XHSC 小华半导体
XIAOHUA SEMICONDUCTOR

# 32 位微控制器

## HC32L12x 自举程序中使用的 UART 协议

# 应用笔记

Rev1.00　2026 年 03 月

# 适用对象

| 产品系列 | 产品型号 |
| --- | --- |
| L 系列 | HC32L12x |

# 声　明

★ 小华半导体有限公司（以下简称：“XHSC”）保留随时更改、更正、增强、修改小华半导体产品和/或本文档的权利，恕不另行通知。用户可在下单前获取最新相关信息。XHSC 产品依据购销基本合同中载明的销售条款和条件进行销售。

★ 客户应针对您的应用选择合适的 XHSC 产品，并设计、验证和测试您的应用，以确保您的应用满足相应标准以及任何安全、安保或其它要求。客户应对此独自承担全部责任。

★ XHSC 在此确认未以明示或暗示方式授予任何知识产权许可。

★ XHSC 产品的转售，若其条款与此处规定不同，XHSC 对此类产品的任何保修承诺无效。

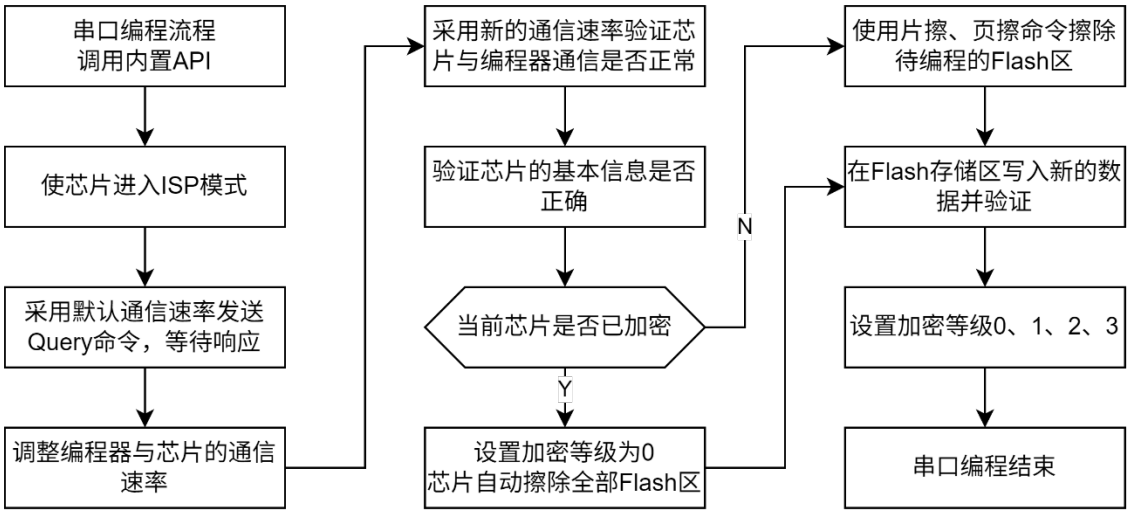
★ 任何带有“®”或“™”标识的图形或字样是 XHSC 的商标。所有其他在 XHSC 产品上显示的产品或服务名称均为其各自所有者的财产。

★ 本通知中的信息取代并替换先前版本中的信息。

# 目　录

# 1 芯片编程流程

芯片上电后，配置 HC32L12x 微控制器进入 ISP 模式，使用 ISP 命令进行编程。



# 2 使目标 MCU 进入 ISP 模式的方法

## 2.1 当该型号封装有 BOOT 引脚时

**Mode1：BOOT 高电平直接进入**

1) 编程器拉低 MCU 的 RST 引脚。
2) 编程器向 MCU 的 BOOT 引脚提供高电平。
3) 编程器拉高 MCU 的 RST 引脚，并延时至少 10ms。
4) 目标 MCU 进入 ISP 编程模式。

**Mode2：BOOT 低电平+握手协议**

注：仅支持 *Rev0.4* 及后续 *BL* 版本。查询 *BL* 版本请查看章节【Query 指令】部分。

1) MCU 的 BOOT 引脚保持低电平。编程器拉低 MCU 的 RST 引脚，并保持 RST 引脚低电平至少 5 ms。
2) 编程器拉高 MCU 的 RST 引脚，并立即向 ISP 接口按 UART 格式循环发送数据序列。UART 参数为 500000-8-N-1，持续时间至少 15 ms，数据序列为 FF-56-FF-56。
3) 目标 MCU 进入 ISP 编程模式。

## 2.2 当该型号封装无 BOOT 引脚时

**Mode3：双线模式**

1) 编程器拉低 MCU 的 RST 引脚，并保持 RST 引脚低电平至少 5 ms。
2) 编程器拉高 MCU 的 RST 引脚，并立即向 ISP 接口按 UART 格式循环发送数据序列。UART 参数为 500000-8-N-1，持续时间至少 15 ms，数据序列为 FF-56-FF-56。
3) 目标 MCU 进入 ISP 编程模式。

**Mode4：单线模式**

1) 编程器拉低 MCU 的 RST 引脚，并保持 RST 引脚低电平至少 5 ms。
2) 编程器拉高 MCU 的 RST 引脚，并立即向 PA14 端口（SWCLK）按 UART 格式循环发送数据序列。UART 参数为 500000-8-N-1，持续时间至少 15 ms，数据序列为 FF-56-FF-56。
3) 目标 MCU 进入 ISP 编程模式。

# 3 UART 通信格式

目标 MCU 进入 ISP 编程模式后，编程器与芯片之间的通信格式如下方所示。

## 3.1 物理层

使用 UART 进行通信，UART 初始通信参数为：**115200 - 8 - N - 1**。

## 3.2 协议层

协议层为**十六进制字节流**，上、下行时数据包格式均为：<**Head**><**Len**><**Info**><**CRC**>

| 协议帧格式 | 帧长度（字节） | 协议帧说明 |
|---|---|---|
| **Head** | 1 | 帧头，其值固定为 0x65 |
| **Len** | 1 | 其值代表 Info 有多少个字节（十六进制），取值范围为 0-255 |
| **Info** | [Len] | 应用层数据信息，其长度为 Len。<br>上行时，Info 的第一个字节为状态字 STA，表示下位机执行信息。 |
| **CRC** | 2 | Head - Len –Info 所有字段的计算值，低字节在前；<br>CRC 参数模型：CRC-16/X25　　$X^{16} + X^{12} + X^5 + 1$<br>多项式 POLY：0x1021<br>初始值：0xFFFF<br>结果异或值：0xFFFF |

收发数据示例：

下行（上位机发给 MCU）：　**650110 65F3**

上行（MCU 返回给上位机）：**650500FF4C2D03 060B**

## 3.3 ISP 指令列表

| 指令名称 | 指令规范（下行） | 指令功能说明 | 适用状态 |
|---|---|---|---|
| Query | <0x10> | 编程器查询目标 MCU 是否能正常通信。 | Level0~2 |
| PPS | <0x11><BRR> | 编程器与目标 MCU 协商通信速率。 | Level0~2 |
| Set BaseAddr | <0x27><BaseAddr> | 设置后续读、写、擦操作基地址为<BaseAddr>。 | Level0~2 |
| FLASH_ChipErase | <0x20> | 擦除目标 MCU 用户闪存区内所有数据。 | Level0~2 |
| FLASH_SectorErase | <0x21><Offset> | 擦除地址“<BaseAddr> + <Offset>”所在页面。 | Level0~2 |
| FLASH_Blank Check | <0x22> | 检查目标 MCU 闪存数据是否全空（全为 0xFF）。 | Level0~2 |
| Verify Data | <0x23><Offset><ByteCnt> | 返回从<BaseAddr>+<Offset>开始<ByteCnt>字节的 CRC 值。 | Level0 |
| CheckSum | <0x24><Offset><ByteCnt> | 返回从<BaseAddr>+<Offset>开始<ByteCnt>字节的 CRC 值。注意：最小校验单位为页。 | Level0~2 |
| WriteData | <0x28><Offset><D1-Dn> | 将 N 字节数据写入<BaseAddr>+ <Offset>处，并读回验证是否与写入数据相同。 | Level0 |

| 指令名称 | 指令规范（下行） | 指令功能说明 | 适用状态 |
|---|---|---|---|
| Read Data | <0x29><Offset><N> | 从<BaseAddr > + <Offset>处，读出N个字节。N = 1 ~ 255。 | Level0 |
| SetProtectLevel | <0x2C><RdLevel> | 设置芯片的读保护等级。<br>支持 Level0~Level3 共四种保护等级。 | Level0~2 |
| Jump | <0x30><Addr> | 跳转到 Addr 所指定的地址开始执行程序。 | Level0 |

## 3.4 ISP 指令

### 3.4.1 Query 指令

- 指令功能：编程器查询目标 MCU 是否能正常通信。
- 下行格式：<0x10>
- 上行格式：<STA><xx xx xx xx>
- 指令说明：<xx xx xx xx>代表 bootloader 版本号。如<**FF 4C 2D 06>，最后一个字节 06 代表版本 Rev0.6**。

### 3.4.2 PPS 指令

- 指令功能：编程器与目标 MCU 协商通信速率。
- 下行格式：<0x11><BRR>
- 上行格式：<STA>
- 指令说明：
  - BRR 为 2 个字节，低字节先发
  - MCU 返回状态字后，调整至指定通信波特率

**表 3-1 波特率配置表**

| 目标波特率 | BRR 值 | 实际波特率 |
|---|---|---|
| 19200 | 0xCF00 | 19231 |
| 115200 | 0x21FB | 115349 |
| 250000 | 0x0F00 | 250000 |
| 500000 | 0x0700 | 500000 |
| 1000000 | 0x0300 | 1000000 |
| 1500000 | 0x01C0 | 1500000 |
| 2000000 | 0x0100 | 2000000 |

### 3.4.3 SetBaseAddr 指令

- 指令功能：设置读、写、擦操作的基地址。
- 下行格式：<0x27><BaseAddr>
- 上行格式：<STA>

- 指令说明：<BasedAddr>为 4 个字节，低字节先发。

### 3.4.4 Flash_ChipErase 指令

- 指令功能：擦除目标 MCU 闪存内的所有数据。
- 下行格式：<0x20>
- 上行格式：<STA>
- 指令说明：擦除完成后，闪存内的数据全部变成 0xFF。

### 3.4.5 Flash_SectorErase 指令

- 指令功能：擦除< BaseAddr + Offset >地址所在的页面。
- 下行格式：<0x21>< Offset >
- 上行格式：<STA>
- 指令说明：Offset 为 4 个字节，低字节先发。

### 3.4.6 Flash_BlankCheck 指令

- 指令功能：检查目标 MCU 闪存内全片数据是否全为 FF。
- 下行格式：<0x22>
- 上行格式：<STA>
- 指令说明：无。

### 3.4.7 WriteData 指令

- 指令功能：将 N 字节的数据写入<BaseAddr>+<Offset>处，并验证读出与写入的数据是否相同。
- 下行格式：<0x28>< Offset ><D1~Dn>
- 上行格式：<STA>
- 指令说明：
  - Offset 为 4 个字节，低字节先发。一次发送的字节数范围为 1 - 248。
  - 当<BaseAddr>+<Offset>与待编程字节数均为 Word 对齐时，BootLoader 调用 Word Program 操作，否则调用 Byte Program 操作。
  - Word Program 仅需 Byte Program 一半的时间。

### 3.4.8 ReadData 指令

- 指令功能：从<BaseAddr>+<Offset>处，读出 N 个字节。
- 下行格式：<0x29><Offset><N>
- 上行格式：<STA><D1~Dn>
- 指令说明：Offset 为 4 个字节，低字节先发。
  N 为 1 个字节长度，取值范围为 1 ~ 255。

当芯片读保护状态解除时，可以读出 Flash 内容。

可读取芯片的唯一识别号 UID，详见参考手册产品唯一身份识别标识 UID 部分。

Level0 支持全部数据的读取。

### 3.4.9 VerifyData 指令

- 指令功能：计算并返回从<BaseAddr>+<Offset>开始的 ByteCnt 字节的 CRC 值。
- 下行格式：<0x23>< Offset > < ByteCnt >
- 上行格式：<STA><CRC>
- 指令说明：Offset 为 4 个字节，低字节先发。

  ByteCnt 为 4 个字节，低字节先发。

  CRC 均为 2 个字节，低字节先发。

### 3.4.10 CheckSum 指令

- 指令功能：计算并返回从<BaseAddr>+<Offset>开始的 ByteCnt 字节的 CRC 值。
- 下行格式：<0x24>< Offset > < ByteCnt >
- 上行格式：<STA><CRC>
- 指令说明：Offset 为 4 个字节，0x200 整数倍扇区对齐，低字节先发。

  ByteCnt 为 4 个字节，0x200 整数倍扇区对齐，低字节先发。

  CRC 均为 2 个字节，低字节先发。

### 3.4.11 SetProtectLevel 指令

- 指令功能：设置或读取芯片的读保据状态，支持 Level0 ~ Level3 共四种保护等级。
- 下行格式：<0x2C><RdLevel>
- 上行格式：<STA><RdState>
- 指令说明：RdLevel 为 0x00~0x03 分别代表 Level0 ~ Level3，保护等级限制如下：

  RdLevel 为 0x55 则不操作当前读保护等级，只返回当前保护等级。

  Level0：SWD 可读写芯片，ISP 可读写芯片。

  Level1：SWD 可降级芯片，ISP 可降级芯片，数据不可读出。

  Level2：SWD 端口物理断开，ISP 可降级芯片，数据不可读出。

  Level3：SWD 端口物理断开，ISP 端口物理断开，数据不可读出。

  **注意**：Level3 保护等级生效后芯片不能再次进行编程，请慎用。

  **注意**：加密等级更新时，只要从其他等级降级到 Level0，就执行全片擦除。

### 3.4.12 Jump 指令

- 指令功能：跳转到 Addr 内指定的地址开始执行程序。
- 下行格式：<0x30><Addr>

- 上行格式：<STA>
- 指令说明：Addr 为 4 个字节，低字节先发。
  跳转地址为 Addr 内所存地址。

### 3.4.13 返回状态字说明

0x00：代表执行成功

0x10：代表通信 CRC 检验错误，需要重发该命令

0x20：代表指令不支持

0x30：代表没有读权限

0x31：代表没有写权限

0x32：代表没有跳转权限

0x33：代表没有擦权限

0x34：代表 Verify 权限

0x40：代表 Write Data 失败

0x41：代表 Blank Check 失败

# 4 附录

CRC 计算例程：

```
//计算 pBuf 所指定的 ByteCnt 个字节的 CRC 值

uint16_t Memery_CalcCRC16( uint8_t pBuf[], uint32_t u32ByteCnt )
{
    uint32_t i;
    uint16_t tmp16;
    uint16_t CRCValue;

    CRCValue = 0xFFFF;
    for ( i=0; i<u32ByteCnt; i++ )
    {
        tmp16 = pBuf[i];
        tmp16 = tmp16 ^ CRCValue;
        tmp16 = tmp16 ^ (tmp16 << 4);
        tmp16 &= 0xFF;
        CRCValue = ((CRCValue >> 8) ^ (tmp16 << 8) ^ (tmp16 << 3) ^ (tmp16 >> 4));
    }

    CRCValue ^= 0xFFFF;

    return( CRCValue );
}
```

# 版本修订记录

| 版本号 | 修订日期 | 修订内容 |
| --- | --- | --- |
| Rev1.00 | 2026/03/25 | 初版发布。 |