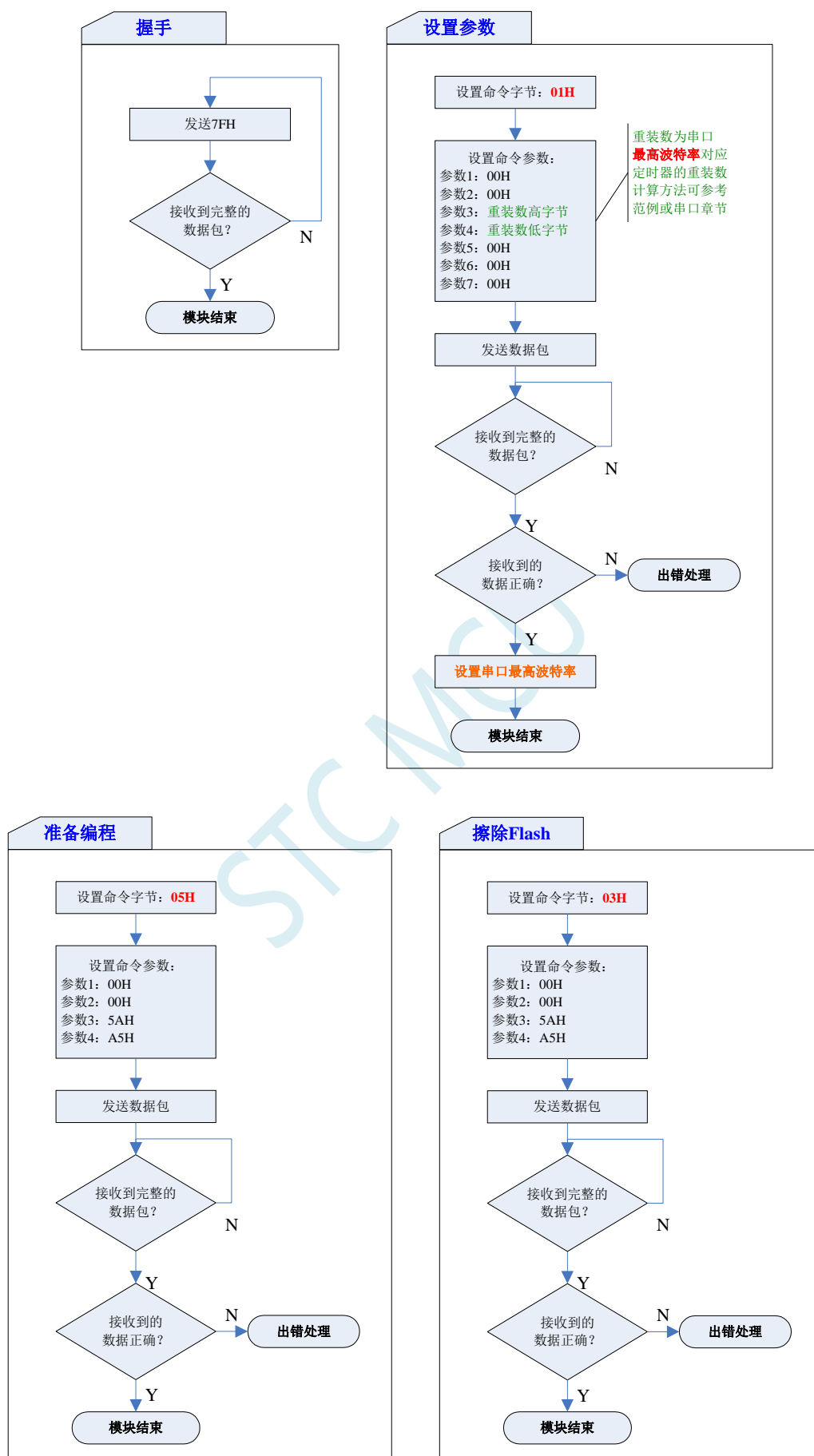
### P.1 电源控制参考电路

STC 芯片 ISP 下载需要对目标进行硬件复位才能进入 ISP 下载模式。当使用第三方 MCU 对 STC 芯片进行 ISP 下载时, 建议使用下面的电源控制电路来实现。

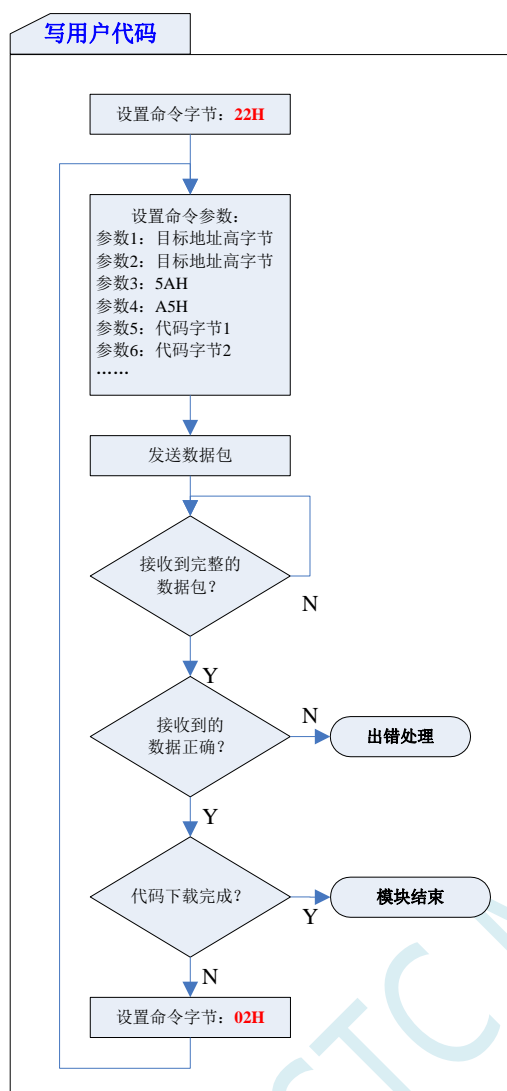


## P.2 通信协议流程图









## P.3 参考代码 (C 语言)

### C 语言代码

//注意: 使用本代码对 STC8H 系列的单片机进行下载时, 必须要执行了 Download 代码之后,  
//才能给目标芯片上电, 否则目标芯片将无法正确下载

```
#include "stc8h.h"
```

```
typedef bit          BOOL;
typedef unsigned char BYTE;
typedef unsigned short WORD;
```

//宏、常量定义

```
#define FALSE        0
#define TRUE         1
#define LOBYTE(w)    ((BYTE)(WORD)(w))
#define HIBYTE(w)    ((BYTE)((WORD)(w) >> 8))
```

```
#define MINBAUD      2400L
#define MAXBAUD      115200L
```

```
#define FOSC          11059200L           //主控芯片工作频率
#define BR(n)         (65536 - (FOSC/(n)+2) /4) //主控芯片串口波特率计算公式
//加2 操作是为了让 Keil 编译器
//自动实现四舍五入运算
#define TMS           (65536 - FOSC/1000) //主控芯片 1ms 定时初值

#define FUSER         24000000L          //STC8H 系列目标芯片工作频率
#define RL(n)         (65536 - (FUSER/(n)+2) /4) //STC8H 系列目标芯片串口波特率计算公式
//加2 操作是为了让 Keil 编译器
//自动实现四舍五入运算
```

//变量定义

```
BOOL f1ms;           //1ms 标志位
BOOL UartBusy;       //串口发送忙标志位
BOOL UartReceived;   //串口数据接收完成标志位
BYTE UartRecvStep;   //串口数据接收控制
BYTE TimeOut;        //串口通讯超时计数器
BYTE xdata TxBuffer[256]; //串口数据发送缓冲区
BYTE xdata RxBuffer[256]; //串口数据接收缓冲区
char code DEMO[256];  //演示代码数据
```

//函数声明

```
void Initial(void);
void DelayXms(WORD x);
BYTE UartSend(BYTE dat);
void CommInit(void);
void CommSend(BYTE size);
BOOL Download(BYTE *pd, long size);
```

//主函数入口

```
void main(void)
{
    P3M0 = 0x00;
    P3M1 = 0x00;
```

```
Initial();
if (Download(DEMO, 256))
{
    // 下载成功
    P3 = 0xff;
    DelayXms(500);
    P3 = 0x00;
    DelayXms(500);
    P3 = 0xff;
    DelayXms(500);
    P3 = 0x00;
    DelayXms(500);
    P3 = 0xff;
    DelayXms(500);
    P3 = 0x00;
    DelayXms(500);
    P3 = 0xff;
}
else
{
    // 下载失败
    P3 = 0xff;
    DelayXms(500);
    P3 = 0xf3;
    DelayXms(500);
    P3 = 0xff;
    DelayXms(500);
    P3 = 0xf3;
    DelayXms(500);
    P3 = 0xff;
    DelayXms(500);
    P3 = 0xf3;
    DelayXms(500);
    P3 = 0xff;
}

while (1);
}
```

//1ms 定时器中断服务程序

void tm0(void) interrupt 1

```
{
    static BYTE Counter100;

    f1ms = TRUE;
    if (Counter100-- == 0)
    {
        Counter100 = 100;
        if (TimeOut) TimeOut--;
    }
}
```

//串口中断服务程序

void uart(void) interrupt 4

```
{
    static WORD RecvSum;
    static BYTE RecvIndex;
    static BYTE RecvCount;
    BYTE dat;
```

```

    if (TI)
    {
        TI = 0;
        UartBusy = FALSE;
    }

    if (RI)
    {
        RI = 0;
        dat = SBUF;
        switch (UartRecvStep)
        {
            case 1:
                if (dat != 0xb9) goto L_CheckFirst;
                UartRecvStep++;
                break;
            case 2:
                if (dat != 0x68) goto L_CheckFirst;
                UartRecvStep++;
                break;
            case 3:
                if (dat != 0x00) goto L_CheckFirst;
                UartRecvStep++;
                break;
            case 4:
                RecvSum = 0x68 + dat;
                RecvCount = dat - 6;
                RecvIndex = 0;
                UartRecvStep++;
                break;
            case 5:
                RecvSum += dat;
                RxBuffer[RecvIndex++] = dat;
                if (RecvIndex == RecvCount) UartRecvStep++;
                break;
            case 6:
                if (dat != HIBYTE(RecvSum)) goto L_CheckFirst;
                UartRecvStep++;
                break;
            case 7:
                if (dat != LOBYTE(RecvSum)) goto L_CheckFirst;
                UartRecvStep++;
                break;
            case 8:
                if (dat != 0x16) goto L_CheckFirst;
                UartReceived = TRUE;
                UartRecvStep++;
                break;
        }
        L_CheckFirst:
        case 0:
        default:
            CommInit();
            UartRecvStep = (dat == 0x46 ? 1 : 0);
            break;
    }
}

```

//系统初始化

**void Initial(void)**

```
{
    UartBusy = FALSE;

    SCON = 0xd0;           //串口数据模式必须为8 位数据+1 位偶检验
    AUXR = 0xc0;
    TMOD = 0x00;
    TH0 = HIBYTE(TIMES);
    TL0 = LOBYTE(TIMES);
    TR0 = 1;
    TH1 = HIBYTE(BR(MINBAUD));
    TL1 = LOBYTE(BR(MINBAUD));
    TR1 = 1;
    ET0 = 1;
    ES = 1;
    EA = 1;
}
```

//Xms 延时程序

**void DelayXms(WORD x)**

```
{
    do
    {
        f1ms = FALSE;
        while (!f1ms);
    } while (x--);
}
```

//串口数据发送程序

**BYTE UartSend(BYTE dat)**

```
{
    while (UartBusy);

    UartBusy = TRUE;
    ACC = dat;
    TB8 = P;
    SBUF = ACC;

    return dat;
}
```

//串口通讯初始化

**void CommInit(void)**

```
{
    UartRecvStep = 0;
    TimeOut = 20;
    UartReceived = FALSE;
}
```

//发送串口通讯数据包

**void CommSend(BYTE size)**

```
{
    WORD sum;
    BYTE i;

    UartSend(0x46);
    UartSend(0xb9);
    UartSend(0x6a);
}
```

```
    UartSend(0x00);
    sum = size + 6 + 0x6a;
    UartSend(size + 6);
    for (i=0; i<size; i++)
    {
        sum += UartSend(TxBuffer[i]);
    }
    UartSend(HIBYTE(sum));
    UartSend(LOBYTE(sum));
    UartSend(0x16);
    while (UartBusy);

    CommInit();
}
```

//对STC15H 系列的芯片进行ISP 下载程序

**BOOL** Download(**BYTE** \*pdat, long size)

```
{
    BYTE arg;
    BYTE offset;
    BYTE cnt;
    WORD addr;

    //握手
    CommInit();
    while (1)
    {
        if (UartRecvStep == 0)
        {
            UartSend(0x7f);
            DelayXms(10);
        }
        if (UartReceived)
        {
            arg = RxBuffer[4];
            if (RxBuffer[0] == 0x50) break;
            return FALSE;
        }
    }
}
```

//设置参数(设置从芯片使用最高的波特率以及等待时间等参数)

```
TxBuffer[0] = 0x01;
TxBuffer[1] = arg;
TxBuffer[2] = 0x40;
TxBuffer[3] = HIBYTE(RL(MAXBAUD));
TxBuffer[4] = LOBYTE(RL(MAXBAUD));
TxBuffer[5] = 0x00;
TxBuffer[6] = 0x00;
TxBuffer[7] = 0x97;
CommSend(8);
while (1)
{
    if (TimeOut == 0) return FALSE;
    if (UartReceived)
    {
        if (RxBuffer[0] == 0x01) break;
        return FALSE;
    }
}
```

*//准备*

```
TH1 = HIBYTE(BR(MAXBAUD));
TL1 = LOBYTE(BR(MAXBAUD));
DelayXms(10);
TxBuffer[0] = 0x05;
TxBuffer[1] = 0x00;
TxBuffer[2] = 0x00;
TxBuffer[3] = 0x5a;
TxBuffer[4] = 0xa5;
CommSend(5);
while (1)
{
    if (TimeOut == 0) return FALSE;
    if (UartReceived)
    {
        if (RxBuffer[0] == 0x05) break;
        return FALSE;
    }
}
```

*//擦除*

```
DelayXms(10);
TxBuffer[0] = 0x03;
TxBuffer[1] = 0x00;
TxBuffer[2] = 0x00;
TxBuffer[3] = 0x5a;
TxBuffer[4] = 0xa5;
CommSend(5);
TimeOut = 100;
while (1)
{
    if (TimeOut == 0) return FALSE;
    if (UartReceived)
    {
        if (RxBuffer[0] == 0x03) break;
        return FALSE;
    }
}
```

*//写用户代码*

```
DelayXms(10);
addr = 0;
TxBuffer[0] = 0x22;
TxBuffer[3] = 0x5a;
TxBuffer[4] = 0xa5;
offset = 5;
while (addr < size)
{
    TxBuffer[1] = HIBYTE(addr);
    TxBuffer[2] = LOBYTE(addr);
    cnt = 0;
    while (addr < size)
    {
        TxBuffer[cnt+offset] = pdat[addr];
        addr++;
        cnt++;
        if (cnt >= 128) break;
    }
}
```

```

    CommSend(cnt + offset);
    while (1)
    {
        if (TimeOut == 0) return FALSE;
        if (UartReceived)
        {
            if ((RxBuffer[0] == 0x02) && (RxBuffer[1] == 'T')) break;
            return FALSE;
        }
    }
    TxBuffer[0] = 0x02;
}

//// 写硬件选项
//// 如果不需要修改硬件选项,此步骤可直接跳过,此时所有的硬件选项
//// 都维持不变,MCU 的频率为上一次所调节频率
//// 若写硬件选项,MCU 的内部 IRC 频率将被固定写为 24M,其他选项恢复为出厂设置
//// 建议:第一次使用 STC-ISP 下载软件将从芯片的硬件选项设置好
//// 以后再使用主芯片对从芯片下载程序时不写硬件选项
//DelayXms(10);
//for (cnt=0; cnt<128; cnt++)
//{
//    TxBuffer[cnt] = 0xff;
//}
//TxBuffer[0] = 0x04;
//TxBuffer[1] = 0x00;
//TxBuffer[2] = 0x00;
//TxBuffer[3] = 0x5a;
//TxBuffer[4] = 0xa5;
//TxBuffer[33] = arg;
//TxBuffer[34] = 0x00;
//TxBuffer[35] = 0x01;
//TxBuffer[41] = 0xbf;
//TxBuffer[42] = 0xbd;
////TxBuffer[42] = 0xad;
//TxBuffer[43] = 0xf7;
//TxBuffer[44] = 0xff;
//CommSend(45);
//while (1)
//{
//    if (TimeOut == 0) return FALSE;
//    if (UartReceived)
//    {
//        if ((RxBuffer[0] == 0x04) && (RxBuffer[1] == 'T')) break;
//        return FALSE;
//    }
//}

// 下载完成
return TRUE;
}

char code DEMO[256] =
{
    0x80,0x00,0x75,0xB2,0xFF,0x75,0xB1,0x00,0x05,0xB0,0x11,0x0E,0x80,0xFA,0xD8,0xFE,
    0xD9,0xFC,0x22,
};

```



STC MCU

## 附录Q 使用第三方应用程序调用 STC 发布项目程序对单片机进行 ISP 下载

使用 STC 的 ISP 下载软件生成的发布项目程序为可执行的 EXE 格式文件, 用户可直接双击发布的项目程序运行进行 ISP 下载, 也可在第三方的应用程序中调用发布项目程序进行 ISP 下载。下面介绍两种调用的方法。

### 简单调用

在第三方应用程序中只是简单创建发布项目程序的进程, 其他的所有下载操作均在发布项目程序中进行, 第三方应用程序此时只需要等待发布项目程序操作完成后, 清理现场即可。

### VC 代码

```
BOOL IspProcess()
{
    //定义相关变量
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    CString path;

    //发布项目程序的完整路径
    path = _T("D:\\Work\\Upgrade.exe");

    //变量初始化
    memset(&si, 0, sizeof(STARTUPINFO));
    memset(&pi, 0, sizeof(PROCESS_INFORMATION));

    //设置启动变量
    si.cb = sizeof(STARTUPINFO);
    GetStartupInfo(&si);
    si.wShowWindow = SW_SHOWNORMAL;
    si.dwFlags = STARTF_USESHOWWINDOW;

    //创建发布项目程序进程
    if (CreateProcess(NULL, (LPTSTR)(LPCTSTR)path, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
    {
        //等待发布项目程序操作完成
        //由于此处会阻塞主进程, 所以建议新建工作进程, 在工作进程中进行等待
        WaitForSingleObject(pi.hProcess, INFINITE);

        //清理工作
        CloseHandle(pi.hThread);
        CloseHandle(pi.hProcess);

        return TRUE;
    }
    else
    {
        AfxMessageBox(_T("创建进程失败 !"));
    }
}
```

```
        return FALSE;
    }
}
```

---

## 高级调用

在第三方应用程序创建发布项目程序的进程，并在第三方应用程序中进行包括选择串口、开始 ISP 编程、停振 ISP 编程以及关闭发布项目程序等的全部 ISP 下载操作，而不需要在发布项目程序中进行界面互动。

### VC 代码

---

*//定义回调函数参数的数据结构*

*struct CALLBACK\_PARAM*

*{*

*DWORD dwProcessId;*

*//主进程ID*

*HWND hMainWnd;*

*//主窗口句柄*

*};*

*//枚举窗口的回调函数，用于获取主窗口句柄*

*BOOL CALLBACK EnumWindowCallBack(HWND hWnd, LPARAM lParam)*

*{*

*CALLBACK\_PARAM \*pcp = (CALLBACK\_PARAM \*)lParam;*

*DWORD id;*

*GetWindowThreadProcessId(hWnd, &id);*

*if ((pcp->dwProcessId == id) && (GetParent(hWnd) == NULL))*

*{*

*pcp->hMainWnd = hWnd;*

*return FALSE;*

*}*

*return TRUE;*

*}*

*BOOL IspProcess()*

*{*

*//定义相关变量*

*STARTUPINFO si;*

*PROCESS\_INFORMATION pi;*

*CALLBACK\_PARAM cp;*

*CString path;*

*//发布项目程序中部分控件的ID*

*const UINT ID\_PROGRAM = 1013;*

*const UINT ID\_STOP = 1012;*

*const UINT ID\_COMPORT = 1001;*

*const UINT ID\_PROGRESS = 1000;*

*//发布项目程序的完整路径*

*path = \_T("D:\\Work\\Upgrade.exe");*

*//变量初始化*

*memset(&si, 0, sizeof(STARTUPINFO));*

*memset(&pi, 0, sizeof(PROCESS\_INFORMATION));*

*memset(&cp, 0, sizeof(CALLBACK\_PARAM));*

```
//设置启动变量
si.cb = sizeof(STARTUPINFO);
GetStartupInfo(&si);
si.wShowWindow = SW_SHOWNORMAL; //此处若设置为SW_HIDE,就不会显示发布项目程序
//的操作界面,所有的ISP 操作都可在后台进行

si.dwFlags = STARTF_USESHOWWINDOW;

//创建发布项目程序进程
if (CreateProcess(NULL, (LPTSTR)(LPCTSTR)path, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
{
    //等待发布项目程序进程初始化完成
    WaitForInputIdle(pi.hProcess, 5000);

    //获取发布项目程序的主窗口句柄
    cp.dwProcessId = pi.dwProcessId;
    cp.hMainWnd = NULL;
    EnumWindows(EnumWindowCallback, (LPARAM)&cp);

    if (cp.hMainWnd != NULL)
    {
        HWND hProgram;
        HWND hStop;
        HWND hPort;

        //获取发布项目程序主窗口中部分控件句柄
        hProgram = ::GetDlgItem(cp.hMainWnd, ID_PROGRAM);
        hStop = ::GetDlgItem(cp.hMainWnd, ID_STOP);
        hPort = ::GetDlgItem(cp.hMainWnd, ID_COMPORT);

        //设置发布项目程序中的串口号, 第3 个参数为0:COM1, 1:COM2, 2:COM3, ...
        ::SendMessage(hPort, CB_SETCURSEL, 0, 0);

        //触发编程按钮开始 ISP 编程
        ::SendMessage(hProgram, BM_CLICK, 0, 0);

        //等待编程完成
        //由于此处会阻塞主进程, 所以建议新建工作进程, 在工作进程中进行等待
        while (!::IsWindowEnabled(hProgram));

        //编程完成后关闭发布项目程序
        ::SendMessage(cp.hMainWnd, WM_CLOSE, 0, 0);
    }

    //等待进程结束
    WaitForSingleObject(pi.hProcess, INFINITE);

    //清理工作
    CloseHandle(pi.hThread);
    CloseHandle(pi.hProcess);

    return TRUE;
}
else
{
    AfxMessageBox(_T("创建进程失败 !"));

    return FALSE;
}
```

/

---

STC MCU

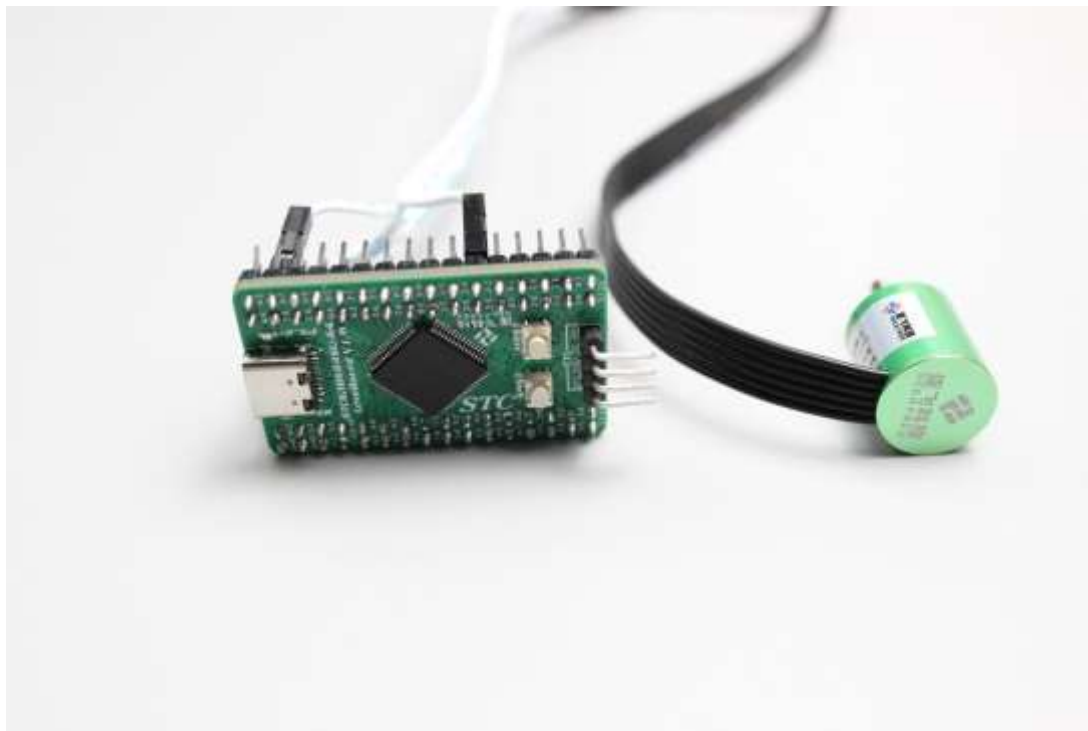
## 附录R STC8H 系列正交解码示例（成都逐飞科技友情提供）

受 STC 委托，本文意在分享使用 STC8H 系列单片机的增强型 PWM 模块实现正交解码功能，以进一步实现对正交编码信号输出的编码器进行双向速度测量。

**硬件平台：**逐飞 STC8H8K-64 脚核心板+逐飞 1024 线 mini 正交编码器

**编译环境：**keil V9.60

**上位机：**串口助手



从逐飞科技之前的开源库中我们可以了解到，开源库里没有正交解码例程，主要原因是因为 STC8H 只有两个 PWM 模块，假如我们推荐使用正交编码编码器，就意味着一个编码器就需要占用一个 PWM 模块，然而今年节能组要求制作平衡小车，就意味着有两个电机，这样就需要两个编码器，那么单片机的两个 PWM 模块就会都被占用，然而小车的电机控制也需要 PWM 功能，所以并没有推荐大家用 PWM 模块来实现正交解码，而是推荐大家使用带方向输出的编码器，这样 PWM 模块就可以留给电机使用。

当然也可以有另一种设想，使用一个 PWM 模块去实现对一个正交编码的编码器进行测速，另一个电机使用带方向信号的编码器，用普通定时器去捕获脉冲，这种方案是可行的，但没必要这么麻烦。

还有一点需要注意，使用 PWM 模块计数和使用定时器模块捕获脉冲计数的方式是不一样的。PWM 模块去捕获编码器数据是通过边沿计数，也就是说这个模块的是在发生上升沿或者下降沿的时候都会计数，而定时器捕获脉冲，是获取高低电平翻转的次数。这里我们通过实验就会发现，用同一个的正交编码编码器转动 360°，PWM 模块采集编码器数据是定时器捕获脉冲数据的两倍，但这个数据并不是精度

变高，只是单片机的计数方式导致了结果翻倍。

下面是使用 STC8H8K64U 采集正交编码信号输出编码器的示例程序：

## C 语言代码

```
#include "headfile.h"

int16 encoder_data;

//-----
//@brief      PWMA 模块正交解码初始化
//@param      void
//@return      void
//@since      v1.0
//Sample usage: PWMA_encoder_init();
//@note
//-----
void PWMA_encoder_init(void)
{
    P_SW2 /= 1<<7;
    PWMA_ARR = 0xFFFF;

    PWMA_CCMR1 /= 1<<0;
    PWMA_CCMR2 /= 1<<0;
    PWMA_SMCR /= 1<<0;

    PWMA_CR1 /= 1<<0;
    PWMA_PS /= 1<<2;
}

//-----
// @brief      PWMA 模块获取正交解码数值
// @param      void
// @return      void
// @since      v1.0
// Sample usage: encoder_data = PWMA_get_encoder();
// @note
//-----
int16 PWMA_get_encoder(void)
{
    int16 res;

    res = PWMA_CNTR;
    PWMA_CNTR = 0;
    return res;
}

//-----
// @brief      定时器0 5ms 中断服务函数
// @param      void
// @return      void
// @since      v1.0
// Sample usage:
// @note
//-----
void TM0_Isr() interrupt 1
{
    encoder_data = PWMA_get_encoder();
}
```

```
void main()
{
    P_SW2 |= 0x80; //使能访问 XFR

    DisableGlobalIRQ();           //关闭总中断
    board_init();                 //初始化内部寄存器, 勿删除此句代码。
    pit_timer_ms(TIM_0, 5);       //初始化定时器, 5ms 执行一次
    PWMA_encoder_init();          //PWMA 模块初始化为正交解码功能
    EnableGlobalIRQ();            //开启总中断
    while(1)
    {
        delay_ms(100);           //每 100ms 输出一次打印信息
        printf("encoder_data = %d \r\n", encoder_data); //串口 1 打印编码器数据
    }
}
```

演示视频链接: <https://www.bilibili.com/video/BV1zT4y177Ht>

视频说明: 我们将编写好的例程编译, 然后下载到单片机, 打开串口助手接收单片机的打印数据, 旋转编码器观察数据变化, 我们发现当编码器不旋转时输出数据为 0, 当编码器朝不同方向旋转时可以输出正负两种数值, 旋转越快, 数值的绝对值越大, 正负用来表示两个旋转方向, 其中哪个方向为正, 哪个方向为负是可以自己定义的。同时我们从程序示例中也看到打印数据是 100ms 一次, 而数据采集是 5ms 一次, 所以打印出来的数据相当于是间断的, 同时因为编码器是 1024 线的高精度, 所以观察到数据变化比较大, 但如果是使用电机空载固定 PWM 占空比驱动, 可以看到编码器的数据输出是十分稳定的。

串口助手接收数据截图:

下图是正交编码的编码器顺时针旋转且角速度逐渐增大的数据



### SSCOM V5.13.1 串口/网络数据调试器

[通讯端口](#)
[串口设置](#)
[显示](#)
[发送](#)
[多字符](#)

```

[12:28:09.163]收←◆encoder_data = 0
[12:28:09.275]收←◆encoder_data = 0
[12:28:09.388]收←◆encoder_data = 0
[12:28:09.501]收←◆encoder_data = 0
[12:28:09.613]收←◆encoder_data = 0
[12:28:09.724]收←◆encoder_data = 6
[12:28:09.838]收←◆encoder_data = 15
[12:28:09.950]收←◆encoder_data = 18
[12:28:10.062]收←◆encoder_data = 44
[12:28:10.175]收←◆encoder_data = 51
[12:28:10.288]收←◆encoder_data = 67
[12:28:10.400]收←◆encoder_data = 67
[12:28:10.513]收←◆encoder_data = 78
[12:28:10.625]收←◆encoder_data = 68
[12:28:10.737]收←◆encoder_data = 63
[12:28:10.850]收←◆encoder_data = 87
[12:28:10.962]收←◆encoder_data = 112

```

下图是正交编码的编码器逆时针旋转且角速度逐渐增大时的数据:

### SSCOM V5.13.1 串口/网络数据调试器

[通讯端口](#)
[串口设置](#)
[显示](#)
[发送](#)
[多字符](#)

```

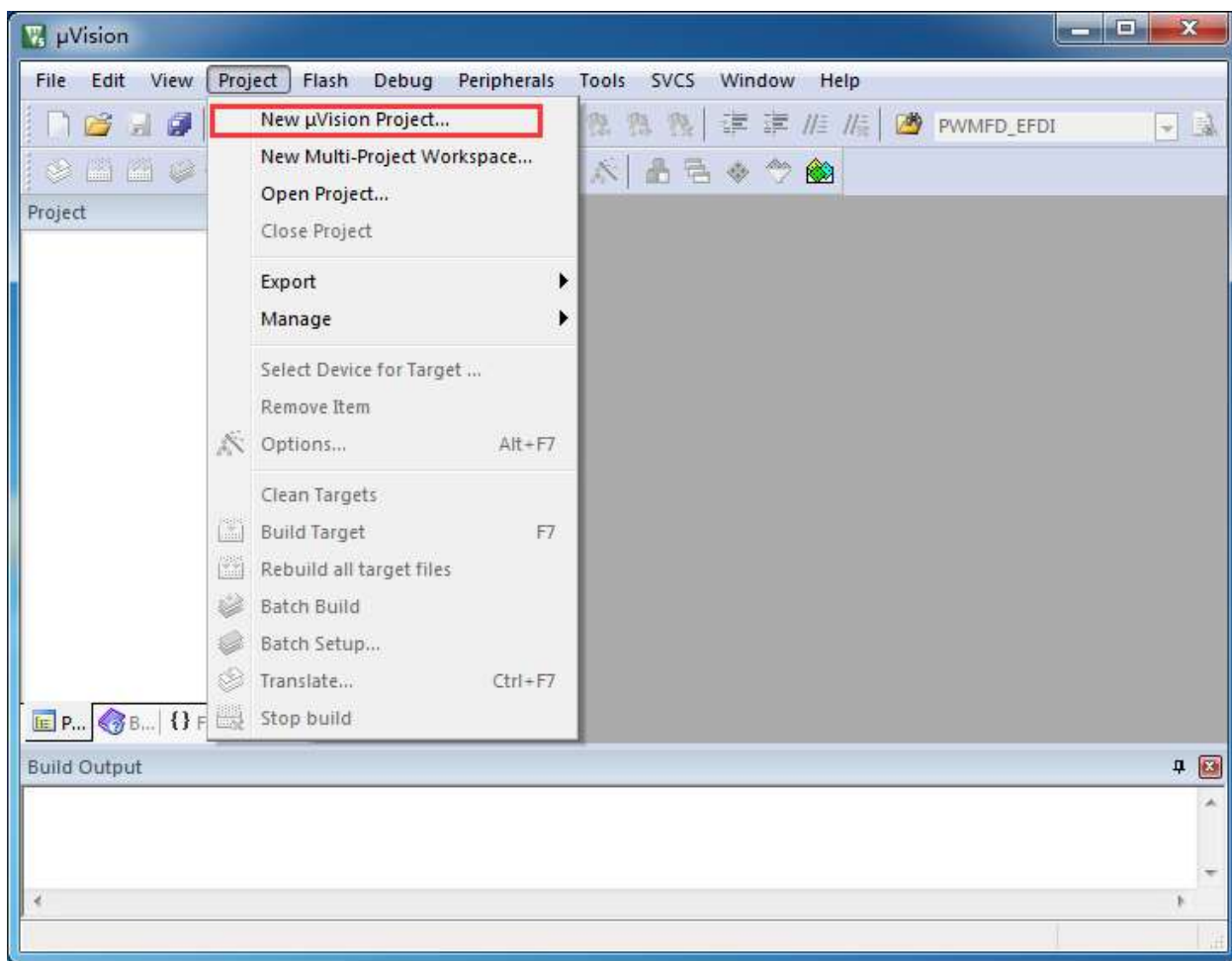
[12:28:23.330]收←◆encoder_data = 0
[12:28:23.443]收←◆encoder_data = 0
[12:28:23.554]收←◆encoder_data = 0
[12:28:23.667]收←◆encoder_data = 0
[12:28:23.780]收←◆encoder_data = 0
[12:28:23.892]收←◆encoder_data = -74
[12:28:24.005]收←◆encoder_data = -120
[12:28:24.117]收←◆encoder_data = -152
[12:28:24.229]收←◆encoder_data = -165
[12:28:24.342]收←◆encoder_data = -210

```

## 附录S 在 Keil 中建立多文件项目的方法

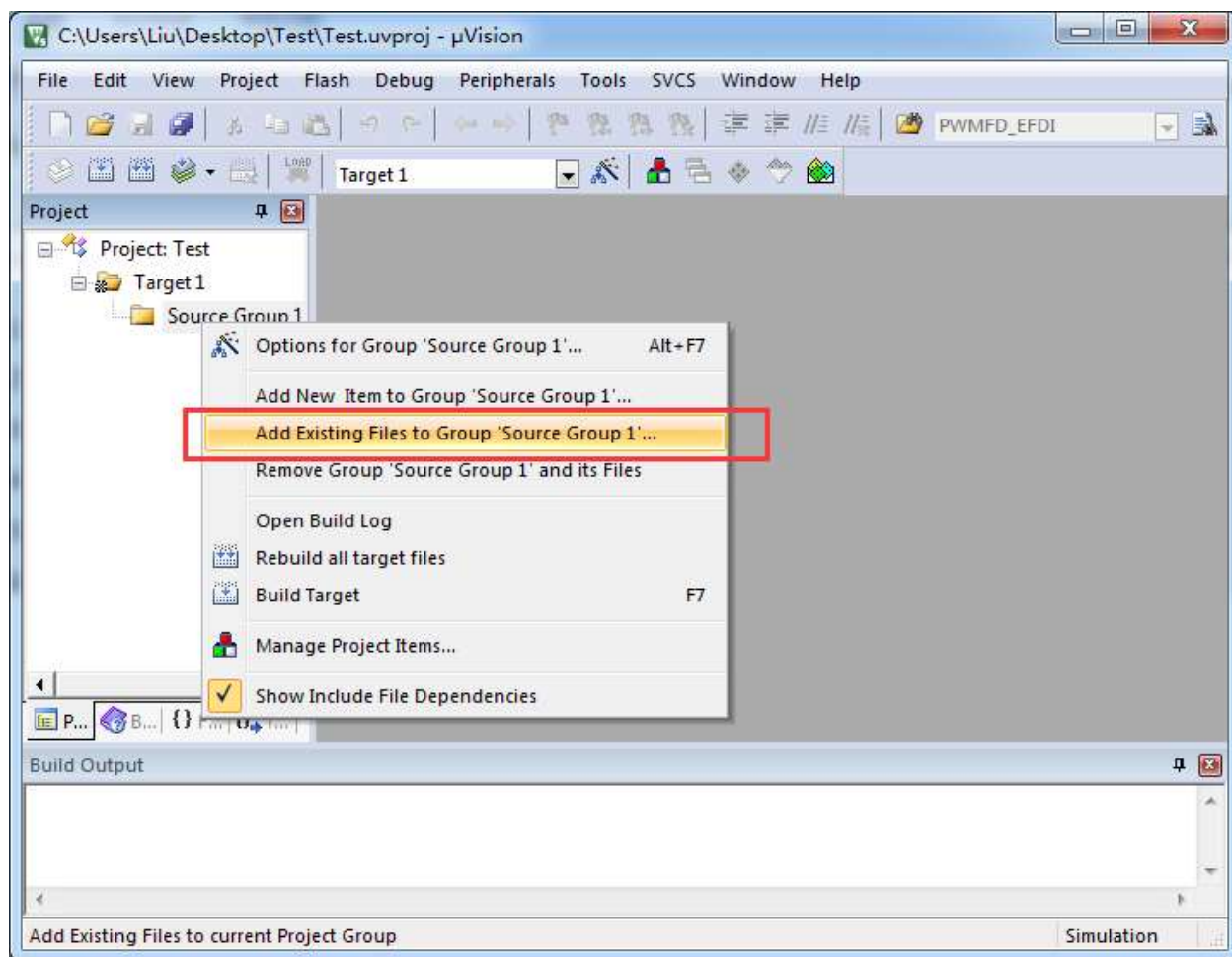
在 Keil 中，一般比较小的项目都只有一个源文件，但对于一些稍微复杂的项目往往需要多个源文件建立多文件项目的方法如下：

1、首先打开 Keil，在菜单“Project”中选择“New uVision Project ...”

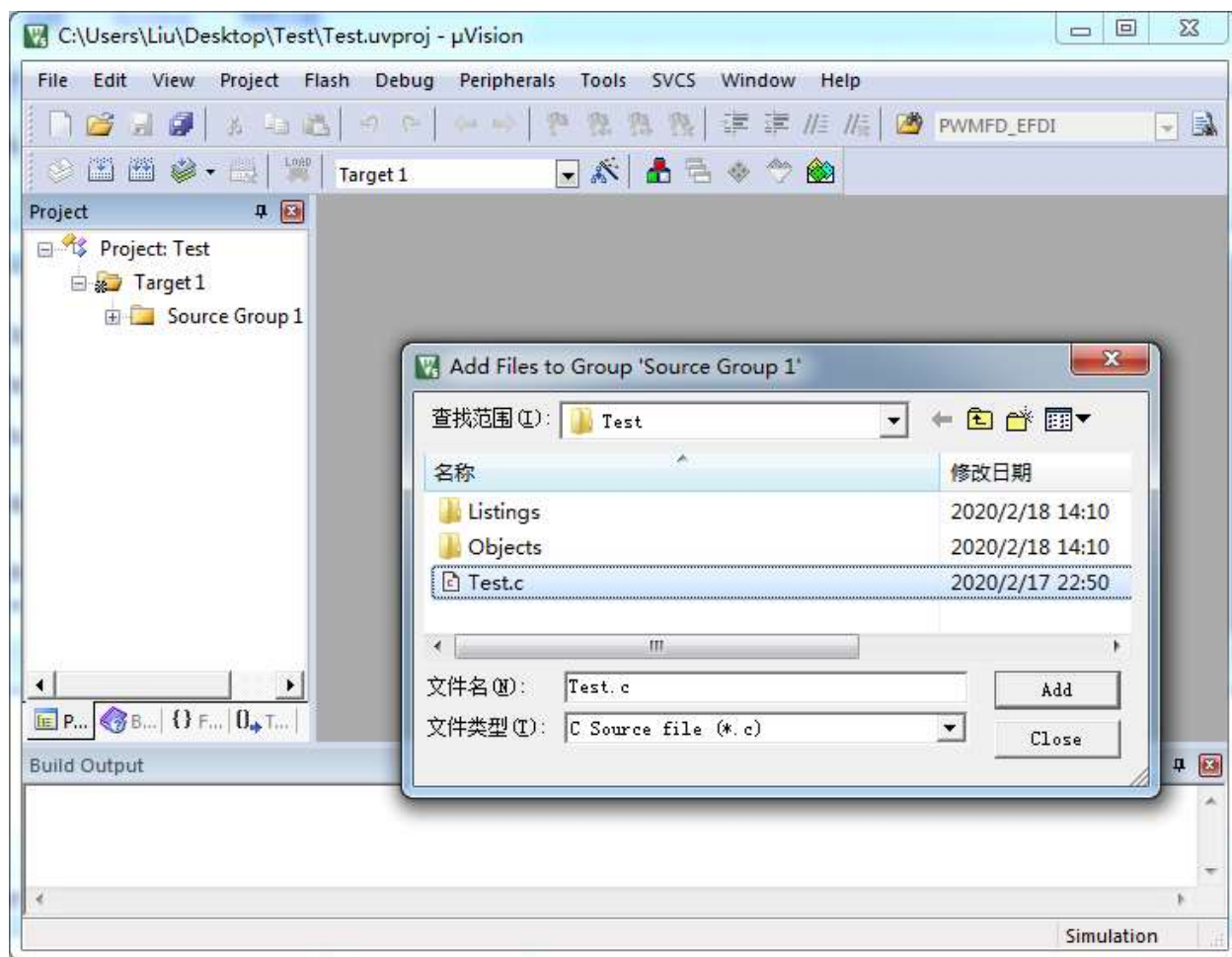


即可完成一个空项目的建立

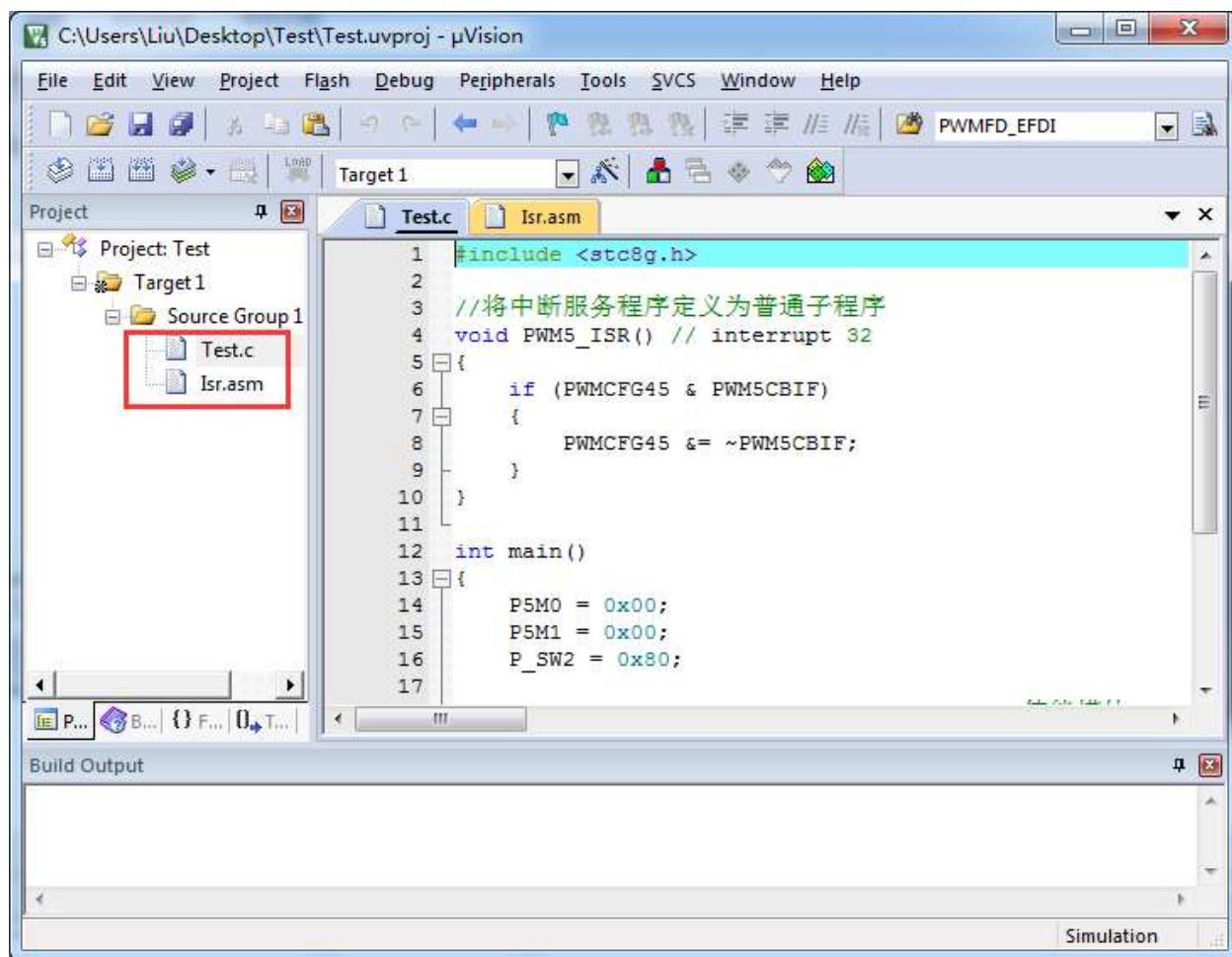
2、在空项目的项目树中，鼠标右键单击“Source Group 1”，并选择右键菜单中的“Add Existing Files to Group "Source Group 1" ...”



3、在弹出的文件对话框中，多次添加源文件



如下图所示即可完成多文件项目的建立

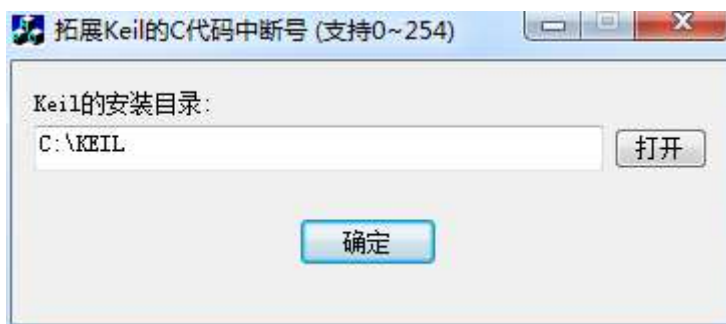


## 附录T 关于中断号大于 31 在 Keil 中编译出错的 处理

注: 目前 Keil 各个版本的 C51 和 C251 编译器均只支持 32 个中断号 (0~31), 经我公司与 Keil 公司多方协商和探讨, Keil 公司答应会在后续某个版本增加我公司对中断号超过 32 个的需求。但对于目前现有的 Keil 版本, 只能使用本章节的方法进行临时解决。

### T.1 使用网上流行的中断号拓展工具

热心网友有提供一个简单的拓展工具, 可将中断号拓展到 254。工具界面如下:



点击“打开”按钮, 定位到 Keil 的安装目录后, 点击“确定”即可。

由于 Keil 的版本在不断更新, 而早期版本过多, 有无法收集齐, 这里列举一下已测试通过的 C51.EXE 版本和 C251.EXE 版本

已测试通过的 C51.EXE 版本:

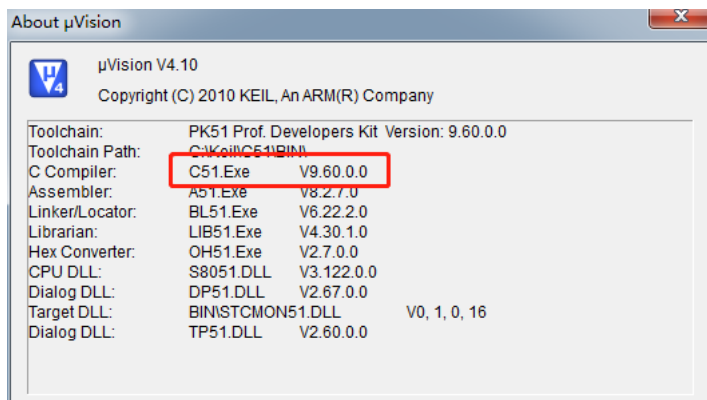
V6.12.0.1  
V8.8.0.1  
V9.0.0.1  
V9.1.0.1  
V9.53.0.0  
V9.54.0.0  
V9.57.0.0  
V9.59.0.0  
V9.60.0.0

已测试通过的 C251.EXE 版本:

V5.57.0.0  
V5.60.0.0

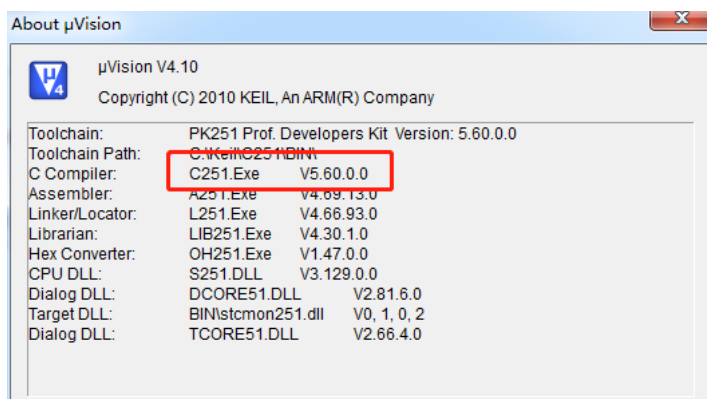
### 查看 C51.EXE 版本的方法:

在 keil 中打开一个基于 STC8 系列或者 STC15 系列单片机的项目, 在 Keil 软件菜单项“Help”中打开“About uVision...”



### 查看 C251.EXE 版本的方法:

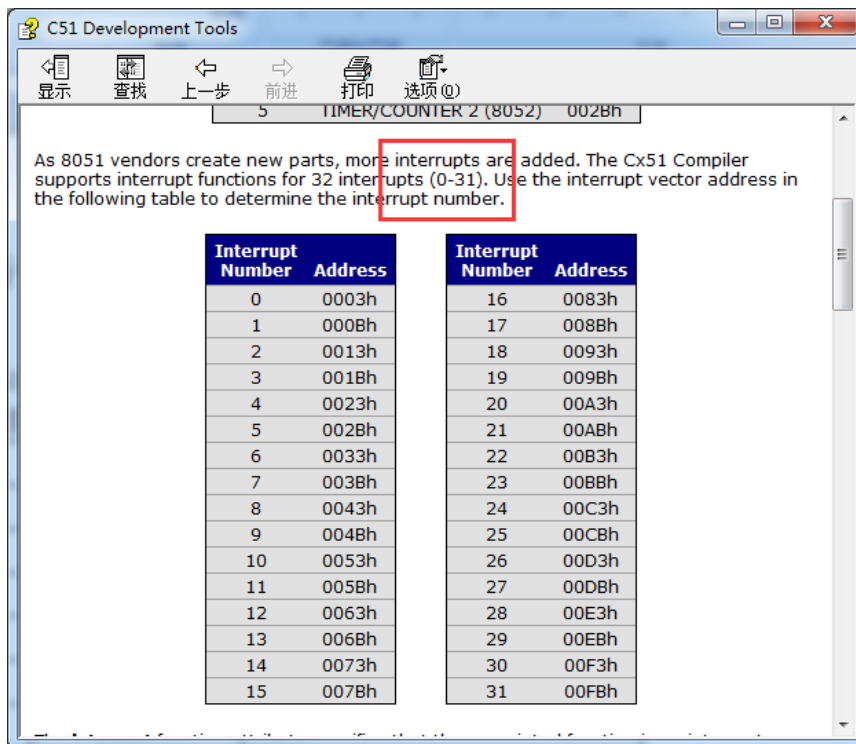
在 keil 中打开一个基于 STC32G 系列单片机的项目, 在 Keil 软件菜单项“Help”中打开“About uVision...”





## T.2 使用保留中断号进行中转

在 Keil 的 C51 编译环境下, 中断号只支持 0~31, 即中断向量必须小于 0100H。



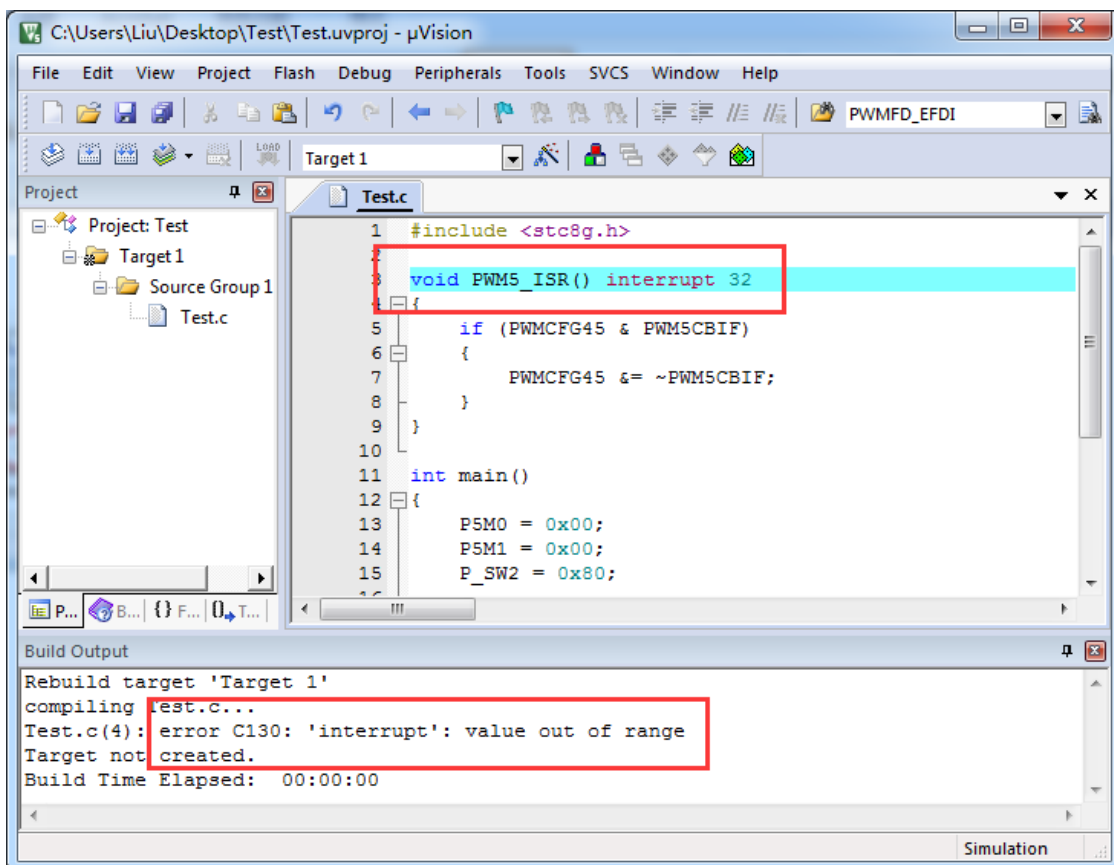
下表是 STC 目前所有系列的中断列表:

| 中断号 | 中断向量   | 中断类型   |
|-----|--------|--------|
| 0   | 0003 H | INT0   |
| 1   | 000B H | 定时器 0  |
| 2   | 0013 H | INT1   |
| 3   | 001B H | 定时器 1  |
| 4   | 0023 H | 串口 1   |
| 5   | 002B H | ADC    |
| 6   | 0033 H | LVD    |
| 7   | 003B H | PCA    |
| 8   | 0043 H | 串口 2   |
| 9   | 004B H | SPI    |
| 10  | 0053 H | INT2   |
| 11  | 005B H | INT3   |
| 12  | 0063 H | 定时器 2  |
| 13  | 006B H |        |
| 14  | 0073 H | 系统内部中断 |
| 15  | 007B H | 系统内部中断 |
| 16  | 0083 H | INT4   |
| 17  | 008B H | 串口 3   |
| 18  | 0093 H | 串口 4   |
| 19  | 009B H | 定时器 3  |
| 20  | 00A3 H | 定时器 4  |



|           |               |                  |
|-----------|---------------|------------------|
| 21        | 00AB H        | 比较器              |
| 22        | 00B3 H        | 波形发生器 0          |
| 23        | 00BB H        | 波形发生器异常 0        |
| 24        | 00C3 H        | I2C              |
| 25        | 00CB H        | USB              |
| 26        | 00D3 H        | PWMA             |
| 27        | 00DB H        | PWMB             |
| 28        | 00E3 H        | 波形发生器 1          |
| 29        | 00EB H        | 波形发生器 2          |
| 30        | 00F3 H        | 波形发生器 3          |
| 31        | 00FB H        | 波形发生器 4          |
| <b>32</b> | <b>0103 H</b> | <b>波形发生器 5</b>   |
| <b>33</b> | <b>010B H</b> | <b>波形发生器异常 2</b> |
| <b>34</b> | <b>0113 H</b> | <b>波形发生器异常 4</b> |
| <b>35</b> | <b>011B H</b> | <b>触摸按键</b>      |
| <b>36</b> | <b>0123 H</b> | <b>RTC</b>       |
| <b>37</b> | <b>012B H</b> | <b>P0 口中断</b>    |
| <b>38</b> | <b>0133 H</b> | <b>P1 口中断</b>    |
| <b>39</b> | <b>013B H</b> | <b>P2 口中断</b>    |
| <b>40</b> | <b>0143 H</b> | <b>P3 口中断</b>    |
| <b>41</b> | <b>014B H</b> | <b>P4 口中断</b>    |
| <b>42</b> | <b>0153 H</b> | <b>P5 口中断</b>    |
| <b>43</b> | <b>015B H</b> | <b>P6 口中断</b>    |
| <b>44</b> | <b>0163 H</b> | <b>P7 口中断</b>    |
| <b>45</b> | <b>016B H</b> | <b>P8 口中断</b>    |
| <b>46</b> | <b>0173 H</b> | <b>P9 口中断</b>    |

不难发现，从波形发生器 5 中断开始，后面所有的中断服务程序，在 keil 中均会编译出错，如下图所示：

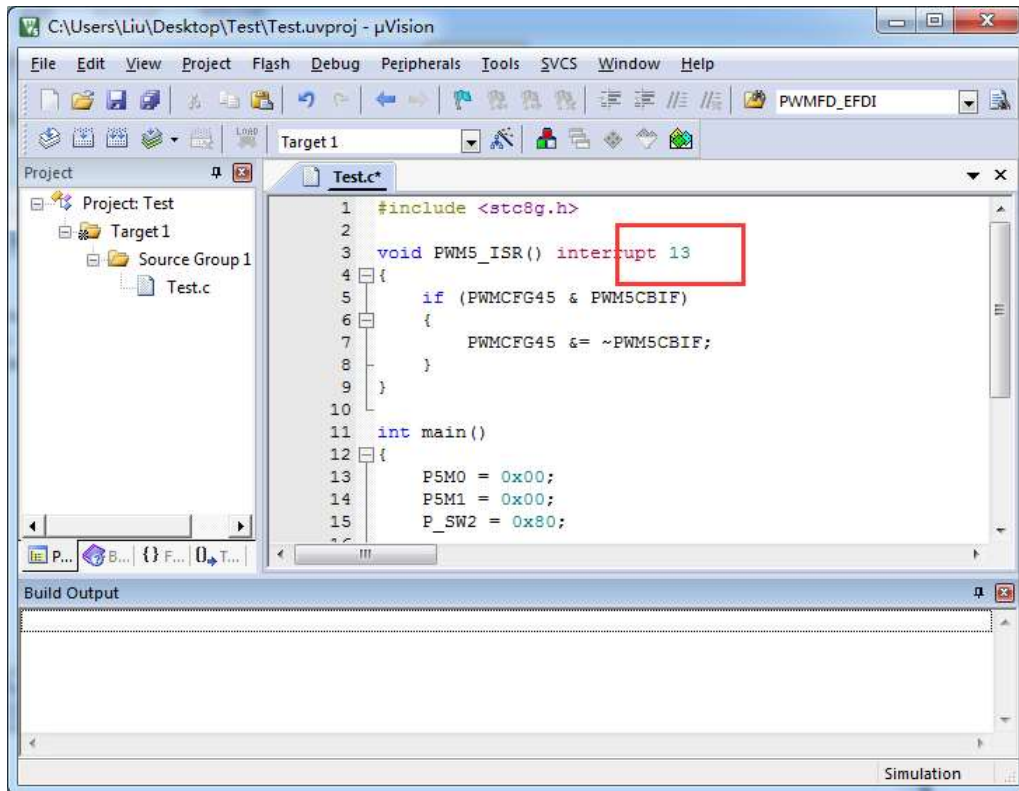


处理这种错误有如下三种方法: (均需要借助于汇编代码, 优先推荐使用方法 1)

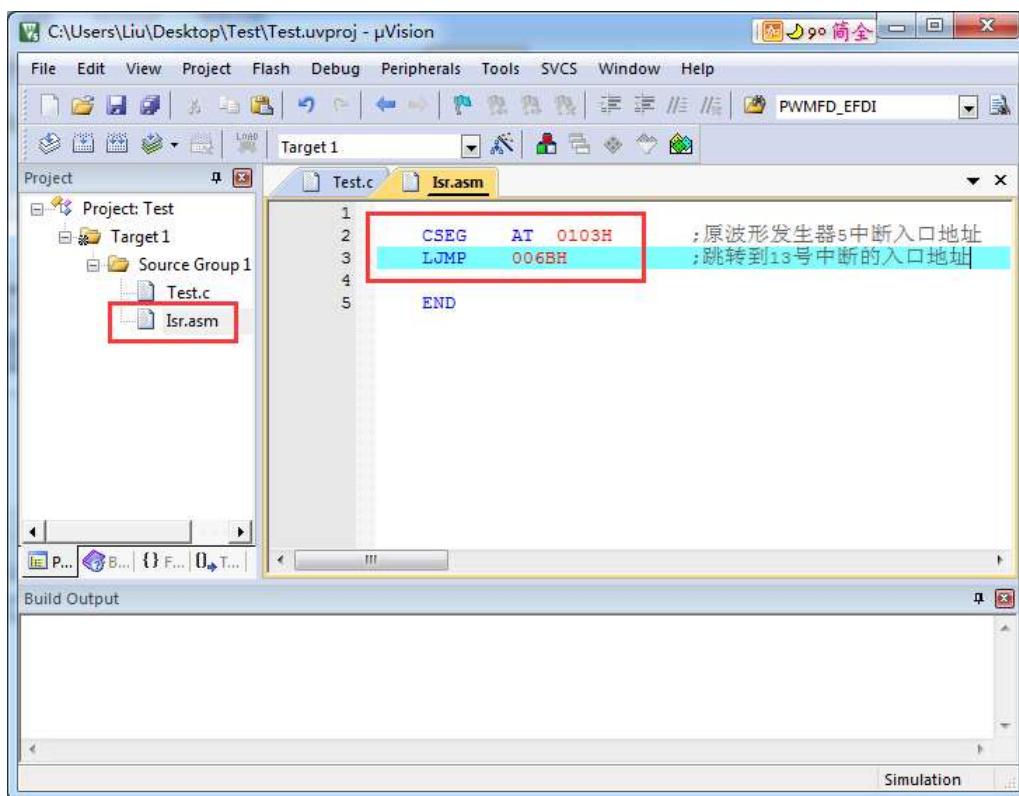
## 方法 1: 借用 13 号中断向量

0~31 号中断中, 第 13 号是保留中断号, 我们可以借用此中断号  
操作步骤如下:

1、将我们报错的中断号改为“13”, 如下图:

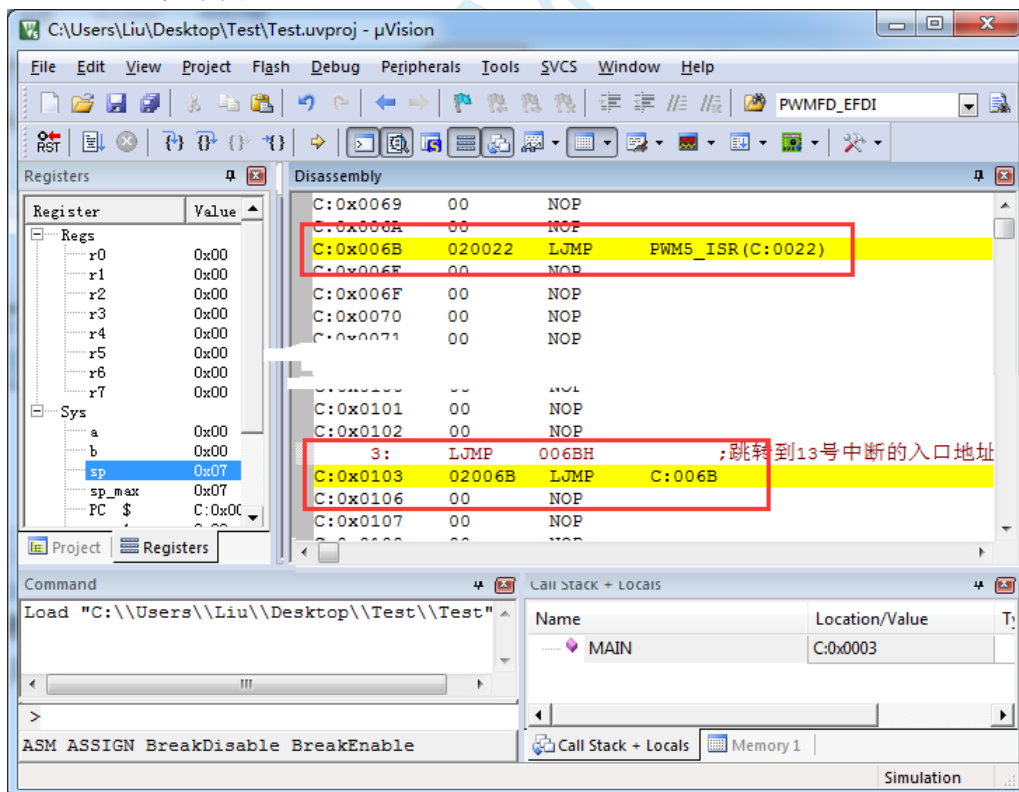


2、新建一个汇编语言文件, 比如“isr.asm”, 加入到项目, 并在地址“0103H”的地方添加一条“LJMP 006BH”, 如下图:

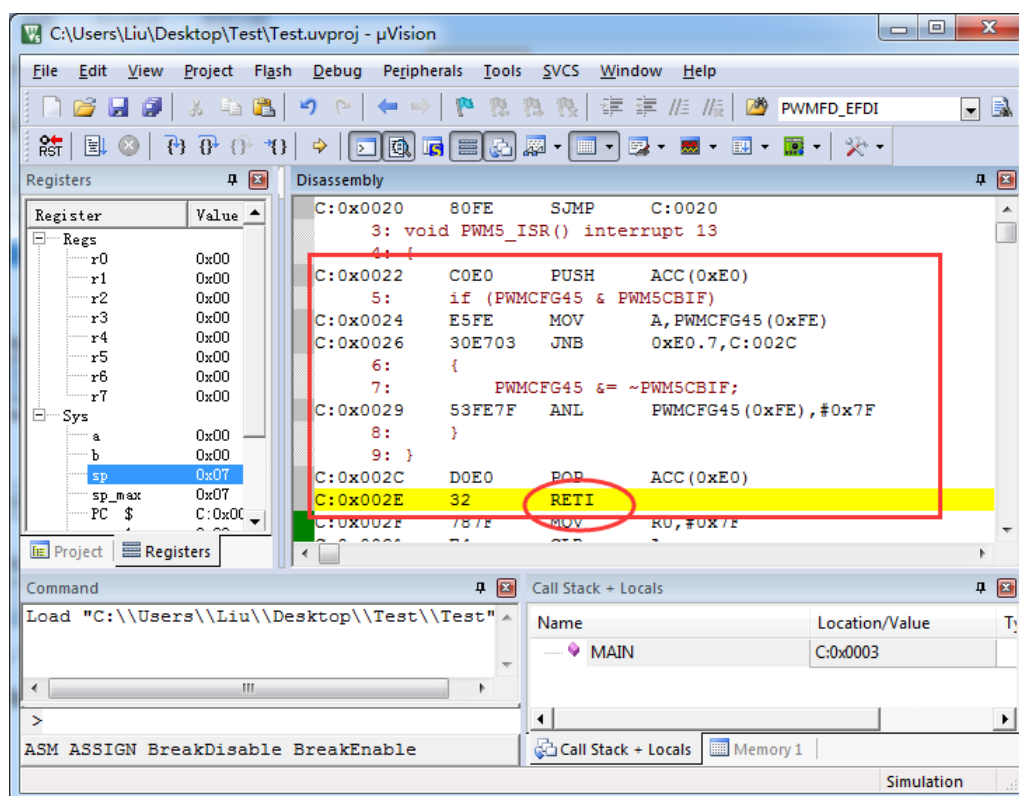


3、编译即可通过。

此时经过 Keil 的 C51 编译器编译后，在 006BH 处有一条“LJMP PWM5\_ISR”，在 0103H 处有一条“LJMP 006BH”，如下图：



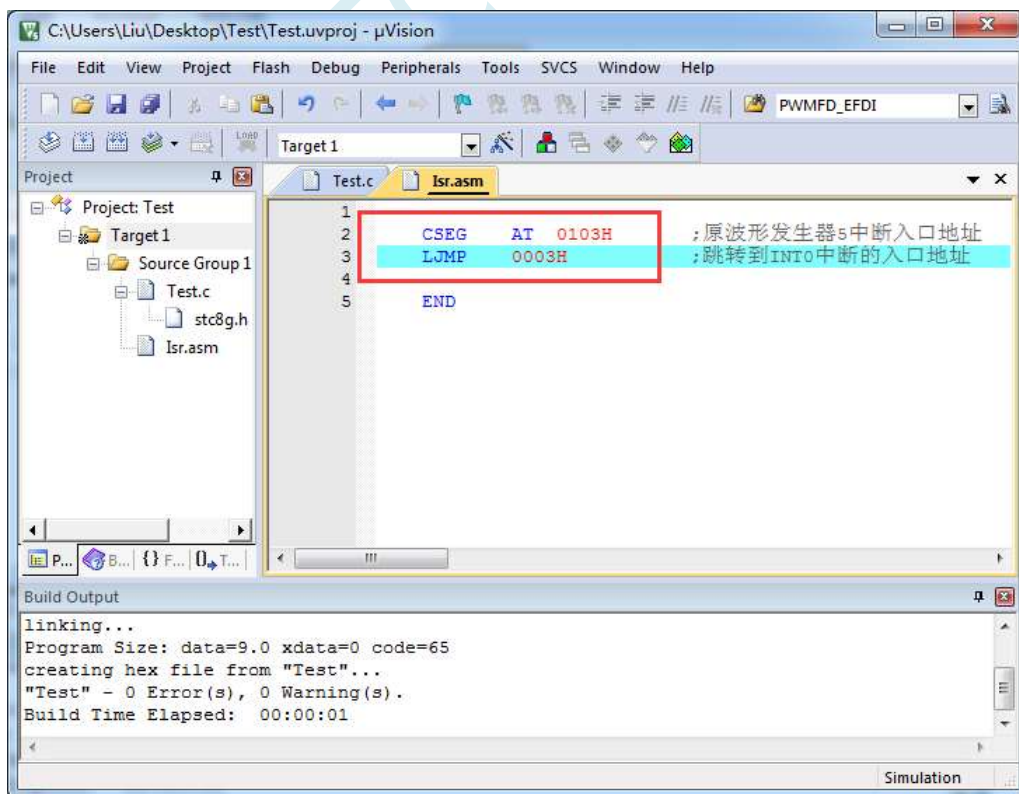
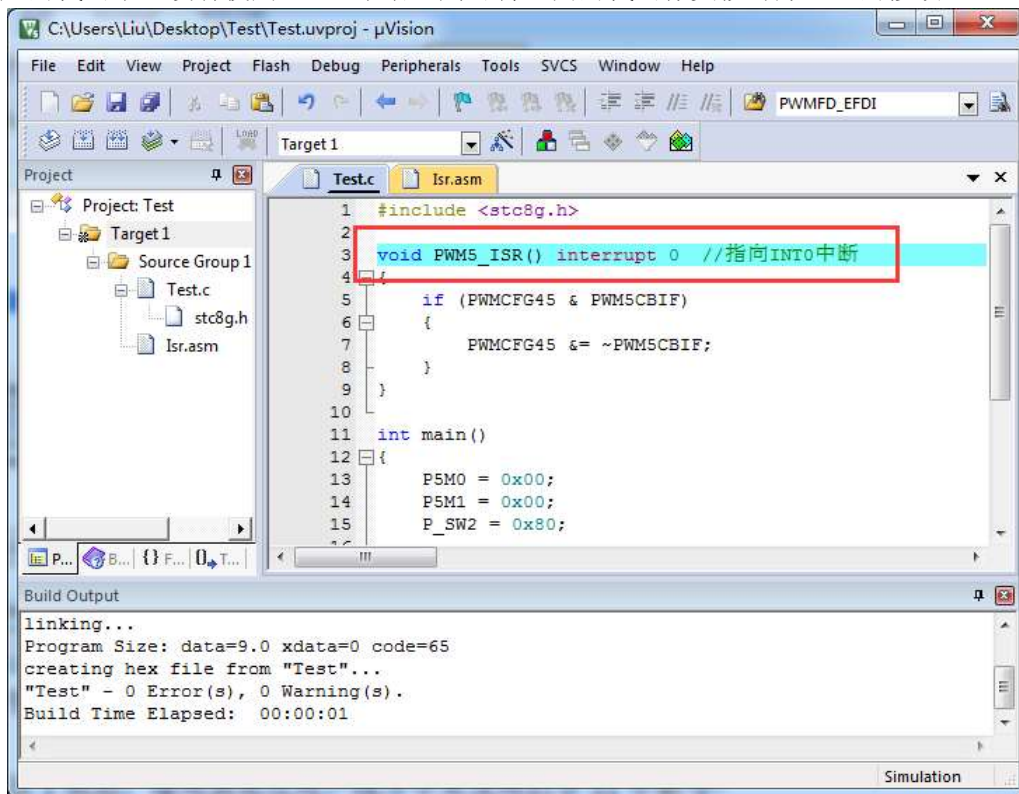
当发生 PWM5 中断时，硬件会自动跳转到 0103H 地址执行“LJMP 006BH”，然后在 006BH 处再执行“LJMP PWM5\_ISR”即可跳转到真正的中断服务程序，如下图：

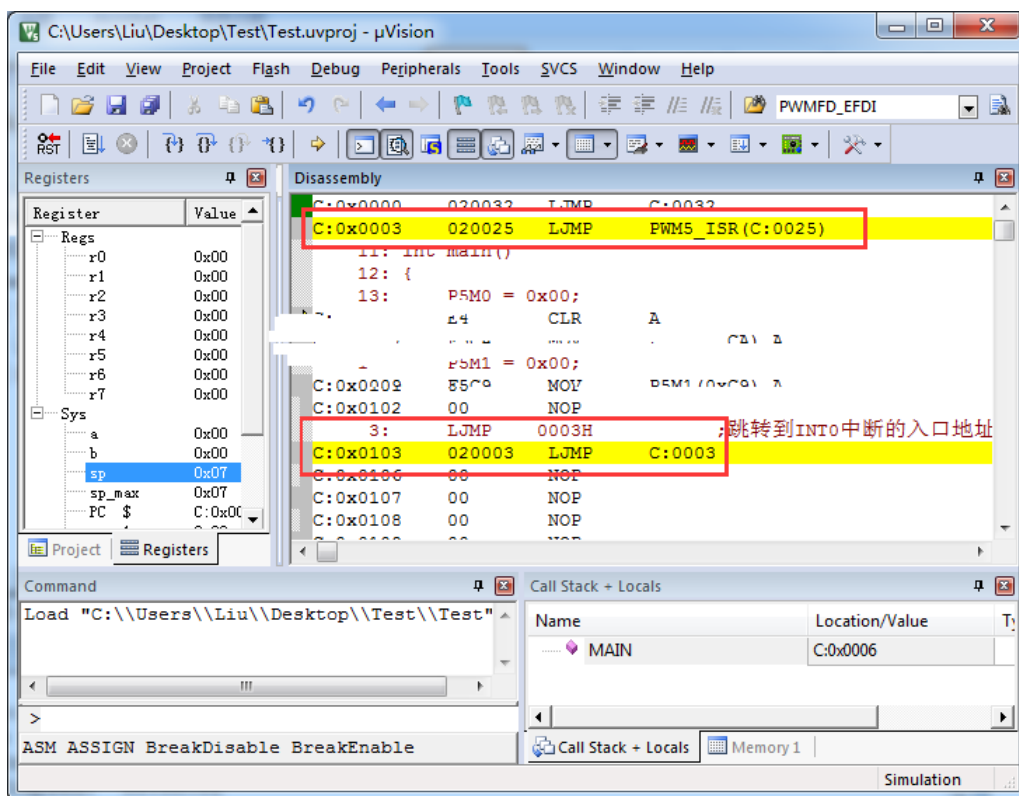


中断服务程序执行完成后，再通过 RETI 指令返回。整个中断响应过程只是多执行了一条 LJMP 语句而已。

## 方法 2: 与方法 1 类似, 借用用户程序中未使用的 0~31 的中断号

比如在用户的代码中, 没有使用 INTO 中断, 则可将上面的代码作类似与方法 1 的修改:





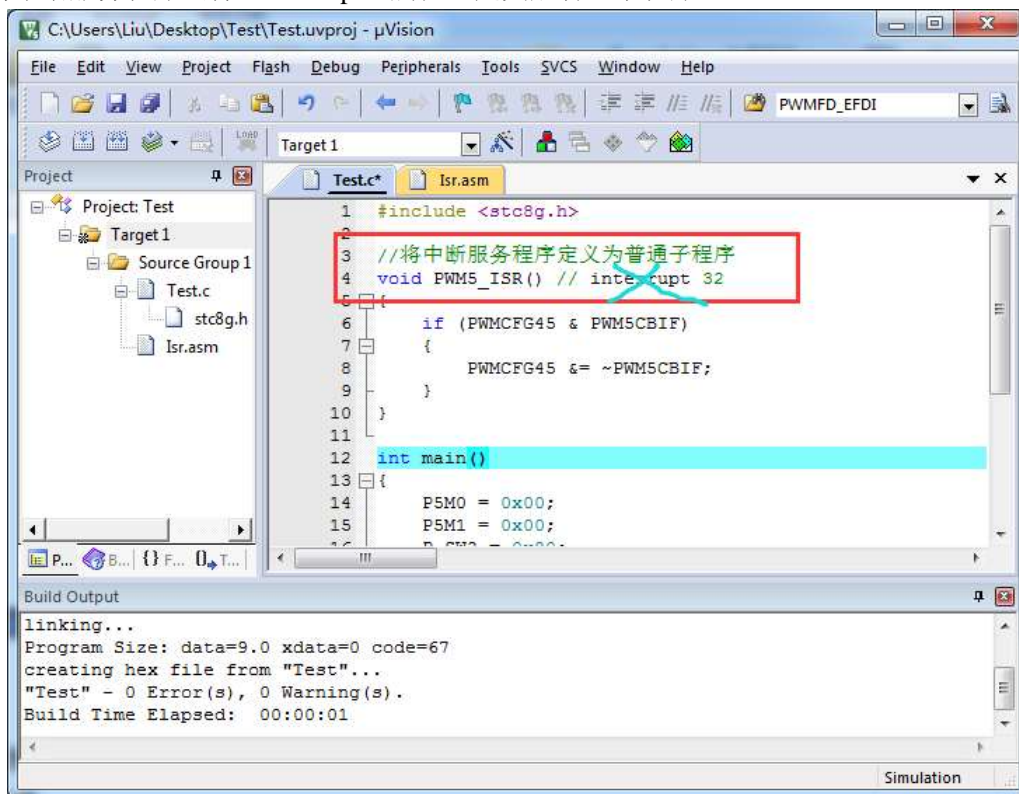
执行效果与方法 1 相同，此方法适用于需要重映射多个中断号大于 31 的情况。



### 方法 3: 将中断服务程序定义成子程序, 然后在汇编代码中的中断入口地址中使用 **LCALL** 指令执行服务程序

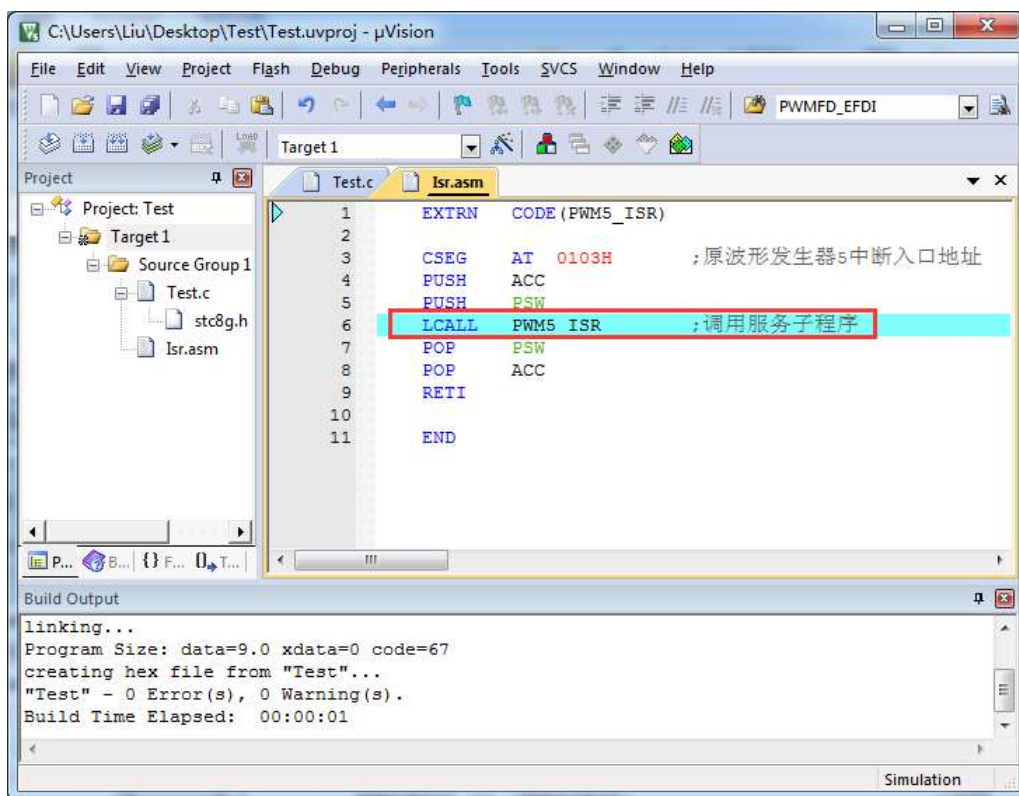
操作步骤如下:

- 1、首先将中断服务程序去掉“interrupt”属性, 定义成普通子程序

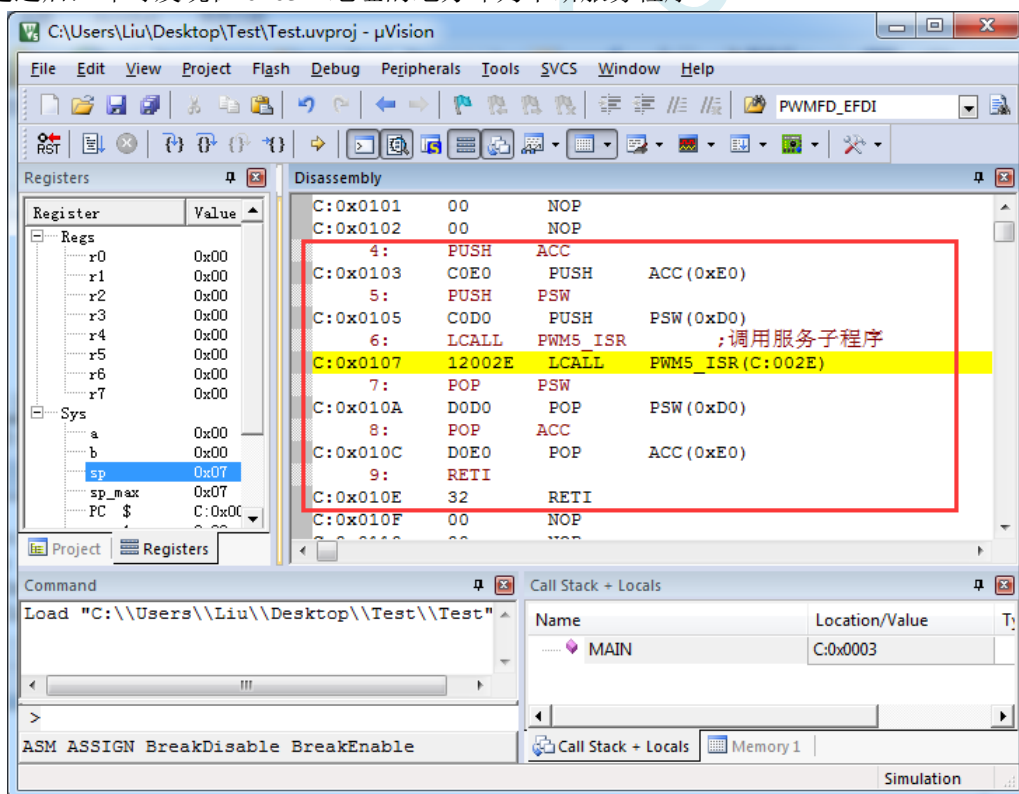


- 2、然后在汇编文件的 0103H 地址输入如下图所示的代码





3、编译通过后，即可发现在 0103H 地址的地方即为中断服务程序



此方法不需要重映射中断入口，不过这种方法有一个问题，在汇编文件中具体需要将哪些寄存器压入堆栈，需要用户查看 C 程序的反汇编代码来确定。一般包括 PSW、ACC、B、DPL、DPH 以及 R0~R7。除 PSW 必须压栈外，其他哪些寄存器在用户子程序中有使用，就必须将哪些寄存器压栈。

## 附录U 电气特性

### U.1 绝对最大额定值

| 参数        | 最小值  | 最大值     | 单位 | 说明   |
|-----------|------|---------|----|--|
| 存储温度      | -55  | +150    | ℃  |  |
| 工作温度      | -40  | +85     | ℃  | <p>若工作温度高于 85℃（如 125℃附近），由于内部 IRC 时钟的频率在高温时的温漂大，建议使用外部高温时钟或晶振。另外温度高时频率跑不快，建议使用 24M 以下的工作频率；如果系统必须运行在较高温度，务请使用外部高可靠低频有源时钟。</p> <p>若工作温度为-55℃附近，则工作电压不能太低，强烈建议 MCU-VCC 电压不要低于 3.0V，另外电源的上升速度也必须尽量快，最好能控制在毫秒级</p> |
| 工作电压      | 1.9  | 5.5     | V  |  |
| VDD 对地电压  | -0.3 | +5.5    | V  |  |
| I/O 口对地电压 | -0.3 | VDD+0.3 | V  |  |

## U.2 直流特性 (3.3V)

(VSS=0V, VDD=3.3V, 测试温度=25℃)

| 标号                | 参数                | 范围   |      |      |    | 测试环境              |
|-------------------|-------------------|------|------|------|----|-------------------|
|                   |                   | 最小值  | 典型值  | 最大值  | 单位 |                   |
| I <sub>PD</sub>   | 掉电模式电流            | -    | 0.4  | -    | uA |                   |
| I <sub>WKT</sub>  | 掉电唤醒定时器           | -    | 1.5  | -    | uA |                   |
| I <sub>LVD</sub>  | 低压检测模块功耗          | -    | 10   | -    | uA |                   |
| I <sub>COMP</sub> | 比较器功耗             | -    | 90   | -    | uA |                   |
| I <sub>IDL</sub>  | 空闲模式电流 (内部 32KHz) | -    | 0.48 | -    | mA | 相当于传统 8051 的 0.5M |
|                   | 空闲模式电流 (6MHz)     | -    | 0.88 | -    | mA | 相当于传统 8051 的 79M  |
|                   | 空闲模式电流 (12MHz)    | -    | 1.00 | -    | mA | 相当于传统 8051 的 158M |
|                   | 空闲模式电流 (24MHz)    | -    | 1.16 | -    | mA | 相当于传统 8051 的 317M |
| I <sub>NOR</sub>  | 正常模式电流 (内部 32KHz) | -    | 0.48 | -    | mA | 相当于传统 8051 的 0.5M |
|                   | 正常模式电流 (500KHz)   | -    | 0.88 | -    | mA | 相当于传统 8051 的 7M   |
|                   | 正常模式电流 (600KHz)   | -    | 0.88 | -    | mA | 相当于传统 8051 的 8M   |
|                   | 正常模式电流 (700KHz)   | -    | 0.90 | -    | mA | 相当于传统 8051 的 9M   |
|                   | 正常模式电流 (800KHz)   | -    | 0.91 | -    | mA | 相当于传统 8051 的 11M  |
|                   | 正常模式电流 (900KHz)   | -    | 0.91 | -    | mA | 相当于传统 8051 的 12M  |
|                   | 正常模式电流 (1MHz)     | -    | 0.94 | -    | mA | 相当于传统 8051 的 13M  |
|                   | 正常模式电流 (2MHz)     | -    | 1.05 | -    | mA | 相当于传统 8051 的 26M  |
|                   | 正常模式电流 (3MHz)     | -    | 1.17 | -    | mA | 相当于传统 8051 的 40M  |
|                   | 正常模式电流 (4MHz)     | -    | 1.26 | -    | mA | 相当于传统 8051 的 53M  |
|                   | 正常模式电流 (5MHz)     | -    | 1.40 | -    | mA | 相当于传统 8051 的 66M  |
|                   | 正常模式电流 (6MHz)     | -    | 1.49 | -    | mA | 相当于传统 8051 的 79M  |
|                   | 正常模式电流 (12MHz)    | -    | 2.09 | -    | mA | 相当于传统 8051 的 158M |
|                   | 正常模式电流 (24MHz)    | -    | 3.16 | -    | mA | 相当于传统 8051 的 317M |
| V <sub>IL1</sub>  | 输入低电平             | -    | -    | 0.99 | V  | 打开施密特触发           |
|                   |                   | -    | -    | 1.07 | V  | 关闭施密特触发           |
| V <sub>IH1</sub>  | 输入高电平 (普通 I/O)    | 1.18 | -    | -    | V  | 打开施密特触发           |
|                   |                   | 1.09 | -    | -    | V  | 关闭施密特触发           |
| V <sub>IH2</sub>  | 输入高电平 (复位脚)       | 0.99 | -    | 1.18 | V  |                   |
| I <sub>OL1</sub>  | 输出低电平的灌电流         | -    | 20   | -    | mA | 端口电压 0.45V        |
| I <sub>OH1</sub>  | 输出高电平电流 (双向模式)    | 200  | 270  | -    | uA |                   |
| I <sub>OH2</sub>  | 输出高电平电流 (推挽模式)    | -    | 20   | -    | mA | 端口电压 2.4V         |
| I <sub>IL</sub>   | 逻辑 0 输入电流         | -    | -    | 50   | uA | 端口电压 0V           |
| I <sub>TL</sub>   | 逻辑 1 到 0 的转移电流    | 100  | 270  | 600  | uA | 端口电压 2.0V         |
| R <sub>PU</sub>   | I/O 口上拉电阻         | 5.8  | 5.9  | 6.0  | KΩ |                   |

|           |                     |   |     |   |     |                  |
|-----------|---------------------|---|-----|---|-----|------------------|
| I/O<br>速度 | I/O 大电流驱动, I/O 快速转换 |   | 25  |   | MHz | PxDR=0, PxSR=0   |
|           | I/O 小电流驱动, I/O 快速转换 |   | 22  |   | MHz | PxDR=1, PxSR=0   |
|           | I/O 大电流驱动, I/O 慢速转换 |   | 16  |   | MHz | PxDR=0, PxSR=1   |
|           | I/O 小电流驱动, I/O 慢速转换 |   | 12  |   | MHz | PxDR=1, PxSR=1   |
| 比较<br>器   | 最快速度                |   | 10  |   | MHz | 关闭所有模拟和数字滤波      |
|           | 模拟滤波时间              |   | 0.1 |   | us  |                  |
|           | 数字滤波时间              |   | 0   |   | 系统  | LCDTY=0          |
|           |                     |   | n+2 |   | 时钟  | LCDTY=n (n=1~63) |
| IPD2      | 使能比较器时掉电模式功耗        | - | 400 | - | uA  |                  |
| IPD3      | 使能 LVD 时掉电模式功耗      | - | 470 | - | uA  |                  |

## U.3 直流特性 (5.0V)

(VSS=0V, VDD=5.0V, 测试温度=25℃)

| 标号                | 参数                | 范围   |      |      |    | 测试环境              |
|-------------------|-------------------|------|------|------|----|-------------------|
|                   |                   | 最小值  | 典型值  | 最大值  | 单位 |                   |
| I <sub>PD</sub>   | 掉电模式电流            | -    | 0.6  | -    | uA |                   |
| I <sub>WKT</sub>  | 掉电唤醒定时器           | -    | 4.4  | -    | uA |                   |
| I <sub>LVD</sub>  | 低压检测模块功耗          | -    | 30   | -    | uA |                   |
| I <sub>COMP</sub> | 比较器功耗             | -    | 90   | -    | uA |                   |
| I <sub>IDL</sub>  | 空闲模式电流 (内部 32KHz) | -    | 0.58 | -    | mA | 相当于传统 8051 的 0.5M |
|                   | 空闲模式电流 (6MHz)     | -    | 0.98 | -    | mA | 相当于传统 8051 的 79M  |
|                   | 空闲模式电流 (12MHz)    | -    | 1.10 | -    | mA | 相当于传统 8051 的 158M |
|                   | 空闲模式电流 (24MHz)    | -    | 1.25 | -    | mA | 相当于传统 8051 的 317M |
| I <sub>NOR</sub>  | 正常模式电流 (内部 32KHz) | -    | 0.58 | -    | mA | 相当于传统 8051 的 0.5M |
|                   | 正常模式电流 (500KHz)   |      | 0.97 |      | mA | 相当于传统 8051 的 7M   |
|                   | 正常模式电流 (600KHz)   |      | 0.97 |      | mA | 相当于传统 8051 的 8M   |
|                   | 正常模式电流 (700KHz)   |      | 1.00 |      | mA | 相当于传统 8051 的 9M   |
|                   | 正常模式电流 (800KHz)   |      | 1.01 |      | mA | 相当于传统 8051 的 11M  |
|                   | 正常模式电流 (900KHz)   |      | 1.01 |      | mA | 相当于传统 8051 的 12M  |
|                   | 正常模式电流 (1MHz)     |      | 1.03 |      | mA | 相当于传统 8051 的 13M  |
|                   | 正常模式电流 (2MHz)     |      | 1.15 |      | mA | 相当于传统 8051 的 26M  |
|                   | 正常模式电流 (3MHz)     |      | 1.27 |      | mA | 相当于传统 8051 的 40M  |
|                   | 正常模式电流 (4MHz)     |      | 1.35 |      | mA | 相当于传统 8051 的 53M  |
|                   | 正常模式电流 (5MHz)     |      | 1.49 |      | mA | 相当于传统 8051 的 66M  |
|                   | 正常模式电流 (6MHz)     | -    | 1.59 | -    | mA | 相当于传统 8051 的 79M  |
|                   | 正常模式电流 (12MHz)    | -    | 2.19 | -    | mA | 相当于传统 8051 的 158M |
|                   | 正常模式电流 (24MHz)    | -    | 3.27 | -    | mA | 相当于传统 8051 的 317M |
| V <sub>IL1</sub>  | 输入低电平             | -    | -    | 1.32 | V  | 打开施密特触发           |
|                   |                   | -    | -    | 1.48 | V  | 关闭施密特触发           |
| V <sub>IH1</sub>  | 输入高电平 (普通 I/O)    | 1.60 | -    | -    | V  | 打开施密特触发           |
|                   |                   | 1.54 | -    | -    | V  | 关闭施密特触发           |
| V <sub>IH2</sub>  | 输入高电平 (复位脚)       | 1.32 | -    | 1.60 | V  |                   |
| I <sub>OL1</sub>  | 输出低电平的灌电流         | -    | 20   | -    | mA | 端口电压 0.45V        |
| I <sub>OH1</sub>  | 输出高电平电流 (双向模式)    | 200  | 270  | -    | uA |                   |
| I <sub>OH2</sub>  | 输出高电平电流 (推挽模式)    | -    | 20   | -    | mA | 端口电压 2.4V         |
| I <sub>IL</sub>   | 逻辑 0 输入电流         | -    | -    | 50   | uA | 端口电压 0V           |
| I <sub>TL</sub>   | 逻辑 1 到 0 的转移电流    | 100  | 270  | 600  | uA | 端口电压 2.0V         |
| R <sub>PU</sub>   | I/O 口上拉电阻         | 4.1  | 4.2  | 4.4  | KΩ |                   |

|           |                     |   |     |   |          |                  |
|-----------|---------------------|---|-----|---|----------|------------------|
| I/O<br>速度 | I/O 大电流驱动, I/O 快速转换 |   | 36  |   | MHz      | PxDR=0, PxSR=0   |
|           | I/O 小电流驱动, I/O 快速转换 |   | 32  |   | MHz      | PxDR=1, PxSR=0   |
|           | I/O 大电流驱动, I/O 慢速转换 |   | 26  |   | MHz      | PxDR=0, PxSR=1   |
|           | I/O 小电流驱动, I/O 慢速转换 |   | 22  |   | MHz      | PxDR=1, PxSR=1   |
| 比较器       | 最快速度                |   | 10  |   | MHz      | 关闭所有模拟和数字滤波      |
|           | 模拟滤波时间              |   | 0.1 |   | us       |                  |
|           | 数字滤波时间              |   | 0   |   | 系统<br>时钟 | LCDTY=0          |
|           |                     |   | n+2 |   |          | LCDTY=n (n=1~63) |
| IPD2      | 使能比较器时掉电模式功耗        | - | 460 | - | uA       |                  |
| IPD3      | 使能 LVD 时掉电模式功耗      | - | 520 | - | uA       |                  |

#### U.4 内部 IRC 温漂特性 (参考温度 25°C)

| 温度           | 范围  |                 |     |
|--------------|-----|-----------------|-----|
|              | 最小值 | 典型值             | 最大值 |
| -40°C ~ 85°C |     | -1.38% ~ +1.42% |     |
| -20°C ~ 65°C |     | -0.88% ~ +1.05% |     |

#### U.5 低压复位门槛电压 (测试温度 25°C)

| 级别   | 电压  |                      |     |
|------|-----|----------------------|-----|
|      | 最小值 | 典型值 (实测值)            | 最大值 |
| POR  |     | (1.69V ~ 1.82V)      |     |
| LVR0 |     | 2.0V (1.88V ~ 1.99V) |     |
| LVR1 |     | 2.4V (2.28V ~ 2.45V) |     |
| LVR2 |     | 2.7V (2.58V ~ 2.76V) |     |
| LVR3 |     | 3.0V (2.86V ~ 3.06V) |     |

## 附录V 应用注意事项

### V.1 关于 STC8H 系列 IO 口的注意事项

1. STC8H 系列芯片的 IO 口，除了 ISP 下载口 P3.0 和 P3.1 外，其余的 IO 口上电后的初始模式均为高阻输入模式，用户无法直接输出电平，所以用户在程序初始化的地方必须要使用 PxM0 和 PxM1 两个寄存器初始化相应的 IO 模式，才能正常使用。
2. STC8H 系列芯片所有的 I/O 口均可以设置为准双向口模式、强推挽输出模式、开漏输出模式或者高阻输入模式，另外每个 I/O 均可独立使能内部 4K 上拉电阻
3. STC8H 系列芯片不会自动为特殊 IO 设置 IO 口模式，如 ADC 口、串口、I2C 口以及 SPI 口，必须用户自行将相应的口设置为合适的模式
4. 若使能 P5.4 管脚为复位脚，则复位电平为低电平
5. **特别注意：**由于 STC8H 系列的所有 I/O（除了 ISP 下载口 P3.0/P3.1 外）在上电后都是高阻输入模式，I/O 外部电平不固定，此时如果 MCU 直接进入掉电模式/停机模式，会导致 I/O 有额外的耗电，所有在 MCU 进入掉电模式/停机模式前，必须将所有 I/O 口都根据实际情况设置好 I/O 口的模式，对于所有没有使用的外部悬空的 I/O 都需要设置为准双向口，并固定输出高电平。特别是部分管脚的芯片，由于芯片内部有部分 I/O 口并没有打线到外部管脚，所以这些 I/O 也是处于悬空状态的，这部分 I/O 也需要设置为准双向口，并固定输出高电平。

### V.2 STC8H8K64U 系列

1. STC8H8K64U 的 B 版本芯片，RTC 不能使用内部 32K 作为时钟源。
  2. STC8H8K64U 的 B 版本芯片，比较器不能选择 ADC 通道作为正极输入。
  3. STC8H8K64U 的 B 版本芯片，进入省电模式后会有 3uA 左右的耗电。
- 上面的 3 个问题，STC8H8K64U 的 C 版本会全部修改正确

### V.3 STC8H3K64S2/S4 系列

1. **STC8H3K64S4 和 STC8H3K64S2 系列的 B 版芯片 I/O 中断经测试发现问题，请暂时不要使用。**（这个问题，C 版芯片将会修改正确）
2. **STC8H3K64S4 和 STC8H3K64S2 系列的 B 版芯片的内部 IRC，32M~37M 可能是盲区，请不要选择这个频率范围的频率**

### V.4 STC8H4K64TLR 系列

1. 当 RTC 的时钟源选择外部 32.768K 的晶振时，需要将晶振接入脚 P1.6 和 P1.7 的数字通道关闭，否则进入 STOP 模式后会有额外的漏电。（将寄存器 P1IE 的 bit6 和 bit7 都设置为 0 即可关闭 P1.6 和 P1.7 的数字通道）

## V.5 STC8H2K64T 系列

1. STC8H2K64T 系列的 A 版芯片, 当设置 P0、P1、P5 口的 I/O 口中断允许位寄存器 P0INTE、P1INTE、P5INTE 时, 会对 P0、P1、P5 的数字输入允许寄存器 P0IE、P1IE、P5IE 造成影响。
2. 测试 STC8H2K64T 系列的 A 版芯片的 ADC 功能时, 不要去设置 P0IE、P1IE 和 P5IE 寄存器, 因为设置是无效的。
3. STC8H2K64T 系列的 A 版芯片的 RTC 功能还有待完善, 请不要使用
4. **特别注意:** 由于 STC8H 系列的所有 I/O (除了 ISP 下载口 P3.0/P3.1 外) 在上电后都是高阻输入模式, I/O 外部电平不固定, 此时如果 MCU 直接进入掉电模式/停机模式, 会导致 I/O 有额外的耗电, 所有在 MCU 进入掉电模式/停机模式前, 必须将所有 I/O 口都根据实际情况设置好 I/O 口的模式, 对于所有没有使用的外部悬空的 I/O 都需要设置为准双向口, 并固定输出高电平。特别是部分管脚的芯片, 由于芯片内部有部分 I/O 口并没有打线到外部管脚, 所以这些 I/O 也是处于悬空状态的, 这部分 I/O 也需要设置为准双向口, 并固定输出高电平。



## 附录W 触摸按键的 PCB 设计指导

触摸按键对 PCB 设计的要求比较严格, 否则其效果会大打折扣甚至失败。建议用户在设计 PCB 时遵循以下几点原则:

1. 遵循通常的数模混合电路设计的基本原则。

电容式触摸按键模块集成了精密电容测量的模拟电路, 因此进行 PCB 设计时应该把它看成一个独立的模拟电路对待。遵循通常的数模混合电路设计的基本原则。

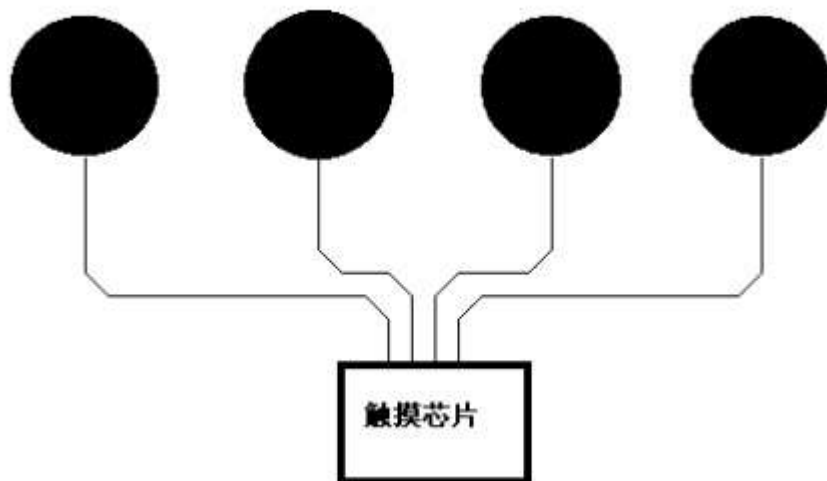
2. 采用星形接地

触摸芯片的地线不要和其他电路公用, 应该单独连到板子电源输入的接地点, 也就是通常说的采用“星形接地”。

3. 电源上产生的噪声对触摸芯片的影响

电源纹波、噪声应该尽量小, 最好用一根独立的走线从板子的供电点取电并增加滤波措施, 不要和其他的电路共用电源回路。

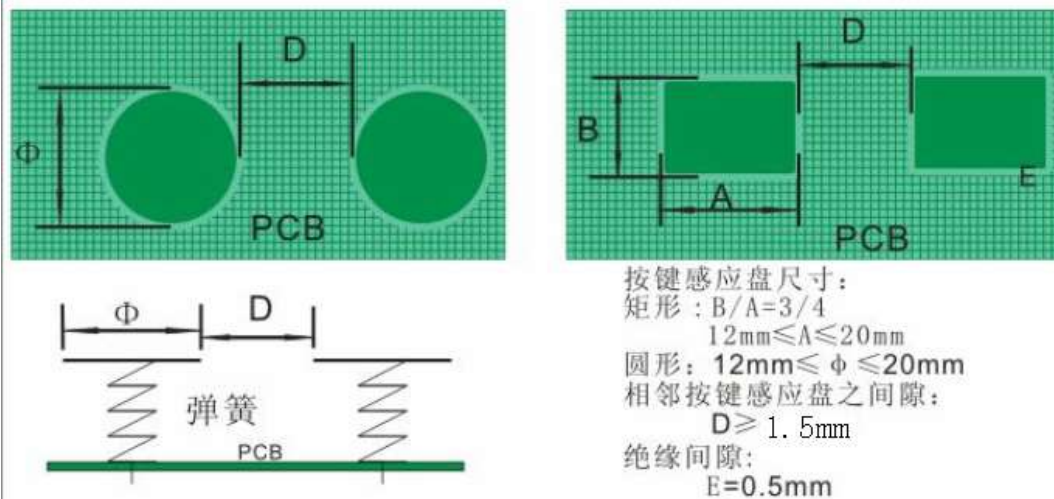
4. IC 与感应盘的连线尽量等长, 让其有近似的分布电容, 入下图所示。



5. 按键感应盘（电容传感器）大小和间隙

在满足面板的美学设计要求的情况下, 必须通过合理安排的感应盘大小和间隔尺寸, 来获得最佳的触摸感应效果。感应盘放在底层, IC 也放在底层, 感应盘与 IC 连线不要有过孔。相邻感应盘边沿的间隔最好在 1.5mm 以上 (下图中的尺寸 D), 如果 PCB 面积允许, 尽量取大一些间隔。铺铜与感应盘的间隔为 0.5mm (下图中的尺寸 E)。

在家用电器应用中，以下推荐的感应盘大小和间距的尺寸可获得最佳触摸感应效果



#### 6. 铺铜处理

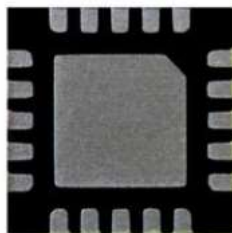
底层可以铺网格铜或实铜均可，注意铺铜与感应盘的间隔为 0.5mm。顶层印刷按键的丝印信息，丝印的外框形状与底层感应盘一致，顶层对应底层感应盘的地方不能铺铜，否则会屏蔽掉触摸动作。顶层铺铜与底层铺铜一样即可。

#### 7. 走线处理

感应盘与 IC 的连线使用比较小的线宽为好，比如 10~15mil 之间。感应盘到触摸芯片的连线不要跨越强干扰、高频、大电流的线。感应盘到触摸芯片的连线周围 1.5mm 内不要走其他信号线，越远离越好。顶层对应底层感应盘和连接线的地方，最好不要放任何线。

## 附录X QFN/DFN 封装元器件焊接方法

STC 产品的封装形式中,增加了现在比较流行的 QFN 和 DFN 的封装。由于这种封装形式的芯片芯片的管脚在芯片底部,手工焊接有一定的难度。市面上有专门做工程样品焊接的小公司,可承接工程样品打样。如用户需要自行焊接,可参考下面的焊接方法。



- 1、 首先需要准备如下工具:电烙铁、热风枪、镊子、固定架等工具
- 2、 需要焊接的 PCB 板和芯片如下图:



- 3、 先给板上芯片的焊盘上锡:



- 4、 然后给芯片底部上锡,这个上完锡后要弄平,尽量减少锡,但不能没有。



- 5、 调整热风枪温度，实际出风大概在 240 度左右，因为风枪质量不一样，根据实际情况调节。



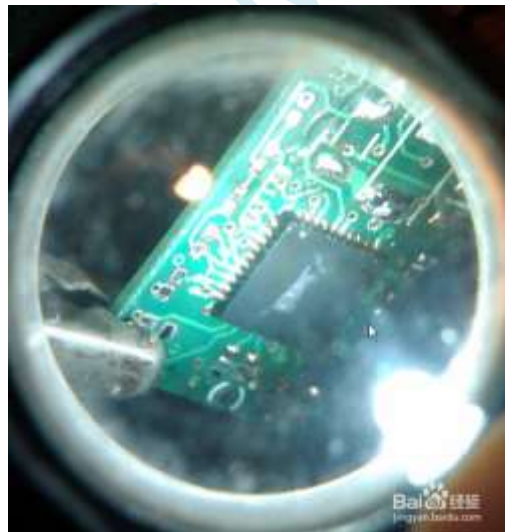
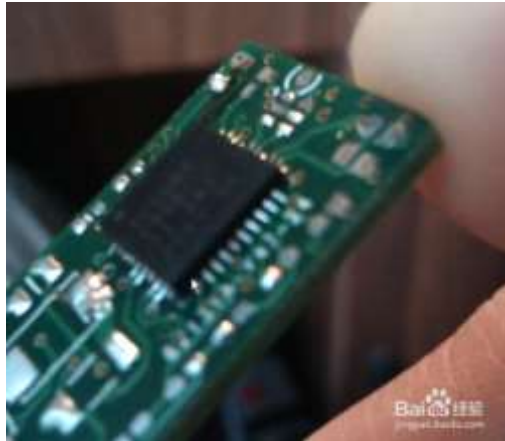
- 6、 把芯片放到焊盘上，一定要放正，然后用热风枪对着它吹，速度要均匀，直到锡溶化，一般 20 秒内。



- 7、 用烙铁给芯片侧引脚上锡



## 8、 焊接完成后的效果



## 附录Y 关于回流焊前是否要烘烤

根据国际湿气敏感性等级 3 (MSL3) 规范的要求, 贴片元器件在拆开真空包装后, 168 小时内, 7 天内, 必须回流焊贴片完成, 如未完成, 必须再次高温烘烤。

SOP/TSSOP 塑料管耐不了 100 度以上的高温, 拆开真空包装后 7 天内必须回流焊贴片完成, 否则回流焊前去除耐不了 100 度以上高温的塑料管, 放到金属托盘中, 重新烘烤: 110~125℃, 4~8 个小时都可以

LQFP/QFN/DFN 托盘能耐 100 度以上的高温, 拆开真空包装后 7 天内必须回流焊贴片完成, 否则回流焊前必须重新烘烤: 110~125℃, 4~8 个小时都可以

STC MCU



## 附录Z 如何使用万用表检测芯片 I/O 口好坏

根据国际湿气敏感性等级 3 (MSL3) 规范的要求, 贴片元器件在拆开真空包装后, 168 小时内, 7 天内, 必须回流焊贴片完成, 如未完成, 必须再次高温烘烤。如果没有高温烘烤的流程, 直接进行回流焊, 则可能由于芯片内外受热不均导致芯片内部金属线被拉断, 最终出现的现象是芯片 I/O 口损坏。

STC 的单片机在芯片设计时, 每个 I/O 口都有两个分别到 VCC 和 GND 的保护二极管, 用万用表的二极管监测档可以进行测量。可使用此方法简单判断 I/O 管脚的好坏情况。使用万用表测量方法如下 (注: 这里使用的是数字万用表)

首先将万用表调到二极管检测挡位, 被测芯片不要供电, 将万用表的**红表笔**连接到被测芯片的**GND 管脚**, **黑表笔**依次测量每个 I/O 口, 如果万用表显示的参数为 0.7V 左右, 则表示芯片的内部 I/O 到 GND 的保护二极管正常, 即打线也是完好的, 若显示的参数为 0V, 则表示芯片内部的打线已被拉断。

上面的方法是检测芯片内部的打线情况的方法。

另外, 如果用户板上, 单片机的管脚没有加保护电路, 一旦出现过流或者过压都可能导致 I/O 烧坏。为检测管脚是否被烧坏, 除了使用上面的方法检测 I/O 口到 GND 的保护二极管外, 还需要检测 I/O 口到 VCC 的保护二极管。使用万用表检测 I/O 口到 VCC 的保护二极管的方法如下:

首先将万用表调到二极管检测挡位, 被测芯片不要供电, 将万用表的**黑表笔**连接到被测芯片的**VCC 管脚**, **红表笔**依次测量每个 I/O 口, 如果万用表显示的参数为 0.7V 左右, 则表示芯片的内部 I/O 到 VCC 的保护二极管正常, 若显示的参数为 0V, 则表示芯片此端口已被损坏。

## 附录AA 大批量生产，如何省去专门的烧录人员， 如何无烧录环节

大批量生产，你在将由 STC 的 MCU 作为主控芯片的控制板组装到设备里面之前在你将 STC MCU 贴片到你的控制板完成之后，你必须测试你的控制板的好坏。不要说 100%，直通无问题，那是抬杠，不是搞生产，只要生产，就会虚焊，短路，部分原件贴错，部分原件采购错。

所以在贴片回来后，组装到外壳里面之前，你必须测试，你的含有 STC MCU 控制板的好坏，好的去组装，坏的去维修抢救。

测试，大批量生产，必须有测试架/下面接上我们的脱机烧录工具 U8W/U8W-Mini/STC-USB Link1，还要接上其他控制部分

通过 USER-VCC、P3.0、P3.1、GND 连接，要工人每次都开电源

通过 S-VCC、P3.0、P3.1、GND 连接，不要你开电源，STC 的脱机工具给你自动供电

外面帮你做一个测试架的成本 500 元以下，就是有机玻璃，夹具，顶针。

1 个测试你控制板是否正常的工人管理 2 - 3 个 测试架

### 操作流程：

- 1、 将你的 STC MCU 控制板 卡到测试架 1 上
  - 2、 将你的 STC MCU 控制板 卡到测试架 2 上，测试架 1 上的程序已烧录完成/感觉不到烧录时间
  - 3、 测试 测试架 1 上的 STC 主控板功能是否正常，正常放到正常区，不正常，放到不正常区
  - 4、 给测试架 1 卡上新的未测试的无程序的控制板
  - 5、 测试 测试架 2 上的未测试控制板/程序不知何时早就不知不觉的烧好了，换新的未测试未烧录的控制板
  - 6、 循环步骤 3 到步骤 5
- =====不需要安排烧录人员



## 附录BB 关于 Keil 软件中 0xFD 问题的说明

众所周知, Keil 软件的 8051 和 80251 编译器的所有版本都有一个叫做 0xFD 的问题, 主要表现在字符串中不能含有带 0xFD 编码的汉字, 否则 Keil 软件在编译时会跳过 0xFD 而出现乱码。

关于这个问题, Keil 官方的回应是: 0xfd、0xfe、0xff 这 3 个字符编码被 Keil 编译器内部使用, 所以代码中若包含有 0xfd 的字符串时, 0xfd 会被编译器自动跳过。

Keil 官方提供的解决方法: 在带有 0xfd 编码的汉字后增加一个 0xfd 即可。例如:

```
printf("数学");           //Keil 编译后打印会显示乱码
printf("数\xfd学");       //显示正常
```

这里的“\xfd”是标准 C 代码中的转义字符, “\x”表示其后的 1~2 个字符为 16 进制数。“\xfd”表示将 16 进制数 0xfd 插入到字符串中。

由于“数”的汉字编码是 0xCAFD, Keil 在编译时会跳过 FD, 而只将 CA 编译到目标文件中, 后面通过转义字符手动再补一个 0xfd 到目标文件中, 就形成完整的 0xCAFD, 从而可正常显示。

关于 0xFD 的补丁网上有很多, 基本只对旧版本的 Keil 软件有效。打补丁的方法均是在可执行文件中查找关键代码[80 FB FD], 并修改为[80 FB FF], 这种修改方法查找的关键代码过于简单, 很容易修改到其它无关的地方, 导致编译出来的目标文件运行时出现莫名其妙的问题。所以, 代码中的字符串有包含如下的汉字时, 建议使用 Keil 官方提供的解决方法进行解决

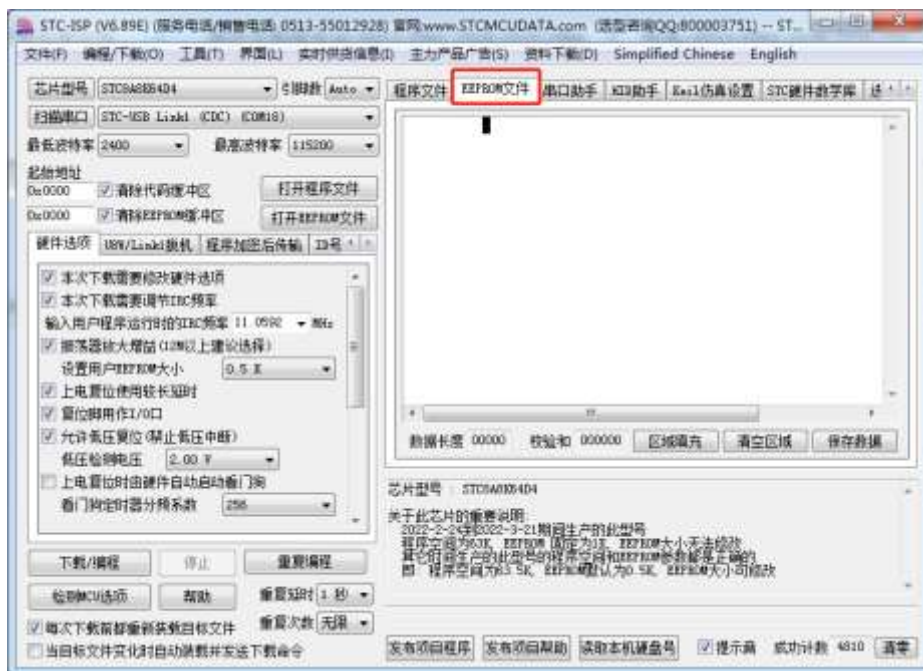
GB2312 中, 包含 0xfd 编码的汉字如下:

褒饼昌除待谍洱俘庚过糊积箭烜君魁  
例笼慢谬凝琵讷驱三升数她听妄锡淆  
旋妖引育札正铸 佚冽邳埠萃蒺掀啐  
幞狃恺泯潺姬纨琮槩犖掌臊忒睚铨稞  
瘕颀螭籴酏觚编鼯

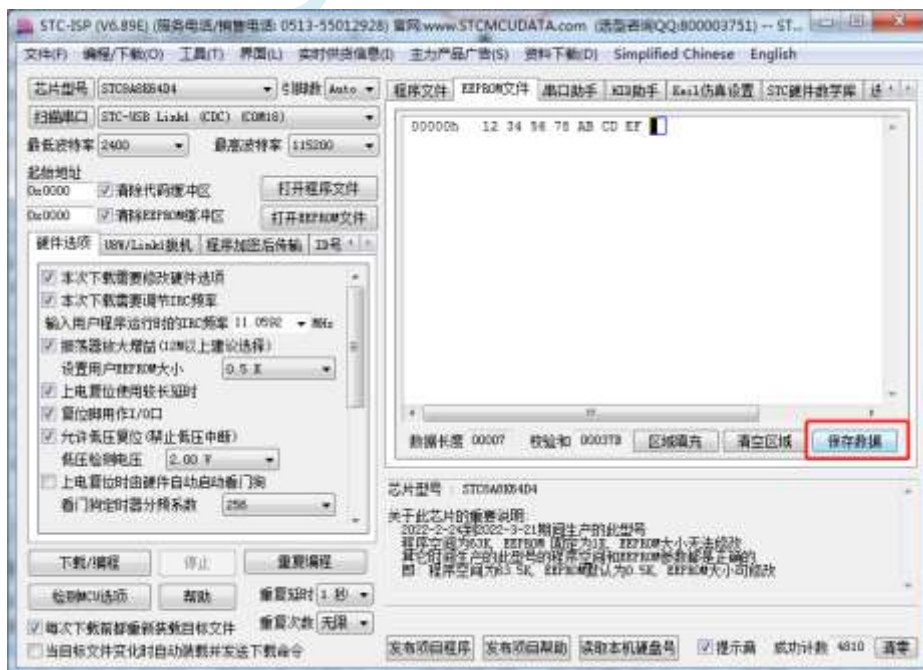
另外, Keil 项目路径名的字符中也不能含有带 0xFD 编码的汉字, 否则 Keil 软件会无法正确编译此项目。

## 附录CC 如何使用 STC-ISP 下载软件制作和编辑 EEPROM 文件

打开任意版本 STC-ISP 下载软件，选择“EEPROM”页面，单击数据窗口，如下所示



当出现黑色长方形的光标时，即可手动输入 16 进制数据，包括数字 0~9、字母 A~F（大小写通用）数据输入完成后，点击“保存数据”按钮即可保存 EEPROM 数据



STC MCU

## 附录DD STC8H 系列头文件定义

```
#ifndef __STC8H_H__
```

```
#define __STC8H_H__
```

```
////////////////////////////////////
```

```
//包含本头文件后,不用另外再包含"REG51.H"
```

```
sfr      P0          =      0x80;
sbit     P00         =      P0^0;
sbit     P01         =      P0^1;
sbit     P02         =      P0^2;
sbit     P03         =      P0^3;
sbit     P04         =      P0^4;
sbit     P05         =      P0^5;
sbit     P06         =      P0^6;
sbit     P07         =      P0^7;
sfr      SP          =      0x81;
sfr      DPL         =      0x82;
sfr      DPH         =      0x83;
sfr      S4CON       =      0x84;
sfr      S4BUF       =      0x85;
sfr      PCON        =      0x87;
sfr      TCON        =      0x88;
sbit     TF1         =      TCON^7;
sbit     TR1         =      TCON^6;
sbit     TF0         =      TCON^5;
sbit     TR0         =      TCON^4;
sbit     IE1         =      TCON^3;
sbit     IT1         =      TCON^2;
sbit     IE0         =      TCON^1;
sbit     IT0         =      TCON^0;
sfr      TMOD        =      0x89;
sfr      TL0         =      0x8a;
sfr      TL1         =      0x8b;
sfr      TH0         =      0x8c;
sfr      TH1         =      0x8d;
sfr      AUXR        =      0x8e;
sfr      INTCLKO     =      0x8f;
sfr      P1          =      0x90;
```

|      |         |   |         |
|------|---------|---|---------|
| sbit | P10     | = | P1^0;   |
| sbit | P11     | = | P1^1;   |
| sbit | P12     | = | P1^2;   |
| sbit | P13     | = | P1^3;   |
| sbit | P14     | = | P1^4;   |
| sbit | P15     | = | P1^5;   |
| sbit | P16     | = | P1^6;   |
| sbit | P17     | = | P1^7;   |
| sfr  | P1M1    | = | 0x91;   |
| sfr  | P1M0    | = | 0x92;   |
| sfr  | P0M1    | = | 0x93;   |
| sfr  | P0M0    | = | 0x94;   |
| sfr  | P2M1    | = | 0x95;   |
| sfr  | P2M0    | = | 0x96;   |
| sfr  | SCON    | = | 0x98;   |
| sbit | SM0     | = | SCON^7; |
| sbit | SM1     | = | SCON^6; |
| sbit | SM2     | = | SCON^5; |
| sbit | REN     | = | SCON^4; |
| sbit | TB8     | = | SCON^3; |
| sbit | RB8     | = | SCON^2; |
| sbit | TI      | = | SCON^1; |
| sbit | RI      | = | SCON^0; |
| sfr  | SBUF    | = | 0x99;   |
| sfr  | S2CON   | = | 0x9a;   |
| sfr  | S2BUF   | = | 0x9b;   |
| sfr  | IRCBAND | = | 0x9d;   |
| sfr  | LIRTRIM | = | 0x9e;   |
| sfr  | IRTRIM  | = | 0x9f;   |
| sfr  | P2      | = | 0xa0;   |
| sbit | P20     | = | P2^0;   |
| sbit | P21     | = | P2^1;   |
| sbit | P22     | = | P2^2;   |
| sbit | P23     | = | P2^3;   |
| sbit | P24     | = | P2^4;   |
| sbit | P25     | = | P2^5;   |
| sbit | P26     | = | P2^6;   |
| sbit | P27     | = | P2^7;   |
| sfr  | P_SW1   | = | 0xa2;   |
| sfr  | IE      | = | 0xa8;   |
| sbit | EA      | = | IE^7;   |
| sbit | ELVD    | = | IE^6;   |
| sbit | EADC    | = | IE^5;   |
| sbit | ES      | = | IE^4;   |
| sbit | ET1     | = | IE^3;   |
| sbit | EX1     | = | IE^2;   |

|      |           |   |       |
|------|-----------|---|-------|
| sbit | ET0       | = | IE^1; |
| sbit | EX0       | = | IE^0; |
| sfr  | SADDR     | = | 0xa9; |
| sfr  | WKTCL     | = | 0xaa; |
| sfr  | WKTCH     | = | 0xab; |
| sfr  | S3CON     | = | 0xac; |
| sfr  | S3BUF     | = | 0xad; |
| sfr  | TA        | = | 0xae; |
| sfr  | IE2       | = | 0xaf; |
| sfr  | P3        | = | 0xb0; |
| sbit | P30       | = | P3^0; |
| sbit | P31       | = | P3^1; |
| sbit | P32       | = | P3^2; |
| sbit | P33       | = | P3^3; |
| sbit | P34       | = | P3^4; |
| sbit | P35       | = | P3^5; |
| sbit | P36       | = | P3^6; |
| sbit | P37       | = | P3^7; |
| sfr  | P3M1      | = | 0xb1; |
| sfr  | P3M0      | = | 0xb2; |
| sfr  | P4M1      | = | 0xb3; |
| sfr  | P4M0      | = | 0xb4; |
| sfr  | IP2       | = | 0xb5; |
| sfr  | IP2H      | = | 0xb6; |
| sfr  | IPH       | = | 0xb7; |
| sfr  | IP        | = | 0xb8; |
| sbit | PPCA      | = | IP^7; |
| sbit | PLVD      | = | IP^6; |
| sbit | PADC      | = | IP^5; |
| sbit | PS        | = | IP^4; |
| sbit | PT1       | = | IP^3; |
| sbit | PX1       | = | IP^2; |
| sbit | PT0       | = | IP^1; |
| sbit | PX0       | = | IP^0; |
| sfr  | SADEN     | = | 0xb9; |
| sfr  | P_SW2     | = | 0xba; |
| sfr  | ADC_CONTR | = | 0xbc; |
| sfr  | ADC_RES   | = | 0xbd; |
| sfr  | ADC_RESL  | = | 0xbe; |
| sfr  | P4        | = | 0xc0; |
| sbit | P40       | = | P4^0; |
| sbit | P41       | = | P4^1; |
| sbit | P42       | = | P4^2; |
| sbit | P43       | = | P4^3; |
| sbit | P44       | = | P4^4; |
| sbit | P45       | = | P4^5; |

|      |           |   |        |
|------|-----------|---|--------|
| sbit | P46       | = | P4^6;  |
| sbit | P47       | = | P4^7;  |
| sfr  | WDT_CONTR | = | 0xc1;  |
| sfr  | IAP_DATA  | = | 0xc2;  |
| sfr  | IAP_ADDRH | = | 0xc3;  |
| sfr  | IAP_ADDRL | = | 0xc4;  |
| sfr  | IAP_CMD   | = | 0xc5;  |
| sfr  | IAP_TRIG  | = | 0xc6;  |
| sfr  | IAP_CONTR | = | 0xc7;  |
| sfr  | P5        | = | 0xc8;  |
| sbit | P50       | = | P5^0;  |
| sbit | P51       | = | P5^1;  |
| sbit | P52       | = | P5^2;  |
| sbit | P53       | = | P5^3;  |
| sbit | P54       | = | P5^4;  |
| sbit | P55       | = | P5^5;  |
| sbit | P56       | = | P5^6;  |
| sbit | P57       | = | P5^7;  |
| sfr  | P5M1      | = | 0xc9;  |
| sfr  | P5M0      | = | 0xca;  |
| sfr  | P6M1      | = | 0xcb;  |
| sfr  | P6M0      | = | 0xcc;  |
| sfr  | SPSTAT    | = | 0xcd;  |
| sfr  | SPCTL     | = | 0xce;  |
| sfr  | SPDAT     | = | 0xcf;  |
| sfr  | PSW       | = | 0xd0;  |
| sbit | CY        | = | PSW^7; |
| sbit | AC        | = | PSW^6; |
| sbit | F0        | = | PSW^5; |
| sbit | RS1       | = | PSW^4; |
| sbit | RS0       | = | PSW^3; |
| sbit | OV        | = | PSW^2; |
| sbit | F1        | = | PSW^1; |
| sbit | P         | = | PSW^0; |
| sfr  | T4T3M     | = | 0xd1;  |
| sfr  | T4H       | = | 0xd2;  |
| sfr  | T4L       | = | 0xd3;  |
| sfr  | T3H       | = | 0xd4;  |
| sfr  | T3L       | = | 0xd5;  |
| sfr  | T2H       | = | 0xd6;  |
| sfr  | T2L       | = | 0xd7;  |
| sfr  | USBCLK    | = | 0xdc;  |
| sfr  | PLLCR     | = | 0xdc;  |
| sfr  | ADCCFG    | = | 0xde;  |
| sfr  | IP3       | = | 0xdf;  |
| sfr  | ACC       | = | 0xe0;  |

|      |          |   |       |
|------|----------|---|-------|
| sfr  | P7M1     | = | 0xe1; |
| sfr  | P7M0     | = | 0xe2; |
| sfr  | DPS      | = | 0xe3; |
| sfr  | DPL1     | = | 0xe4; |
| sfr  | DPH1     | = | 0xe5; |
| sfr  | CMPCR1   | = | 0xe6; |
| sfr  | CMPCR2   | = | 0xe7; |
| sfr  | P6       | = | 0xe8; |
| sbit | P60      | = | P6^0; |
| sbit | P61      | = | P6^1; |
| sbit | P62      | = | P6^2; |
| sbit | P63      | = | P6^3; |
| sbit | P64      | = | P6^4; |
| sbit | P65      | = | P6^5; |
| sbit | P66      | = | P6^6; |
| sbit | P67      | = | P6^7; |
| sfr  | USBDAT   | = | 0xec; |
| sfr  | IP3H     | = | 0xee; |
| sfr  | AUXINTIF | = | 0xef; |
| sfr  | B        | = | 0xf0; |
| sfr  | USBCON   | = | 0xf4; |
| sfr  | IAP_TPS  | = | 0xf5; |
| sfr  | P7       | = | 0xf8; |
| sbit | P70      | = | P7^0; |
| sbit | P71      | = | P7^1; |
| sbit | P72      | = | P7^2; |
| sbit | P73      | = | P7^3; |
| sbit | P74      | = | P7^4; |
| sbit | P75      | = | P7^5; |
| sbit | P76      | = | P7^6; |
| sbit | P77      | = | P7^7; |
| sfr  | USBADR   | = | 0xfc; |
| sfr  | RSTCFG   | = | 0xff; |

//如下特殊功能寄存器位于扩展 RAM 区域

//访问这些寄存器,需先将 P\_SW2 的 BIT7 设置为 1,才可正常读写

```

////////////////////////////////////
//FF00H-FFFFH
////////////////////////////////////

```

```

////////////////////////////////////
//FE00H-FEFFFH
////////////////////////////////////

```



```
#define CLKSEL          (*(unsigned char volatile xdata *)0xfe00)
#define CLKDIV          (*(unsigned char volatile xdata *)0xfe01)
#define HIRCCR          (*(unsigned char volatile xdata *)0xfe02)
#define XOSCCR          (*(unsigned char volatile xdata *)0xfe03)
#define IRC32KCR        (*(unsigned char volatile xdata *)0xfe04)
#define MCLKOCR         (*(unsigned char volatile xdata *)0xfe05)
#define IRCDB           (*(unsigned char volatile xdata *)0xfe06)
#define X32KCR          (*(unsigned char volatile xdata *)0xfe08)
#define HSCLKDIV        (*(unsigned char volatile xdata *)0xfe0b)

#define P0PU            (*(unsigned char volatile xdata *)0xfe10)
#define P1PU            (*(unsigned char volatile xdata *)0xfe11)
#define P2PU            (*(unsigned char volatile xdata *)0xfe12)
#define P3PU            (*(unsigned char volatile xdata *)0xfe13)
#define P4PU            (*(unsigned char volatile xdata *)0xfe14)
#define P5PU            (*(unsigned char volatile xdata *)0xfe15)
#define P6PU            (*(unsigned char volatile xdata *)0xfe16)
#define P7PU            (*(unsigned char volatile xdata *)0xfe17)
#define P0NCS           (*(unsigned char volatile xdata *)0xfe18)
#define P1NCS           (*(unsigned char volatile xdata *)0xfe19)
#define P2NCS           (*(unsigned char volatile xdata *)0xfe1a)
#define P3NCS           (*(unsigned char volatile xdata *)0xfe1b)
#define P4NCS           (*(unsigned char volatile xdata *)0xfe1c)
#define P5NCS           (*(unsigned char volatile xdata *)0xfe1d)
#define P6NCS           (*(unsigned char volatile xdata *)0xfe1e)
#define P7NCS           (*(unsigned char volatile xdata *)0xfe1f)
#define P0SR            (*(unsigned char volatile xdata *)0xfe20)
#define P1SR            (*(unsigned char volatile xdata *)0xfe21)
#define P2SR            (*(unsigned char volatile xdata *)0xfe22)
#define P3SR            (*(unsigned char volatile xdata *)0xfe23)
#define P4SR            (*(unsigned char volatile xdata *)0xfe24)
#define P5SR            (*(unsigned char volatile xdata *)0xfe25)
#define P6SR            (*(unsigned char volatile xdata *)0xfe26)
#define P7SR            (*(unsigned char volatile xdata *)0xfe27)
#define P0DR            (*(unsigned char volatile xdata *)0xfe28)
#define P1DR            (*(unsigned char volatile xdata *)0xfe29)
#define P2DR            (*(unsigned char volatile xdata *)0xfe2a)
#define P3DR            (*(unsigned char volatile xdata *)0xfe2b)
#define P4DR            (*(unsigned char volatile xdata *)0xfe2c)
#define P5DR            (*(unsigned char volatile xdata *)0xfe2d)
#define P6DR            (*(unsigned char volatile xdata *)0xfe2e)
#define P7DR            (*(unsigned char volatile xdata *)0xfe2f)
#define P0IE            (*(unsigned char volatile xdata *)0xfe30)
#define P1IE            (*(unsigned char volatile xdata *)0xfe31)
#define P2IE            (*(unsigned char volatile xdata *)0xfe32)
```

```
#define P3IE          (*(unsigned char volatile xdata *)0xfe33)
#define P4IE          (*(unsigned char volatile xdata *)0xfe34)
#define P5IE          (*(unsigned char volatile xdata *)0xfe35)
#define P6IE          (*(unsigned char volatile xdata *)0xfe36)
#define P7IE          (*(unsigned char volatile xdata *)0xfe37)

#define LCMIFCFG       (*(unsigned char volatile xdata *)0xfe50)
#define LCMIFCFG2      (*(unsigned char volatile xdata *)0xfe51)
#define LCMIFCR        (*(unsigned char volatile xdata *)0xfe52)
#define LCMIFSTA       (*(unsigned char volatile xdata *)0xfe53)
#define LCMIFDATL      (*(unsigned char volatile xdata *)0xfe54)
#define LCMIFDATH      (*(unsigned char volatile xdata *)0xfe55)

#define RTCCR          (*(unsigned char volatile xdata *)0xfe60)
#define RTCCFG         (*(unsigned char volatile xdata *)0xfe61)
#define RTCIEN         (*(unsigned char volatile xdata *)0xfe62)
#define RTCIF          (*(unsigned char volatile xdata *)0xfe63)
#define ALAHOUR        (*(unsigned char volatile xdata *)0xfe64)
#define ALAMIN         (*(unsigned char volatile xdata *)0xfe65)
#define ALASEC         (*(unsigned char volatile xdata *)0xfe66)
#define ALASSEC        (*(unsigned char volatile xdata *)0xfe67)
#define INIYEAR        (*(unsigned char volatile xdata *)0xfe68)
#define INIMONTH       (*(unsigned char volatile xdata *)0xfe69)
#define INIDAY         (*(unsigned char volatile xdata *)0xfe6a)
#define INIhour        (*(unsigned char volatile xdata *)0xfe6b)
#define INIMIN         (*(unsigned char volatile xdata *)0xfe6c)
#define INISEC         (*(unsigned char volatile xdata *)0xfe6d)
#define INISSEC        (*(unsigned char volatile xdata *)0xfe6e)
#define YEAR           (*(unsigned char volatile xdata *)0xfe70)
#define MONTH          (*(unsigned char volatile xdata *)0xfe71)
#define DAY            (*(unsigned char volatile xdata *)0xfe72)
#define HOUR           (*(unsigned char volatile xdata *)0xfe73)
#define MIN            (*(unsigned char volatile xdata *)0xfe74)
#define SEC            (*(unsigned char volatile xdata *)0xfe75)
#define SSEC           (*(unsigned char volatile xdata *)0xfe76)

#define I2CCFG         (*(unsigned char volatile xdata *)0xfe80)
#define I2CMSCR        (*(unsigned char volatile xdata *)0xfe81)
#define I2CMSST        (*(unsigned char volatile xdata *)0xfe82)
#define I2CSLCR        (*(unsigned char volatile xdata *)0xfe83)
#define I2CSLST        (*(unsigned char volatile xdata *)0xfe84)
#define I2CSLADR       (*(unsigned char volatile xdata *)0xfe85)
#define I2CTXD         (*(unsigned char volatile xdata *)0xfe86)
#define I2CRXD         (*(unsigned char volatile xdata *)0xfe87)
#define I2CMSAUX       (*(unsigned char volatile xdata *)0xfe88)
#define TM2PS          (*(unsigned char volatile xdata *)0xfea2)
```

```

#define TM3PS                (*(unsigned char volatile xdata *)0xfea3)
#define TM4PS                (*(unsigned char volatile xdata *)0xfea4)
#define ADCTIM               (*(unsigned char volatile xdata *)0xfea8)
#define T3T4PIN              (*(unsigned char volatile xdata *)0xfeac)

#define PWM1_ETRPS           (*(unsigned char volatile xdata *)0xfeb0)
#define PWM1_ENO             (*(unsigned char volatile xdata *)0xfeb1)
#define PWM1_PS              (*(unsigned char volatile xdata *)0xfeb2)
#define PWM1_IOAUX           (*(unsigned char volatile xdata *)0xfeb3)
#define PWM2_ETRPS           (*(unsigned char volatile xdata *)0xfeb4)
#define PWM2_ENO             (*(unsigned char volatile xdata *)0xfeb5)
#define PWM2_PS              (*(unsigned char volatile xdata *)0xfeb6)
#define PWM2_IOAUX           (*(unsigned char volatile xdata *)0xfeb7)
#define PWM1_CR1             (*(unsigned char volatile xdata *)0xfec0)
#define PWM1_CR2             (*(unsigned char volatile xdata *)0xfec1)
#define PWM1_SMCR            (*(unsigned char volatile xdata *)0xfec2)
#define PWM1_ETR             (*(unsigned char volatile xdata *)0xfec3)
#define PWM1_IER             (*(unsigned char volatile xdata *)0xfec4)
#define PWM1_SR1             (*(unsigned char volatile xdata *)0xfec5)
#define PWM1_SR2             (*(unsigned char volatile xdata *)0xfec6)
#define PWM1_EGR             (*(unsigned char volatile xdata *)0xfec7)
#define PWM1_CCMR1           (*(unsigned char volatile xdata *)0xfec8)
#define PWM1_CCMR2           (*(unsigned char volatile xdata *)0xfec9)
#define PWM1_CCMR3           (*(unsigned char volatile xdata *)0xfeca)
#define PWM1_CCMR4           (*(unsigned char volatile xdata *)0xfecb)
#define PWM1_CCER1           (*(unsigned char volatile xdata *)0xfecb)
#define PWM1_CCER2           (*(unsigned char volatile xdata *)0xfecd)
#define PWM1_CNTR            (*(unsigned int volatile xdata *)0xfece)
#define PWM1_CNTRH           (*(unsigned char volatile xdata *)0xfece)
#define PWM1_CNTRL           (*(unsigned char volatile xdata *)0xfecf)
#define PWM1_PSCR            (*(unsigned int volatile xdata *)0xfed0)
#define PWM1_PSCRH           (*(unsigned char volatile xdata *)0xfed0)
#define PWM1_PSCRL           (*(unsigned char volatile xdata *)0xfed1)
#define PWM1_ARR             (*(unsigned int volatile xdata *)0xfed2)
#define PWM1_ARRH            (*(unsigned char volatile xdata *)0xfed2)
#define PWM1_ARRL            (*(unsigned char volatile xdata *)0xfed3)
#define PWM1_RCR             (*(unsigned char volatile xdata *)0xfed4)
#define PWM1_CCR1            (*(unsigned int volatile xdata *)0xfed5)
#define PWM1_CCR1H           (*(unsigned char volatile xdata *)0xfed5)
#define PWM1_CCR1L           (*(unsigned char volatile xdata *)0xfed6)
#define PWM1_CCR2            (*(unsigned int volatile xdata *)0xfed7)
#define PWM1_CCR2H           (*(unsigned char volatile xdata *)0xfed7)
#define PWM1_CCR2L           (*(unsigned char volatile xdata *)0xfed8)
#define PWM1_CCR3            (*(unsigned int volatile xdata *)0xfed9)
#define PWM1_CCR3H           (*(unsigned char volatile xdata *)0xfed9)
#define PWM1_CCR3L           (*(unsigned char volatile xdata *)0xfeda)

```

```
#define PWM1_CCR4          (*(unsigned int volatile xdata *)0xfedb)
#define PWM1_CCR4H         (*(unsigned char volatile xdata *)0xfedb)
#define PWM1_CCR4L         (*(unsigned char volatile xdata *)0xfedc)
#define PWM1_BKR           (*(unsigned char volatile xdata *)0xfedd)
#define PWM1_DTR           (*(unsigned char volatile xdata *)0xfede)
#define PWM1_OISR          (*(unsigned char volatile xdata *)0xfedf)
#define PWM2_CR1           (*(unsigned char volatile xdata *)0xfee0)
#define PWM2_CR2           (*(unsigned char volatile xdata *)0xfee1)
#define PWM2_SMCR          (*(unsigned char volatile xdata *)0xfee2)
#define PWM2_ETR           (*(unsigned char volatile xdata *)0xfee3)
#define PWM2_IER           (*(unsigned char volatile xdata *)0xfee4)
#define PWM2_SR1           (*(unsigned char volatile xdata *)0xfee5)
#define PWM2_SR2           (*(unsigned char volatile xdata *)0xfee6)
#define PWM2_EGR           (*(unsigned char volatile xdata *)0xfee7)
#define PWM2_CCMR1         (*(unsigned char volatile xdata *)0xfee8)
#define PWM2_CCMR2         (*(unsigned char volatile xdata *)0xfee9)
#define PWM2_CCMR3         (*(unsigned char volatile xdata *)0xfeea)
#define PWM2_CCMR4         (*(unsigned char volatile xdata *)0xfeeb)
#define PWM2_CCER1         (*(unsigned char volatile xdata *)0xfeec)
#define PWM2_CCER2         (*(unsigned char volatile xdata *)0xfeed)
#define PWM2_CNTR          (*(unsigned int volatile xdata *)0xfeee)
#define PWM2_CNTRH         (*(unsigned char volatile xdata *)0xfeee)
#define PWM2_CNTRL         (*(unsigned char volatile xdata *)0xfef0)
#define PWM2_PSCR          (*(unsigned int volatile xdata *)0xfef0)
#define PWM2_PSCRH         (*(unsigned char volatile xdata *)0xfef0)
#define PWM2_PSCRL         (*(unsigned char volatile xdata *)0xfef1)
#define PWM2_ARR           (*(unsigned int volatile xdata *)0xfef2)
#define PWM2_ARRH          (*(unsigned char volatile xdata *)0xfef2)
#define PWM2_ARRL          (*(unsigned char volatile xdata *)0xfef3)
#define PWM2_RCR           (*(unsigned char volatile xdata *)0xfef4)
#define PWM2_CCR1          (*(unsigned int volatile xdata *)0xfef5)
#define PWM2_CCR1H         (*(unsigned char volatile xdata *)0xfef5)
#define PWM2_CCR1L         (*(unsigned char volatile xdata *)0xfef6)
#define PWM2_CCR2          (*(unsigned int volatile xdata *)0xfef7)
#define PWM2_CCR2H         (*(unsigned char volatile xdata *)0xfef7)
#define PWM2_CCR2L         (*(unsigned char volatile xdata *)0xfef8)
#define PWM2_CCR3          (*(unsigned int volatile xdata *)0xfef9)
#define PWM2_CCR3H         (*(unsigned char volatile xdata *)0xfef9)
#define PWM2_CCR3L         (*(unsigned char volatile xdata *)0xfefa)
#define PWM2_CCR4          (*(unsigned int volatile xdata *)0xfefb)
#define PWM2_CCR4H         (*(unsigned char volatile xdata *)0xfefb)
#define PWM2_CCR4L         (*(unsigned char volatile xdata *)0xfefc)
#define PWM2_BKR           (*(unsigned char volatile xdata *)0xfefd)
#define PWM2_DTR           (*(unsigned char volatile xdata *)0xfefe)
#define PWM2_OISR          (*(unsigned char volatile xdata *)0xfeff)
```

```

#define PWMA_ETRPS      (*(unsigned char volatile xdata *)0xfeb0)
#define PWMA_ENO        (*(unsigned char volatile xdata *)0xfeb1)
#define PWMA_PS         (*(unsigned char volatile xdata *)0xfeb2)
#define PWMA_IOAUX      (*(unsigned char volatile xdata *)0xfeb3)
#define PWMB_ETRPS      (*(unsigned char volatile xdata *)0xfeb4)
#define PWMB_ENO        (*(unsigned char volatile xdata *)0xfeb5)
#define PWMB_PS         (*(unsigned char volatile xdata *)0xfeb6)
#define PWMB_IOAUX      (*(unsigned char volatile xdata *)0xfeb7)
#define PWMA_CR1        (*(unsigned char volatile xdata *)0xfec0)
#define PWMA_CR2        (*(unsigned char volatile xdata *)0xfec1)
#define PWMA_SMCR       (*(unsigned char volatile xdata *)0xfec2)
#define PWMA_ETR        (*(unsigned char volatile xdata *)0xfec3)
#define PWMA_IER        (*(unsigned char volatile xdata *)0xfec4)
#define PWMA_SR1        (*(unsigned char volatile xdata *)0xfec5)
#define PWMA_SR2        (*(unsigned char volatile xdata *)0xfec6)
#define PWMA_EGR        (*(unsigned char volatile xdata *)0xfec7)
#define PWMA_CCMR1      (*(unsigned char volatile xdata *)0xfec8)
#define PWMA_CCMR2      (*(unsigned char volatile xdata *)0xfec9)
#define PWMA_CCMR3      (*(unsigned char volatile xdata *)0xfeca)
#define PWMA_CCMR4      (*(unsigned char volatile xdata *)0xfecb)
#define PWMA_CCER1      (*(unsigned char volatile xdata *)0xfecb)
#define PWMA_CCER2      (*(unsigned char volatile xdata *)0xfecd)
#define PWMA_CNTR       (*(unsigned int volatile xdata *)0xfece)
#define PWMA_CNTRH      (*(unsigned char volatile xdata *)0xfece)
#define PWMA_CNTRL      (*(unsigned char volatile xdata *)0xfecf)
#define PWMA_PSCR       (*(unsigned int volatile xdata *)0xfed0)
#define PWMA_PSCRH      (*(unsigned char volatile xdata *)0xfed0)
#define PWMA_PSCRL      (*(unsigned char volatile xdata *)0xfed1)
#define PWMA_ARR        (*(unsigned int volatile xdata *)0xfed2)
#define PWMA_ARRH       (*(unsigned char volatile xdata *)0xfed2)
#define PWMA_ARRL       (*(unsigned char volatile xdata *)0xfed3)
#define PWMA_RCR        (*(unsigned char volatile xdata *)0xfed4)
#define PWMA_CCR1       (*(unsigned int volatile xdata *)0xfed5)
#define PWMA_CCR1H      (*(unsigned char volatile xdata *)0xfed5)
#define PWMA_CCR1L      (*(unsigned char volatile xdata *)0xfed6)
#define PWMA_CCR2       (*(unsigned int volatile xdata *)0xfed7)
#define PWMA_CCR2H      (*(unsigned char volatile xdata *)0xfed7)
#define PWMA_CCR2L      (*(unsigned char volatile xdata *)0xfed8)
#define PWMA_CCR3       (*(unsigned int volatile xdata *)0xfed9)
#define PWMA_CCR3H      (*(unsigned char volatile xdata *)0xfed9)
#define PWMA_CCR3L      (*(unsigned char volatile xdata *)0xfeda)
#define PWMA_CCR4       (*(unsigned int volatile xdata *)0xfedb)
#define PWMA_CCR4H      (*(unsigned char volatile xdata *)0xfedb)
#define PWMA_CCR4L      (*(unsigned char volatile xdata *)0xfedc)
#define PWMA_BKR        (*(unsigned char volatile xdata *)0xfedd)
#define PWMA_DTR        (*(unsigned char volatile xdata *)0xfede)

```



```

#define PWMA_OISR          (*(unsigned char volatile xdata *)0xfedf)
#define PWMB_CR1           (*(unsigned char volatile xdata *)0xfef0)
#define PWMB_CR2           (*(unsigned char volatile xdata *)0xfef1)
#define PWMB_SMCR          (*(unsigned char volatile xdata *)0xfef2)
#define PWMB_ETR           (*(unsigned char volatile xdata *)0xfef3)
#define PWMB_IER           (*(unsigned char volatile xdata *)0xfef4)
#define PWMB_SR1           (*(unsigned char volatile xdata *)0xfef5)
#define PWMB_SR2           (*(unsigned char volatile xdata *)0xfef6)
#define PWMB_EGR           (*(unsigned char volatile xdata *)0xfef7)
#define PWMB_CCMR1         (*(unsigned char volatile xdata *)0xfef8)
#define PWMB_CCMR2         (*(unsigned char volatile xdata *)0xfef9)
#define PWMB_CCMR3         (*(unsigned char volatile xdata *)0xfefa)
#define PWMB_CCMR4         (*(unsigned char volatile xdata *)0xfefb)
#define PWMB_CCER1         (*(unsigned char volatile xdata *)0xfefc)
#define PWMB_CCER2         (*(unsigned char volatile xdata *)0xfefd)
#define PWMB_CNTR          (*(unsigned int volatile xdata *)0xfefe)
#define PWMB_CNTRH         (*(unsigned char volatile xdata *)0xfef0)
#define PWMB_CNTRL         (*(unsigned char volatile xdata *)0xfef1)
#define PWMB_PSCR          (*(unsigned int volatile xdata *)0xfef2)
#define PWMB_PSCRH         (*(unsigned char volatile xdata *)0xfef3)
#define PWMB_PSCRL         (*(unsigned char volatile xdata *)0xfef4)
#define PWMB_ARR           (*(unsigned int volatile xdata *)0xfef5)
#define PWMB_ARRH          (*(unsigned char volatile xdata *)0xfef6)
#define PWMB_ARRL          (*(unsigned char volatile xdata *)0xfef7)
#define PWMB_RCR           (*(unsigned char volatile xdata *)0xfef8)
#define PWMB_CCR5          (*(unsigned int volatile xdata *)0xfef9)
#define PWMB_CCR5H         (*(unsigned char volatile xdata *)0xfefa)
#define PWMB_CCR5L         (*(unsigned char volatile xdata *)0xfefb)
#define PWMB_CCR6          (*(unsigned int volatile xdata *)0xfefc)
#define PWMB_CCR6H         (*(unsigned char volatile xdata *)0xfefd)
#define PWMB_CCR6L         (*(unsigned char volatile xdata *)0xfefe)
#define PWMB_CCR7          (*(unsigned int volatile xdata *)0xfef0)
#define PWMB_CCR7H         (*(unsigned char volatile xdata *)0xfef1)
#define PWMB_CCR7L         (*(unsigned char volatile xdata *)0xfef2)
#define PWMB_CCR8          (*(unsigned int volatile xdata *)0xfef3)
#define PWMB_CCR8H         (*(unsigned char volatile xdata *)0xfef4)
#define PWMB_CCR8L         (*(unsigned char volatile xdata *)0xfef5)
#define PWMB_BKR           (*(unsigned char volatile xdata *)0xfef6)
#define PWMB_DTR           (*(unsigned char volatile xdata *)0xfef7)
#define PWMB_OISR          (*(unsigned char volatile xdata *)0xfef8)

```

```

////////////////////////////////////
//FD00H-FDFFH
////////////////////////////////////

```

```

#define P0INTE              (*(unsigned char volatile xdata *)0xfd00)

```

```
#define P1INTE      (*(unsigned char volatile xdata *)0xfd01)
#define P2INTE      (*(unsigned char volatile xdata *)0xfd02)
#define P3INTE      (*(unsigned char volatile xdata *)0xfd03)
#define P4INTE      (*(unsigned char volatile xdata *)0xfd04)
#define P5INTE      (*(unsigned char volatile xdata *)0xfd05)
#define P6INTE      (*(unsigned char volatile xdata *)0xfd06)
#define P7INTE      (*(unsigned char volatile xdata *)0xfd07)
#define P0INTF      (*(unsigned char volatile xdata *)0xfd10)
#define P1INTF      (*(unsigned char volatile xdata *)0xfd11)
#define P2INTF      (*(unsigned char volatile xdata *)0xfd12)
#define P3INTF      (*(unsigned char volatile xdata *)0xfd13)
#define P4INTF      (*(unsigned char volatile xdata *)0xfd14)
#define P5INTF      (*(unsigned char volatile xdata *)0xfd15)
#define P6INTF      (*(unsigned char volatile xdata *)0xfd16)
#define P7INTF      (*(unsigned char volatile xdata *)0xfd17)
#define P0IM0       (*(unsigned char volatile xdata *)0xfd20)
#define P1IM0       (*(unsigned char volatile xdata *)0xfd21)
#define P2IM0       (*(unsigned char volatile xdata *)0xfd22)
#define P3IM0       (*(unsigned char volatile xdata *)0xfd23)
#define P4IM0       (*(unsigned char volatile xdata *)0xfd24)
#define P5IM0       (*(unsigned char volatile xdata *)0xfd25)
#define P6IM0       (*(unsigned char volatile xdata *)0xfd26)
#define P7IM0       (*(unsigned char volatile xdata *)0xfd27)
#define P0IM1       (*(unsigned char volatile xdata *)0xfd30)
#define P1IM1       (*(unsigned char volatile xdata *)0xfd31)
#define P2IM1       (*(unsigned char volatile xdata *)0xfd32)
#define P3IM1       (*(unsigned char volatile xdata *)0xfd33)
#define P4IM1       (*(unsigned char volatile xdata *)0xfd34)
#define P5IM1       (*(unsigned char volatile xdata *)0xfd35)
#define P6IM1       (*(unsigned char volatile xdata *)0xfd36)
#define P7IM1       (*(unsigned char volatile xdata *)0xfd37)
#define P0WKUE      (*(unsigned char volatile xdata *)0xfd40)
#define P1WKUE      (*(unsigned char volatile xdata *)0xfd41)
#define P2WKUE      (*(unsigned char volatile xdata *)0xfd42)
#define P3WKUE      (*(unsigned char volatile xdata *)0xfd43)
#define P4WKUE      (*(unsigned char volatile xdata *)0xfd44)
#define P5WKUE      (*(unsigned char volatile xdata *)0xfd45)
#define P6WKUE      (*(unsigned char volatile xdata *)0xfd46)
#define P7WKUE      (*(unsigned char volatile xdata *)0xfd47)
#define PIN_IP      (*(unsigned char volatile xdata *)0xfd60)
#define PIN_IPH     (*(unsigned char volatile xdata *)0xfd61)

#define CHIPID0     (*(unsigned char volatile xdata *)0xfde0)
#define CHIPID1     (*(unsigned char volatile xdata *)0xfde1)
#define CHIPID2     (*(unsigned char volatile xdata *)0xfde2)
#define CHIPID3     (*(unsigned char volatile xdata *)0xfde3)
```

```

#define CHIPID4          (*(unsigned char volatile xdata *)0xfde4)
#define CHIPID5          (*(unsigned char volatile xdata *)0xfde5)
#define CHIPID6          (*(unsigned char volatile xdata *)0xfde6)
#define CHIPID7          (*(unsigned char volatile xdata *)0xfde7)
#define CHIPID8          (*(unsigned char volatile xdata *)0xfde8)
#define CHIPID9          (*(unsigned char volatile xdata *)0xfde9)
#define CHIPID10         (*(unsigned char volatile xdata *)0xfdea)
#define CHIPID11         (*(unsigned char volatile xdata *)0xfdeb)
#define CHIPID12         (*(unsigned char volatile xdata *)0xfdec)
#define CHIPID13         (*(unsigned char volatile xdata *)0xfded)
#define CHIPID14         (*(unsigned char volatile xdata *)0xfdee)
#define CHIPID15         (*(unsigned char volatile xdata *)0xfdef)
#define CHIPID16         (*(unsigned char volatile xdata *)0xfdf0)
#define CHIPID17         (*(unsigned char volatile xdata *)0xfdf1)
#define CHIPID18         (*(unsigned char volatile xdata *)0xfdf2)
#define CHIPID19         (*(unsigned char volatile xdata *)0xfdf3)
#define CHIPID20         (*(unsigned char volatile xdata *)0xfdf4)
#define CHIPID21         (*(unsigned char volatile xdata *)0xfdf5)
#define CHIPID22         (*(unsigned char volatile xdata *)0xfdf6)
#define CHIPID23         (*(unsigned char volatile xdata *)0xfdf7)
#define CHIPID24         (*(unsigned char volatile xdata *)0xfdf8)
#define CHIPID25         (*(unsigned char volatile xdata *)0xfdf9)
#define CHIPID26         (*(unsigned char volatile xdata *)0xfdfa)
#define CHIPID27         (*(unsigned char volatile xdata *)0xfdfb)
#define CHIPID28         (*(unsigned char volatile xdata *)0xfdfc)
#define CHIPID29         (*(unsigned char volatile xdata *)0xfdfd)
#define CHIPID30         (*(unsigned char volatile xdata *)0xfdfе)
#define CHIPID31         (*(unsigned char volatile xdata *)0xfdff)

```

```

////////////////////////////////////
//FC00H-FCFFH
////////////////////////////////////

```

```

#define MD3              (*(unsigned char volatile xdata *)0xfcf0)
#define MD2              (*(unsigned char volatile xdata *)0xfcf1)
#define MD1              (*(unsigned char volatile xdata *)0xfcf2)
#define MD0              (*(unsigned char volatile xdata *)0xfcf3)
#define MD5              (*(unsigned char volatile xdata *)0xfcf4)
#define MD4              (*(unsigned char volatile xdata *)0xfcf5)
#define ARCON            (*(unsigned char volatile xdata *)0xfcf6)
#define OPCON            (*(unsigned char volatile xdata *)0xfcf7)

```

```

////////////////////////////////////
//FB00H-FBFFH
////////////////////////////////////

```



```
#define COMEN                (*(unsigned char volatile xdata *)0xfb00)
#define SEGENL                (*(unsigned char volatile xdata *)0xfb01)
#define SEGENH                (*(unsigned char volatile xdata *)0xfb02)
#define LEDCTRL               (*(unsigned char volatile xdata *)0xfb03)
#define LEDCKS                (*(unsigned char volatile xdata *)0xfb04)
#define COM0_DA_L             (*(unsigned char volatile xdata *)0xfb10)
#define COM1_DA_L             (*(unsigned char volatile xdata *)0xfb11)
#define COM2_DA_L             (*(unsigned char volatile xdata *)0xfb12)
#define COM3_DA_L             (*(unsigned char volatile xdata *)0xfb13)
#define COM4_DA_L             (*(unsigned char volatile xdata *)0xfb14)
#define COM5_DA_L             (*(unsigned char volatile xdata *)0xfb15)
#define COM6_DA_L             (*(unsigned char volatile xdata *)0xfb16)
#define COM7_DA_L             (*(unsigned char volatile xdata *)0xfb17)
#define COM0_DA_H             (*(unsigned char volatile xdata *)0xfb18)
#define COM1_DA_H             (*(unsigned char volatile xdata *)0xfb19)
#define COM2_DA_H             (*(unsigned char volatile xdata *)0xfb1a)
#define COM3_DA_H             (*(unsigned char volatile xdata *)0xfb1b)
#define COM4_DA_H             (*(unsigned char volatile xdata *)0xfb1c)
#define COM5_DA_H             (*(unsigned char volatile xdata *)0xfb1d)
#define COM6_DA_H             (*(unsigned char volatile xdata *)0xfb1e)
#define COM7_DA_H             (*(unsigned char volatile xdata *)0xfb1f)
#define COM0_DC_L             (*(unsigned char volatile xdata *)0xfb20)
#define COM1_DC_L             (*(unsigned char volatile xdata *)0xfb21)
#define COM2_DC_L             (*(unsigned char volatile xdata *)0xfb22)
#define COM3_DC_L             (*(unsigned char volatile xdata *)0xfb23)
#define COM4_DC_L             (*(unsigned char volatile xdata *)0xfb24)
#define COM5_DC_L             (*(unsigned char volatile xdata *)0xfb25)
#define COM6_DC_L             (*(unsigned char volatile xdata *)0xfb26)
#define COM7_DC_L             (*(unsigned char volatile xdata *)0xfb27)
#define COM0_DC_H             (*(unsigned char volatile xdata *)0xfb28)
#define COM1_DC_H             (*(unsigned char volatile xdata *)0xfb29)
#define COM2_DC_H             (*(unsigned char volatile xdata *)0xfb2a)
#define COM3_DC_H             (*(unsigned char volatile xdata *)0xfb2b)
#define COM4_DC_H             (*(unsigned char volatile xdata *)0xfb2c)
#define COM5_DC_H             (*(unsigned char volatile xdata *)0xfb2d)
#define COM6_DC_H             (*(unsigned char volatile xdata *)0xfb2e)
#define COM7_DC_H             (*(unsigned char volatile xdata *)0xfb2f)

#define TSCHEN1               (*(unsigned char volatile xdata *)0xfb40)
#define TSCHEN2               (*(unsigned char volatile xdata *)0xfb41)
#define TSCFG1                (*(unsigned char volatile xdata *)0xfb42)
#define TSCFG2                (*(unsigned char volatile xdata *)0xfb43)
#define TSWUTC                (*(unsigned char volatile xdata *)0xfb44)
#define TSCTRL                (*(unsigned char volatile xdata *)0xfb45)
#define TSSTA1                (*(unsigned char volatile xdata *)0xfb46)
#define TSSTA2                (*(unsigned char volatile xdata *)0xfb47)
```

```
#define TSRT (* (unsigned char volatile xdata *)0xfb48)
#define TSDAT (* (unsigned int volatile xdata *)0xfb49)
#define TSDATH (* (unsigned char volatile xdata *)0xfb49)
#define TSDATL (* (unsigned char volatile xdata *)0xfb4A)
#define TSTH00 (* (unsigned int volatile xdata *)0xfb50)
#define TSTH00H (* (unsigned char volatile xdata *)0xfb50)
#define TSTH00L (* (unsigned char volatile xdata *)0xfb51)
#define TSTH01 (* (unsigned int volatile xdata *)0xfb52)
#define TSTH01H (* (unsigned char volatile xdata *)0xfb52)
#define TSTH01L (* (unsigned char volatile xdata *)0xfb53)
#define TSTH02 (* (unsigned int volatile xdata *)0xfb54)
#define TSTH02H (* (unsigned char volatile xdata *)0xfb54)
#define TSTH02L (* (unsigned char volatile xdata *)0xfb55)
#define TSTH03 (* (unsigned int volatile xdata *)0xfb56)
#define TSTH03H (* (unsigned char volatile xdata *)0xfb56)
#define TSTH03L (* (unsigned char volatile xdata *)0xfb57)
#define TSTH04 (* (unsigned int volatile xdata *)0xfb58)
#define TSTH04H (* (unsigned char volatile xdata *)0xfb58)
#define TSTH04L (* (unsigned char volatile xdata *)0xfb59)
#define TSTH05 (* (unsigned int volatile xdata *)0xfb5a)
#define TSTH05H (* (unsigned char volatile xdata *)0xfb5a)
#define TSTH05L (* (unsigned char volatile xdata *)0xfb5b)
#define TSTH06 (* (unsigned int volatile xdata *)0xfb5c)
#define TSTH06H (* (unsigned char volatile xdata *)0xfb5c)
#define TSTH06L (* (unsigned char volatile xdata *)0xfb5d)
#define TSTH07 (* (unsigned int volatile xdata *)0xfb5e)
#define TSTH07H (* (unsigned char volatile xdata *)0xfb5e)
#define TSTH07L (* (unsigned char volatile xdata *)0xfb5f)
#define TSTH08 (* (unsigned int volatile xdata *)0xfb60)
#define TSTH08H (* (unsigned char volatile xdata *)0xfb60)
#define TSTH08L (* (unsigned char volatile xdata *)0xfb61)
#define TSTH09 (* (unsigned int volatile xdata *)0xfb62)
#define TSTH09H (* (unsigned char volatile xdata *)0xfb62)
#define TSTH09L (* (unsigned char volatile xdata *)0xfb63)
#define TSTH10 (* (unsigned int volatile xdata *)0xfb64)
#define TSTH10H (* (unsigned char volatile xdata *)0xfb64)
#define TSTH10L (* (unsigned char volatile xdata *)0xfb65)
#define TSTH11 (* (unsigned int volatile xdata *)0xfb66)
#define TSTH11H (* (unsigned char volatile xdata *)0xfb66)
#define TSTH11L (* (unsigned char volatile xdata *)0xfb67)
#define TSTH12 (* (unsigned int volatile xdata *)0xfb68)
#define TSTH12H (* (unsigned char volatile xdata *)0xfb68)
#define TSTH12L (* (unsigned char volatile xdata *)0xfb69)
#define TSTH13 (* (unsigned int volatile xdata *)0xfb6a)
#define TSTH13H (* (unsigned char volatile xdata *)0xfb6a)
#define TSTH13L (* (unsigned char volatile xdata *)0xfb6b)
```

```
#define TSTH14          (*(unsigned int volatile xdata *)0xfb6c)
#define TSTH14H         (*(unsigned char volatile xdata *)0xfb6c)
#define TSTH14L         (*(unsigned char volatile xdata *)0xfb6d)
#define TSTH15          (*(unsigned int volatile xdata *)0xfb6e)
#define TSTH15H         (*(unsigned char volatile xdata *)0xfb6e)
#define TSTH15L         (*(unsigned char volatile xdata *)0xfb6f)

#define LCDCFG          (*(unsigned char volatile xdata *)0xfb80)
#define LCDCFG2         (*(unsigned char volatile xdata *)0xfb81)
#define DBLNTH          (*(unsigned char volatile xdata *)0xfb82)
#define COMLNTHL        (*(unsigned char volatile xdata *)0xfb83)
#define COMLNTHM        (*(unsigned char volatile xdata *)0xfb84)
#define COMLNTHH        (*(unsigned char volatile xdata *)0xfb85)
#define BLINKFRPS       (*(unsigned char volatile xdata *)0xfb86)
#define LCDCR           (*(unsigned char volatile xdata *)0xfb87)
#define COM_ON_A        (*(unsigned char volatile xdata *)0xfb88)
#define COM_ON_B        (*(unsigned char volatile xdata *)0xfb89)
#define SEG_ON_A        (*(unsigned char volatile xdata *)0xfb8a)
#define SEG_ON_B        (*(unsigned char volatile xdata *)0xfb8b)
#define SEG_ON_C        (*(unsigned char volatile xdata *)0xfb8c)
#define SEG_ON_D        (*(unsigned char volatile xdata *)0xfb8d)
#define SEG_ON_E        (*(unsigned char volatile xdata *)0xfb8e)
#define C0SEGV0         (*(unsigned char volatile xdata *)0xfb90)
#define C0SEGV1         (*(unsigned char volatile xdata *)0xfb91)
#define C0SEGV2         (*(unsigned char volatile xdata *)0xfb92)
#define C0SEGV3         (*(unsigned char volatile xdata *)0xfb93)
#define C0SEGV4         (*(unsigned char volatile xdata *)0xfb94)
#define C1SEGV0         (*(unsigned char volatile xdata *)0xfb98)
#define C1SEGV1         (*(unsigned char volatile xdata *)0xfb99)
#define C1SEGV2         (*(unsigned char volatile xdata *)0xfb9a)
#define C1SEGV3         (*(unsigned char volatile xdata *)0xfb9b)
#define C1SEGV4         (*(unsigned char volatile xdata *)0xfb9c)
#define C2SEGV0         (*(unsigned char volatile xdata *)0xfba0)
#define C2SEGV1         (*(unsigned char volatile xdata *)0xfba1)
#define C2SEGV2         (*(unsigned char volatile xdata *)0xfba2)
#define C2SEGV3         (*(unsigned char volatile xdata *)0xfba3)
#define C2SEGV4         (*(unsigned char volatile xdata *)0xfba4)
#define C3SEGV0         (*(unsigned char volatile xdata *)0xfba8)
#define C3SEGV1         (*(unsigned char volatile xdata *)0xfba9)
#define C3SEGV2         (*(unsigned char volatile xdata *)0xfbaa)
#define C3SEGV3         (*(unsigned char volatile xdata *)0xfbab)
#define C3SEGV4         (*(unsigned char volatile xdata *)0xfbac)
#define HSPWMA_CFG      (*(unsigned char volatile xdata *)0xfbfb)
#define HSPWMA_ADR      (*(unsigned char volatile xdata *)0xfbfb1)
#define HSPWMA_DAT      (*(unsigned char volatile xdata *)0xfbfb2)
#define HSPWMB_CFG      (*(unsigned char volatile xdata *)0xfbfb4)
```

```

#define HSPWMB_ADR      (*(unsigned char volatile xdata *)0xfb5)
#define HSPWMB_DAT      (*(unsigned char volatile xdata *)0xfb6)

////////////////////////////////////
//FA00H-FAFFH
////////////////////////////////////

#define DMA_ADC_CFG      (*(unsigned char volatile xdata *)0xfa10)
#define DMA_ADC_CR       (*(unsigned char volatile xdata *)0xfa11)
#define DMA_ADC_STA      (*(unsigned char volatile xdata *)0xfa12)
#define DMA_ADC_RXA      (*(unsigned int volatile xdata *)0xfa17)
#define DMA_ADC_RXAH     (*(unsigned char volatile xdata *)0xfa17)
#define DMA_ADC_RXAL     (*(unsigned char volatile xdata *)0xfa18)
#define DMA_ADC_CFG2     (*(unsigned char volatile xdata *)0xfa19)
#define DMA_ADC_CHSW0    (*(unsigned char volatile xdata *)0xfa1a)
#define DMA_ADC_CHSW1    (*(unsigned char volatile xdata *)0xfa1b)

#define DMA_SPI_CFG      (*(unsigned char volatile xdata *)0xfa20)
#define DMA_SPI_CR       (*(unsigned char volatile xdata *)0xfa21)
#define DMA_SPI_STA      (*(unsigned char volatile xdata *)0xfa22)
#define DMA_SPI_AMT      (*(unsigned char volatile xdata *)0xfa23)
#define DMA_SPI_DONE     (*(unsigned char volatile xdata *)0xfa24)
#define DMA_SPI_TXA      (*(unsigned int volatile xdata *)0xfa25)
#define DMA_SPI_TXAH     (*(unsigned char volatile xdata *)0xfa25)
#define DMA_SPI_TXAL     (*(unsigned char volatile xdata *)0xfa26)
#define DMA_SPI_RXA      (*(unsigned int volatile xdata *)0xfa27)
#define DMA_SPI_RXAH     (*(unsigned char volatile xdata *)0xfa27)
#define DMA_SPI_RXAL     (*(unsigned char volatile xdata *)0xfa28)
#define DMA_SPI_CFG2     (*(unsigned char volatile xdata *)0xfa29)

#define DMA_UR1T_CFG     (*(unsigned char volatile xdata *)0xfa30)
#define DMA_UR1T_CR      (*(unsigned char volatile xdata *)0xfa31)
#define DMA_UR1T_STA     (*(unsigned char volatile xdata *)0xfa32)
#define DMA_UR1T_AMT     (*(unsigned char volatile xdata *)0xfa33)
#define DMA_UR1T_DONE    (*(unsigned char volatile xdata *)0xfa34)
#define DMA_UR1T_TXA     (*(unsigned int volatile xdata *)0xfa35)
#define DMA_UR1T_TXAH    (*(unsigned char volatile xdata *)0xfa35)
#define DMA_UR1T_TXAL    (*(unsigned char volatile xdata *)0xfa36)
#define DMA_UR1R_CFG     (*(unsigned char volatile xdata *)0xfa38)
#define DMA_UR1R_CR      (*(unsigned char volatile xdata *)0xfa39)
#define DMA_UR1R_STA     (*(unsigned char volatile xdata *)0xfa3a)
#define DMA_UR1R_AMT     (*(unsigned char volatile xdata *)0xfa3b)
#define DMA_UR1R_DONE    (*(unsigned char volatile xdata *)0xfa3c)
#define DMA_UR1R_RXA     (*(unsigned int volatile xdata *)0xfa3d)
#define DMA_UR1R_RXAH    (*(unsigned char volatile xdata *)0xfa3d)
#define DMA_UR1R_RXAL    (*(unsigned char volatile xdata *)0xfa3e)

```

```
#define DMA_UR2T_CFG      (*(unsigned char volatile xdata *)0xfa40)
#define DMA_UR2T_CR       (*(unsigned char volatile xdata *)0xfa41)
#define DMA_UR2T_STA      (*(unsigned char volatile xdata *)0xfa42)
#define DMA_UR2T_AMT      (*(unsigned char volatile xdata *)0xfa43)
#define DMA_UR2T_DONE     (*(unsigned char volatile xdata *)0xfa44)
#define DMA_UR2T_TXA      (*(unsigned int volatile xdata *)0xfa45)
#define DMA_UR2T_TXAH     (*(unsigned char volatile xdata *)0xfa45)
#define DMA_UR2T_TXAL     (*(unsigned char volatile xdata *)0xfa46)
#define DMA_UR2R_CFG      (*(unsigned char volatile xdata *)0xfa48)
#define DMA_UR2R_CR       (*(unsigned char volatile xdata *)0xfa49)
#define DMA_UR2R_STA      (*(unsigned char volatile xdata *)0xfa4a)
#define DMA_UR2R_AMT      (*(unsigned char volatile xdata *)0xfa4b)
#define DMA_UR2R_DONE     (*(unsigned char volatile xdata *)0xfa4c)
#define DMA_UR2R_RXA      (*(unsigned int volatile xdata *)0xfa4d)
#define DMA_UR2R_RXAH     (*(unsigned char volatile xdata *)0xfa4d)
#define DMA_UR2R_RXAL     (*(unsigned char volatile xdata *)0xfa4e)

#define DMA_UR3T_CFG      (*(unsigned char volatile xdata *)0xfa50)
#define DMA_UR3T_CR       (*(unsigned char volatile xdata *)0xfa51)
#define DMA_UR3T_STA      (*(unsigned char volatile xdata *)0xfa52)
#define DMA_UR3T_AMT      (*(unsigned char volatile xdata *)0xfa53)
#define DMA_UR3T_DONE     (*(unsigned char volatile xdata *)0xfa54)
#define DMA_UR3T_TXA      (*(unsigned int volatile xdata *)0xfa55)
#define DMA_UR3T_TXAH     (*(unsigned char volatile xdata *)0xfa55)
#define DMA_UR3T_TXAL     (*(unsigned char volatile xdata *)0xfa56)
#define DMA_UR3R_CFG      (*(unsigned char volatile xdata *)0xfa58)
#define DMA_UR3R_CR       (*(unsigned char volatile xdata *)0xfa59)
#define DMA_UR3R_STA      (*(unsigned char volatile xdata *)0xfa5a)
#define DMA_UR3R_AMT      (*(unsigned char volatile xdata *)0xfa5b)
#define DMA_UR3R_DONE     (*(unsigned char volatile xdata *)0xfa5c)
#define DMA_UR3R_RXA      (*(unsigned int volatile xdata *)0xfa5d)
#define DMA_UR3R_RXAH     (*(unsigned char volatile xdata *)0xfa5d)
#define DMA_UR3R_RXAL     (*(unsigned char volatile xdata *)0xfa5e)

#define DMA_UR4T_CFG      (*(unsigned char volatile xdata *)0xfa60)
#define DMA_UR4T_CR       (*(unsigned char volatile xdata *)0xfa61)
#define DMA_UR4T_STA      (*(unsigned char volatile xdata *)0xfa62)
#define DMA_UR4T_AMT      (*(unsigned char volatile xdata *)0xfa63)
#define DMA_UR4T_DONE     (*(unsigned char volatile xdata *)0xfa64)
#define DMA_UR4T_TXA      (*(unsigned int volatile xdata *)0xfa65)
#define DMA_UR4T_TXAH     (*(unsigned char volatile xdata *)0xfa65)
#define DMA_UR4T_TXAL     (*(unsigned char volatile xdata *)0xfa66)
#define DMA_UR4R_CFG      (*(unsigned char volatile xdata *)0xfa68)
#define DMA_UR4R_CR       (*(unsigned char volatile xdata *)0xfa69)
#define DMA_UR4R_STA      (*(unsigned char volatile xdata *)0xfa6a)
```

```
#define DMA_UR4R_AMT      (*(unsigned char volatile xdata *)0xfa6b)
#define DMA_UR4R_DONE    (*(unsigned char volatile xdata *)0xfa6c)
#define DMA_UR4R_RXA     (*(unsigned int volatile xdata *)0xfa6d)
#define DMA_UR4R_RXAH    (*(unsigned char volatile xdata *)0xfa6d)
#define DMA_UR4R_RXAL    (*(unsigned char volatile xdata *)0xfa6e)
```

```
#define DMA_LCM_CFG      (*(unsigned char volatile xdata *)0xfa70)
#define DMA_LCM_CR       (*(unsigned char volatile xdata *)0xfa71)
#define DMA_LCM_STA      (*(unsigned char volatile xdata *)0xfa72)
#define DMA_LCM_AMT      (*(unsigned char volatile xdata *)0xfa73)
#define DMA_LCM_DONE     (*(unsigned char volatile xdata *)0xfa74)
#define DMA_LCM_TXA      (*(unsigned int volatile xdata *)0xfa75)
#define DMA_LCM_TXAH     (*(unsigned char volatile xdata *)0xfa75)
#define DMA_LCM_TXAL     (*(unsigned char volatile xdata *)0xfa76)
#define DMA_LCM_RXA      (*(unsigned int volatile xdata *)0xfa77)
#define DMA_LCM_RXAH     (*(unsigned char volatile xdata *)0xfa77)
#define DMA_LCM_RXAL     (*(unsigned char volatile xdata *)0xfa78)
```

```
////////////////////////////////////
```

```
#endif
```

## 附录EE 单片机是否可以提供裸芯

Q: 单片机是否可以提供裸芯?

A: 暂不提供裸芯。若需要芯片面积小, 可使用用 DFN8、QFN20、QFN32、QFN48 等小体积封装

STC MCU



# 附录FF STC8H 系列单片机取代 STC15 系列的注意事项

## ■ 单片机指令

STC8H 系列的指令码与 STC15 系列是完全一致的, 所以 STC15 系列的代码移植到 STC8H 上, 运行依然正确, 但 **STC8H 系列的指令速度比 STC15 系列要快**, STC15 系列的指令系统属于 STC-Y5 系列指令, 而 STC8H 系列的指令属于 STC-Y6 系列指令, STC-Y6 系列的大部分指令执行都只需要一个 CPU 时钟。如果用户代码中有指令延时的代码, 则需要进行调整。有个每条指令的对比可参考 STC 下载软件的指令表, 如下图:

| 助记符            | 长度 | 时钟 (STC-Y6) | 时钟 (STC-Y5) | 时钟 (STC-Y3) |
|----------------|----|-------------|-------------|-------------|
| ADD A, Rn      | 1  | 1           | 1           | 2           |
| ADD A, direct  | 2  | 1           | 2           | 3           |
| ADD A, @Ri     | 1  | 1           | 2           | 3           |
| ADD A, #data   | 2  | 1           | 2           | 2           |
| ADDC A, Rn     | 1  | 1           | 1           | 2           |
| ADDC A, direct | 2  | 1           | 2           | 3           |
| ADDC A, @Ri    | 1  | 1           | 2           | 3           |
| ADDC A, #data  | 2  | 1           | 2           | 2           |
| SUBB A, Rn     | 1  | 1           | 1           | 2           |
| SUBB A, direct | 2  | 1           | 2           | 3           |
| SUBB A, @Ri    | 1  | 1           | 2           | 3           |
| SUBB A, #data  | 2  | 1           | 2           | 2           |
| INC A          | 1  | 1           | 1           | 2           |
| INC Rn         | 1  | 1           | 2           | 3           |
| INC direct     | 2  | 1           | 3           | 4           |
| INC @Ri        | 1  | 1           | 3           | 4           |

## ■ I/O 口

STC8H 系列单片机上电后, I/O 的模式与 STC15 系列不一样。STC15 系列单片机所有 I/O 口上电后都是 8051 的准双向口模式, 而 **STC8H 系列单片机的 I/O 中, 除了 ISP 下载脚 P3.0/P3.1 为准双向口模式外, 其余的所有 I/O 口在上电后都是高阻输入模式**。传统的 8051 和 STC 的 15 系列单片机上电后即为准双向口模式并输出高电平, 经常有客户的系统中使用 I/O 驱动马达或者 LED 灯, 因此会出现单片机上电的瞬间马达会动一下或者 LED 会闪一下。STC8H 系列的 I/O 上电后为高阻输入模式, 就可避免马达和 LED 的这种误动作。



由于 STC8H 系列单片机的 I/O 中, 除了 ISP 下载脚 P3.0/P3.1 为准双向口模式外, 其余的所有 I/O 口在上电后都是高阻输入模式, 所以当用户需要 STC8H 系列的 I/O 口向外输出信号前, 必须先使用 PxM0 和 PxM1 两个寄存器对 I/O 的工作模式进行设置。

## ■ 复位脚

STC8H 系列和 STC15 系列的 P5.4 口一般情况下是当作普通 I/O 口使用的, 当用户在 ISP 下载时设置了 P5.4 为复位脚功能时, P5.4 口则为单片机的复位脚 (RESET 脚)。对于 STC15H 系列, 复位脚为高电平时单片机处于复位状态, 低电平时单片机解除复位状态。而 STC8 系列与 STC15H 系列的复位电平是向反的, 即对于 STC8H 系列, 复位脚为低电平时, 单片机处于复位状态, 高电平时单片机解除复位状态。

所以当用户使能 P5.4 口的复位脚功能是需要注意复位电平的问题。

## ■ ADC

STC8H 系列和 STC15 系列的 ADC\_CONTR、ADC\_RES、ADC\_RES13 个寄存器地址相同的。但 STC8H 系列另外新增加了两个寄存器: ADCCFG 和 ADCTIM。

STC15 系列开始 ADC 转换位 ADC\_START 位于寄存器 ADC\_CONTR 的 BIT3, 而 STC8H 系列的位于 ADC\_CONTR 的 BIT6

STC15 系列 ADC 转换完成标志位 ADC\_FLAG 位于寄存器 ADC\_CONTR 的 BIT4, 而 STC8H 系列的位于 ADC\_CONTR 的 BIT5

STC15 系列 ADC 速度控制为 ADC\_SPEED 位于寄存器 ADC\_CONTR 的 BIT6-BIT5, 而 STC8H 系列的位于 ADCCFG 的 BIT3-BIT0

STC15 系列 ADC 转换结果的对齐控制位 ADRJ 位于寄存器 CLK\_DIV 的 BIT5, 而 STC8H 系列的对齐控制位 RESFMT 位于 ADCCFG 的 BIT5

STC8H 系列新增了更为精准的 ADC 转换时序控制机制, 通过寄存器 ADCTIM 进行设置

## ■ EEPROM

STC15 系列的 EEPROM 擦除和编程的等待时间用寄存器 IAP\_CONTR 的 Bit2-Bit0 设置, 设置的只是一个大概的频率范围值, STC8H 系列新增了一个寄存器 IAP\_TPS (SFR 地址: 0F5H), 专用于设置 EEPROM 擦除和编程的等待时间, 且用户不需要去计算, 只需要根据当前 CPU 的工作频率, 直接填入 IAP\_TPS 即可, 硬件会自动计算等待时间。(比如: 当前 CPU 的工作频率为 24MHz, 则只需要向 IAP\_TPS 填入 24 即可)

# 附录GGSTC8H 系列单片机取代 STC8A/8F 系列的注意事项

## ■ I/O 口

STC8H 系列单片机上电后, I/O 的模式与 STC8A/8F 系列不一样。STC8A/8F 系列单片机所有 I/O 口上电后都是 8051 的准双向口模式, 而 STC8H 系列单片机的 I/O 中, 除了 ISP 下载脚 P3.0/P3.1 为准双向口模式外, 其余的所有 I/O 口在上电后都是高阻输入模式。传统的 8051 和 STC 的 15/8A/8F 系列单片机上电后即为准双向口模式并输出高电平, 经常有客户的系统中使用 I/O 驱动马达或者 LED 灯, 因此会出现单片机上电的瞬间马达会动一下或者 LED 会闪一下。STC8H 系列的 I/O 上电后为高阻输入模式, 就可避免马达和 LED 的这种误动作。

由于 STC8H 系列单片机的 I/O 中, 除了 ISP 下载脚 P3.0/P3.1 为准双向口模式外, 其余的所有 I/O 口在上电后都是高阻输入模式, 所以当用户需要 STC8H 系列的 I/O 口向外输出信号前, 必须先使用 PxM0 和 PxM1 两个寄存器对 I/O 的工作模式进行设置。

## ■ 复位脚

STC8H 系列和 STC8A/8F 系列的 P5.4 口一般情况下是当作普通 I/O 口使用的, 当用户在 ISP 下载时设置了 P5.4 为复位脚功能时, P5.4 口则为单片机的复位脚 (RESET 脚)。对于 STC8A/8F 系列, 复位脚为高电平时单片机处于复位状态, 低电平时单片机解除复位状态。而 STC8H 系列与 STC8A/8F 系列的复位电平是向反的, 即对于 STC8H 系列, 复位脚为低电平时, 单片机处于复位状态, 高电平时单片机解除复位状态。

所以当用户使能 P5.4 口的复位脚功能是需要注意复位电平的问题。

## ■ EEPROM

STC8A/8F 系列的 EEPROM 擦除和编程的等待时间用寄存器 IAP\_CONTR 的 Bit2-Bit0 设置, 设置的只是一个大概的频率范围值, STC8H 系列新增了一个寄存器 IAP\_TPS (SFR 地址: 0F5H), 专用于设置 EEPROM 擦除和编程的等待时间, 且用户不需要去计算, 只需要根据当前 CPU 的工作频率, 直接填入 IAP\_TPS 即可, 硬件会自动计算等待时间。(比如: 当前 CPU 的工作频率为 24MHz, 则只需要向 IAP\_TPS 填入 24 即可)

# 附录HH 内部测试型号

## HH.1 STC8H2K64T-35I-LQFP48/QFN48

### HH.1.1 特性及价格

➤ 选型价格（不需要外部晶振、不需要外部复位，12 位 ADC，15 通道）

| 2020 年新品供货信息                              |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
|---|------------------|-----|-----|----|---|-----|----|---|---|---|---|---|---|---|---|---|---|---|---|------|---|---|---|-----|---|---|---|---|---|---|--|--|
| 价格及封装                                     | QFN48 <6mm*6mm>  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
|   | LQFP48 <9mm*9mm> |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 本身就可在线仿真                                  |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 支持软件模拟硬件 USB 直接下载                         |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 支持 RS485 下载                               |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 可设置下次更新程序需口令                              |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 程序加密后传输（防拦截）                              |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 可对外输出时钟及复位                                |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 内部高精度时钟（35MHz 以下可调）追频                     |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 内部高可靠复位（可选复位门檻电压）                         |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 看门狗 复位定时器                                 |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 内部低压检测中断并可掉电唤醒                            |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 比较器（可当 1 路 A/D <sup>*</sup> 可作外部掉电检测）     |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 15 路高速 ADC（8 路 PWM 可当 8 路 D/A 使用）         |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 掉电唤醒专用定时器                                 |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 16 位高级 PWM 定时器 互补对称死区控制                   |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 定时器/计数器（T0/T1/T2/T3/T4 外部管脚也可掉电唤醒）        |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| MDU16 硬件 16 位乘法器                          |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| RTC 实时时钟                                  |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| LED 驱动                                    |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 触摸按键                                      |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| I <sup>2</sup> C 并可掉电唤醒                   |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| SPI 并可掉电唤醒                                |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 串口并可掉电唤醒                                  |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 所有的 I/O 口均支持中断并可掉电唤醒                      |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 传统 I/O 中断（INT0/INT1/INT2/INT3/INT4）并可掉电唤醒 |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| I/O 口最多数量                                 |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| EEPROM 10 万次 字节                           |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 强大的双 DPTR 可增可减                            |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| xdata <sup>*</sup> 内部大容量扩展 SRAM 字节        |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| idata <sup>*</sup> 内部传统 8051 RAM 字节       |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| Flash 程序存储器 10 万次 字节                      |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 工作电压（V）                                   |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| 单片机型号                                     |                  |     |     |    |   |     |    |   |   |   |   |   |   |   |   |   |   |   |   |      |   |   |   |     |   |   |   |   |   |   |  |  |
| STC8H2K32T                                | 1.9-5.5          | 32K | 256 | 2K | 2 | 32K | 44 | 有 | 有 | 4 | 有 | 有 | 有 | 有 | 有 | 有 | 5 | 8 | 有 | 12 位 | 有 | 有 | 有 | 4 级 | 有 | 是 | 有 | 是 | 是 | 是 |  |  |
| STC8H2K60T                                | 1.9-5.5          | 60K | 256 | 2K | 2 | 4K  | 44 | 有 | 有 | 4 | 有 | 有 | 有 | 有 | 有 | 有 | 5 | 8 | 有 | 12 位 | 有 | 有 | 有 | 4 级 | 有 | 是 | 有 | 是 | 是 | 是 |  |  |
| STC8H2K64T                                | 1.9-5.5          | 64K | 256 | 2K | 2 | IAP | 44 | 有 | 有 | 4 | 有 | 有 | 有 | 有 | 有 | 有 | 5 | 8 | 有 | 12 位 | 有 | 有 | 有 | 4 级 | 有 | 是 | 有 | 是 | 是 | 是 |  |  |

➤ 内核

- ✓ 超高速 8051 内核 (1T)，比传统 8051 约快 12 倍以上
- ✓ 指令代码完全兼容传统 8051
- ✓ 29 个中断源，4 级中断优先级
- ✓ 支持在线仿真

➤ 工作电压

- ✓ 1.9V~5.5V

➤ 工作温度

- ✓ -40℃~85℃ (芯片为-40℃~125℃制程，超温度范围应用请参考电气特性章节说明)

➤ Flash 存储器

- ✓ 最大 64K 字节 FLASH 程序存储器 (ROM)，用于存储用户代码
- ✓ 支持用户配置 EEPROM 大小，512 字节单页擦除，擦写次数可达 10 万次以上
- ✓ 支持在系统编程方式 (ISP) 更新用户应用程序，无需专用编程器
- ✓ 支持单芯片仿真，无需专用仿真器，理论断点个数无限制

### ➤ SRAM

- ✓ 128 字节内部直接访问 RAM (DATA, C 语言程序中使用 data 关键字进行声明)
- ✓ 128 字节内部间接访问 RAM (IDATA, C 语言程序中使用 idata 关键字进行声明)
- ✓ 2048 字节内部扩展 RAM (内部 XDATA, C 语言程序中使用 xdata 关键字进行声明)

### ➤ 时钟控制

- ✓ 内部高精度 IRC (4MHz~45MHz, ISP 编程时选择或手动输入, 还可以用户软件分频到较低的频率工作, 如 100KHz)
  - ⊕ 误差±0.3% (常温下 25℃)
  - ⊕ -1.35%~+1.30%温漂 (全温度范围, -40℃~85℃)
  - ⊕ -0.76%~+0.98%温漂 (温度范围, -20℃~65℃)
- ✓ 内部 32KHz 低速 IRC (误差较大)
- ✓ 外部晶振 (4MHz~45MHz) 和外部时钟  
用户可自由选择上面的 3 种时钟源

### ➤ 复位

- ✓ 硬件复位
  - ⊕ 上电复位, 实测电压值为 1.69V~1.82V。 (**在芯片未使能低压复位功能时有效**)  
上电复位电压由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到上电复位的下限门槛电压时, 芯片处于复位状态; 当电压从 0V 上升到上电复位的上限门槛电压时, 芯片解除复位状态。
  - ⊕ 复位脚复位, 出厂时 P5.4 默认为 I/O 口, ISP 下载时可将 P5.4 管脚设置为复位脚 (**注意: 当设置 P5.4 管脚为复位脚时, 复位电平为低电平**)
  - ⊕ 看门狗溢出复位
  - ⊕ 低压检测复位, 提供 4 级低压检测电压: 2.0V (实测为 1.90V~2.04V)、2.4V (实测为 2.30V~2.50V)、2.7V (实测为 2.61V~2.82V)、3.0V (实测为 2.90V~3.13V)。  
每级低压检测电压都是由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到低压检测的下限门槛电压时, 低压检测生效; 当电压从 0V 上升到低压检测的上限门槛电压时, 低压检测生效。
- ✓ 软件复位
  - ⊕ 软件方式写复位触发寄存器

### ➤ 中断

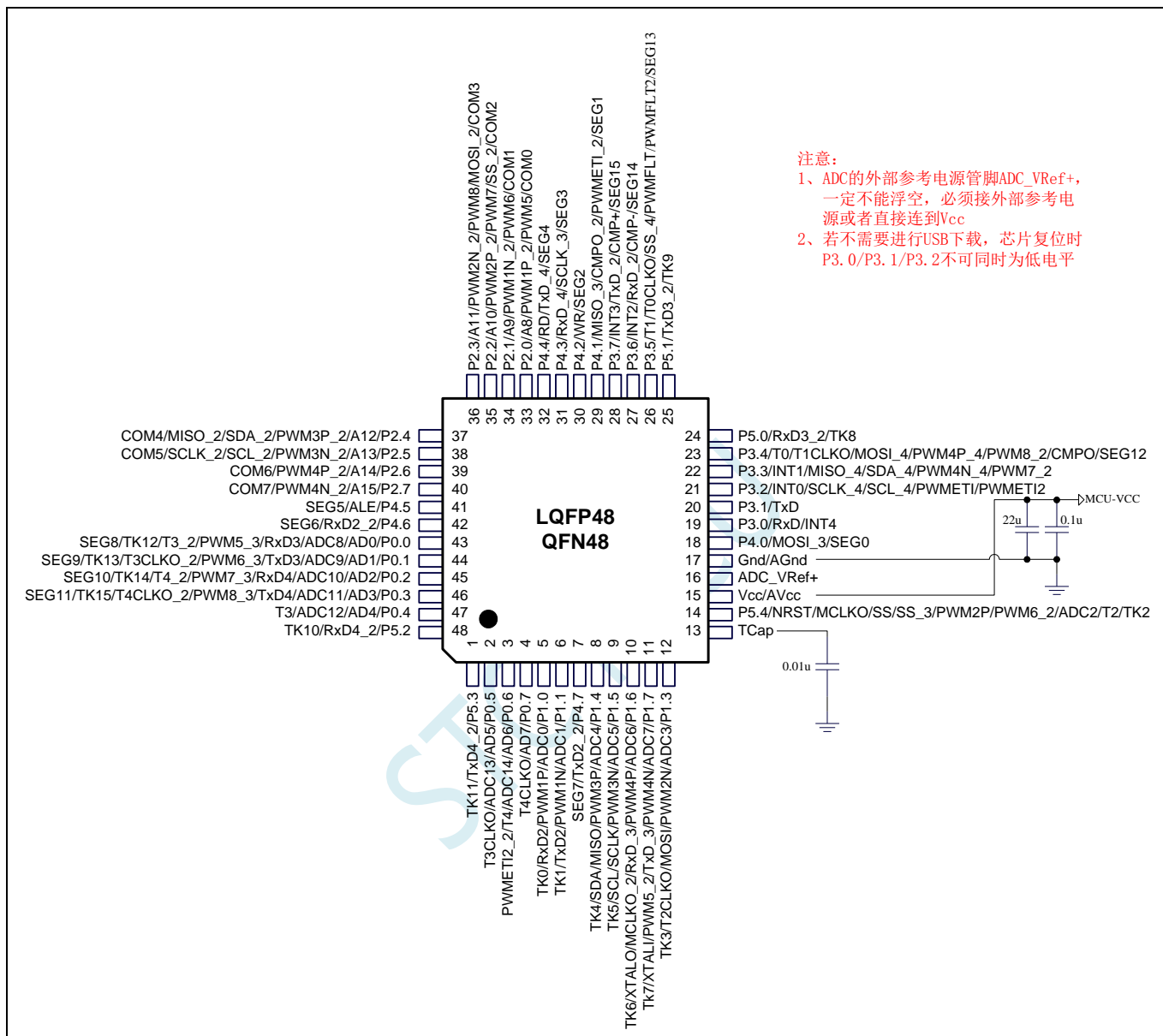
- ✓ 提供 29 个中断源: INT0 (支持上升沿和下降沿中断)、INT1 (支持上升沿和下降沿中断)、INT2 (只支持下降沿中断)、INT3 (只支持下降沿中断)、INT4 (只支持下降沿中断)、定时器 0、定时器 1、定时器 2、定时器 3、定时器 4、串口 1、串口 2、串口 3、串口 4、ADC 模数转换、LVD 低压检测、SPI、I<sup>2</sup>C、比较器、PWMA、PWMB、**RTC、TKS、EXP0、EXP1、EXP2、EXP3、EXP4、EXP5**
- ✓ 提供 4 级中断优先级
- ✓ 时钟停振模式下可以唤醒的中断: INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、T3(P0.4)、T4(P0.6)、RXD(P3.0/P3.6/P1.6/P4.3)、RXD2(P1.0/P4.6)、RXD3(P0.0/P5.0)、RXD4(P0.2/P5.2)、I2C\_SDA(P1.4/P2.4/P3.3)以及比较器中断、低压检测中断、掉电唤醒定时器唤醒。

### ➤ 数字外设

- ✓ 5 个 16 位定时器: 定时器 0、定时器 1、定时器 2、定时器 3、定时器 4, 其中定时器 0 的模式 3 具有 NMI (不可屏蔽中断) 功能, 定时器 0 和定时器 1 的模式 0 为 16 位自动重载模式
- ✓ 4 个高速串口: 串口 1、串口 2、串口 3、串口 4, 波特率时钟源最快可为 FOSC/4
- ✓ **8 路/2 组高级 PWM, 可实现带死区的控制信号, 并支持外部异常检测功能, 另外还支持 16 位定时器、8 个外部中断、8 路外部捕获测量脉宽等功能**

- ✓ SPI: 支持主机模式和从机模式以及主机/从机自动切换
  - ✓ I<sup>2</sup>C: 支持主机模式和从机模式
  - ✓ MDU16: 硬件 16 位乘除法器 (支持 32 位除以 16 位、16 位除以 16 位、16 位乘 16 位、数据移位以及数据规格化等运算)
  - ✓ I/O 口中断: 所有的 I/O 均支持中断, 每组 I/O 中断有独立的中断入口地址, 所有的 I/O 中断可支持 4 种中断模式: 高电平中断、低电平中断、上升沿中断、下降沿中断
- 模拟外设
- ✓ 超高速 ADC, 支持 12 位高精度 15 通道 (通道 0~通道 14) 的模数转换, 速度最快能达到 800K (每秒进行 80 万次 ADC 转换)
  - ✓ ADC 的通道 15 用于测试内部 1.19V 参考信号源 (芯片在出厂时, 内部参考信号源已调整为 1.19V)
  - ✓ 比较器, 一组比较器 (比较器的正端可选择 CMP+端口和所有的 ADC 输入端口, 所以比较器可当作多路比较器进行分时复用)
  - ✓ 触摸按键: 最多支持 16 个触摸按键; 每个触摸按键能够独立使能; 内部参考电压 4 级可调; 灵活的充放电时间设置以及内部工作频率设置; 支持低功耗触摸唤醒
  - ✓ LED 驱动: 最多可驱动 256 (8\*16\*2) 个 LED; 可支持共阴模式、共阳模式以及共阴/共阳模式; 支持 8 级灰度调节 (亮度调节)
  - ✓ DAC: 8 路高级 PWM 定时器可当 8 路 DAC 使用
- GPIO
- ✓ 最多可达 44 个 GPIO: P0.0~P0.7、P1.0~P1.7 (无 P1.2)、P2.0~P2.7、P3.0~P3.7、P4.0~P4.7、P5.0~P5.4
  - ✓ 所有的 GPIO 均支持如下 4 种模式: 准双向口模式、强推挽输出模式、开漏输出模式、高阻输入模式
  - ✓ 除 P3.0 和 P3.1 外, 其余所有 IO 口上电后的状态均为高阻输入状态, 用户在使用 IO 口时必须先设置 IO 口模式。另外每个 I/O 均可独立使能内部 4K 上拉电阻
- 封装
- ✓ LQFP48 <9mm\*9mm>、QFN48 <6mm\*6mm>

## HH.1.2 管脚图，最小系统



## HH.2 STC8H4K64LCD-45I-LQFP64/QFN64/LQFP48/QFN48

### HH.2.1 特性及价格（有 16 位硬件乘除法器 MDU16，准 16 位单片机）

➤ 选型价格（不需要外部晶振、不需要外部复位，12 位 ADC，15 通道）

| 价格及封装                                     |  | 2021 年新品供货信息    |                  |                 |                    |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |  |
|---|--|-----------------|------------------|-----------------|--------------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|--|
|   |  |                 |                  |                 |                    |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |  |
|   |  |                 |                  |                 |                    |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |  |
|   |  | QFN48 <6mm*6mm> | LQFP48 <9mm*9mm> | QFN64 <8mm*8mm> | LQFP64 <12mm*12mm> |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |  |
| 本身就可在线仿真                                  |  | 是               | 是                | 是               | 是                  | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       |  |
| 支持软件模拟硬件 USB 直接下载                         |  | 是               | 是                | 是               | 是                  | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       |  |
| 支持 RS485 下载                               |  | 是               | 是                | 是               | 是                  | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       |  |
| 可设置下次更新程序需口令                              |  | 是               | 是                | 是               | 是                  | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       |  |
| 程序加密后传输（防拦截）                              |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| 可对外输出时钟及复位                                |  | 是               | 是                | 是               | 是                  | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       | 是       |  |
| 内部高精度时钟（45MHz 以下可调）追频                     |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| 内部高可靠复位（可选复位门檻电压）                         |  | 4 级             | 4 级              | 4 级             | 4 级                | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     | 4 级     |  |
| 看门狗 复位定时器                                 |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| 内部低压检测中断并可掉电唤醒                            |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| 比较器（可当 1 路 A/D，可作外部掉电检测）                  |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| DMA 15 路高速 ADC（8 路 PWM 可当 8 路 D/A 使用）     |  | 12 位            | 12 位             | 12 位            | 12 位               | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    | 12 位    |  |
| 掉电唤醒专用定时器                                 |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| 16 位高级 PWM 定时器 互补对称死区控制                   |  | 8               | 8                | 8               | 8                  | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       | 8       |  |
| 定时器计数器（T0/T1/T2/T3/T4 外部引脚也可掉电唤醒）         |  | 5               | 5                | 5               | 5                  | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       | 5       |  |
| MDU16 硬件 16 位乘除法器                         |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| I <sup>2</sup> C 并可掉电唤醒（暂无 DMA）           |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| DMA SPI 并可掉电唤醒                            |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| RTC 实时时钟                                  |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| LCD 液晶驱动（4COM*40SEG）                      |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| DMA 8080/6800 接口 LCM 模块驱动(8 位和 16 位)      |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| DMA 串口并可掉电唤醒                              |  | 4               | 4                | 4               | 4                  | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       | 4       |  |
| 所有的 I/O 口均支持中断并可掉电唤醒                      |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| 传统 I/O 中断（INT0/INT1/INT2/INT3/INT4）并可掉电唤醒 |  | 有               | 有                | 有               | 有                  | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       | 有       |  |
| I/O 口最多数量                                 |  | 61              | 61               | 61              | 61                 | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      | 61      |  |
| EEPROM 10 万次 字节                           |  | 32K             | 4K               | 4K              | 2                  | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       |  |
| 强大的双 DPTR 可增可减                            |  | 2               | 4K               | 4K              | 2                  | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       | 2       |  |
| xdata <sup>*</sup> 内部大容量扩展 SRAM 字节        |  | 4K              | 256              | 256             | 4K                 | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      | 4K      |  |
| idata <sup>*</sup> 内部传统 8051 RAM 字节       |  | 256             | 60K              | 256             | 256                | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     |  |
| Flash 程序存储器 10 万次 字节                      |  | 32K             | 64K              | 256             | 256                | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     | 256     |  |
| 工作电压（V）                                   |  | 1.9-5.5         | 1.9-5.5          | 1.9-5.5         | 1.9-5.5            | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 | 1.9-5.5 |  |
| 单片机型号                                     |  | STC8H4K32LCD    | STC8H4K60LCD     | STC8H4K64LCD    |                    |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |         |  |

#### ➤ 内核

- ✓ 超高速 8051 内核 (1T)，比传统 8051 约快 12 倍以上
- ✓ 指令代码完全兼容传统 8051
- ✓ 42 个中断源，4 级中断优先级
- ✓ 支持在线仿真

#### ➤ 工作电压

- ✓ 1.9V~5.5V

#### ➤ 工作温度

- ✓ -40℃~85℃（芯片为-40℃~125℃制程，超温度范围应用请参考电气特性章节说明）

#### ➤ Flash 存储器

- ✓ 最大 64K 字节 FLASH 程序存储器 (ROM)，用于存储用户代码
- ✓ 支持用户配置 EEPROM 大小，512 字节单页擦除，擦写次数可达 10 万次以上
- ✓ 支持在系统编程方式 (ISP) 更新用户应用程序，无需专用编程器
- ✓ 支持单芯片仿真，无需专用仿真器，理论断点个数无限制

#### ➤ SRAM

- ✓ 128 字节内部直接访问 RAM (DATA，C 语言程序中使用 data 关键字进行声明)



- ✓ 128 字节内部间接访问 RAM (IDATA, C 语言程序中使用 idata 关键字进行声明)
- ✓ 4096 字节内部扩展 RAM (内部 XDATA, C 语言程序中使用 xdata 关键字进行声明)

#### ➤ 时钟控制

- ✓ 内部高精度 IRC (4MHz~45MHz, ISP 编程时选择或手动输入, 还可以用户软件分频到较低的频率工作, 如 100KHz)
  - ⊕ 误差±0.3% (常温下 25℃)
  - ⊕ -1.35%~+1.30%温漂 (全温度范围, -40℃~85℃)
  - ⊕ -0.76%~+0.98%温漂 (温度范围, -20℃~65℃)
- ✓ 内部 32KHz 低速 IRC (误差较大)
- ✓ 外部晶振 (4MHz~45MHz) 和外部时钟  
用户可自由选择上面的 3 种时钟源

#### ➤ 复位

- ✓ 硬件复位
  - ⊕ 上电复位, 实测电压值为 1.69V~1.82V。 (在芯片未使能低压复位功能时有效)  
上电复位电压由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到上电复位的下限门槛电压时, 芯片处于复位状态; 当电压从 0V 上升到上电复位的上限门槛电压时, 芯片解除复位状态。
  - ⊕ 复位脚复位, 出厂时 P5.4 默认为 I/O 口, ISP 下载时可将 P5.4 管脚设置为复位脚 (注意: 当设置 P5.4 管脚为复位脚时, 复位电平为低电平)
  - ⊕ 看门狗溢出复位
  - ⊕ 低压检测复位, 提供 4 级低压检测电压: 1.9V、2.3V、2.8V、3.7V。  
每级低压检测电压都是由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到低压检测的下限门槛电压时, 低压检测生效; 当电压从 0V 上升到低压检测的上限门槛电压时, 低压检测生效。
- ✓ 软件复位
  - ⊕ 软件方式写复位触发寄存器

#### ➤ 中断

- ✓ 提供 42 个中断源: INT0 (支持上升沿和下降沿中断)、INT1 (支持上升沿和下降沿中断)、INT2 (只支持下降沿中断)、INT3 (只支持下降沿中断)、INT4 (只支持下降沿中断)、定时器 0、定时器 1、定时器 2、定时器 3、定时器 4、串口 1、串口 2、串口 3、串口 4、ADC 模数转换、LVD 低压检测、SPI、I<sup>2</sup>C、比较器、PWMA、PWMB、RTC、P1 口中断、P2 口中断、P3 口中断、P4 口中断、P5 口中断、P6 口中断、P7 口中断、LCM 驱动中断、串口 1 的 DMA 接收和发送中断、串口 2 的 DMA 接收和发送中断、串口 3 的 DMA 接收和发送中断、串口 4 的 DMA 接收和发送中断、SPI 的 DMA 中断、ADC 的 DMA 中断、LCD 驱动的 DMA 中断以及存储器到存储器的 DMA 中断。
- ✓ 提供 4 级中断优先级
- ✓ 时钟停振模式下可以唤醒的中断: INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、T3(P0.4)、T4(P0.6)、RXD(P3.0/P3.6/P1.6/P4.3)、RXD2(P1.0/P4.6)、RXD3(P0.0/P5.0)、RXD4(P0.2/P5.2)、I2C\_SDA(P1.4/P2.4/P3.3)、SPI\_SS(P5.4/P2.2/P3.5)以及比较器中断、低压检测中断、掉电唤醒定时器唤醒以及所有端口的 I/O 中断。

#### ➤ 数字外设

- ✓ 5 个 16 位定时器: 定时器 0、定时器 1、定时器 2、定时器 3、定时器 4, 其中定时器 0 的模式 3 具有 NMI (不可屏蔽中断) 功能, 定时器 0 和定时器 1 的模式 0 为 16 位自动重载模式
- ✓ 4 个高速串口: 串口 1、串口 2、串口 3、串口 4, 波特率时钟源最快可为 FOSC/4
- ✓ 8 路/2 组高级 PWM, 可实现带死区的控制信号, 并支持外部异常检测功能, 另外还支持 16 位定时器、8



个外部中断、8 路外部捕获测量脉宽等功能

- ✓ SPI: 支持主机模式和从机模式以及主机/从机自动切换
- ✓ I<sup>2</sup>C: 支持主机模式和从机模式
- ✓ MDU16: 硬件 16 位乘法器 (支持 32 位除以 16 位、16 位除以 16 位、16 位乘 16 位、数据移位以及数据规格化等运算)
- ✓ RTC: 支持年、月、日、时、分、秒、次秒 (1/128 秒), 并支持时钟中断和一组闹钟
- ✓ I/O 口中断: 所有的 I/O 均支持中断, 每组 I/O 中断有独立的中断入口地址, 所有的 I/O 中断可支持 4 种中断模式: 高电平中断、低电平中断、上升沿中断、下降沿中断。提供 4 级中断优先级并支持掉电唤醒功能。
- ✓ DMA: 支持 Memory-To-Memory、SPI、UART1TX/UART1RX、UART2TX/UART2RX、UART3TX/UART3RX、UART4TX/UART4RX、ADC (自动计算多次 ADC 结果的平均值)、LCM
- ✓ LCM (TFT 彩屏) 驱动: 支持 8080 和 6800 接口, 支持 8 位和 16 位数据宽度
  - ⊕ 8 位 8080 总线: 8 为数据线 (TD0~TD7), 读信号 (TRD), 写信号 (TWR), RS 线 (TRS)
  - ⊕ 16 为 8080 总线: 16 为数据线 (TD0~TD15), 读信号 (TRD), 写信号 (TWR), RS 线 (TRS)
  - ⊕ 8 为 6800 总线: 8 为数据线 (TD0~TD7), 使能信号 (TE), 读写信号 (TRW), RS 线 (TRS)
  - ⊕ 16 位 6800 总线: 16 为数据线 (TD0~TD15), 使能信号 (TE), 读写信号 (TRW), RS 线 (TRS)
  - ⊕ 注意: 如果使用 8 位数据线控制 TFT 屏, 一般需要 TD0~D7, TRD/TWR/TRS, 11 根数据及控制线, 外加 2 个普通 I/O 控制片选及复位 (很多 TFT 彩屏的片选及复位厂商已进行了自动处理, 并不需要软件控制)
- ✓ LCD 液晶驱动: 最大支持 4COM\*40SEG; 支持 8 级灰度调节

#### ➤ 模拟外设

- ✓ 超高速 ADC, 支持 12 位高精度 15 通道 (通道 0~通道 14) 的模数转换, 速度最快能达到 800K (每秒进行 80 万次 ADC 转换)
- ✓ ADC 的通道 15 用于测试内部 1.19V 参考信号源 (芯片在出厂时, 内部参考信号源已调整为 1.19V)
- ✓ 比较器, 一组比较器 (比较器的正端可选择 CMP+端口和所有的 ADC 输入端口, 所以比较器可当作多路比较器进行分时复用)
- ✓ DAC: 8 路高级 PWM 定时器可当 8 路 DAC 使用

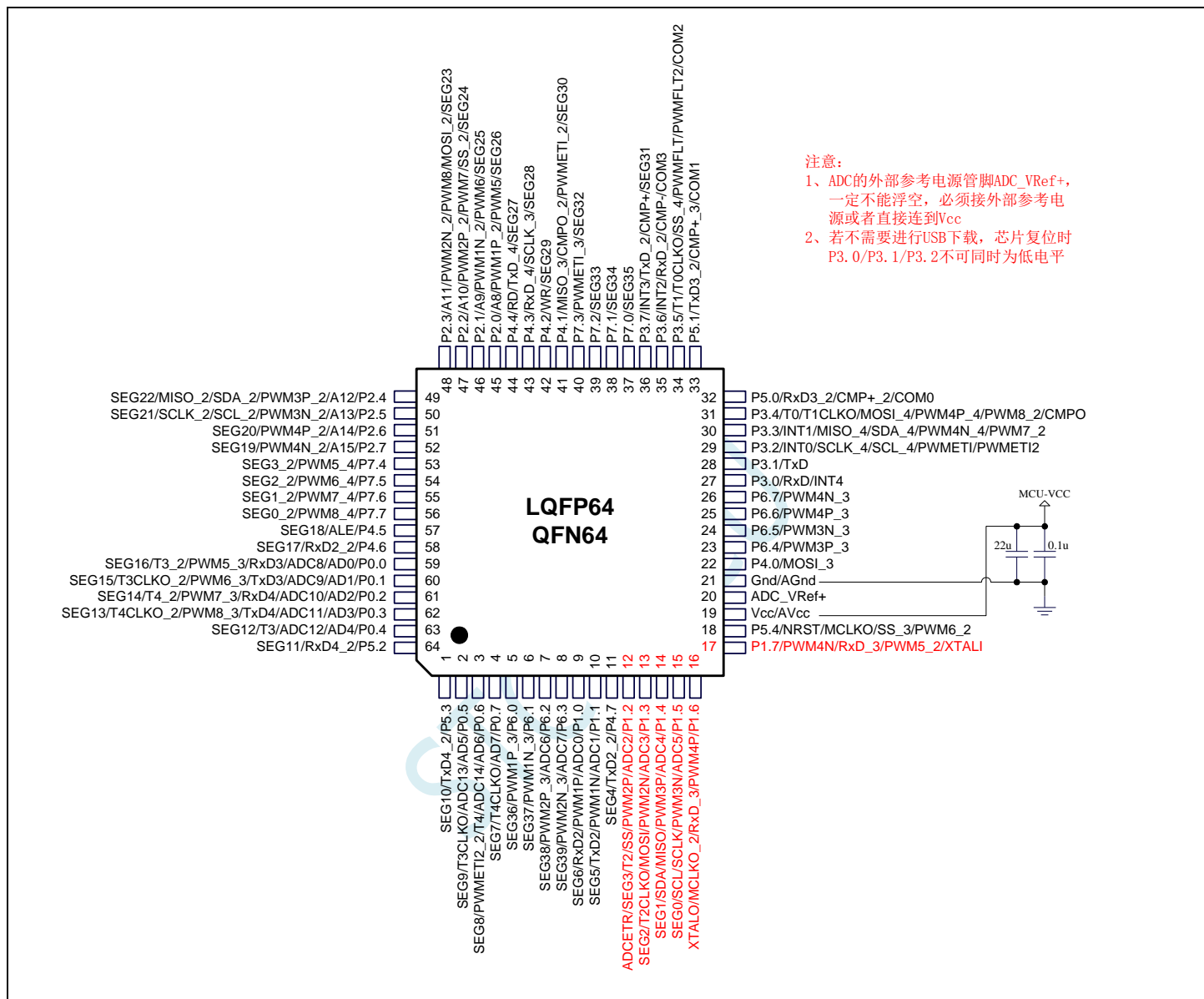
#### ➤ GPIO

- ✓ 最多可达 61 个 GPIO: P0.0~P0.7、P1.0~P1.7、P2.0~P2.7、P3.0~P3.7、P4.0~P4.7、P5.0~P5.4、P6.0~P6.7、P7.0~P7.7
- ✓ 所有的 GPIO 均支持如下 4 种模式: 准双向口模式、强推挽输出模式、开漏输出模式、高阻输入模式
- ✓ 除 P3.0 和 P3.1 外, 其余所有 IO 口上电后的状态均为高阻输入状态, 用户在使用 IO 口时必须先设置 IO 口模式。另外每个 I/O 均可独立使能内部 4K 上拉电阻

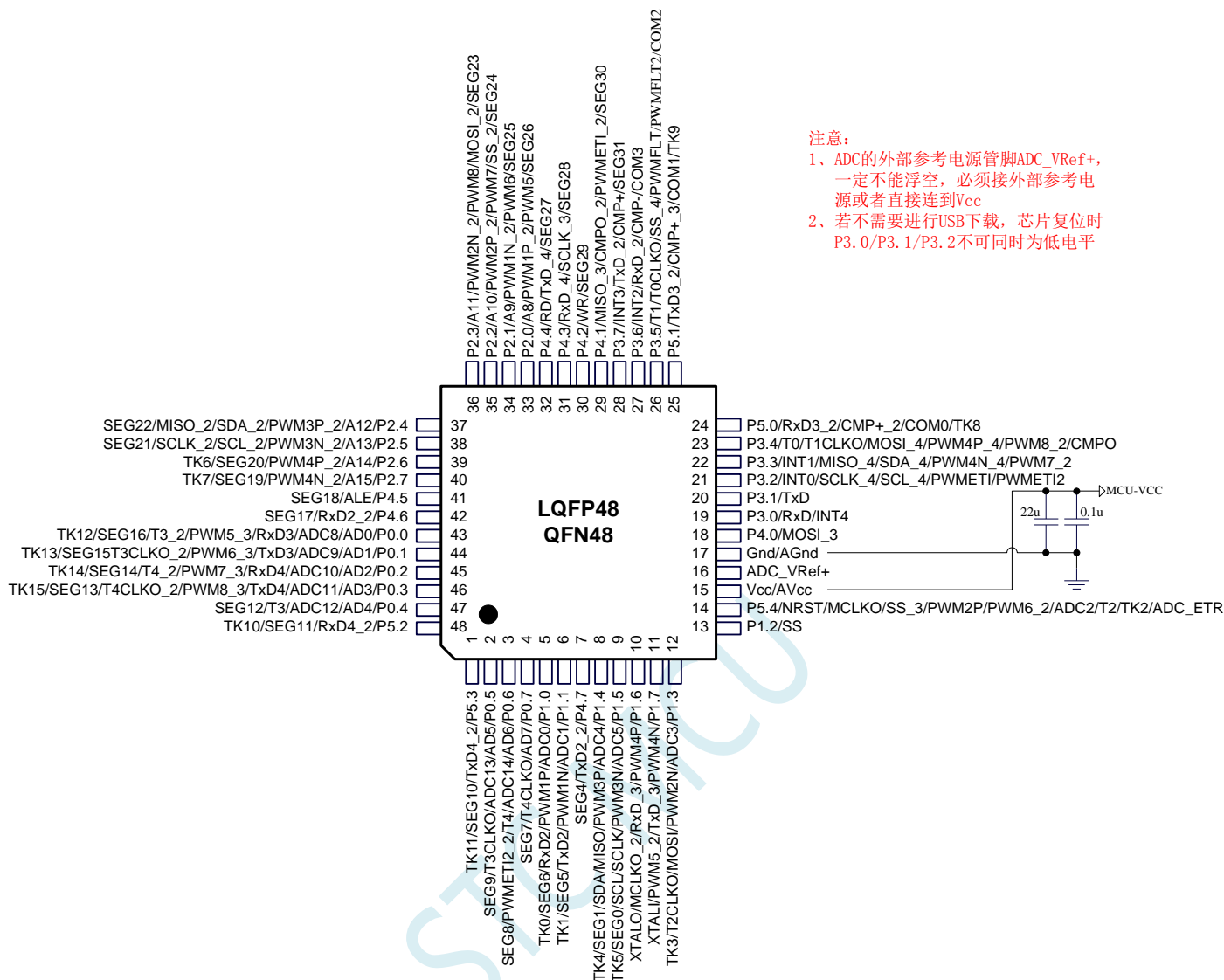
#### ➤ 封装

- ✓ LQFP64 <12mm\*12mm>、QFN64 <8mm\*8mm>、LQFP48 <9mm\*9mm>、QFN48 <6mm\*6mm>

## HH.2.2 管脚图，最小系统



**注意: 上图中 STC8H4K64LCD 的红色部分管脚与 STC8H4K64TLCD 不兼容**





- ✓ 支持在系统编程方式 (ISP) 更新用户应用程序, 无需专用编程器
- ✓ 支持单芯片仿真, 无需专用仿真器, 理论断点个数无限制

➤ **SRAM**

- ✓ 128 字节内部直接访问 RAM (DATA, C 语言程序中使用 data 关键字进行声明)
- ✓ 128 字节内部间接访问 RAM (IDATA, C 语言程序中使用 idata 关键字进行声明)
- ✓ 4096 字节内部扩展 RAM (内部 XDATA, C 语言程序中使用 xdata 关键字进行声明)

➤ **时钟控制**

- ✓ 内部高精度 IRC (4MHz~45MHz, ISP 编程时选择或手动输入, 还可以用户软件分频到较低的频率工作, 如 100KHz)
  - ⊕ 误差±0.3% (常温下 25℃)
  - ⊕ -1.35%~+1.30%温漂 (全温度范围, -40℃~85℃)
  - ⊕ -0.76%~+0.98%温漂 (温度范围, -20℃~65℃)
- ✓ 内部 32KHz 低速 IRC (误差较大)
- ✓ 外部晶振 (4MHz~45MHz) 和外部时钟  
用户可自由选择上面的 3 种时钟源

➤ **复位**

- ✓ 硬件复位
  - ⊕ 上电复位, 实测电压值为 1.69V~1.82V。(在芯片未使能低压复位功能时有效)  
上电复位电压由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到上电复位的下限门槛电压时, 芯片处于复位状态; 当电压从 0V 上升到上电复位的上限门槛电压时, 芯片解除复位状态。
  - ⊕ 复位脚复位, 出厂时 P5.4 默认为 I/O 口, ISP 下载时可将 P5.4 管脚设置为复位脚 (注意: 当设置 P5.4 管脚为复位脚时, 复位电平为低电平)
  - ⊕ 看门狗溢出复位
  - ⊕ 低压检测复位, 提供 4 级低压检测电压: 1.9V、2.3V、2.8V、3.7V。  
每级低压检测电压都是由一个上限电压和一个下限电压组成的电压范围, 当工作电压从 5V/3.3V 向下掉到低压检测的下限门槛电压时, 低压检测生效; 当电压从 0V 上升到低压检测的上限门槛电压时, 低压检测生效。
- ✓ 软件复位
  - ⊕ 软件方式写复位触发寄存器

➤ **中断**

- ✓ 提供 41 个中断源: INT0 (支持上升沿和下降沿中断)、INT1 (支持上升沿和下降沿中断)、INT2 (只支持下降沿中断)、INT3 (只支持下降沿中断)、INT4 (只支持下降沿中断)、定时器 0、定时器 1、定时器 2、定时器 3、定时器 4、串口 1、串口 2、串口 3、串口 4、ADC 模数转换、LVD 低压检测、SPI、I<sup>2</sup>C、比较器、PWMA、PWMB、RTC、TKS、P0 口中断、P1 口中断、P2 口中断、P3 口中断、P4 口中断、P5 口中断、LCM 驱动中断、串口 1 的 DMA 接收和发送中断、串口 2 的 DMA 接收和发送中断、串口 3 的 DMA 接收和发送中断、串口 4 的 DMA 接收和发送中断、SPI 的 DMA 中断、ADC 的 DMA 中断、LCM 驱动的 DMA 中断以及存储器到存储器的 DMA 中断。
- ✓ 提供 4 级中断优先级
- ✓ 时钟停振模式下可以唤醒的中断: INT0(P3.2)、INT1(P3.3)、INT2(P3.6)、INT3(P3.7)、INT4(P3.0)、T0(P3.4)、T1(P3.5)、T2(P1.2)、T3(P0.4)、T4(P0.6)、RXD(P3.0/P3.6/P1.6/P4.3)、RXD2(P1.0/P4.6)、RXD3(P0.0/P5.0)、RXD4(P0.2/P5.2)、I2C\_SDA(P1.4/P2.4/P3.3)、SPI\_SS(P5.4/P2.2/P3.5)以及比较器中断、低压检测中断、掉电唤醒定时器唤醒以及所有端口的 I/O 中断。

## ➤ 数字外设

- ✓ 5 个 16 位定时器: 定时器 0、定时器 1、定时器 2、定时器 3、定时器 4, 其中定时器 0 的模式 3 具有 NMI (不可屏蔽中断) 功能, 定时器 0 和定时器 1 的模式 0 为 16 位自动重载模式
- ✓ 4 个高速串口: 串口 1、串口 2、串口 3、串口 4, 波特率时钟源最快可为 FOSC/4
- ✓ 8 路/2 组高级 PWM, 可实现带死区的控制信号, 并支持外部异常检测功能, 另外还支持 16 位定时器、8 个外部中断、8 路外部捕获测量脉宽等功能
- ✓ SPI: 支持主机模式和从机模式以及主机/从机自动切换
- ✓ I<sup>2</sup>C: 支持主机模式和从机模式
- ✓ MDU16: 硬件 16 位乘法器 (支持 32 位除以 16 位、16 位除以 16 位、16 位乘 16 位、数据移位以及数据规格化等运算)
- ✓ RTC: 支持年、月、日、时、分、秒、次秒 (1/128 秒), 并支持时钟中断和一组闹钟
- ✓ I/O 口中断: 所有的 I/O 均支持中断, 每组 I/O 中断有独立的中断入口地址, 所有的 I/O 中断可支持 4 种中断模式: 高电平中断、低电平中断、上升沿中断、下降沿中断。提供 4 级中断优先级并支持掉电唤醒功能。
- ✓ DMA: 支持 Memory-To-Memory、SPI、UART1TX/UART1RX、UART2TX/UART2RX、UART3TX/UART3RX、UART4TX/UART4RX、ADC (自动计算多次 ADC 结果的平均值)、LCM
- ✓ LCM (TFT 彩屏) 驱动: 支持 8080 和 6800 接口, 支持 8 位和 16 位数据宽度
  - ✦ 8 位 8080 总线: 8 为数据线 (TD0~TD7), 读信号 (TRD), 写信号 (TWR), RS 线 (TRS)
  - ✦ 16 为 8080 总线: 16 为数据线 (TD0~TD15), 读信号 (TRD), 写信号 (TWR), RS 线 (TRS)
  - ✦ 8 为 6800 总线: 8 为数据线 (TD0~TD7), 使能信号 (TE), 读写信号 (TRW), RS 线 (TRS)
  - ✦ 16 位 6800 总线: 16 为数据线 (TD0~TD15), 使能信号 (TE), 读写信号 (TRW), RS 线 (TRS)
  - ✦ 注意: 如果使用 8 位数据线控制 TFT 屏, 一般需要 TD0~D7, TRD/TWR/TRS, 11 根数据及控制线, 外加 2 个普通 I/O 控制片选及复位 (很多 TFT 彩屏的片选及复位厂商已进行了自动处理, 并不需要软件控制)

## ➤ 模拟外设

- ✓ 超高速 ADC, 支持 12 位高精度 15 通道 (通道 0~通道 14) 的模数转换, 速度最快能达到 800K (每秒进行 80 万次 ADC 转换)
- ✓ ADC 的通道 15 用于测试内部 1.19V 参考信号源 (芯片在出厂时, 内部参考信号源已调整为 1.19V)
- ✓ 比较器, 一组比较器 (比较器的正端可选择 CMP+端口和所有的 ADC 输入端口, 所以比较器可当作多路比较器进行分时复用)
- ✓ 触摸按键: 最多支持 16 个触摸按键; 每个触摸按键能够独立使能; 内部参考电压 4 级可调; 灵活的充放电时间设置以及内部工作频率设置; 支持低功耗触摸唤醒
- ✓ LED 驱动: 最多可驱动 256 (8\*16\*2) 个 LED; 可支持共阴模式、共阳模式以及共阴/共阳模式; 支持 8 级灰度调节 (亮度调节)
- ✓ DAC: 8 路高级 PWM 定时器可当 8 路 DAC 使用

## ➤ GPIO

- ✓ 最多可达 44 个 GPIO: P0.0~P0.7、P1.0~P1.7 (无 P1.2)、P2.0~P2.7、P3.0~P3.7、P4.0~P4.7、P5.0~P5.4
- ✓ 所有的 GPIO 均支持如下 4 种模式: 准双向口模式、强推挽输出模式、开漏输出模式、高阻输入模式
- ✓ 除 P3.0 和 P3.1 外, 其余所有 IO 口上电后的状态均为高阻输入状态, 用户在使用 IO 口时必须先设置 IO 口模式。另外每个 I/O 均可独立使能内部 4K 上拉电阻

## ➤ 封装

- ✓ LQFP48 <9mm\*9mm>、QFN48 <6mm\*6mm>

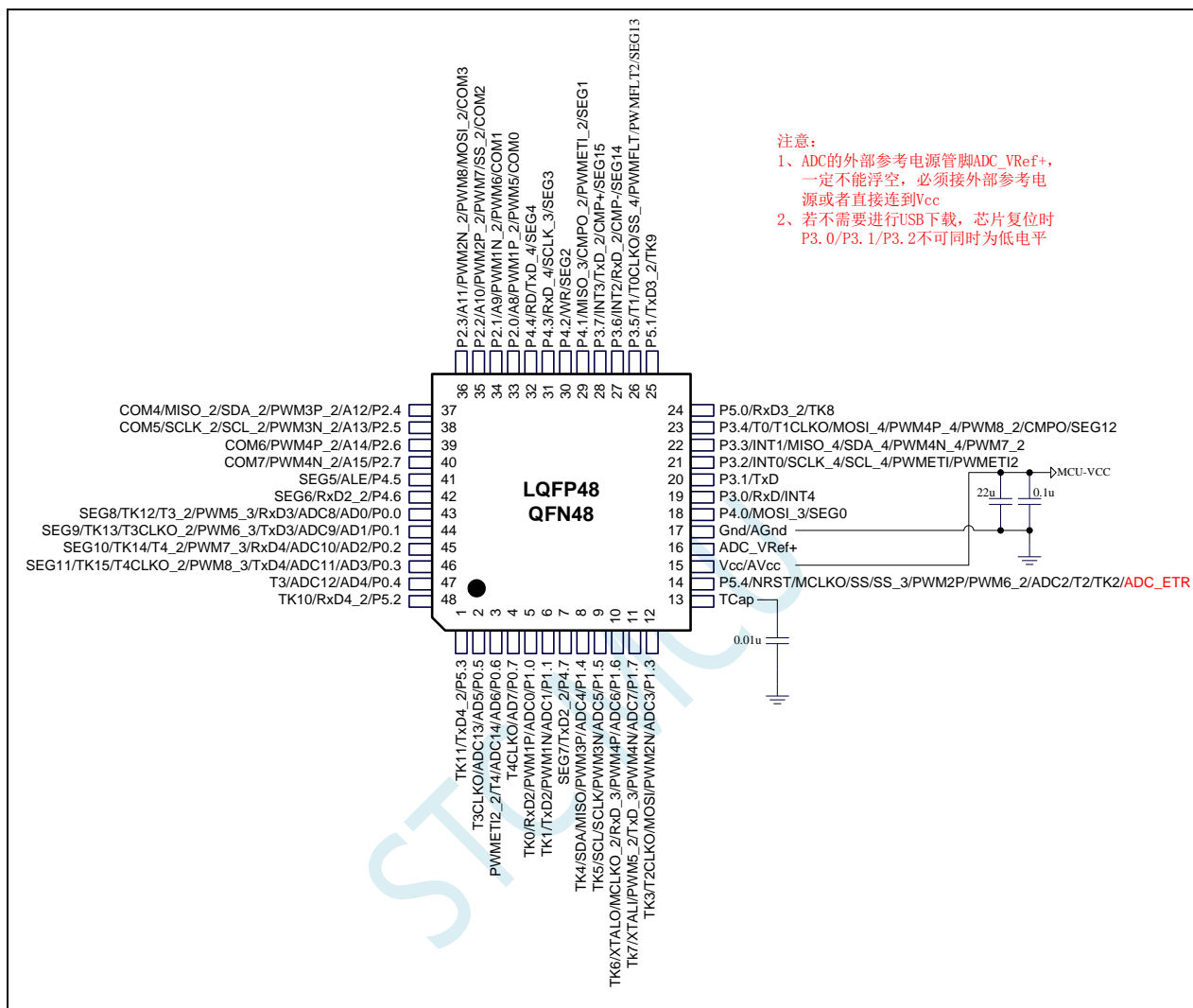
## ✓ 特别说明

- ✓ 本系列芯片有 CHIPID 功能

- ✓ 本系列芯片有所有的 I/O 口中断功能, 有 4 级中断优先级, 可掉电唤醒
- ✓ 本系列芯片的比较器为 4P+2N 版本

STC MCU

## HH.3.2 管脚图，最小系统



正看芯片丝印左下方小圆点处为第一脚

正看芯片丝印最下面一行最后一个字母为芯片版本号

典型下载线路见下一页





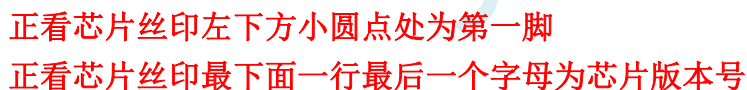
### 通用USB转串口工具（人民币30元）

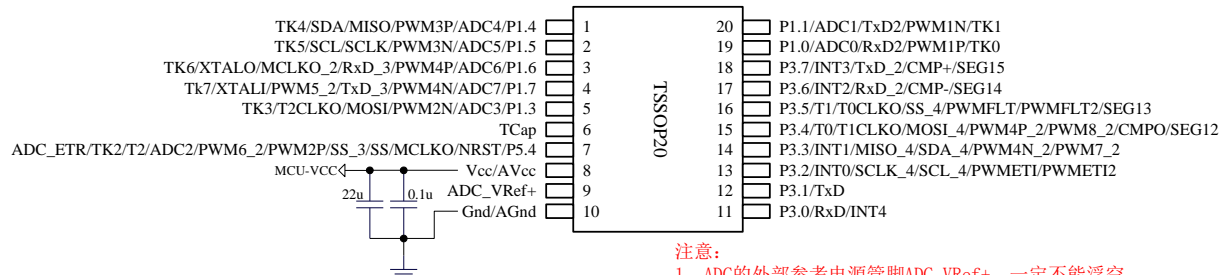
#### ISP下载步骤:

- 1、按照如图所示的连接方式将通用USB转串口工具和目标芯片连接
- 2、按下电源按钮，确定目标芯片处于**停电状态**（上电指示灯为灭的状态）。  
**注意：工具第一次上电时是不对外供电的，因此若是第一次上电使用此工具，可跳过此步。**
- 3、点击STC-ISP下载软件中的“下载/编程”按钮
- 4、再次按下电源按钮，给目标芯片上电（上电指示灯为亮的状态）
- 5、开始ISP下载

**注意：目前有发现使用USB线供电进行ISP下载时，由于USB线太细，在USB线上的压降过大，导致ISP下载时供电不足，所以请在使用USB线供电进行ISP下载时，务必使用USB加强线。**

- 注意：**
- 1、除 P3.0 和 P3.1 外，其余所有 I/O 口上电后的状态均为高阻输入状态，用户在使用 I/O 口时必须先设置 I/O 口模式
  - 2、所有的 I/O 口均可以设置为准双向口模式、强推挽输出模式、开漏输出模式或者高阻输入模式，另外每个 I/O 均可独立使能内部 4K 上拉电阻
  - 3、当使能 P5.4 口为复位脚时，复位电平为低电平





### 通用USB转串口工具 (人民币30元)

#### ISP下载步骤:

- 1、按照如图所示的连接方式将通用USB转串口工具和目标芯片连接
- 2、按下电源按钮, 确定目标芯片处于**停电状态** (上电指示灯为灭的状态)。  
**注意: 工具第一次上电时是不对外供电的, 因此若是第一次上电使用此工具, 可跳过此步。**
- 3、点击STC-ISP下载软件中的“下载/编程”按钮
- 4、再次按下电源按钮, 给目标芯片上电 (上电指示灯为亮的状态)
- 5、开始ISP下载

**注意:** 目前有发现使用USB线供电进行ISP下载时, 由于USB线太细, 在USB线上的压降过大, 导致ISP下载时供电不足, 所以请在使用USB线供电进行ISP下载时, 务必使用USB加强线。

## 附录II 更新记录

### ● 2023/3/1

1. STC8H1K28 系列增加 TSSOP20 管脚图
2. 更新选型价格表
3. RTC 章节增加“内部 RTC 时钟低功耗休眠唤醒-比较器检测电压程序”范例

### ● 2023/1/31

1. 更新 USB 参考线路图中 UCap 脚的电容的参考值
2. 更新 ADC 章节中高精度 ADC 参考线路图
3. RTC 章节增加不停电下载保持 RTC 参数的参考范例
4. 增加 USB-CDC 虚拟串口说明章节

### ● 2023/1/17

1. 增加“用 STC 一箭双雕之 USB 转双串口仿真 STC8 系列 MCU”章节

### ● 2023/1/13

1. 时钟复位章节增加 STC8H8K64U 系列和 STC8H1K08T 系列的时钟树结构图
2. 增加高速高级 PWM 说明章节
3. 增加高速高级 PWM 使用 PLL 高速时钟的范例程序

### ● 2023/1/10

1. 高级 PWM 章节增加系统时钟分频输出范例
2. 更新参考线路图说明

### ● 2022/12/23

1. 更新选型价格表
2. 更新 I2C 从机代码范例程序（提高代码兼容性）
3. 更新 USB 下载线路图中 SIP4 的电源脚的接线网络

### ● 2022/12/19

1. 更正 USB 章节中寄存器错误的名称
2. 更新 USB 参考线路图中 UCap 脚的电容的参考值

## ● 2022/11/24

1. LED 驱动章节增加分时驱动 32 个 8-LED 数码管的参考电路图
2. P5.1.5 章节增加 STC-USB Link1D 全部配线图片
3. 文档首页增加官方论坛网址

## ● 2022/11/14

1. ISP 下载参考线路图中的电容统一建议使用 22uF+0.1uF (104) 的组合

## ● 2022/11/3

1. 附录中增加最小系统电源保护参考电路图
2. 修正部分管脚图中的乱码

## ● 2022/10/31

3. 更新 STC8H8K64U 管脚图中的参考线路图

## ● 2022/10/27

1. 增加 ISP 下载通信协议流程图
2. 增加 STC-USB Link1D 工具配套的连接线的使用说明章节 (P5.1.5)

## ● 2022/10/21

1. 更新 RTC 功耗参数
2. 增加自动追频说明章节
3. 更新管脚图 (增加 USB-TypeA 下载接口示意图)
4. 更新 USB 下载参考线路图
5. 所有串口示例的波特率计算都增加四舍五入处理

## ● 2022/10/12

1. STC8H8K64U 增加 3.3V 的 USB 下载参考线路图
2. 增加 RTC 实战线路图说明
3. 更新部分芯片的实时供货价格

## ● 2022/9/26

1. 高级 PWM 章节增加硬件方式实现脉冲计数的范例程序
2. 增加 STC8H1K08T 系列以及管脚图
3. 更新 STC8H4K64TLCD 管脚图 (增加统计表格)

## ● 2022/9/21

1. 高级 PWM 章节增加“产生 3 路 120 度相位差的 PWM”的范例程序

## ● 2022/9/20

1. 更新选型价格表
2. 增加 STC-ISP 高级应用章节
3. 增加关闭驱动程序强制数字签名说明章节

## ● 2022/9/16

1. 更新官方网址

## ● 2022/9/9

1. 修正 I/O 章节中的笔误
2. I/O 内部结构图中增加保护二极管示意图
3. 各个系列的最小系统下添加“Vcc 和 Gnd 增加去耦电容”的建议
4. ADC 章节和比较器章节增加扩展功能的说明表格

## ● 2022/8/26

1. 将外设的功能脚切换寄存器的描述添加到各个外设章节中
2. 增加 LCD 显示 RTC 时钟的范例程序的链接地址
3. 修正 I2C 章节中部分寄存器的错误描述
4. STC8H8K64U 增加 TSSOP20 封装的 USB 下载线路图
5. 增加 STC8H3K64S2 系列和 STC8H3K64S4 系列的内部 IRC 频率盲区的建议

## ● 2022/8/4

1. 修改 DMA 章节中的部分描述笔误
2. 修正高级 PWM 章节中部分受 LOCK 位影响的 SFR 的描述
3. 更新 FLASH 中 4 频段 IRC 的相关参数
4. 增加用户从 FLASH 中读 4 频段 IRC 参数自定义频率的范例程序
5. 更新使用高精度 ADC 的参考线路图
6. 各章节中去掉 STC8H4K64TLR 系列和 STC8H4K64LCD 系列
7. 更新选型价格表

## ● 2022/7/6

1. 更新选型价格表
2. 修正 ADC\_DMA 通道使能寄存器描述错误
3. 增加复位寄存器 RSTFLAG 中, SWRSTF 寄存器位的使用注意事项

## ● 2022/6/13

1. 附录中增加自动批量生产流程说明

2. 增加使用 STC-ISP 软件制作、编辑 EEPROM 文件的方法
3. 附录中增加 STC8H 的头文件定义
4. 更新定时器外部计数范例程序
5. USB 下载参考线路图中增加 Type-C 接口连线图

## ● 2022/6/6

1. 高级 PWM 章节的捕获范例程序中, 增加原理、范例说明以及注意事项说明
2. 高级 PWM 章节增加同时捕获 4 路信号的周期和占空比的范例程序

## ● 2022/5/31

1. 修正高级 PWM 章节捕获方框图中网络名的错误
2. 增加 PDIP40 封装芯片的 USB 直接下载的参考线路图
3. 修正 STC8H4K64LCD 管脚图
4. 存储器章节增加 EAXFR 寄存器使用说明小节
5. 增加使能 I/O 口内部上拉电阻的范例
6. 更新所有范例程序, 默认包含 STC8H.H 头文件
7. 更新所有范例程序, 初始化时打开 EAXFR

## ● 2022/5/16

1. 修正比较器章节的错别字
2. 更新 USB 章节中端点数据包大小寄存器的描述

## ● 2022/5/6

1. 增加输出占空比为 100% 和 0% 的 PWM 波形的范例程序

## ● 2022/5/5

1. 增加用户程序中若添加了 STARTUP.A51 文件时的仿真说明
2. 更新 STC8K1K28 系列的特性格表 (保留 STC8H1K28, 去掉 STC8H1K33)
3. 在管脚图下面增加通过芯片丝印识别第一脚和辨别芯片版本的方法
4. 增加使用 USB 直接下载时 P3.2 口的说明

## ● 2022/4/22

1. 更新芯片型号以及供货信息
2. 更新串口波特率计算说明
3. 附录中增加“如何使用万用表检测芯片 I/O 口好坏”章节

## ● 2022/4/7

1. 串口章节增加 MODBUS 协议范例程序
2. 高级 PWM 章节增加编码器范例程序

3. 更正 USB 章节寄存器错误描述 (INCSR1)
4. 增加高级 PWM 周期触发 ADC 的范例程序

## ● 2022/3/19

1. 更正: STC8H4K64TL 系列无 CHIPID 功能

## ● 2022/3/18

1. 增加 STC8H4K64TL 系列 (STC8H4K64TL 系列完全兼容 STC8H4K64TLR 系列)

## ● 2022/3/9

1. I/O 章节增加 “使用 I/O 和 R-2R 电阻分压实现 DAC 的经典线路图”
2. 修正 STC8H 系列内部 IRC 频率说明
3. 修正 DMA 章节中寄存器命名错误的问题
4. 更新 BLDC 无刷直流电机驱动线路图

## ● 2022/2/28

1. ADC 章节增加 ADCEXCFG 寄存器的描述
2. 复位章节增加 RSTFLAG 寄存器的描述

## ● 2022/2/18

3. 修改 LED 章节的寄存器地址描述错误部分
4. 修正 DMA 章节中 ADC 数据结构的描述错误部分

## ● 2022/2/8

1. 建议用户代码中不要使用 pdata, 统一使用 xdata 在扩展 RAM 中声明变量。

## ● 2022/1/25

1. 更新文档中有关 TL431/CD431 的参考线路图

## ● 2022/1/20

1. 更新高级 PWM 驱动 BLDC 无刷直流电机 (无 HALL) 的范例程序
2. 增加第三方 MCU 对 STC 芯片进行 ISP 下载时的电源控制参考电路图

## ● 2022/1/7

1. 移除 STC8H2K64T 型号, 后续 STC8H2K64T 会完全由 STC8H4K64TLR 型号替代
2. 增加 STC8H3K64S2 系列 B 版本芯片和 STC8H3K64S4 系列 B 版本芯片的 I/O 口中断问题的应用注意事项



## ● 2021/12/21

1. STC8H8K64U 增加 LQFP32 和 TSSOP20 管脚图
2. 更正文档中的笔误
3. 增加新品单片机预告信息

## ● 2021/12/17

1. 修正 SPI 主模式到从模式自动切换的描述
2. 更新 RTC 范例程序
3. 所有管脚图中复位脚名称修改为 NRST
4. 修正定时器 2/3/4 的定时计算公式
5. 更新选型价格表
6. 更新 STC8H8K64U 的管脚图和管脚描述

## ● 2021/11/23

1. 将 EEPROM 应用范例程序进行重新排版
2. 增加附录“STC 仿真使用说明书”章节
3. 对只读特殊功能寄存器 (CHIPID) 增加详细说明
4. 增加 USB 产品开发中有关 VID 的使用和 PID 的分配问题
5. 增加 STC8H4K64TLR 系列的 LQFP32 和 TSSOP20 的管脚图

## ● 2021/11/16

1. 增加“只读特殊功能寄存器中存储的唯一 ID 号和重要参数”章节
2. 增加从只读特殊功能寄存器中读取重要参数的范例程序
3. 增加“USB 仿真步骤演示”附录章节
4. 增加 EEPROM 应用范例程序

## ● 2021/10/27

1. 更正 LCD 工作电压设置寄存器的描述
2. 更新中断结构图
3. 更正文档中的错别字
4. BMM 全部更名为 DMA
5. 更新 STC8H3K64S2 系列和 STC8H3K64S2 系列的供货信息
6. 增加 STC8H4K64TLR 系统的 RTC 使用外部晶振的应用注意事项

## ● 2021/10/8

1. 更新 LCM 章节中 8 位数据和 16 位数据的端口切换表格
2. 更正文档中已发现的错别字
3. 在中断系统章节的中断源表格中增加外部中断说明
4. 附录中仿真器章节的使用新版本软件进行截图

5. 在附录中增加“如何测试 I/O 口”章节

## ● 2021/9/26

1. 增加串口 DMA 的超时处理和数据校验的范例程序

## ● 2021/8/30

1. 修改部分章节的标题
2. 在附录中增加“关于回流焊前烘烤的注意事项”章节

## ● 2021/8/26

1. 修正 ADC 章节范例程序中的注释错误
2. 高级 PWM 章节增加范例程序

## ● 2021/8/9

1. 修正 STC8H3K64S4-LQFP32 和 STC8H3K64S2-LQFP32 管脚图中的错误
2. 修正 STC8H3K64S4 系列中 ADC 通道数的错误描述

## ● 2021/7/13

1. 比较器章节增加新版比较器的结构图和寄存器说明
2. 删除 STC8H3K64S2 系列的 QFN20 封装脚位图

## ● 2021/7/7

1. 比较器章节添加 STC8H4K 系列比较器使用范例程序
2. RTC 实时时钟章节添加 STC8H4K64TLR 型号芯片 RTC 使用范例程序
3. 触摸按键控制器章节添加范例程序触摸按键配置软件介绍

## ● 2021/7/6

1. 修改部分章节的标题
2. 更新 STC8H3K64S4 系列的 LQFP32 和 QFN32 封装脚位图以及管脚说明
3. 更新 STC8H3K64S2 系列的 LQFP32、QFN32、TSSOP20 和 QFN20 封装脚位图以及管脚说明

## ● 2021/7/5

1. 增加 I/O 口中断掉电唤醒特殊功能寄存器描述
2. 增加 I/O 口中断优先级特殊功能寄存器描述
3. 修改部分章节标题的描述

## ● 2021/6/29

4. 增加 STC8H3K64S4 系列的 LQFP32 和 TSSOP20 封装脚位

## ● 2021/6/26

1. 增加 STC8H3K48S4 型号
2. 增加 STC8H3K48S2 型号
3. 增加 STC8H8K48U 型号
4. 增加 STC8H4K48TLR 型号
5. 增加 STC8H4K48TLCD 型号

## ● 2021/6/10

1. 增加 STC8H4K64TLR 系列以及管脚图
2. 增加 STC8H4K64TLCD 系列以及管脚图
3. 增加 STC8H4K64LCD 系列以及管脚图
4. 增加 LCD 控制说明章节
5. 增加 LCM 控制说明章节
6. 增加 DMA 控制说明章节

## ● 2021/5/10

1. 增加 ADC 电源开关延时说明
2. 增加使用 ADC 第 15 通道反推外部通道输入电压原理说明及计算公式
3. 修改部分系列的最大可用 FLASH 大小的错误描述
4. 增加了定时器 2/3/4 中断标志位的相关说明

## ● 2021/3/8

1. 增加 RTC 参考线路图
2. 增加 STC8H3K64S4 系列和 STC8H3K64S2 系列说明
3. 增加 STC8H4K64R4-48Pin RTC 系列新产品预告
4. 附录中增加 STC8 系列命名花絮

## ● 2021/2/26

1. 增加有关软件模拟硬件 USB 下载的说明
2. 增加定时器 2、定时器 3、定时器 4 的 8 位时钟预分频寄存器说明
3. 触摸按键芯片的触摸按键灵敏度调节电容说明（建议使用独石电容）

## ● 2021/2/4

1. 更正 CLKDIV 寄存器的复位初始值
2. 更新驱动共阴/共阳 LED 参考线路图
3. 增加 STC8H1K24 系列的全球唯一 ID 号以及重要测试参数的存储地址信息
4. 增加特殊功能寄存器初始值说明
5. 修正 P6M1/P7M1 复位初始值

6. 修改 I/O 口章节中 3V/5V 器件混连系统应用中的描述错误
7. 增加晶振滤波控制寄存器说明（仅 STC8H3K64S4-B 版芯片有效）
8. 在管脚图下增加应用参考线路图

## ● 2020/11/25

1. 更新中断结构图
2. 更正文档中的描述错误
3. 增加串口转 LIN 总线范例程序

## ● 2020/10/30

1. 更正部分范例程序中的错误
2. 更正 STC8H3K 系列的 I/O 口数量
3. 增加 STC8H4K64T-48PIN 系列提前预告

## ● 2020/10/16

1. 更新应用注意事项
2. 增加外部晶振电路的负载电容说明

## ● 2020/9/4

1. 修改文档中的部分错别字
2. 定时器章节完善定时器 0/1/2/3/4 内部硬件框图
3. 更正定时器 0 的模式 3 停止说明（定时器 0 的模式 3 为不可屏蔽中断，一旦启动，软件将无法停止，必须芯片复位才能停止）
4. 整理文档章节顺序
5. 典型应用线路图章节中增加“单机电源控制参考电路”
6. 7.3 章“存储器中的特殊参数”的特殊参数列表中，增加 20M、27M、30M、33.1776M、35M 以及 36.864MHz 的 IRC 参数
7. 更新 7.3.7 章“用户自定义内部 IRC 频率”的范例代码

## ● 2020/8/26

1. 增加定时器计算公式章节
2. 增加串口波特率计算公式章节
3. 增加 16 位高级 PWM 输出频率计算公式章节
4. 增加 ADC 相关计算公式章节
5. 增加 12 位 ADC 静态参数参考数据
6. 增加 MDU16 运算所需时钟数参数
7. 增加 EEPROM 操作所需时间参数
8. 将所有章节中特殊功能寄存器都单独列为目录小节，方便查找
9. 增加 STC8H 系列单片机取代 STC8A/8F 系列的注意事项

## ● 2020/8/21

1. 修改文档中的部分描述有误的地方
2. 16 位高级 PWM 定时器章节增加说明
3. 第一组 16 位高级 PWM 定时器 PWM1 改名为 PWMA
4. 第二组 16 位高级 PWM 定时器 PWM2 改名为 PWMB
5. 增加“利用 PWM 实现互补 SPWM”的范例程序

## ● 2020/8/10

1. 增加看门狗定时器章节
2. 整理掉电唤醒定时器章节
3. 增加有关 STC 下载工具使用说明附录章节

## ● 2020/8/6

1. 对工作温度进行说明
2. 增加 16 位高级 PWM 定时器应用于测量周期、占空比、高电平、低电平宽度的示例程序
3. 增加外部 32.768KHz 晶振控制寄存器 X32KCR 的描述
4. 增加使用通用 USB 转串口工具下载的应用线路图
5. 更新应用注意事项

## ● 2020/7/16

1. 增加 BUS\_SPEED 寄存器的说明
2. 增加 QFN/DFN 封装形式芯片的焊接指导
3. 增加 BLDC 无刷直流电机驱动参考电路图(无 HALL)
4. 增加正交编码器模式示例程序
5. 增加成都逐飞科技友情提供的正交解码示例, 见附录 L
6. 增加 EEPROM 编程说明
7. 在下载应用线路图章节中增加设置 U8W/U8-Mini 为直通模式的方法

## ● 2020/7/3

1. 修改文件中部分文字排版乱的问题

## ● 2020/7/2

1. 增加 STC8H2K64T 系列
2. 增加附录章节, “不停电用户程序复位到系统区进行 ISP 下载的方法”
3. 增加附录章节, “使用 STC 的 IAP 系列单片机开发自己的 ISP 程序”
4. 增加附录章节, “STC8H 系列单片机取代 STC15 系列的注意事项”
5. 增加附录章节, “官方网址说明”
6. 增加 LED 驱动说明章节
7. 增加触摸按键说明章节
8. 增加 RTC 实时时钟说明章节

9. ADC 章节中, 增加 ADC 转换时序图

## ● 2020/6/15

1. 增加 ADC\_VRef+管脚说明
2. 增加使用 USB 转串口参考电路中的二极管和电阻的说明
3. 修正 I/O 口驱动电流控制寄存器 PxDR 的描述 (1: 普通驱动电流; 0: 强驱动电流)
4. 增加 I2C 从机设备地址的说明

## ● 2020/6/8

1. 增加 ADC 最快转换速度的说明
2. 详细说明 I<sup>2</sup>C 总线速度设置
3. 更新软件模拟硬件 USB 和硬件 USB 模式 ISP 下载参考线路图
4. 增加硬件 USB 下载时的预置的用户可选内部 IRC 频率说明
5. 在封装图中增加 DFN8、QFN20、QFN32、QFN48、QFN64 衬底说明
6. 增加比较器多路复用 (比较器+ADC 输入) 应用的范例程序

## ● 2020/5/29

1. ADC 章节增加加法电路应用
2. 增加寄存器 EAXFR 的说明
3. 修正 DFN8 封装尺寸图中的错误
4. 增加使用第三方应用程序调用发布项目程序的方法

## ● 2020/5/25

1. ADC 章节增加负压检测电路
2. 修正部分图片中的乱码

## ● 2020/5/20

1. 更新电气特性中的使能低压检测唤醒功能时, 时钟停振模式的功耗参数
2. 更新电气特性中的使能比较器掉电唤醒功能时, 时钟停振模式的功耗参数
3. ADC 范例程序中增加设置 ADCTIM 寄存器用于控制 ADC 内部时序
4. 更正文档中的部分笔误
5. 在每个单片机系列的特性中增加可用于时钟停振模式唤醒的中断
6. 增加 I/O 口中断的范例程序
7. 在 ISP 下载应用线路图中增加 ISP 下载步骤指导
8. 增加掉电唤醒定时器唤醒省电模式范例程序

## ● 2020/5/14

1. 增加比较器多路复用说明
2. 增加 PWM 触发 ADC 范例程序
3. ADC 章节中增加 ADC 工作时钟频率描述

4. ADC 章节中增加 ADC 参考线路图
5. 更新电气特性中的低压检测、比较器等功耗参数
6. 更新 STC8H1K08 系列的参考价格
7. 增加 PWM 当作 DAC 使用的参考线路图
8. 更新管脚图以及管脚说明中有关 PWM 外部触发脚 PWMETI 的说明
9. 增加上电复位和按键复位参考电路图

## ● 2020/4/29

1. 将串口下载参考电路图中, MCU 的 TxD 脚上串联的电阻由 300 欧姆改为 100 欧姆
2. 修正使用 PL2303GL 进行 ISP 下载的参考电路图中供电部分的错误

## ● 2020/4/26

1. 更新电气特性中 I/O 速度参数
2. 增加 STC8H8K64U 系列的 PDIP40 的参考管脚图
3. 更新电气特性中比较器速度参数
4. 更正 14.6 章节串口注意事项中设置 TI 和 RI 的时间点
5. 增加比较器章节中有关模拟滤波和数字滤波的说明
6. 修正附录 E 中, MAX232 与 RS485 连线错误

## ● 2020/4/8

1. 增加掉电唤醒定时器寄存器的使用说明
2. 更新 I/O 口章节中有关整体驱动电流的内容

## ● 2020/3/27

1. 删除 STC8H8K64S2U 系列
2. STC8H8K64S4U 系列更名为 STC8H8K64U 系列
3. IRC24MCR 寄存器更名为 HIRCCR
4. 增加 STC8H8K64S4U 型号使用 PL2303GL 下载参考线路图
5. 增加 STC8H8K64S4U 型号直接硬件 USB 下载参考线路图
6. 按照统一风格重命名所有芯片的电源相关管脚名称
7. 更新直流特性中的芯片在不同工作频率下的功耗
8. 在高级 PWM 定时器章节开头增加说明部分
9. 更新 STC8H8K64U 系列选型价格表
10. 更正“利用 ADC 第 15 通道测量外部电压或电池电压”章节中计算电压的公式

## ● 2020/3/13

1. 重新校对高级 PWM 定时器章节中的内容
2. 增加高级 PWM 定时器当外部中断使用的范例程序

## ● 2020/3/6



1. 更正文档中有关内部参考信号源描述错误的部分
2. 增加一般精度 ADC 和高精度 ADC 的应用线路图
3. 增加 ADC 模块的静态参数
4. 增加 STC8H8K64S4U 系列 LQFP48 管脚图与管脚说明
5. 增加 STC8H8K64S2U 系列 LQFP48 管脚图与管脚说明
6. 删除 STC8H3K64S4 系列
7. 删除 STC8H3K64S2 系列
8. 增加 I/O 口中断章节
9. 中断系统章节中增加 I/O 中断的说明部分
10. 增加 USB 范例程序 (HID 界面)
11. 重新整理管脚图章节结构
12. 更正 STC8H1K17 型号的芯片特性

## ● 2020/1/20

1. 增加高级 PWM 相关的范例代码
2. 增加“一种典型三极管控制”电路
3. 增加“典型发光二极管控制”电路
4. 增加“混合电压供电系统 3V/5V 器件 I/O 口互连”参考电路
5. 增加“如何让 I/O 口上电复位时为低电平”参考电路
6. 增加“利用 74HC595 驱动 8 个数码管(串行扩展,3 根线)”参考电路
7. 增加“I/O 口直接驱动 LED 数码管”参考电路
8. 增加“运行用户程序时收到用户命令后自动启动 ISP 下载”说明

## ● 2020/1/17

1. 增加 MDU16 运算时钟数说明
2. 增加 STC8H1K08 系列 QFN20 的管脚图

## ● 2020/1/15

1. 增加“ADC 做电容感应触摸按键”章节
2. 增加“ADC 作按键扫描应用线路图”章节
3. 增加附录“RS485 自动控制或 I/O 口控制线路图”
4. 增加附录“U8W 下载工具中 RS485 部分线路图”

## ● 2019/12/31

1. 修正 STC8H8K64S2U 系列和 STC8H8K64S4U 系列 I/O 口数量, 实际为最多 60 个 I/O

## ● 2019/12/30

1. 创建 STC8H 系列单片机技术参考手册文档
2. 增加 STC8H1K28 系列
3. 增加 STC8H1K08 系列
4. 增加 STC8H3K64S4 系列



5. 增加 STC8H3K64S2 系列
6. 增加 STC8H8K64S4U 系列
7. 增加 STC8H8K64S2U 系列
8. 增加 MDU16 乘除单元说明

STC MCU

## 附录JJ STC8 系列命名花絮

**STC8A:** 字母“A”代表 ADC，是 STC 12 位 ADC 的起航产品

**STC8F:** 无 ADC、PWM 和 PCA 功能，现 STC8F 的改版芯片与原始的 STC8F 管脚完全兼容，但内部设计进行了优化和更新，用户需要修改程序，所以命名为 STC8C

**STC8C:** 字母“C”代表改版，是 STC8F 的改版芯片

**STC8G:** 字母“G”最初是芯片生产时打错字了，后来将错就错，定义 G 系列为“GOOD”系列，STC8G 系列简单易学

**STC8H:** 字母“H”取自“高”的英文单词“High”的首字母，“高”表示“16 位高级 PWM”

STC MCU

# 本系列产品标准销售合同

- 一. 产品质量标准：货物为全新正品。符合 ROHS 质量标准。
  - 二. 供方责任：如是供方质量问题，经双方确认后，需方退回芯片，有一换一，质保一年。
  - 三. 需方责任：
    - A、验收：在快递送货到时，需方确认数量无误，无芯片散落，无管脚变形，无其他品质异常情况后再签收。如有异常需方不能签收，由快递公司承担责任。一经需方签收，需方就是认可供方已按要求完成该订单，不再有其他连带责任。
    - B、保管及贴片加工：根据国际湿敏度 3（MSL3）规范的要求，贴片元器件在拆开真空包装后，168 小时内，7 天内，必须回流焊贴片完成。LQFP/QFN/DFN 托盘能耐 100 度以上的高温，拆开真空包装后 7 天内必须回流焊贴片完成，如未完成，回流焊前必须重新烘烤：110~125℃，4~8 个小时都可以 SOP/TSSOP 塑料管耐不了 100 度以上的高温，拆开真空包装后 7 天内必须回流焊贴片完成，否则回流焊前先去除耐不了 100 度以上高温的塑料管，放到金属托盘中，重新烘烤：110~125℃，4~8 个小时都可以
- 由于经常有客户退回来的货物中含有来历不明产品，且贴片原器件拆开真空包装后，需要在 168 小时/7 天内完成回流焊贴片工序。
- 我司无产能对退回器件再进行重新详细检测，再进行重新烘烤，无能力对客户退回的所谓未拆封芯片进行评估，为保证全体客户的利益，产品一经出库，概不退换，以确保品质，确保所有客户的安全。
- 四. 解决纠纷方式：对本合同不详尽之处或产生争议，双方协商解决。协商不成在供方所在地申请仲裁。
  - 五. 其他条款：合同一式两份。自双方签署起生效。供方若因外力因素而导致无法交货，供方应及时通知需方，并重新协商本合同相关事宜，需方免除供方应承担的义务。本合同未能列入条款可在合同附件详细列入。
  - 六. 本合同双方代表签字且款到后方可生效。

备注：如特殊情况，买方买的型号要更换成其他型号，供方也同意的：

- 1, 开机 13 小时高温烘烤， 1000 元一次
- 2, 开机测试 RMB500 一次， +0.2 元/片

# 产 品 授 权 书

**致：江苏国芯科技有限公司**

STC8G 系列和 STC8H 系列产品的知识产权归深圳国芯人工智能有限公司所有。现授权江苏国芯科技有限公司可从事 STC8G 系列和 STC8H 系列产品在中国的推广和销售工作。

**授权单位：深圳国芯人工智能有限公司**

**授权时限：2019 年 10 月 24 日 - 2024 年 12 月 31 日**



# 自主产权，生产可控

深圳国芯人工智能有限公司是中华人民共和国大陆独资企业，按中国法律法规独立运营的企业，注册地址在深圳市前海深港合作区前湾一路1号A栋201室。

本手册所描述的器件是在中国境内自主研发，具备独立自主知识产权。

产品核心研发在中国境内，具备芯片设计、封装设计、结构设计、可靠性设计、器件仿真、工艺模拟等全部设计能力；产品核心研发团队人员及带头人全部为我国境内人员组成，其中研发团队带头人研发从业年限十年以上，具备长期、稳定的后续支持能力，具有在我国境内申请的专利证书及软件著作权等。

晶圆制造：本器件设计完成后的晶圆制造加工，在中华人民共和国大陆境内的晶圆厂加工制造完成，受中华人民共和国法律法规管理监管和控制，完全可控。

封装制造：本器件设计完成后的封装制造，在中华人民共和国大陆境内的封装厂加工完成，受中华人民共和国法律法规管理监管和控制，完全可控。

测试：本器件设计完成后的测试，在中华人民共和国大陆境内测试完成，受中华人民共和国法律法规管理监管和控制，完全可控。

本器件全部关键工艺均在我国自有生产线上完成，可以长期供货，无被断供的困扰。

特此说明。

