AN-32002

FT32 串口 ISP 编程说明文档

Rev1.00
http://www.touchmcu.com

文档修改历史

版本	内容	修订日期
V1.00	初始版本	2021-10-20

目录

表目录

图目录

1. 文档简介

本文档主要描述了用于FT32 串口编程（ISP）协议，是编写串口 ISP 烧录上位机的参考文档，其内容包含 ISP 硬件配置、串口通信协议以及详细的编程指令说明。

2. 硬件配置

进行 ISP 烧录操作时，需保证以下的硬件配置，保证芯片复位上电后，系统能够从芯片内部的系统存储区启动，进入ISP 引导烧录模式。

- ISP 烧录串口：USART1-TX(PA9)、USART1-RX(PA10)。
- BOOT0 引脚：上拉为高电平。
- 系统选项字中：nBOOT1 = 1。
- 串口波特率： 4800~119200bps。

3. ISP 串口编码序列

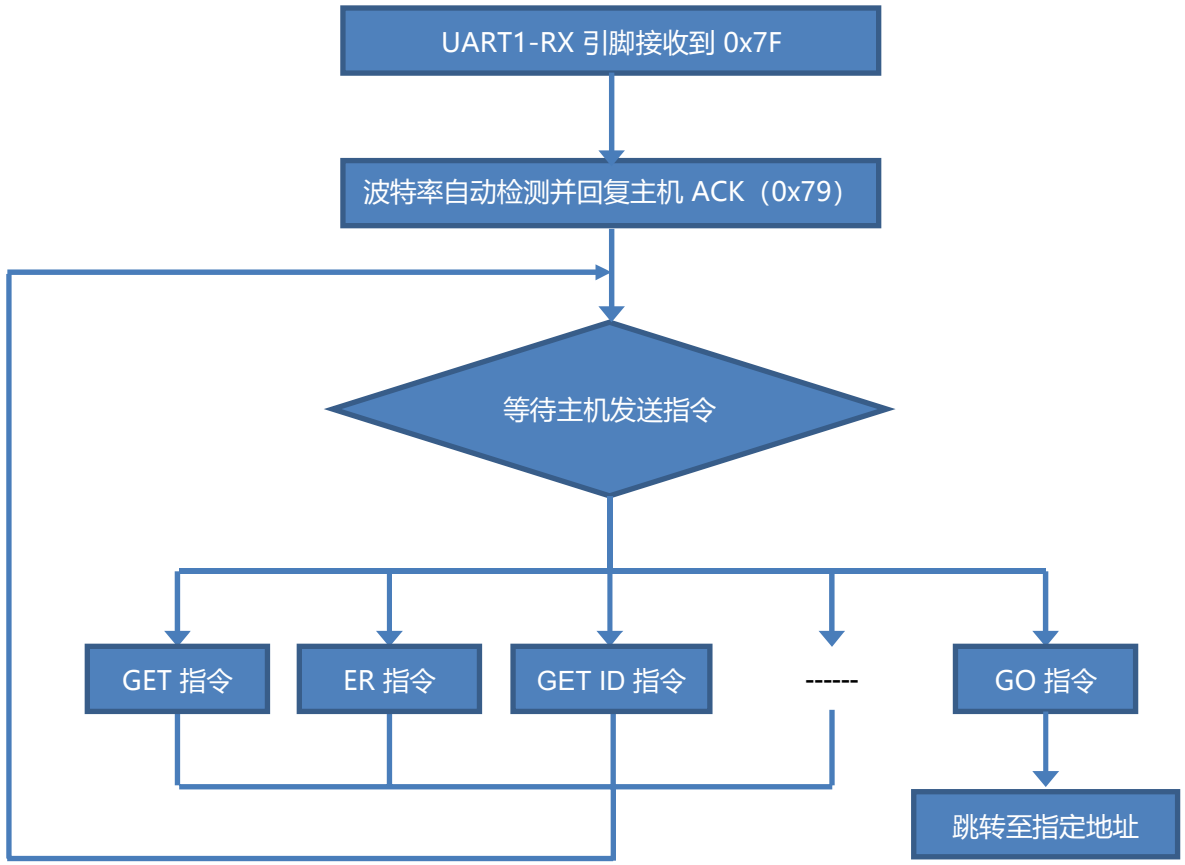

图 3-1 ISP 编程序列

当从系统存储区（0x1FFFE800~0x1FFFF7FF）启动时，存储于此引导区的代码将会不断的扫描 UART1-RX 引脚，等待引脚上出现 0x7F 数据帧（1 起始位，8 数据位，1 偶校验位，1 停止位），当检测到此数据帧后，引导程序内部实现波特率自动检测，并与上位机波特率进行对准，此时芯片的USART-TX 会回复ACK(0x79)，表示芯片已接收初始化完毕，可以进入后续的编码序列。

4. Bootloader 指令集

指令名称	指令码	指令描述
GET	0x00	读取Bootloader版本及其当前版本所支持的指令集
Get Version & Read Protection Status	0x01	读取Bootloader版本，以及Flash的读保护设置状态
Get ID	0x02	读取芯片识别PID
Read Memory	0x11	读取存储区指定地址的内容，最大支持256bytes
Go	0X21	跳转至位于main flash或SRAM中指定的地址区域，并从此处开始运行存储的用户代码
Write Memory	0x31	向flash或sram中指定的地址区域写入数据，最大支持256bytes
Extended Erase	0x44	将此命令后的相应数量及地址所对应的flash page进行擦除，1次命令可指定擦除1至127个page（每个page 0.5kB）。
Write Protect	0x63	使能指定flash扇区的写保护功能，F072共16个扇区，每扇区4k大小
Write Unprotect	0x73	失能芯片所有扇区的写保护功能
Readout Protect	0x82	使能读保护功能
Readout Unprotect	0x92	失能芯片的读保护功能

表 4-1 bootloader 支持指令集说明

注意：

1. 若接收到的指令不属于上述指令，或上述指令执行中产生错误，Bootloader 会回应主机NACK(0x1F)，并退回到初始等待指令的接收状态。

2. 读保护使能后，GET、Get Version & Read Protection Status、Get ID、Readout Unprotect 等指令不受影响，其他指令Bootloader 只会回应主机NACK(0x1F)，不对指令进行解读操作。

为保证串口通信安全传输，所有由主机发送的数据均要进行保护校验，校验方式如下：

1. checksum：接收到的数据块通过异或和进行校验，有效数据块后面会跟随 1Byte 的校验和，其值为此数据块的异或和。

2. 主机发送的每个指令均为 2Byte：1Byte 的指令、1Byte 的指令取反数（两数异或结果应为 0xFF）。

3. 每个数据包发送后，Bootloader 会回应接受指令应答ACK（0x79）或拒绝指令应答NACK(0x1F)。

4.1 GET 指令（0x00）

Get 指令用于用户读取BootLoader 版本及其所支持的指令集，当 BootLoader 接收到 Get 指令，它会向主机回发此信息项。上位机端 Get 指令处理如下：

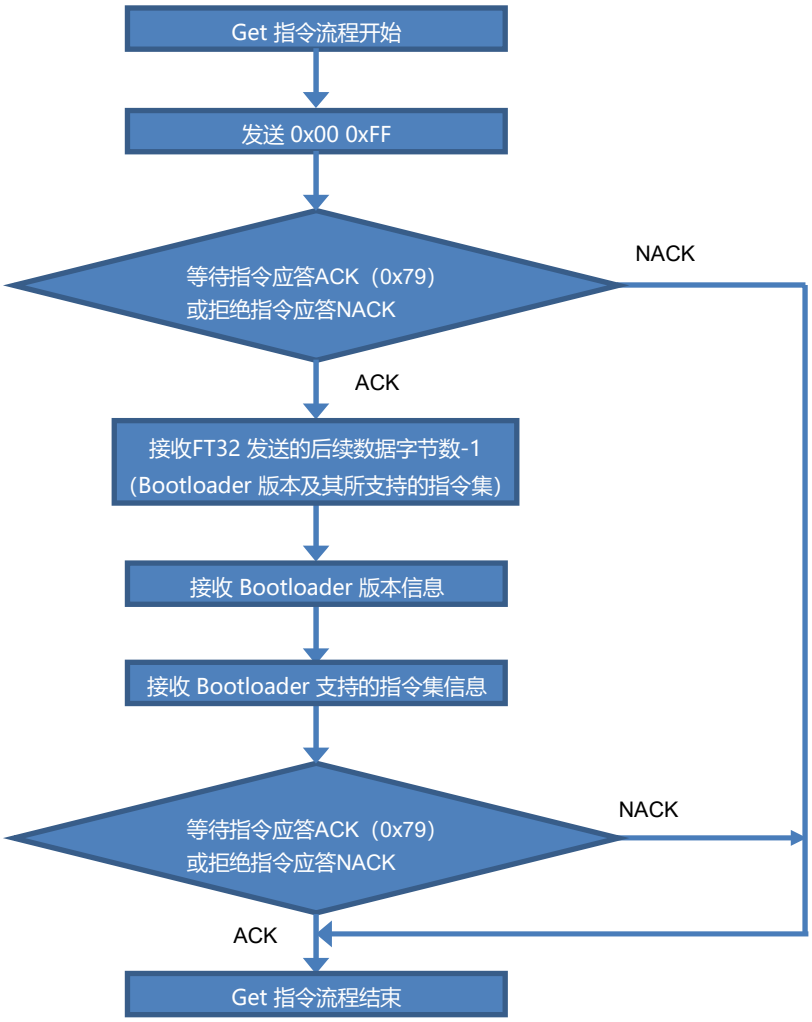

图 4-1 Get 指令流程

Get 指令FT32 接收回复流程示例：

示例流程开始

上位机端发送Byte1~2： 0x00 0xFF（Get 指令）

从机端回复Byte1： 0x79（ACK）

从机端回复Byte2： 0x0B（后续字节数是 11+1 = 12 个，即Byte3~Byte14）

从机端回复Byte3： 0x31（Bootloader 版本V3.1）

从机端回复Byte4~14：0x00 0x01 0x02 0x11 0x21 0x31 0x44 0x63 0x73 0x82 0x92

（每字节均表示 1 条支持的指令：Get、Get version……）

从机端回复Byte15：0x79（ACK）

示例流程结束。

4.2 Get Version & Read Protection Status 指令（0x01）

Get GV 指令用于用户读取 BootLoader 版本及芯片读保护状态，当 BootLoader 接收到此指令，它会向主机回发此信息项。上位机端指令处理如下（Get GV = Get Version & Read Protection Status 指令）

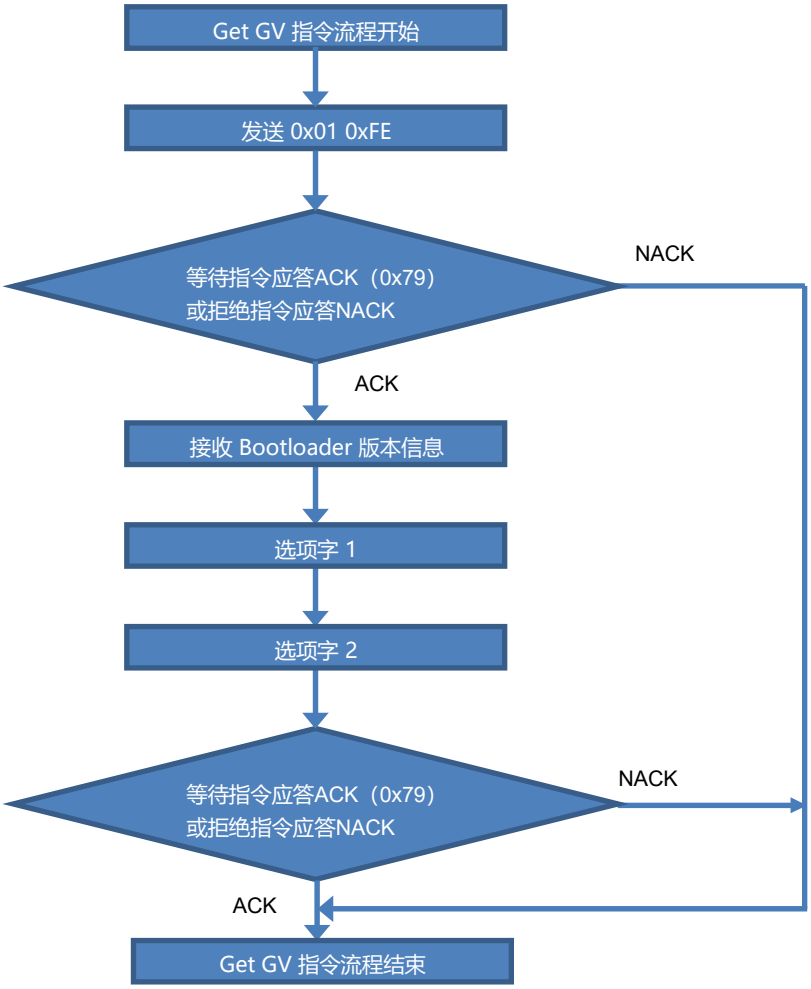

图 4-2 Get Version & Read Protection Status 指令流程

目前版本中选项字1~2 为 0x00 0x00 表示读保护解除状态，0x01 0x01 表示芯片处于读保护状态。

Get Version & Read Protection Status 指令FT32 接收回复流程示例：

示例流程开始

上位机端发送Byte1~2： 0x01 0xFE（Get Version & Read Protection Status 指令）

从机端回复Byte1： 0x79（ACK）

从机端回复Byte2： 0x31（Bootloader 版本V3.1）

从机端回复Byte3： 0x00（与 Byte4 组合表示读保护解除状态）

从机端回复Byte4： 0x00（与 Byte3 组合表示读保护解除状态）

从机端回复Byte5： 0x79（ACK）

示例流程结束。

4.3 Get ID 指令（0x02）

Get ID 指令用于用户读取芯片的内置 ID，当 BootLoader 接收到此指令，它会向主机回发此信息项。上位机端指令处理如下：

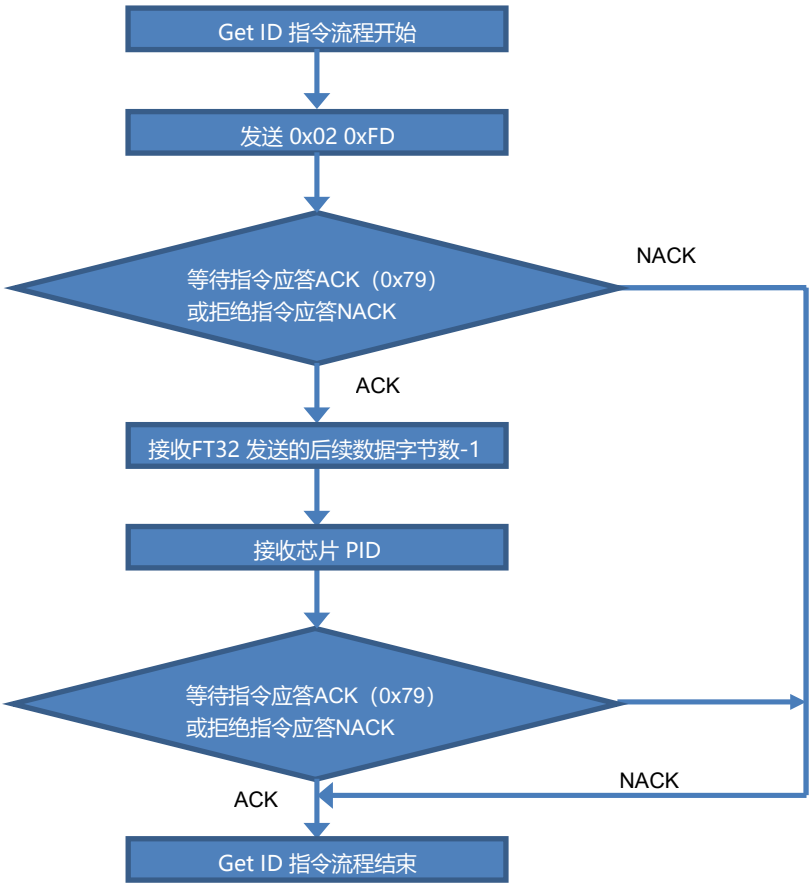

图 4-3 Get ID 指令流程

Get ID 指令 FT32 接收回复流程示例：

示例流程开始

上位机端发送Byte1~2： 0x02 0xFD（Get ID 指令）

从机端回复Byte1： 0x79（ACK）

从机端回复Byte2： 0x01（后续字节数是 1+1 = 2 个，即 Byte3~4）

从机端回复Byte3： 0x04（芯片 PID）

从机端回复Byte4： 0x48（芯片 PID）

从机端回复Byte5： 0x79（ACK）

示例流程结束。

4.4 Read Memory 指令（0x11）

Read Memory 指令用于用户读取芯片内指定的有效存储区，包括 main flash，选项字区以及 SRAM。如下所示：

名称	地址范围	大小
Main Flash	0x08000000~0x0800FFFF	64kbytes
User Option	0x1FFFF800~0x1FFFF813	20bytes
SRAM	0x20000000~0x20001FFF	8kbytes

表 4-2 Read Memory 指令可读取区域

当 Bootloader 接收到此指令后，会回复上位机 ACK，并进入等待上位机发送待读取地址状态，地址为 4 字节，大端存储，地址后面跟随 1 字节的地址异或校验和（即地址格式为 4Bytes Address + 1Bytes XOR checksum = 5Bytes）。当 Bootloader 接收到地址信息后，进行校验及地址有效性分析，分析后，回复上位机 ACK（地址校验通过且合理）或 NACK（地址校验未通过或不合理）。

当地址校验通过且合理，Bootloader 进入等待上位机发送待读取字节数（N-1），以及字节数的取反校验数，1Bytes N-1 + 1Bytes N-1`s Checksum = 2Bytes （两者异或和应为 0xFF），当 Bootloader 校验 2 字节数据合理后，Bootloader 开始发送指定地址、指定数量（N）的内容到上位机，若校验未通过，回复上位机 NACK 后结束此指令流程。

具体流程如下图（**图 4-4**）所示：

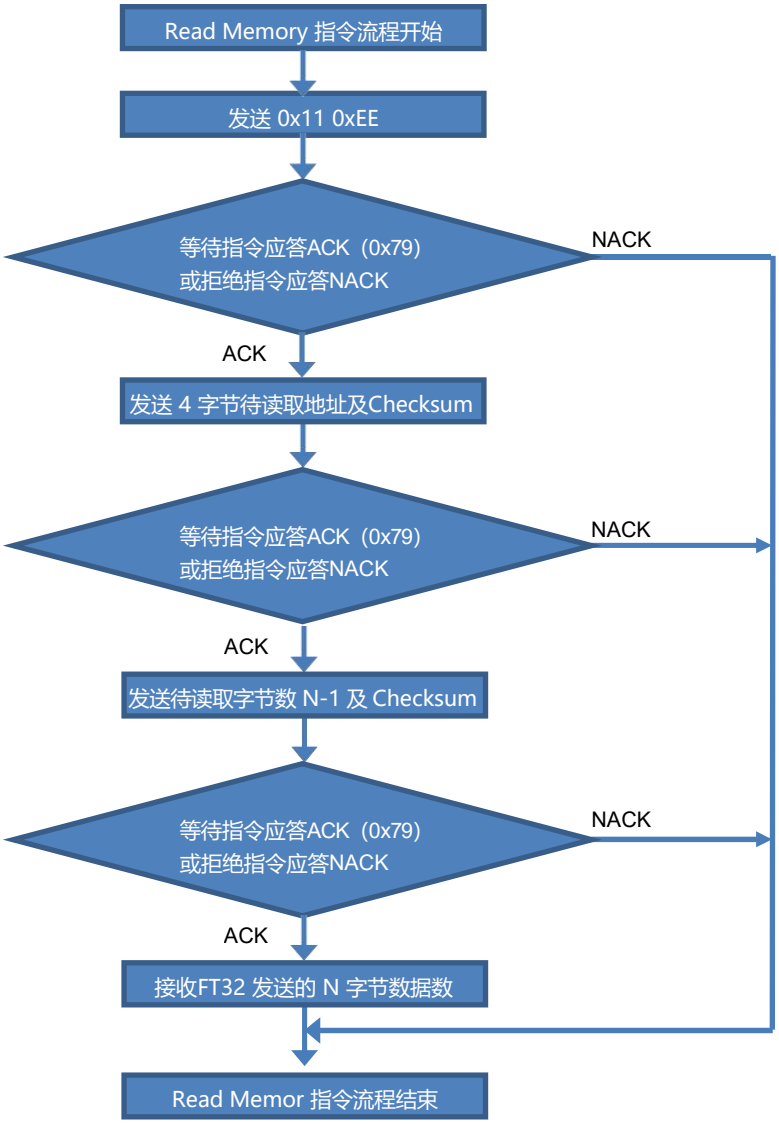

图 4-4 Read Memory 指令流程

Read Memory 指令 FT32 接收回复流程示例（从地址 0x08002000 读取 64 字节数据）：

示例流程开始

上位机端发送Byte1~2： 0x11 0xEE（Read Memory 指令）

从机端回复Byte1： 0x79（ACK）

上位机端发送Byte3~6： 0x08 0x00 0x20 0x00（待读取地址为 0x08002000）

上位机端发送Byte7： 0x28（Byte3 ~ Byte6 校验结果为 0x28）

从机端回复Byte2： 0x79（ACK）

上位机端发送Byte8~9： 0x3F 0xC0（待读取字节为 63+1=64 字节，校验字节为 0xC0）

从机端回复Byte3： 0x79（ACK）

从机端回复Byte4~67： 0xFF…0xFF（以地址 0x08002000 为起始地址的闪存区数据 64 字节）

示例流程结束。

4.5 Go 指令（0x21）

Go 指令用于跳转执行已经烧录的代码，或者执行指定地址区域的代码。（包括main flash 区域 0x08000000~0x0800FFFF 及 SRAM 区域 0x20000000~0x20001FFF）。

当 Bootloader 接收到此指令后，会回复上位机 ACK，并进入等待上位机发送待跳转地址状态，地址为 4 字节，大端存储，地址后面跟随 1 字节的地址异或校验和（即地址格式为 4Bytes Address + 1Bytes XOR checksum = 5Bytes）。

当 Bootloader 接收到地址信息后，进行校验及地址有效性分析，分析后，回复上位机 ACK（地址校验通过且合理）或 NACK（地址校验未通过或不合理），当地址校验通过且合理，Bootloader 复位其所用的外设资源，并重置堆栈，跳转至指定程序地址运行程序。

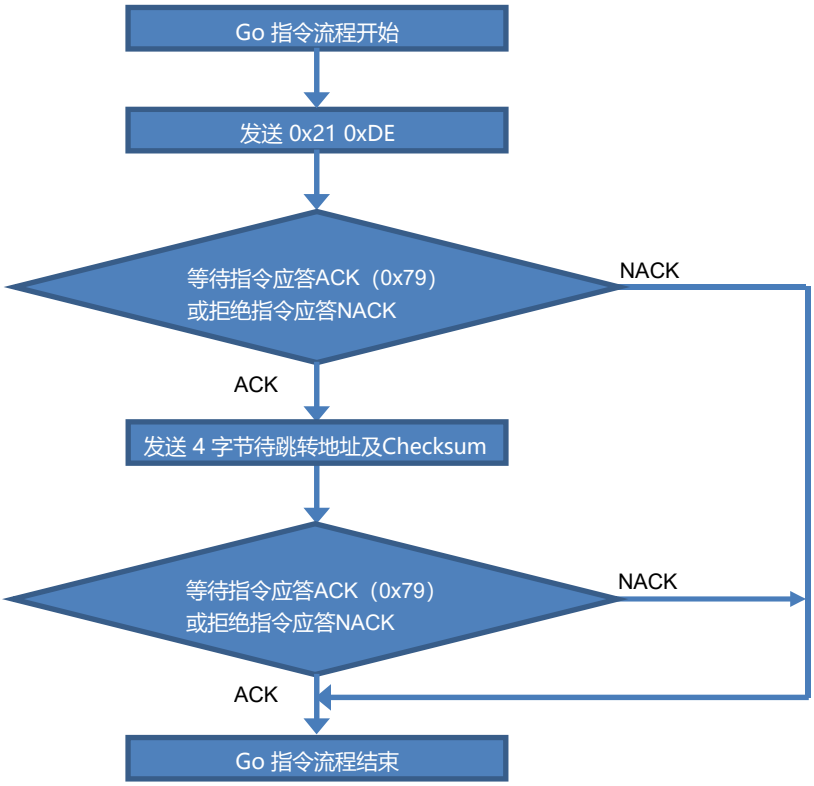

图 4-5 Go 指令流程

Go 指令 FT32 接收回复流程示例（跳转至main flash 区的 0x08000000 执行程序）：

示例流程开始

上位机端发送Byte1~2： 0x21 0xDE（Go 指令）

从机端回复Byte1： 0x79（ACK）

上位机端发送Byte3~6： 0x08 0x00 0x00 0x00（待跳转地址为 0x08000000）

上位机端发送Byte7： 0x08（Byte3 ~ Byte6 校验结果为 0x08）

从机端回复Byte2： 0x79（ACK）

示例流程结束。

4.6 Write Memory 指令（0x31）

Write Memory 指令用于用户向芯片内指定的有效存储区（包括 main flash，选项字区以及 SRAM）写入数据。

名称	地址范围	大小
Main Flash	0x08000000~0x0800FFFF	64kbytes
User Option	0x1FFFF800~0x1FFFF813	20bytes
SRAM	0x20000000~0x20001FFF	8kbytes

表 4-3 Write Memory 指令可存储区域

注意：对 Main Flash 及 UserOption 操作时，地址需按字对齐（即地址是 4 的倍数），写入数量应为 4 的倍数字节。

当 Bootloader 接收到此指令后，会回复上位机 ACK，并进入等待上位机发送待读取地址状态，地址为 4 字节，大端存储，地址后面跟随 1 字节的地址异或校验和（即地址格式为 4Bytes Address + 1Bytes XOR checksum = 5Bytes）。当 Bootloader 接收到地址信息后，进行校验及地址有效性分析，分析后，回复上位机 ACK（地址校验通过且合理）或 NACK（地址校验未通过或不合理）

当地址校验通过且合理，Bootloader 进入等待上位机发送待写入的字节数（1byte 表示 N-1,一次烧录支持的最大字节数是 256），以及N 个 Bytes 待写入数据，后面跟随 1byte 的异或校验和：

checksum = （N-1）^ data1 ^ data2…. dataN

当 Bootloader 校验烧录字节数及字节数据合理后，Bootloader 开始向指定地址区域写入数据，写入成功后回复上位机ACK（烧写成功），若写入未成功，回复上位机 NACK 后结束此指令流程。

具体流程如下图（**图 4-6**）所示：

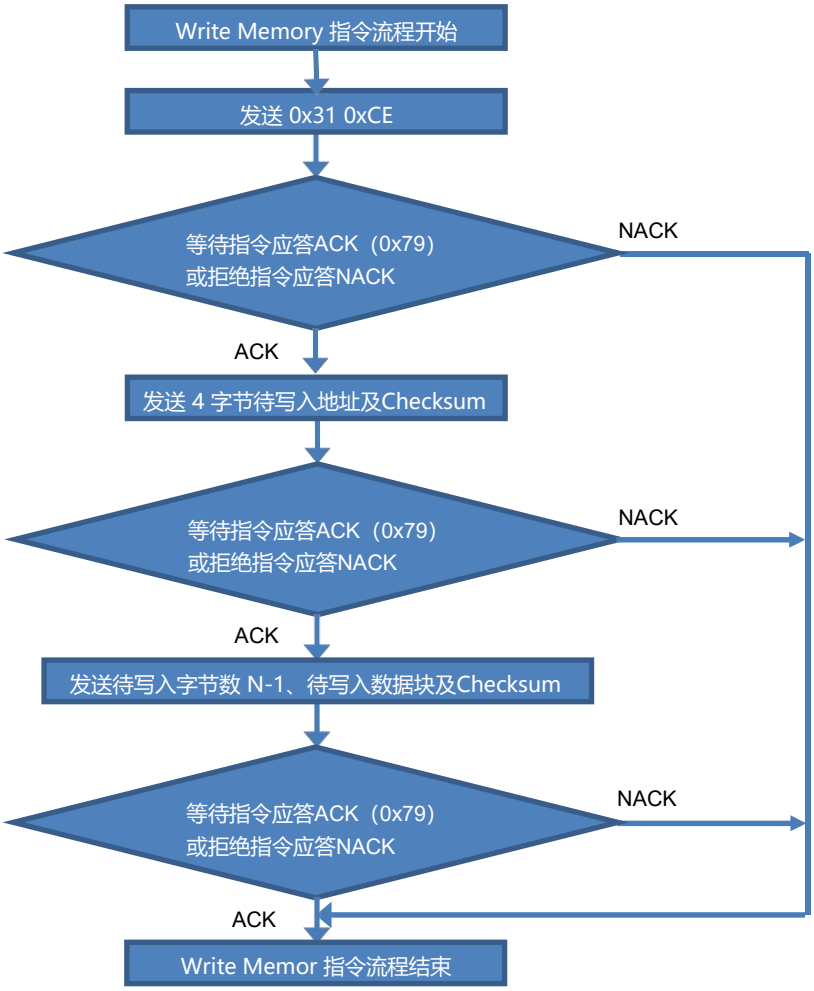

图 4-6 Write Memory 指令流程

Write Memory 指令 FT32 接收回复流程示例（向地址 0x08002000 写入 64 字节数据）：

示例流程开始

上位机端发送Byte1~2： 0x31 0xCE（WriteMemory 指令）

从机端回复Byte1： 0x79（ACK）

上位机端发送Byte3~6： 0x08 0x00 0x20 0x00（待写入地址为 0x08002000）

上位机端发送Byte7： 0x28（Byte3 ~ Byte6 校验结果为 0x28）

从机端回复Byte2： 0x79（ACK）

上位机端发送 Byte8： 0x3F （待写入数据大小-1，即实际需写入数量为 63+1 = 64）

上位机端发送 Byte9~ Byte72： 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07

0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F

0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17

0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F

0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27

0x28 0x29 0x2A 0x2B 0x2C 0x2D 0x2E 0x2F

0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37

0x38 0x39 0x3A 0x3B 0x3C 0x3D 0x3E 0x3F

（64 字节数据块）

上位机端发送Byte73： 0x3F（待读取数量及数据块异或校验和，即 Byte8~Byte72 异或校验和）

从机端回复Byte3： 0x79（ACK）

示例流程结束。

4.7 Extended Erase Memory 指令（0x44）

Extended Erase Memory 指令用于上位机擦除芯片内全部或指定的有效存储区（main flash），其擦除最小单位为 1page（FT32F072X 1page 为 0.5k bytes，总计 127 个 page）。

当 Bootloader 接收到此指令后，会回复上位机ACK，并进入等待上位机发送待擦除的 page 数量状态，擦除的 page 数量(N-1)为 2 字节，大端存储，Bootloader 会根据接收的（N-1）page 数量进行如下流程。

- 若 N-1 = 0xFFFF，且紧随其后的 1Byte 校验码为 0x00， Bootloader 则会进行main flash 的整片擦除；
- 若 N-1 小于等于 127，Bootloader 会继续接收 N x 2 个字节的 page 号，其每两个字节表示待指定擦除的 page，而后 Bootloader 会再接收 1 字节的异或和校验码 checksum。

Checksum = 2byte（待擦除数量 N-1）^ N x 2 bytes （所有待擦除 page 号）

当 Bootloader 校验擦除 page 数及各个待擦除 page 号合理后，Bootloader 开始擦除指定数量指定号码的 page，擦除成功回复上位机 ACK（擦除成功），未成功，回复上位机NACK 后结束此指令流程。

具体流程如下图（**图 4-7**）所示：

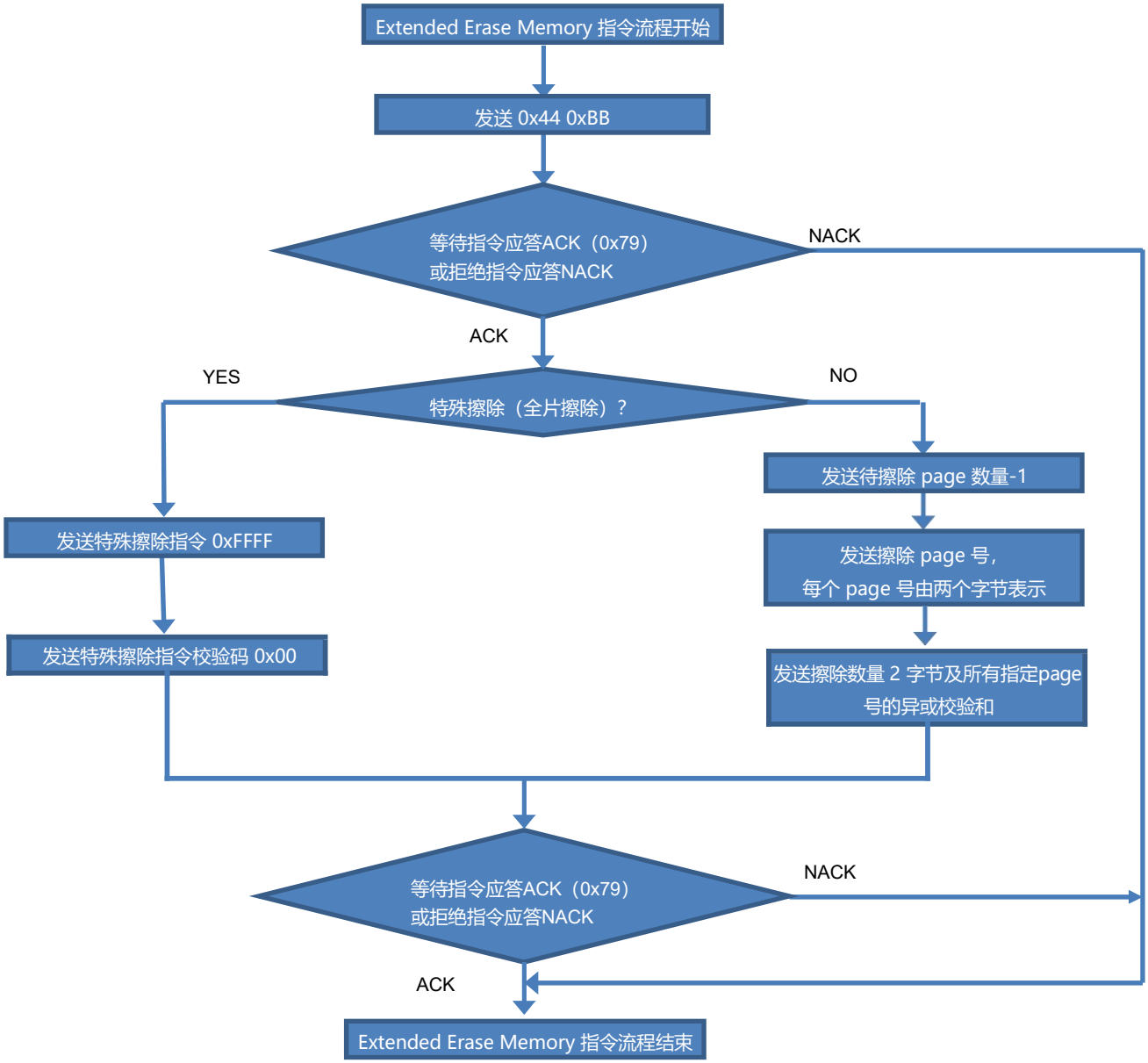

图 4-7 Extended Erase Memory 指令流程

Extended Erase Memory 指令FT32 接收回复流程示例 1（擦除整片main flash）：

示例流程开始：

上位机端发送Byte1~2： 0x44 0xBB（Extended Erase Memory 指令）

从机端回复Byte1： 0x79（ACK）

上位机端发送 Byte3~4： 0xFF 0xFF（上位机全片擦除指令码）

上位机端发送 Byte5： 0x00（校验字节 0xFF xor 0xFF = 0x00）

从机端回复 Byte2： 0x79（ACK）

示例流程结束。

Extended Erase Memory 指令FT32 接收回复流程示例 2 （擦除main flash 内指定数量的指定 page 区）：

示例流程开始：

上位机端发送Byte1~2： 0x44 0xBB（Extended Erase Memory 指令）

从机端回复Byte1： 0x79（ACK）

上位机端发送Byte3~4：0x00 0x04（上位机待擦除page 数量-1，即实际待擦除数量为4+1=5 page）

上位机端发送Byte5~14：0x00 0x10 0x00 0x29 0x00 0x56 0x00 0x58 0x00 0x36

（上位机指定擦除的 5 个指定 page 号，即第 16page，第 41page，第 86page，第 88page 以及第 54page）

上位机端发送Byte15：05（校验字节，即上述 byte3~byte14 十二个字节的异或校验和）

从机端回复Byte2： 0x79（ACK）

示例流程结束。

4.8 Write Protect 指令（0x63）

Write Protect 指令用于用户对main flash 的某些指定扇区或整片进行写保护处理。FT32 每个扇区（sector）组织为 4k bytes。

当 Bootloader 接收到此指令后，会回复上位机 ACK，并进入等待上位机发送待写保护扇区数量，上位机应发送 1byte 待保护扇区数量 N-1（实际发送待保护扇区数量 -1），后面跟随N 字节待保护的各个扇区号（每个扇区号 1 字节），最后发送扇区数量及各个扇区号的异或校验和（即格式为 1Bytes sector 数量(-1) + NBytes 扇区号 + 1Bytes XOR checksum = N+2 Bytes）。当 Bootloader 接收到信息后，进行校验，并回复上位机 ACK（校验通过）或 NACK（校验未通过）。

当校验通过且合理，Bootloader 对相应的扇区进行写保护开启，全部指定扇区开启写保护后，Bootloader 回复上位机ACK，并通过复位对写保护状态进行加载，以便于新的写保护状态生效。

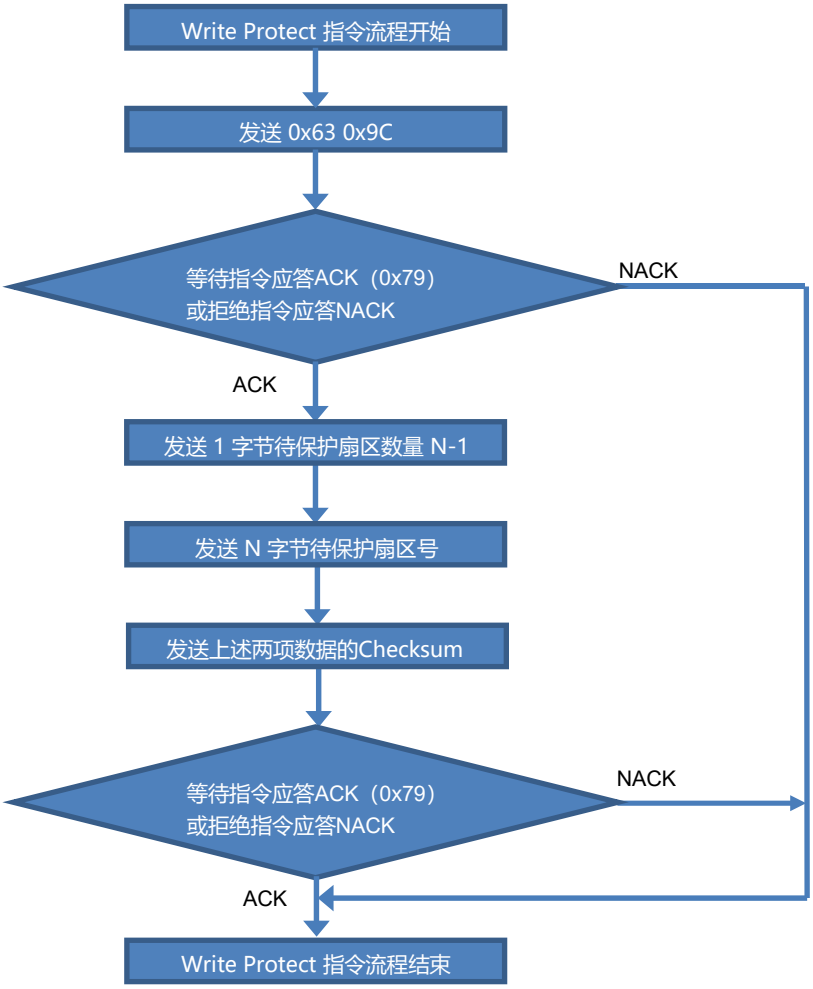

图 4-8 Write Protect 指令流程

Write Protect 指令 FT32 接收回复流程示例（写保护指定的 3 个扇区）：

示例流程开始：

上位机端发送Byte1~2： 0x63 0x9C（Write Protect 指令）

从机端回复Byte1： 0x79（ACK）

上位机端发送Byte3： 0x02（写保护扇区数量-1，即实际指定写保护扇区数为 2+1 = 3 个扇区）

上位机端发送Byte4： 0x02（指定 2 号扇区写保护）

上位机端发送Byte5： 0x03（指定 3 号扇区写保护）

上位机端发送Byte6： 0x04（指定 4 号扇区写保护）

上位机端发送Byte7： 0x07（校验字节，即 byte3~byte6 的异或校验和）

从机端回复Byte2： 0x79（ACK）

示例流程结束。

4.9 Write Unprotect 指令（0x73）

Write Unprotect 指令用于用户对main flash 整区进行写保护解除处理。

当 Bootloader 接收到此指令后，会回复上位机 ACK，并执行全片写保护解除，全部扇区解除写保护后，Bootloader 回复上位机ACK，并通过复位对写保护状态进行加载，以便于写保护解除状态生效。

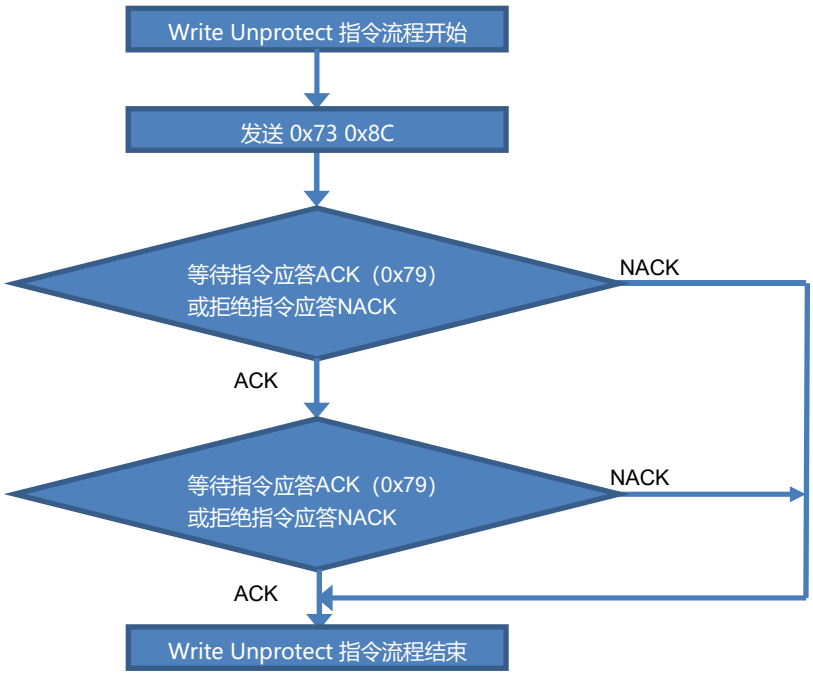

图 4-9 Write Unprotect 指令流程

Write Unprotect 指令 FT32 接收回复流程示例（main flash 全部写保护解除）：

示例流程开始：

上位机端发送Byte1~2： 0x73 0x8C（Write Unprotect 指令）

从机端回复Byte1： 0x79（ACK）

从机端回复Byte2： 0x79（ACK）

示例流程结束。

4.10 Readout Protect 指令（0x82）

Readout Protect 指令用于用户对main flash 整区进行读出保护处理。

当 Bootloader 接收到此指令后，会回复上位机 ACK，并执行全片读出保护，完成读出保护后，Bootloader 回复上位机ACK，并通过复位对新的保护状态进行加载，以便于读保护状态生效。

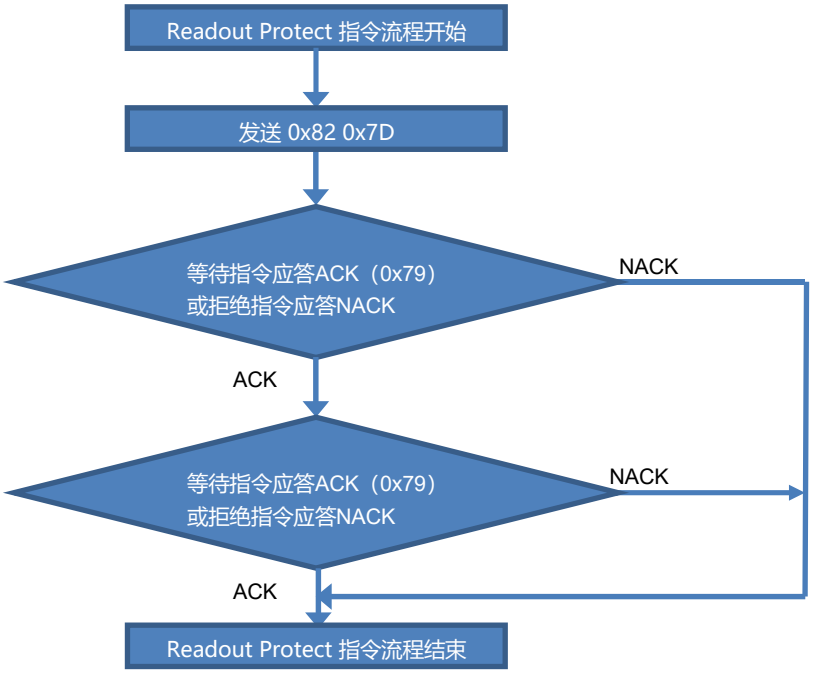

图 4-10 Readout Protect 指令流程

Readout Protect 指令FT32 接收回复流程示例（main flash 读保护）：

示例流程开始：

上位机端发送Byte1~2： 0x82 0x7D（Readout Protect 指令）

从机端回复Byte1： 0x79（ACK）

从机端回复Byte2： 0x79（ACK）

示例流程结束。

4.11 Readout Unprotect 指令（0x92）

Readout Unprotect 指令用于用户对main flash 的读出保护功能进行解除。

当 Bootloader 接收到此指令后，会回复上位机 ACK，并执行全片读出保护解除，此时会擦除 main flash 全部数据，若擦除动作或保护解除动作完成异常，Bootloader 回复上位机NACK；若正常完成读出保护解除动作，Bootloader 回复上位机ACK，并通过复位对新的保护状态进行加载，以便于读保护解除状态生效。

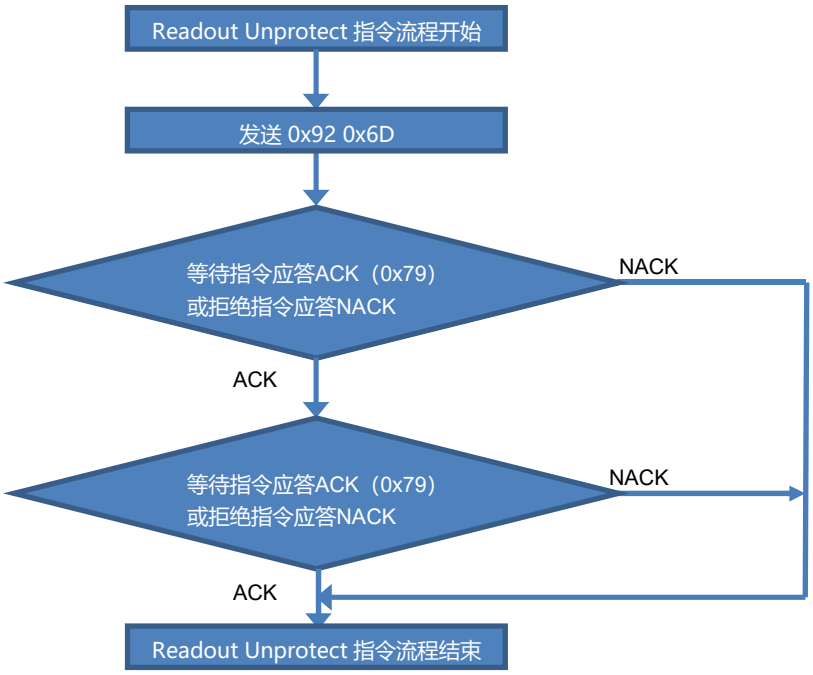

图 4-11 Readout Unprotect 指令流程

Readout Unprotect 指令 FT32 接收回复流程示例（main flash 读保护解除）：

示例流程开始：

上位机端发送Byte1~2： 0x92 0x6D（Readout Unprotect 指令）

从机端回复Byte1： 0x79（ACK）

从机端回复Byte2： 0x79（ACK）

示例流程结束。

联系信息

4th Floor, Building 11,
Zhongke Innovation Plaza, 150 Pubin Road, Jiangbei New District,
Nanjing city, Jiangsu Province

Tel: 025-58101616
Fax: 025-58263220

http://www.touchmcu.com