AN3154
应用笔记

STM32自举程序中使用的CAN协议

前言

本应用笔记说明了STM32微控制器自举程序中使用的CAN协议。它详细说明了每个支持的指令。

本文档适用于内置V3.x、V7.x和V9.x版自举程序的STM32产品，如应用笔记AN2606“*STM32微控制器系统存储器自举模式*”（参见*www.st.com*网站）中所述。这些产品已在*表 1*中列出，在整篇文档中统称为STM32。

关于所使用器件自举程序的CAN硬件资源和要求的更多信息，请参考上文提到的AN2606。

表1. 适用产品

类型	料号或产品系列
微控制器	STM32F1系列： – STM32F105xx、STM32F107xx STM32F2系列 STM32F4系列： – STM32F405xx、STM32F407xx、STM32F412xx、STM32F415xx、STM32F417xx、STM32F427xx、STM32F429xx、STM32F437xx、STM32F439xx、STM32F446xx、STM32F469xx、STM32F479xx STM32F7系列： – STM32F745xx、STM32F746xx、STM32F756xx、STM32F765xx、STM32F767xx、STM32F769xx、STM32F777xx、STM32F779xx STM32L4系列： – STM32L431xx、STM32L432xx、STM32L433xx、STM32L442xx、STM32L443xx、STM32L476xx、STM32L486xx

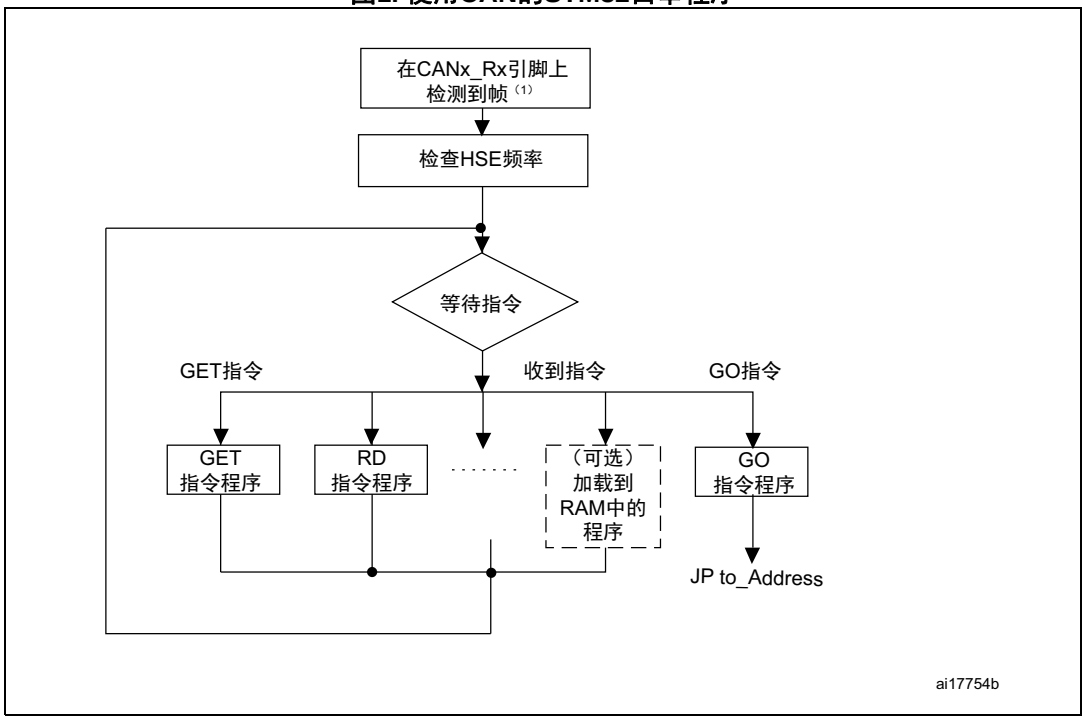
目录

表格索引

图片索引

1 自举程序代码序列

图1. 使用CAN的STM32自举程序

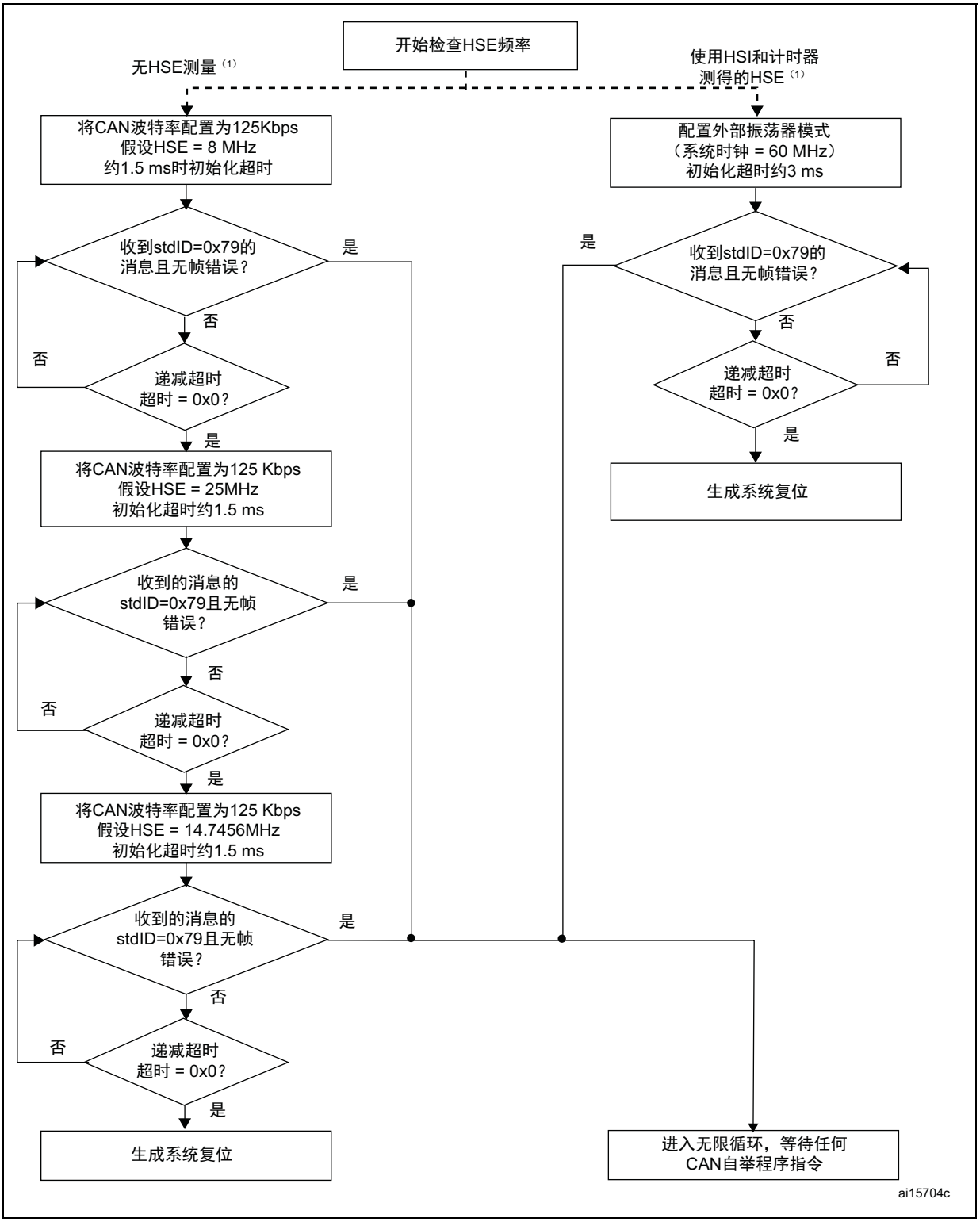


1. 建议使用标准ID = 0x79发送帧。

在进入系统存储器自举模式并且STM32器件配置完毕后（更多详细信息，请参见AN2606），自举程序代码会等待CANx_Rx引脚上的帧。当检测发生时，CAN自举程序固件开始检查外部时钟频率。

*图 2*显示了频率检查的流程图。

图2. 检查HSE频率



1. 对于某些器件，使用连接到计时器的HSI振荡器计算HSE频率。对于其他器件，不实施此类测量。对于无HSE频率测量的器件，仅执行左侧的流程，而对于有HSE频率测量的器件，则仅执行右侧的流程。预知哪个流程适用于所使用器件，请参见AN2606。

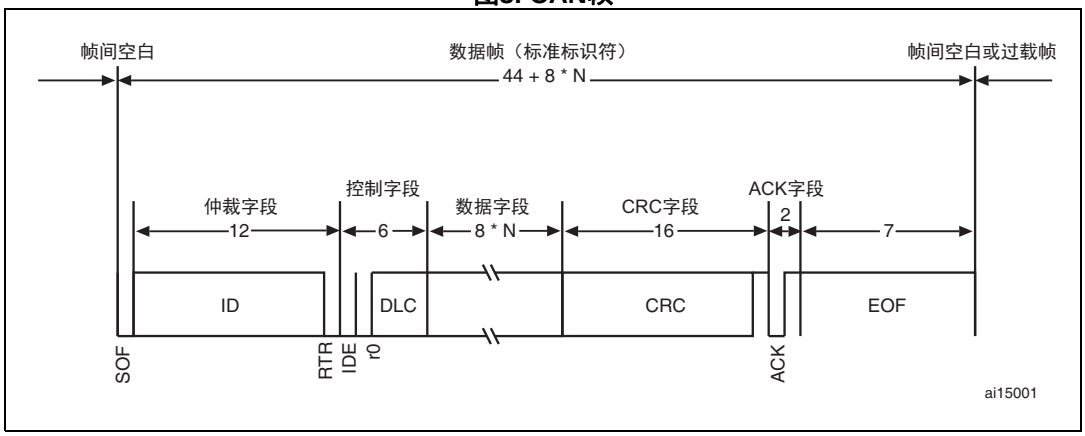
随后，代码将相应初始化串行接口。使用此计算波特率向主机返回确认字节（*0x79*），表明STM32已准备接收指令。

2 CAN设置

STM32 CAN与2.0A和B（主动）规范兼容，比特率最高达1 Mbit/s。它可接收和发送包含 11 位标识符的标准帧和包含 29 位标识符的扩展帧。

*图 3*显示了仅使用标准标识符的CAN帧。

图3. CAN帧



此应用中的CAN设置如下：

- 标准标识符（未延长）
- 比特率：开始时为125 kbps；运行期间可通过速度指令进行更改，以达到最大比特率1 Mbps。

发送设置（从STM32至主机）如下：

- Tx mailbox0：开
- Tx mailbox1和Tx mailbox2：关
- Tx标识符：（0x00、0x01、0x02、v03、0x11、0x21、0x31、0x43、0x63、0x73、0x82、0x92）

接收设置（从主机至STM32）如下：

- 同步字节0x79在RX标识符而不是数据字段内。
- RX标识符取决于指令（0x00、0x01、0x02、0x03、0x11、0x21、0x31、0x43、0x63、0x73、0x82、0x92）。
- 错误检查：如果错误字段（CAN_ESR寄存器中的[6:4]位）不是000b，则丢弃消息并向主机发送NACK。
- 当发生FIFO过载时，丢弃消息并向主机发送NACK。
- 传入消息可包含1至8个数据字节。

注： *CAN自举程序固件每次仅支持一个节点。这意味着固件不支持CAN网络管理。*

3 自举程序指令集

*表 2*中列出了支持的指令，本节将对其中的每一个进行说明。

表2. CAN自举程序指令

指令	指令代码	指令说明
Get(1)	0x00	获取自举程序当前版本所支持的版本和允许的指令。
Get Version & Read Protection Status(1)	0x01	获取自举程序版本和闪存的读保护状态
Get ID(1)	0x02	获取芯片ID
速度	0x03	速度指令用于更改CAN运行时间的波特率。
Read Memory(2)	0x11	从存储器读取最多256字节，由应用指定起始地址
Go(2)	0x21	跳转到内部Flash存储器或SRAM中的用户应用代码
Write Memory(2)	0x31	向RAM或Flash存储器写入最多256字节，由应用指定起始地址
擦除(2)	0x43	擦除一个到所有的闪存扇区
Write Protect	0x63	对一些扇区使能写保护
Write Unprotect	0x73	对所有Flash扇区禁用写保护
Readout Protect(1)	0x82	使能读保护
Readout Unprotect(1)	0x92	禁用读保护

1. 读保护 - 当RDP（读保护）选项激活时，仅能使用此有限子集的指令。所有其它指令都会被NACK，对器件没有作用。取消RDP之后，其它指令变为激活。
2. 若需了解哪些存储器空间可执行这些指令，请参见STM32产品数据手册和AN2606。

通信安全

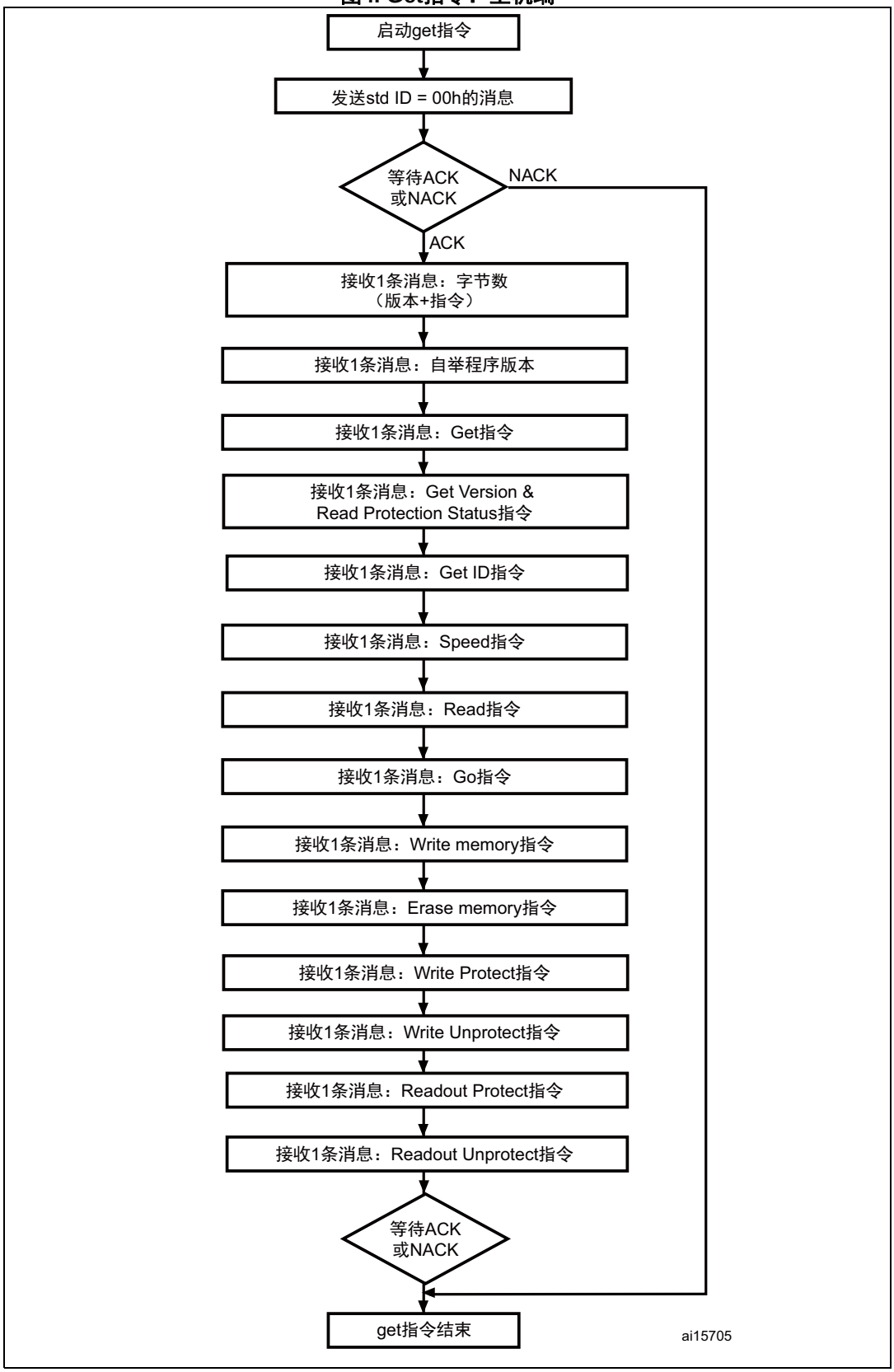
每个包或被接受（ACK应答）或丢弃（NACK应答）：

- ACK 消息 = 0x79
- NACK 消息 = 0x1F

3.1 Get指令

Get指令可帮主机得到自举程序版本及所支持的指令。当自举程序收到get指令时，它将自举程序版本和所支持的指令代码发送给主机。

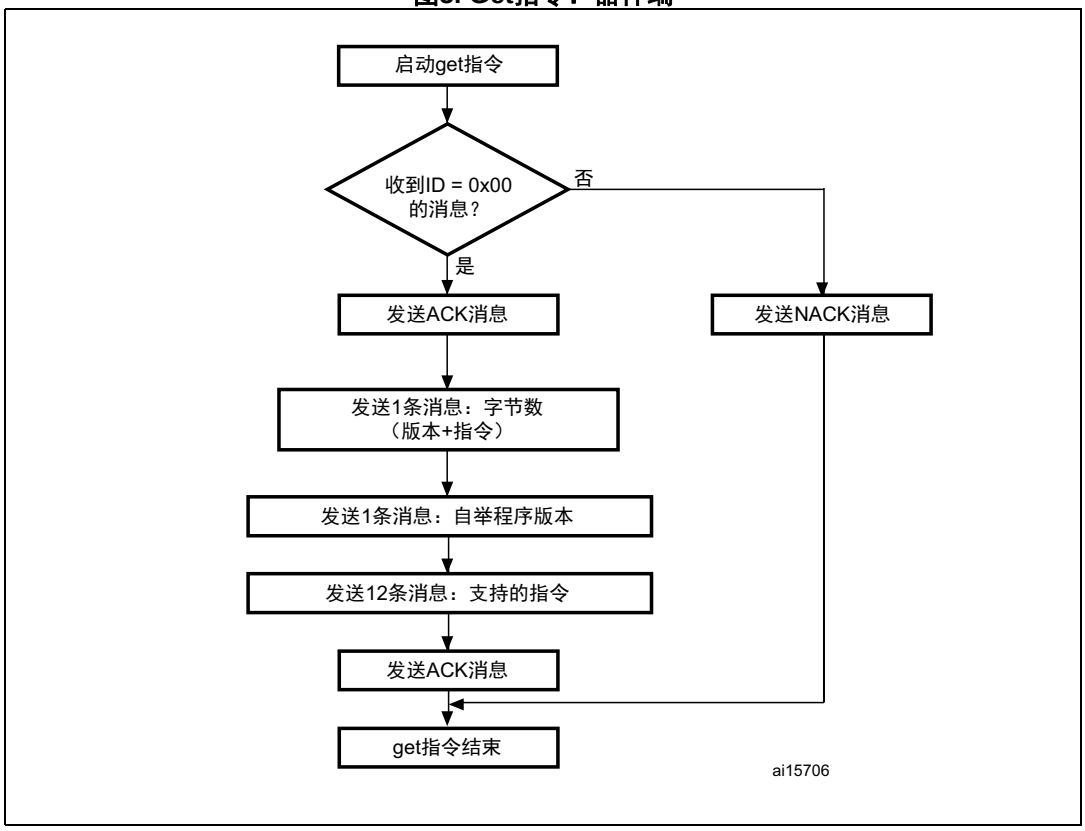
图4. Get指令：主机端


启动get指令
发送std ID = 00h的消息
等待ACK
或NACK
NACK
ACK
接收1条消息：字节数
（版本+指令）
接收1条消息：自举程序版本
接收1条消息：Get指令
接收1条消息：Get Version &
Read Protection Status指令
接收1条消息：Get ID指令
接收1条消息：Speed指令
接收1条消息：Read指令
接收1条消息：Go指令
接收1条消息：Write memory指令
接收1条消息：Erase memory指令
接收1条消息：Write Protect指令
接收1条消息：Write Unprotect指令
接收1条消息：Readout Protect指令
接收1条消息：Readout Unprotect指令
等待ACK
或NACK
get指令结束
ai15705

主机发送的消息如下：

指令消息：Std ID = 0x00，数据长度代码（DLC）=“not important”。

图5. Get指令：器件端



STM32发送的消息如下：

消息1：Std ID = 0x00，DLC = 1，数据 = 0x79 - ACK

消息2：Std ID = 0x00，DLC = 1，数据 = N = 12 = 要发送的字节数 - 1 （1 ≤ N +1 ≤ 256）

消息3：Std ID = 0x00，DLC = 1，数据 = 自举程序版本（0 < 版本 ≤ 255）

消息4：Std ID = 0x00，DLC = 1，数据 = 0x00 - Get指令

消息5：Std ID = 0x00，DLC = 1，数据 = 0x01 - Get Version & Read Protection Status指令

消息6：Std ID = 0x00，DLC = 1，数据 = 0x02 - Get ID指令

消息7：Std ID = 0x00，DLC = 1，数据 = 0x03 - Speed指令

消息8：Std ID = 0x00，DLC = 1，数据 = 0x11 - Read memory指令

消息9：Std ID = 0x00，DLC = 1，数据 = 0x21 - Go指令

消息10：Std ID = 0x00，DLC = 1，数据 = 0x31 - Write memory指令

消息11：Std ID = 0x00，DLC = 1，数据 = 0x43 - Erase memory指令

消息12：Std ID = 0x00，DLC = 1，数据 = 0x63 - Write Protect指令

消息13：Std ID = 0x00，DLC = 1，数据 = 0x73 - Write Unprotect指令

消息14：Std ID = 0x00，DLC = 1，数据 = 82h - Readout Protect指令

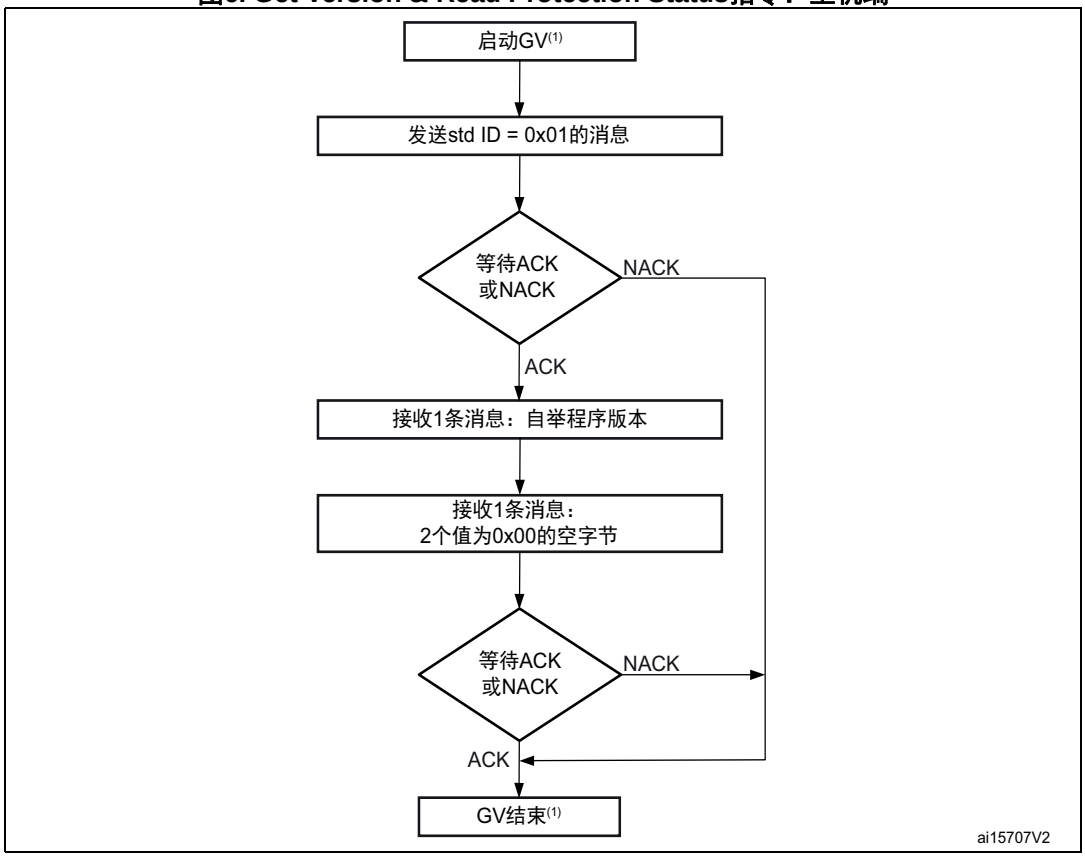
消息15：Std ID = 0x00，DLC = 1，数据 = 92h - Readout Unprotect指令

消息16：Std ID = 0x00，DLC = 1，数据 = 0x79- ACK

3.2 Get Version & Read Protection Status指令

Get Version & Read Protection Status指令用于获取自举程序版本和读保护状态。当自举程序收到该指令时，它会向主机发送如下信息（版本和2个值为0x00的空字节）。

图6. Get Version & Read Protection Status指令：主机端



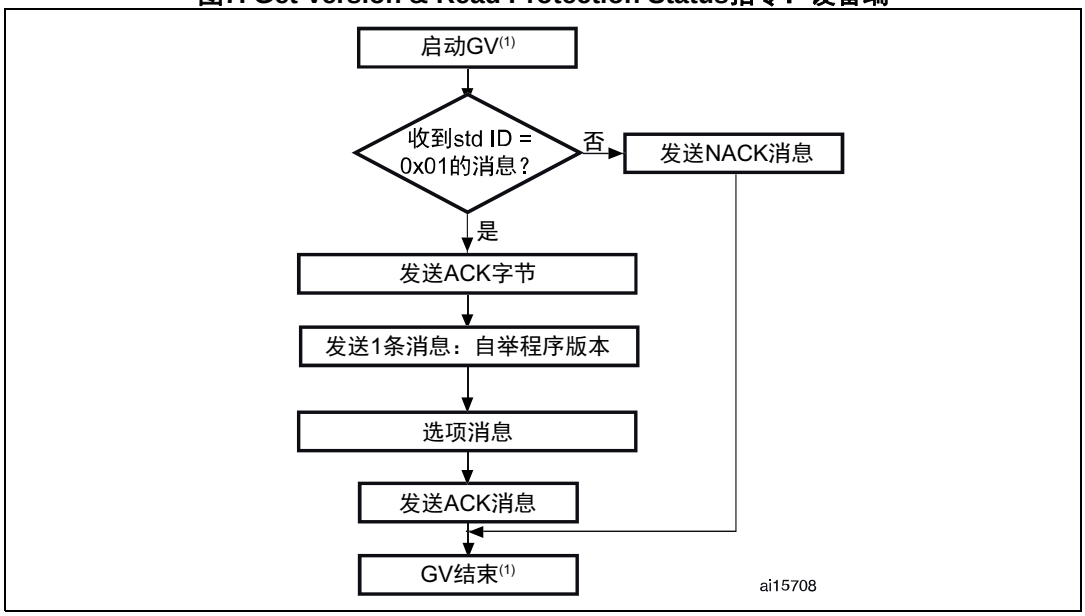
1. GV = Get Version & Read Protection Status。

主机发送的消息如下：

指令消息：Std ID = 0x01，数据长度代码（DLC）=“not important”。

ACK消息包含：Std ID = 0x01，DLC = 1，数据 = 0x79 - ACK

图7. Get Version & Read Protection Status指令：设备端



1. GV = Get Version & Read Protection Status。

STM32发送的消息如下：

消息1： Std ID = 0x01，DLC = 1，数据 = ACK

消息2： Std ID = 0x01，DLC = 1，data[0] = 自举程序版本（0 < 版本 ≤ 255），示例：0x10 = 版本1.0

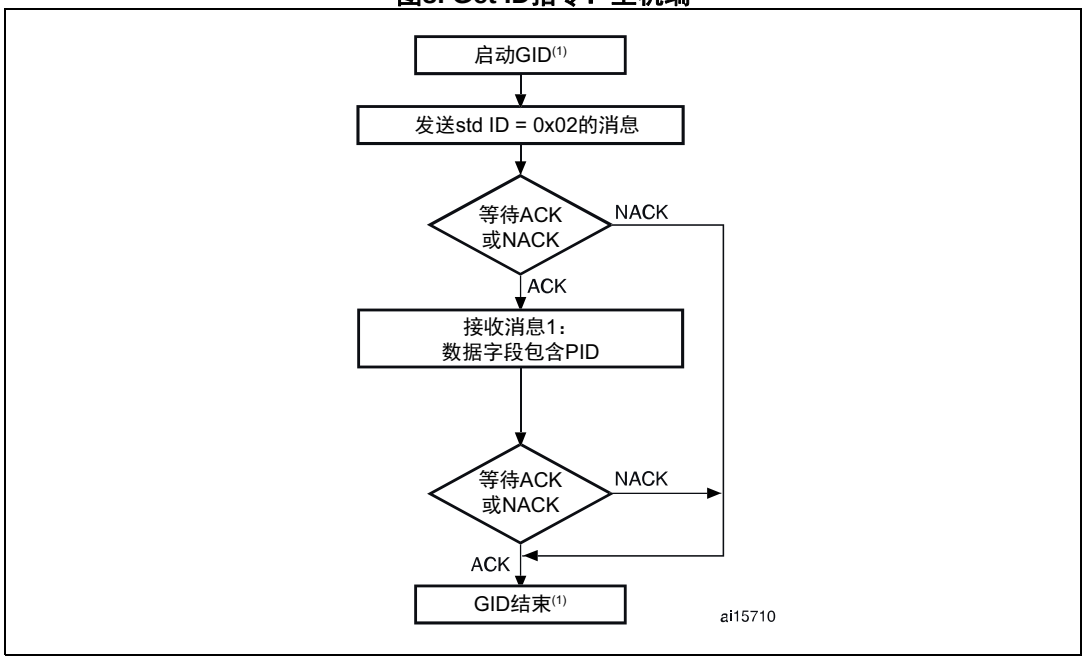
消息3： 选项消息1：Std ID = 0x01，DLC = 2，数据 = 0x00（byte1和byte2）

消息4： Std ID = 0x01，DLC = 1，数据 = ACK

3.3 Get ID指令

Get ID指令用于得到芯片ID（标识）的版本。当自举程序收到该指令时，它会向主机发送产品ID。

图8. Get ID指令：主机端



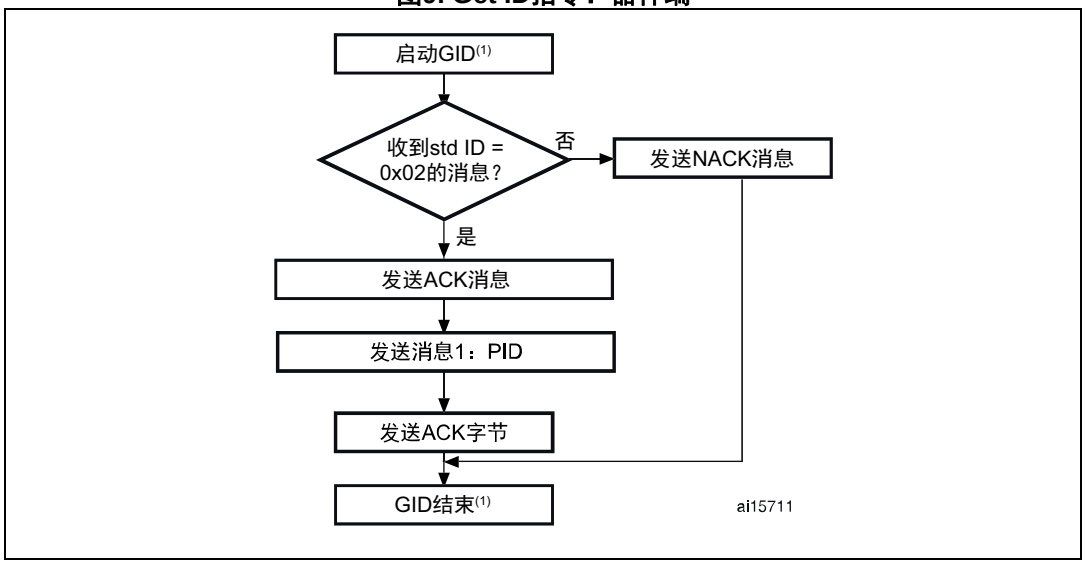
1. GID = Get ID.
2. PID代表产品ID。字节1为MSB，字节2为地址的LSB。更多关于所使用器件的PID的信息，请参见*第 3.1 节：Get指令*。

主机发送的消息如下：

指令消息：Std ID = 0x02，数据长度代码（DLC）=“not important”。

ACK消息包含：Std ID = 0x02，DLC = 1，数据 = 0x79 - ACK

图9. Get ID指令：器件端



1. GID = Get ID.
2. PID代表产品ID。字节1为MSB，字节2为地址的LSB。

STM32发送的字节如下：

消息1： Std ID = 0x02，DLC = 1，数据 = ACK，DLC不包括当前消息和ACK。

消息2： Std ID=0x02，DLC=N（字节数-1。对于STM32，N=1），数据=PID，字节0为MSB，字节N为产品ID的LSB

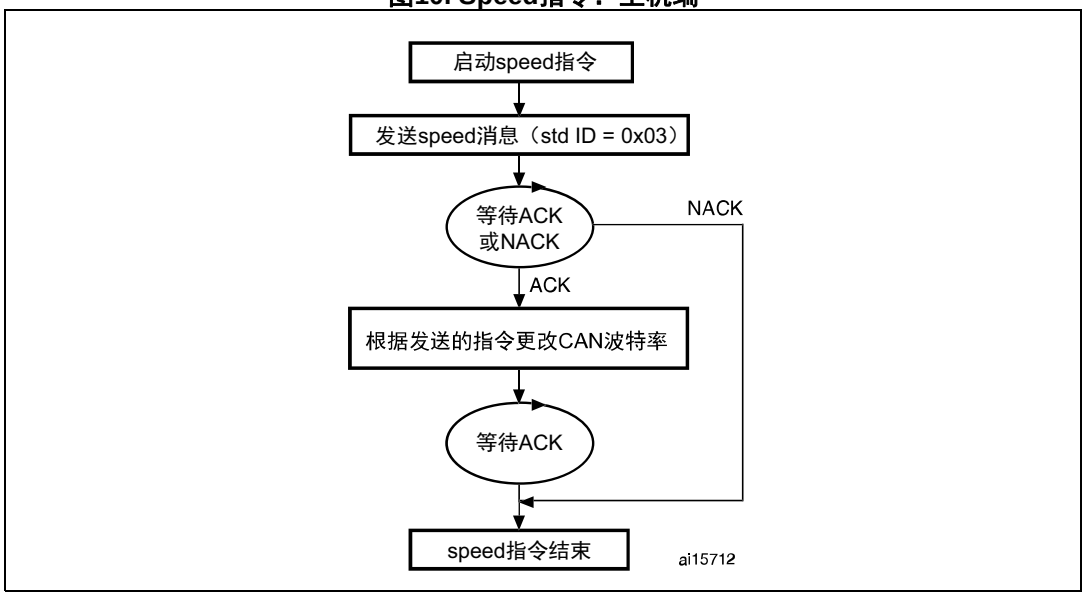
消息3： Std ID = 0x02，DLC = 1，数据 = ACK = 0x79

3.4 Speed指令

速度指令用于更改CAN运行时间的波特率。仅当CAN是正在使用的外设时，才可以使用它。

如果CAN收到正确消息但设置新波特率的操作失败（这会阻止其进入或离开初始化模式），则生成系统复位。

图10. Speed指令：主机端



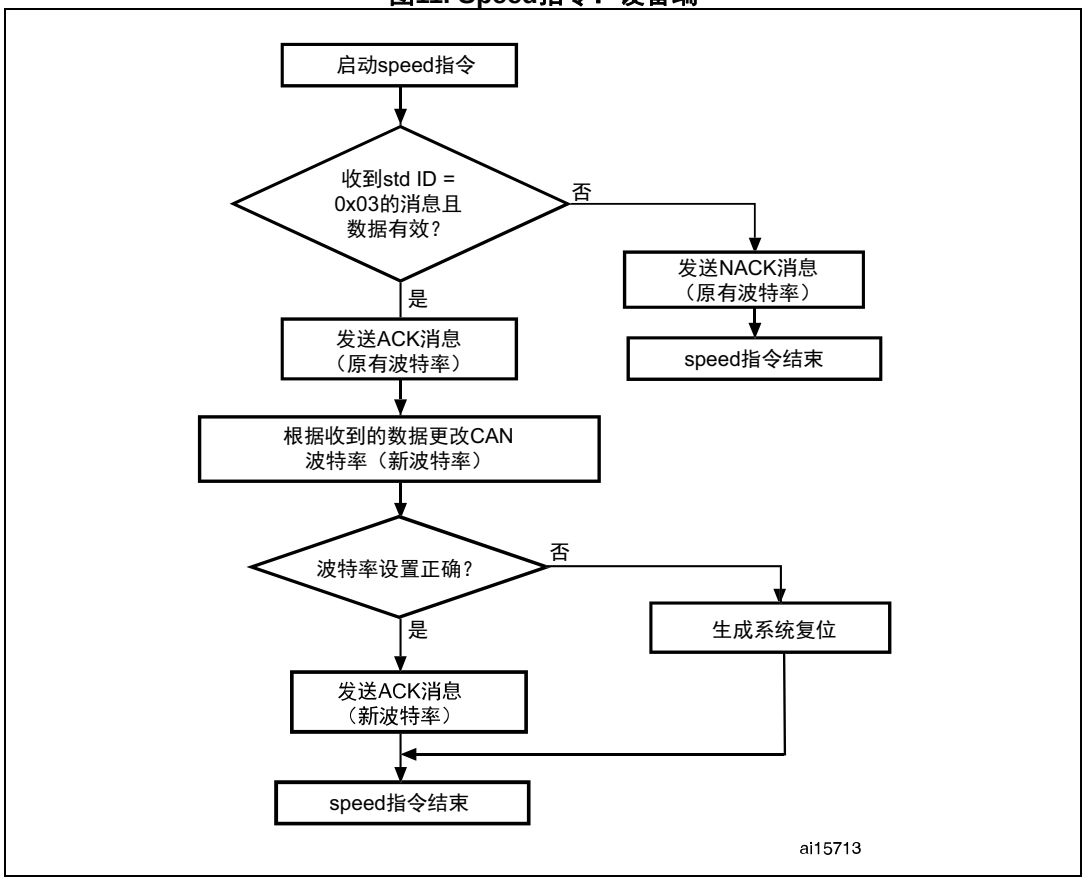
1. 在设置新波特率后，自举程序发送ACK消息。因此，主机在等待ACK时设置其波特率。

主机发送的消息如下：

指令消息：Std ID = 0x03，DLC = 0x01，data[0] = XXh，XXh根据要设置的波特率取以下值：

- 0x01：波特率 = 125 kbps
- 0x02：波特率 = 250 kbps
- 0x03：波特率 = 500 kbps
- 0x04：波特率 = 1 Mbps

图11. Speed指令：设备端



STM32发送的字节如下：

消息1： Std ID = 0x03，DLC = 1，data[0] = ACK = 0x79：使用原有波特率，如果收到的消息正确；否则data[0] = NACK = 0x1F

消息2： Std ID = 0x03，DLC = 1，data[0] = ACK = 0x79，使用新波特率

3.5 Read Memory指令

Read Memory指令用于从RAM、闪存和信息块（系统存储器或选项字节区）中的任何有效存储器地址（参见注释）读取数据。

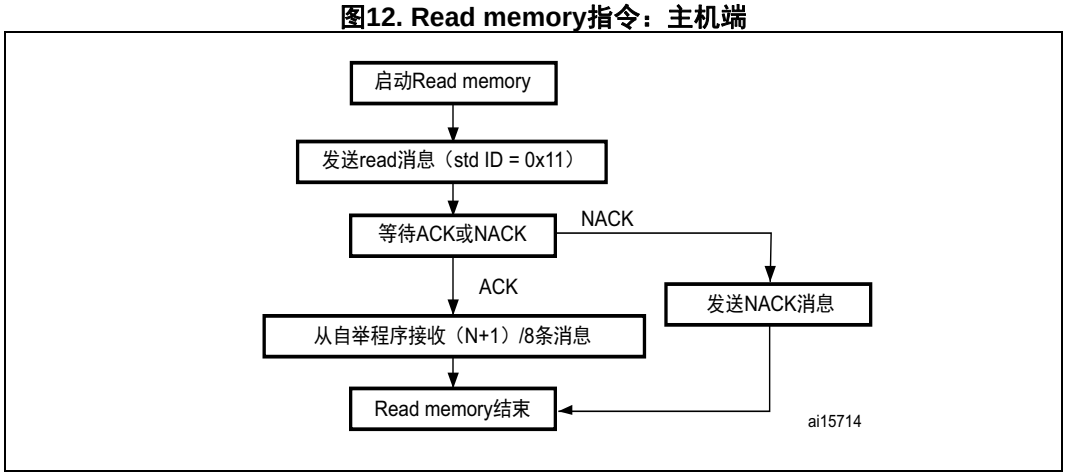
当自举程序收到Read Memory指令时，它将开始验证消息内容：

- 指令ID是否正确
- ReadOutProtection禁用还是使能
- 要读取的地址是否有效

若消息内容正确，它将发送ACK消息，否则发送NACK消息。

在发送ACK消息后，它将向应用发送所需数据通过（N+1）/8（因为每条消息包含8个字节）条消息接收（（N + 1）字节），从接收到的地址开始。

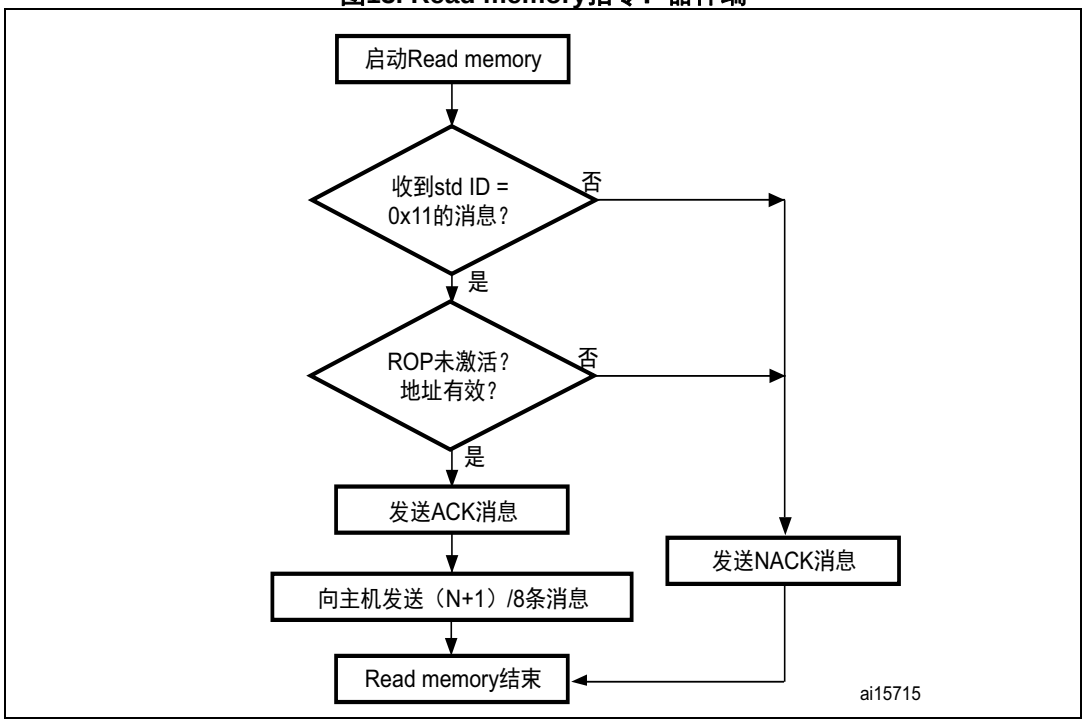
图12. Read memory指令：主机端



主机发送的消息如下：

指令消息：
Std ID = 0x11，DLC = 0x05，data[0] = 0xXX：地址的MSB，...，data[3] = 0xYY：地址的LSB，data[4] = N：要读取的字节数（其中0 < N ≤ 255）。

图13. Read memory指令：器件端



STM32发送的消息如下：

ACK消息：StdID=0x11，DLC=1，data[0]=ACK：如果指令内容正确；否则data[0]=NACK

数据消息（N+1）/ 8：Std ID = 0x11，DLC = 字节数，data[0] = 0xXX，...，data[字节数 - 1] = 0xYY

ACK消息：Std ID = 0x11，DLC = 1，data[0] = ACK

3.6 Go指令

Go指令用于执行下载的代码或任何其它代码，由应用指定要跳转到的地址。当自举程序收到Go指令时，它将开始验证消息是否包含下列有效信息：

- 指令ID是否正确
- ReadOutProtection禁用还是使能
- 分支目标地址是否有效（data[0]为地址MSB，data[3] 4为LSB）

若消息内容正确，它将发送ACK消息，否则发送NACK消息。

在向应用发送ACK消息后，自举程序固件：

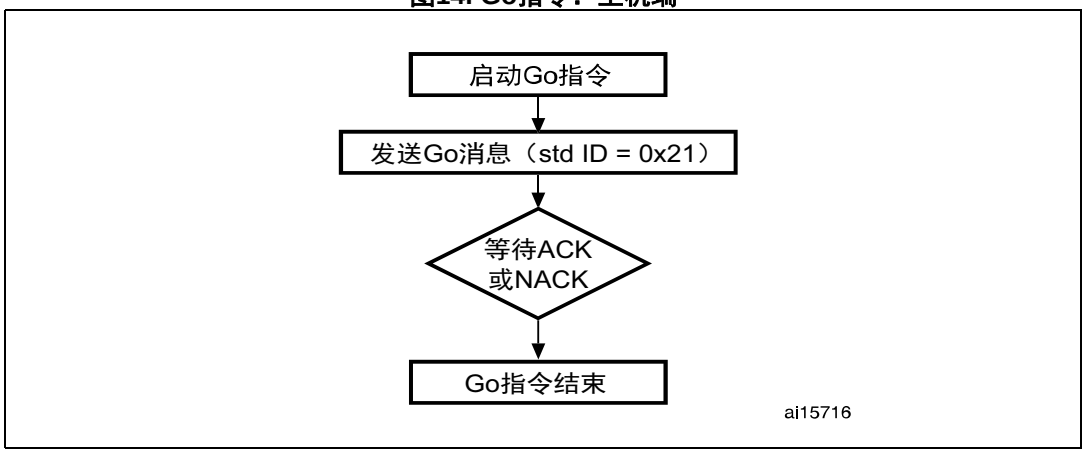
- 将自举程序所用外设的寄存器初始化至其默认复位值
- 初始化用户应用的主堆栈指针
- 跳转至收到的'地址 + 4'所编程的存储器位置（对应于应用复位处理程序的地址）。
  例如，若收到的地址为0x0800 0000，则自举程序跳转至编程为0x0800 0004地址的存储器位置。
  总之，主机发送基址，应用编程跳转。

注： *1 仅当用户应用正确设置了指向应用地址的向量表时，跳转到应用才能工作。*

2 Go指令的有效地址在RAM还是闪存中（请参见第 3.1 节：Get指令了解更多关于所使用器件的有效存储器地址的信息）。所有其他地址均被视为无效，并被器件NACK。

3 如果将应用加载到RAM然后对其执行跳转，则必须将程序配置为以一定的偏移运行，从而避免与自举程序固件使用的第一个RAM存储器重叠（请参见第 3.1 节：Get指令了解更多关于所使用器件的RAM偏移的信息）。

图14. Go指令：主机端

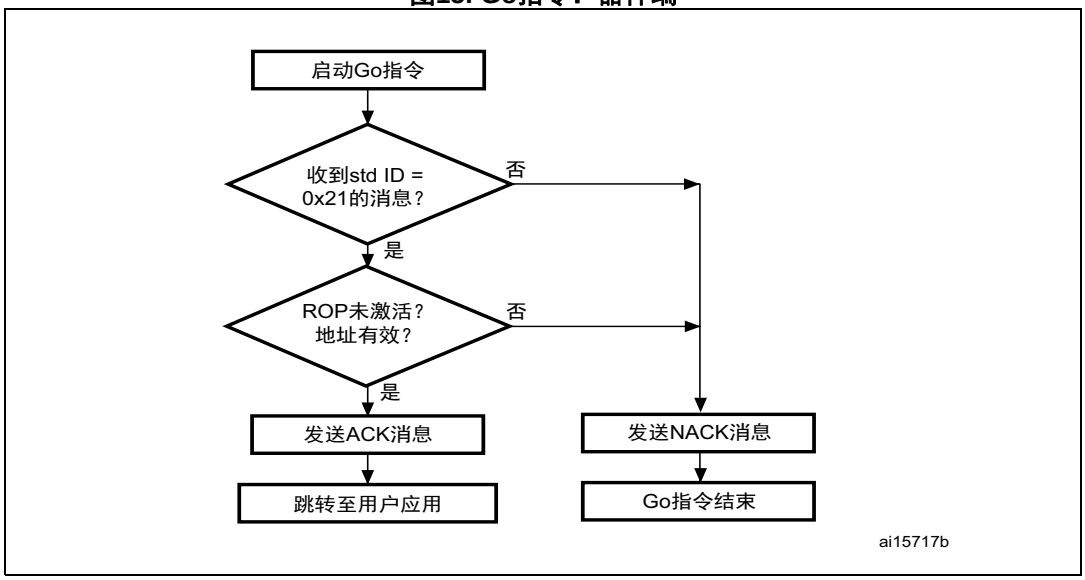


1. 参见产品数据手册了解有效地址。

主机发送的字节如下

Go指令消息：Std ID = 0x21，DLC = 0x04，data[0] = 0xXX：MSB地址，...，data[3] = 0xYY：LSB地址。

图15. Go指令：器件端



STM32发送的消息如下：

ACK消息：StdID=0x21，DLC=1，data[0]=ACK：如果指令内容正确；否则data[0]=NACK

3.7 Write Memory指令

Write Memory指令用于向RAM、Flash、选项字节区域的任何有效存储器地址（见注释）写入数据。当自举程序收到Write Memory指令时（数据长度为5字节的消息，data[0]为地址MSB，data[3]为LSB，data[4]为要接收的数据字节数），它将检查收到的地址。对于选项字节区，开始地址必须是选项字节区的基址（参见注释），从而避免在该区的不合适写入。

注： *更多关于所使用的器件的有效存储器地址信息，请参见第 3.1 节：Get指令。*

若收到的地址有效，则自举程序发送ACK消息；否则它发送NACK消息并终止该指令。如果地址有效，则自举程序会：

- 接收用户数据（N个字节），因此器件收到N/8条消息（每条消息包含8个数据字节）
- 从收到的地址开始将用户数据编程至存储器
- 在该指令结束时，若写入操作成功，则自举程序向应用发送ACK消息；否则它发送NACK消息并终止指令。

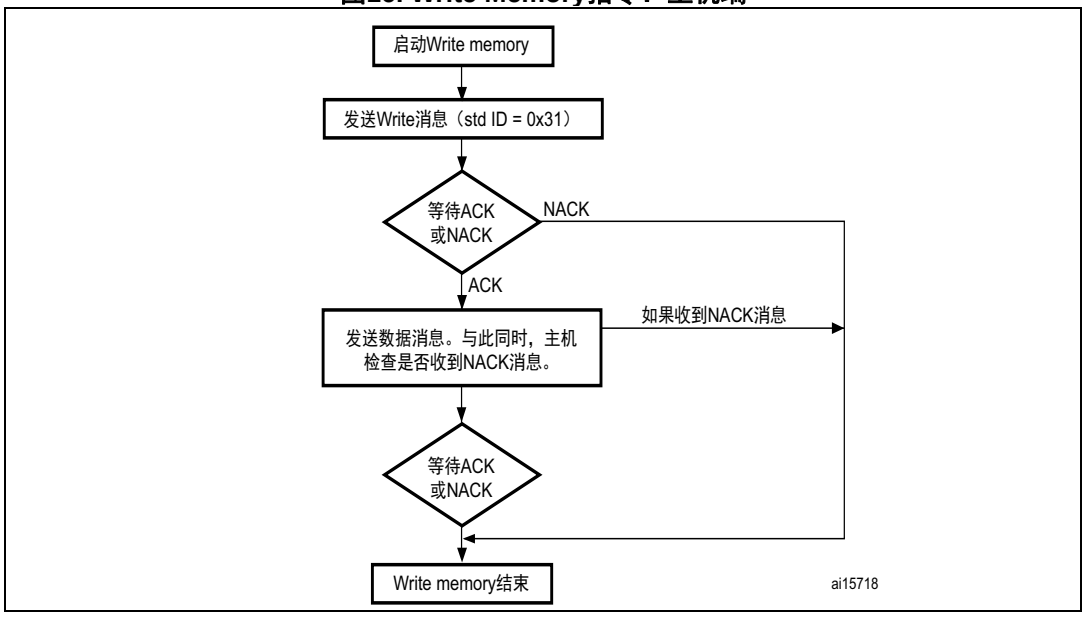
对于STM32，要写入的块的最大长度为256字节。

若Write Memory指令用于选项字节区域，则在写入新值之前会擦除所有选项。在指令末尾，自举程序会生成系统复位，以使选项字节的新配置生效。

注： 1 当写入RAM时，用户应注意不要与自举程序固件使用的第一个RAM存储器重叠。

2 当向写保护的扇区执行写操作时，不会返回错误。

图16. Write Memory指令：主机端



注： 如果开始地址无效，指令会被设备NACK。

主机发送的消息如下：

指令消息：Std ID = 0x31，DLC = 0x05，data[0] = 0xXX：MSB地址，...，data[3] = 0xYY：LSB地址，data[4] = N-1（要写入的字节数 - 1），0 < N ≤ 255）。

然后，主机发送N/8条消息

数据消息：Std ID = 0x31，DLC_1 = 1至8，数据 = byte_11，…，byte_18…

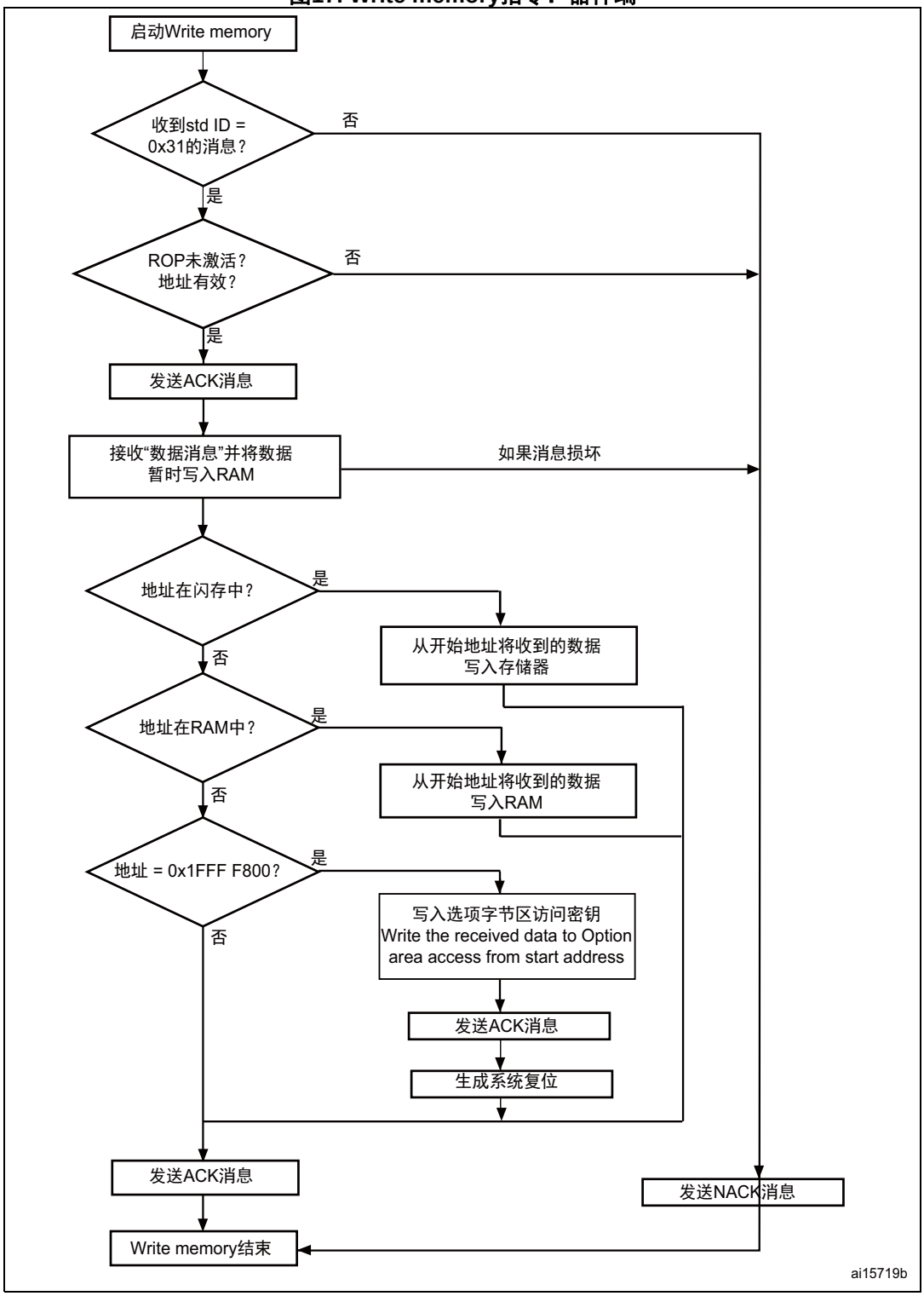
数据消息_M：Std ID = 0x04，DLC_M = 1至8，数据 = byte_m1，…，byte_M8

注： 1 DLC_1 + DLC_2 + ... DLC_M = 256（最大值）

2 在每条消息之后，主机会收到来自设备的ACK或NACK消息

3 自举程序不检查数据的标准ID，因此可以使用从0h至0xFF的任何ID。建议使用0x04。

图17. Write memory指令：器件端



STM32发送的消息如下：

ACK消息：StdID=0x31，DLC=1，data[0]=ACK：如果指令内容正确；否则data[0]=NACK

在接收每一条消息后，如果指令内容正确，设备将发送ACK，否则发送NACK。但是，如*图 17*中所述，在收到所有数据消息（N/8条消息）并将数据暂时写入RAM后，如果消息内容均无损坏，自举程序将在请求的地址（闪存、RAM或选项字节）写入N个字节。然后，如果写操作成功完成，设备将向主机发送ACK消息。

换句话说，在发送N/8条消息后，主机将相继收到两条ACK；如果N/8条消息的最后一条消息被正确接收，设备将发送第一条ACK，在请求的地址正确写入N/8条消息后，将发送第二条ACK。

3.8 Erase Memory指令

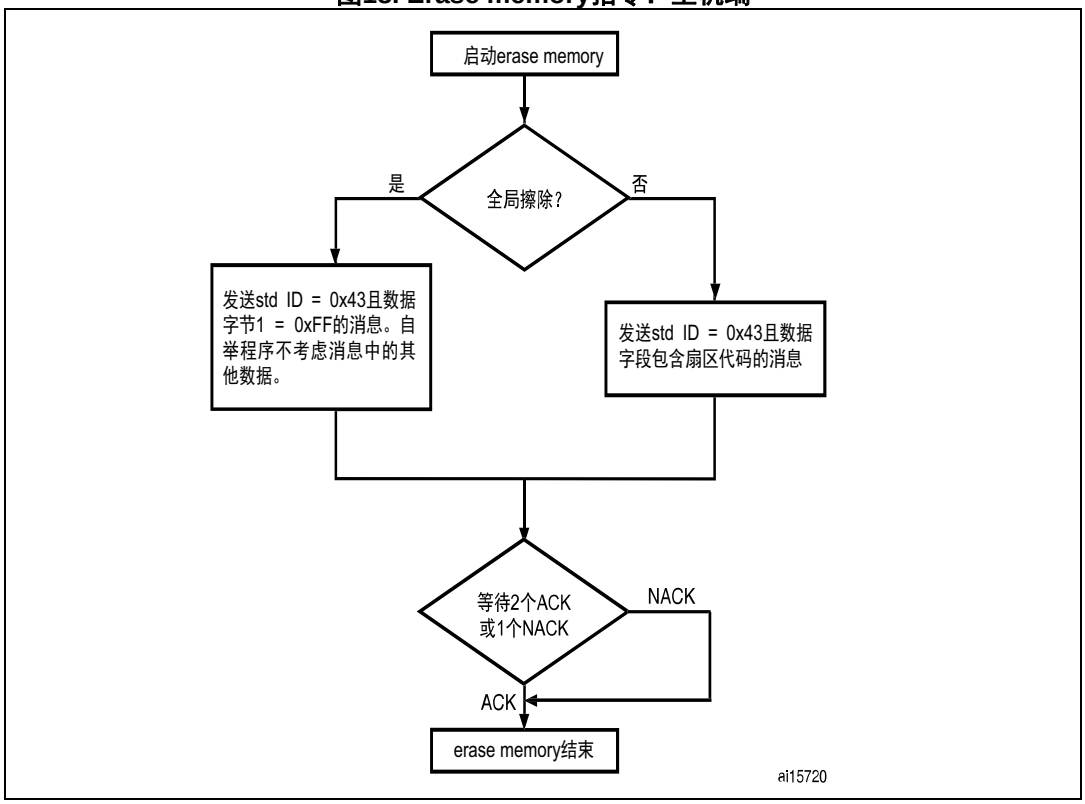
EraseMemory指令使主机能够擦除闪存页面。当自举程序收到EraseMemory指令且ROP禁用时，它会向主机发送ACK消息。在发送ACK消息后，自举程序检查data[0]是否等于0xFF，如果是，将启动全局存储器擦除操作，并在操作完成后发送ACK消息。否则（data[0]不等于0xFF），自举程序将按照主机的定义启动存储器页面擦除，并在擦除每一页后发送ACK或NACK消息。

Erase Memory指令规范：

1. 自举程序收到一条包含N（要擦除的页面数 - 1）的消息。
   N = 255预留给全局擦除请求。对于0 ≤ N ≤ 254，擦除N + 1个页面。
2. 自举程序收到（N + 1）个字节，每个字节均包含一个页号

注： *当向写保护的扇区执行擦除操作时，不会返回错误。*

图18. Erase memory指令：主机端



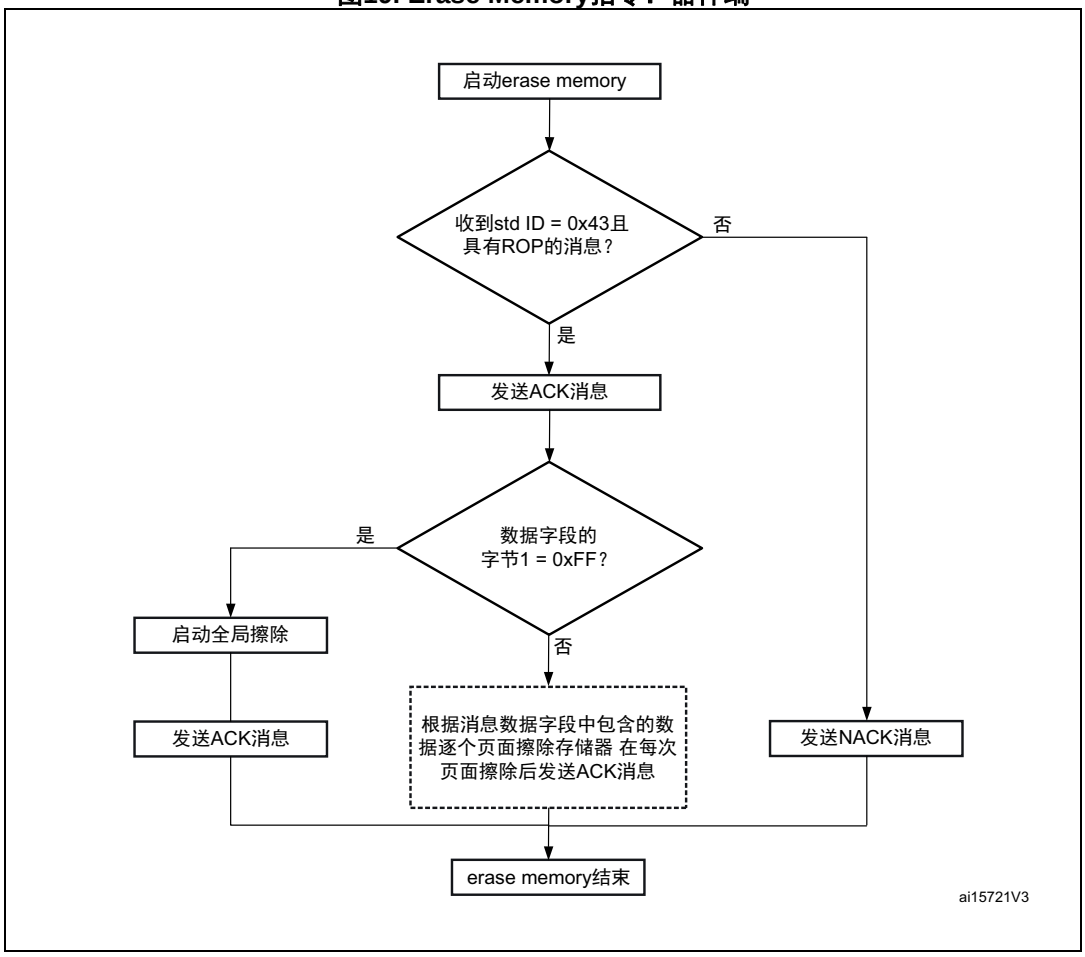
主机发送的消息如下：

ID包含的指令类型（0x43）：

- 全局擦除消息：Std ID = 0x43，DLC = 0x01，数据 = 0xFF。
- 逐个扇区擦除消息：Std ID = 0x43，DLC = 0x01至0x08，数据 = 参见产品数据手册。

如果逐个页面擦除，在每条消息之后，主机会收到来自设备的ACK或NACK消息。

图19. Erase Memory指令：器件端



STM32发送的消息如下：

ACK消息：Std ID = 0x43，DLC = 1，data[0] = ACK：如果指令内容正确且ROP未激活；否则data[0] = NACK。

3.9 Write Protect指令

Write Protect指令用于对部分或全部Flash扇区使能写保护。当自举程序收到Write Protect指令时，如果ROP禁用，它会向主机发送ACK消息，否则发送NACK。

发送ACK字节之后，自举程序等待从应用接收闪存扇区码。

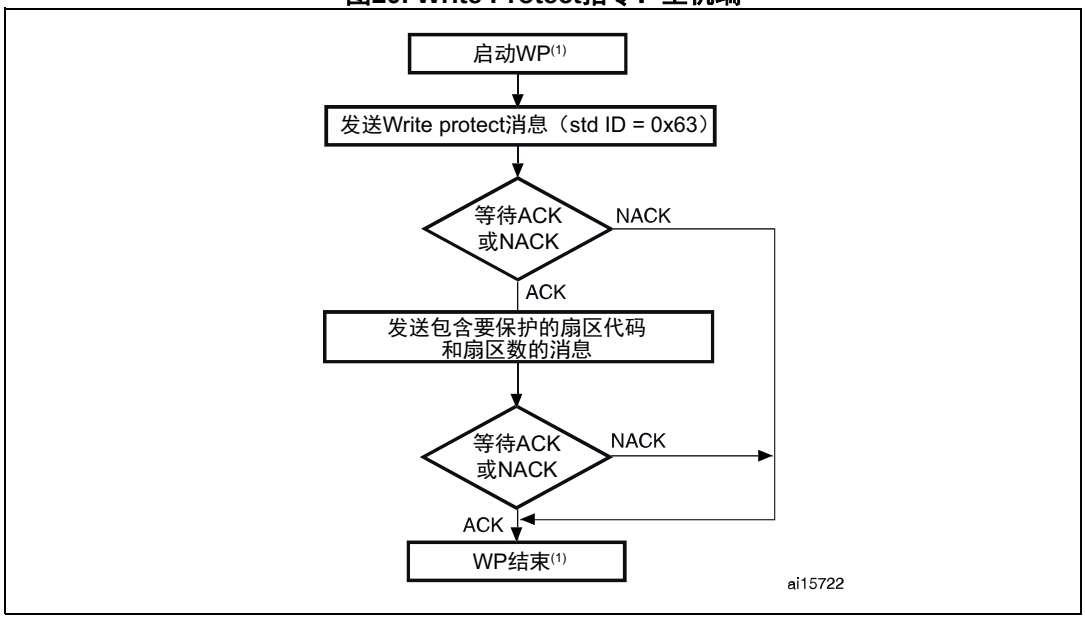
在Write Protect指令末尾，自举程序会发送ACK消息并生成系统复位，以使选项字节的新配置生效。

注： 1 *更多关于所使用器件的扇区大小的信息，请参见第 3.1 节：Get指令。*

2 *要保护的扇区总数和扇区号不经过校验，这意味着若指令中要保护的扇区总数和扇区号错误，也不会返回错误。*

若执行了第二个Write Protect指令，则第一个指令已经保护的Flash扇区会被解除保护，只有第二个Write Protect指令内的扇区才会被保护。

图20. Write Protect指令：主机端



1. WP = Write Protect.

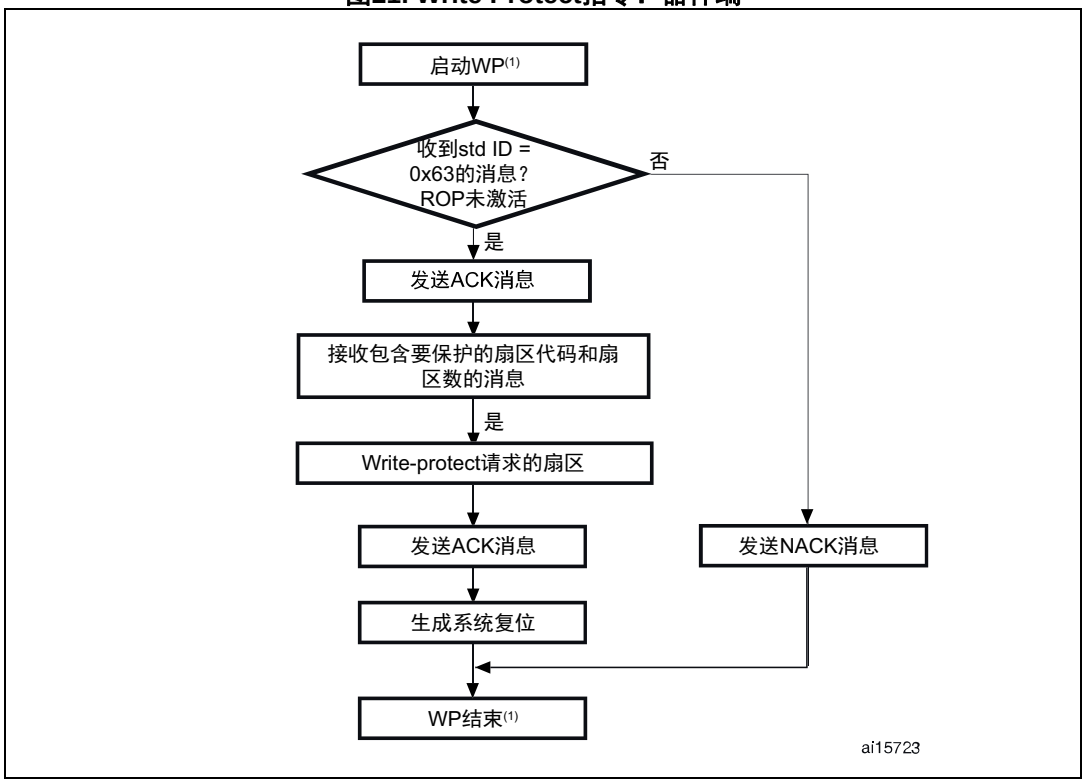
主机发送的消息如下：

指令消息：Std ID = 0x63，DLC = 0x01，data[0] = N（0 < N ≤ 255）。

指令消息：Std ID = 0x63，DLC = 0x01..08，data[0] = N（0 < N ≤ 255）。

在每条消息之后，主机会收到来自设备的ACK或NACK消息。

图21. Write Protect指令：器件端



1. WP = Write Protect

STM32发送的消息如下：

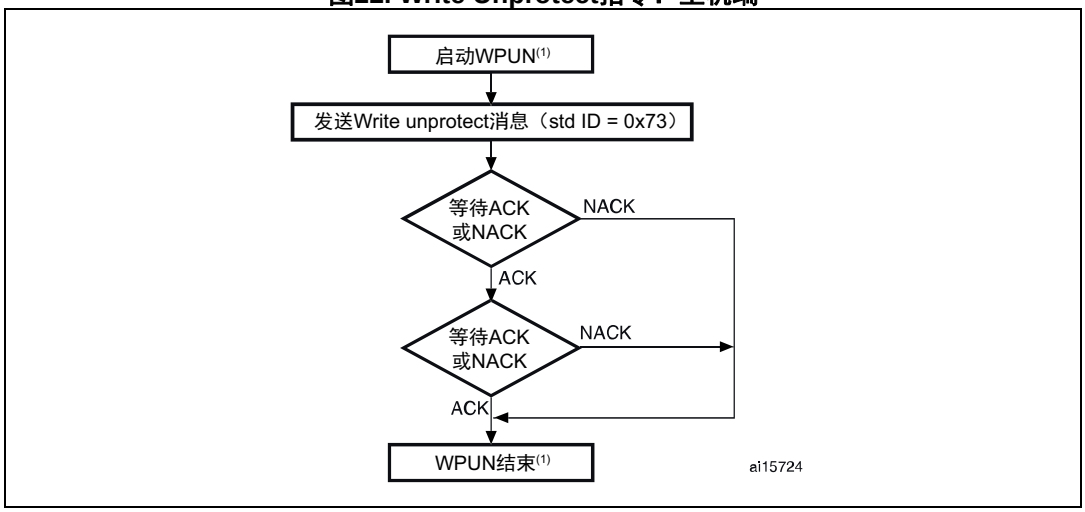
ACK消息：Std ID = 0x63，DLC = 1，data[0] = ACK：如果指令内容正确且ROP未激活；否则data[0] = NACK。

3.10 Write Unprotect指令

Write Unprotect指令用于对全部Flash扇区禁用写保护。当自举程序收到Write Unprotect指令时，如果ROP禁用，它会向主机发送ACK消息，否则发送NACK。发送完ACK消息之后，自举程序禁用所有闪存扇区的写保护。

在Write Unprotect指令末尾，自举程序会发送ACK消息并生成系统复位，以使选项字节的新配置生效。

图22. Write Unprotect指令：主机端

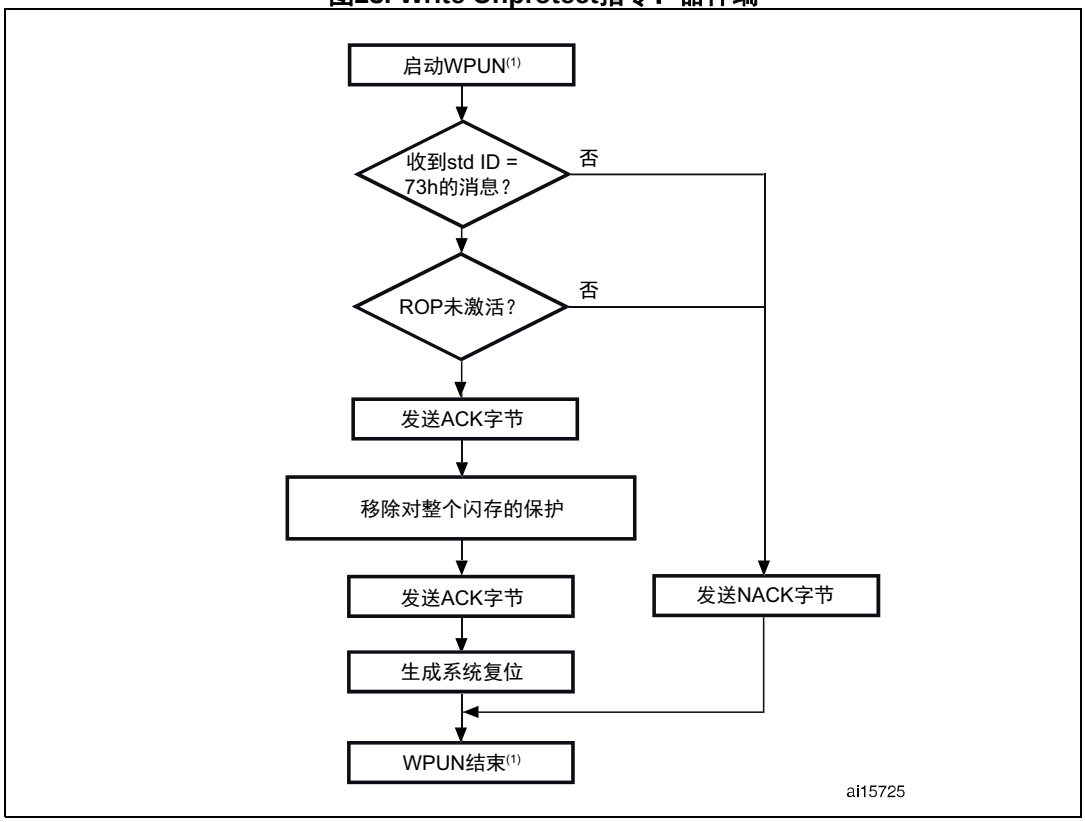


1. WPUN = Write Unprotect.

主机发送的消息如下：

指令消息：Std ID = 0x73，DLC = 0x01，数据 = 00。

图23. Write Unprotect指令：器件端



1. WPUN = Write Unprotect.

STM32发送的消息如下：

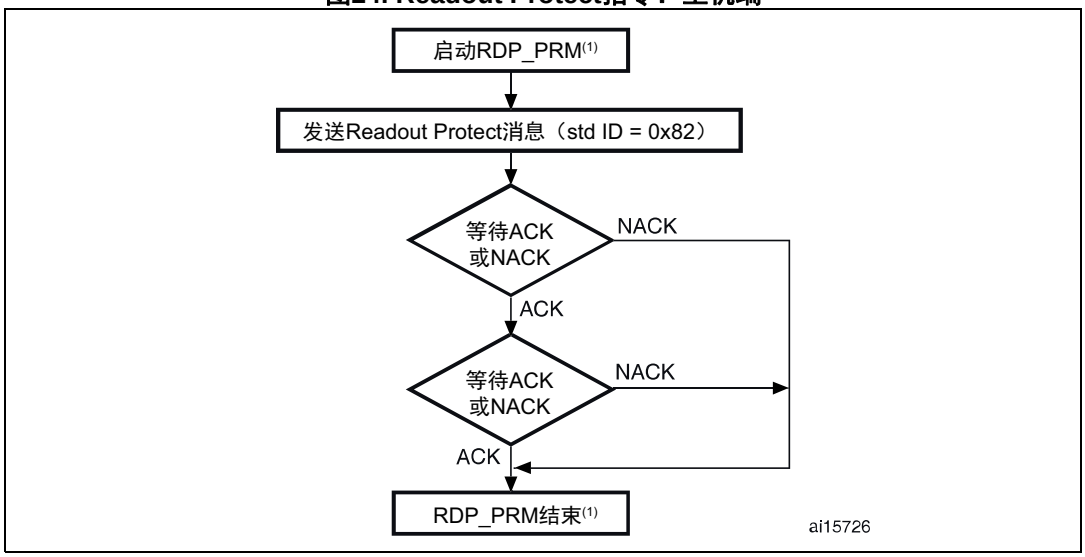
ACK消息：Std ID = 0x73，DLC = 1，data[0] = ACK：如果指令内容正确且ROP未激活；否则data[0] = NACK。

3.11 Readout Protect指令

Readout Protect指令用于使能Flash的读保护。当自举程序收到Readout Protect指令时，如果ROP禁用，它会向主机发送ACK消息，否则发送NACK。发送完ACK消息之后，自举程序使能闪存的读保护。

在Readout Protect指令末尾，自举程序会发送ACK消息并生成系统复位，以使选项字节的新配置生效。

图24. Readout Protect指令：主机端

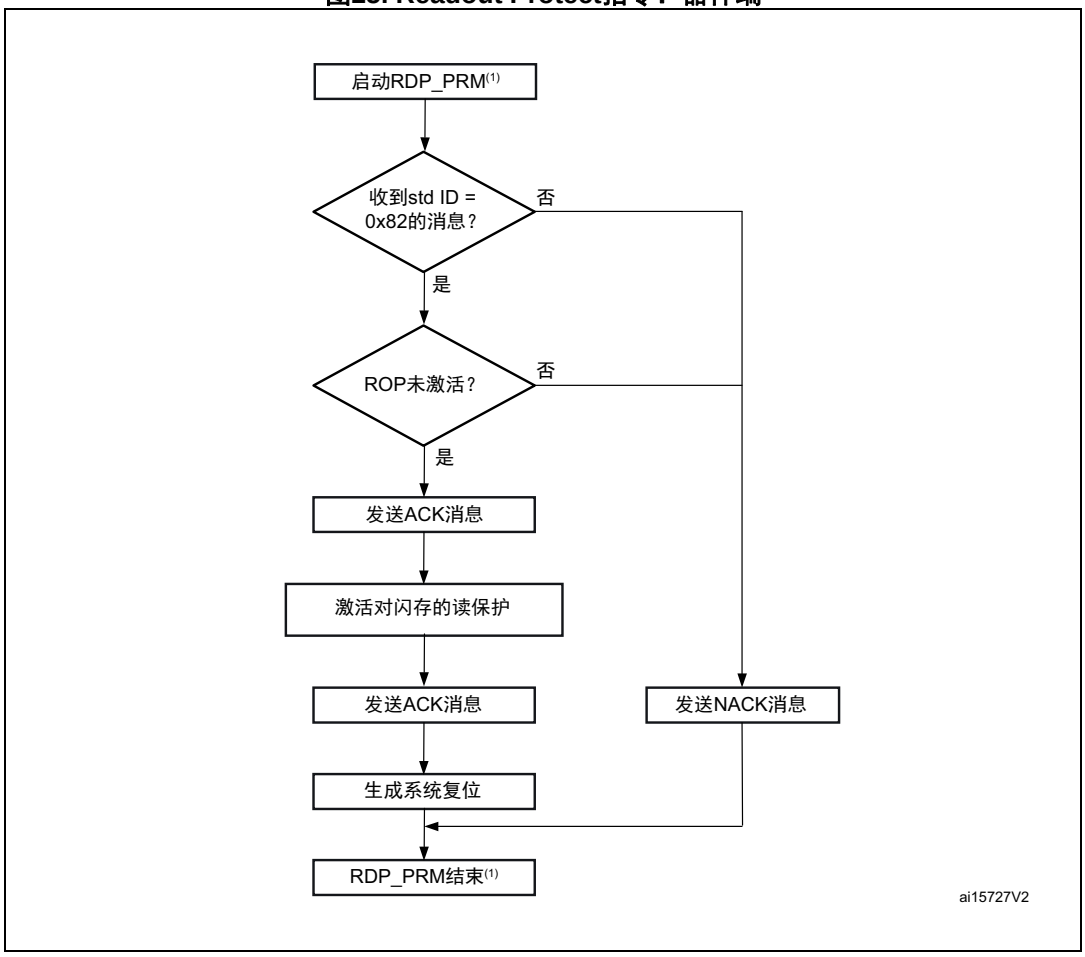


1. RDP_PRM = Readout Protect.

主机发送的消息如下

指令消息：Std ID = 0x82，DLC = 0x01，data[0] = 00。

图25. Readout Protect指令：器件端



1. RDP_PRM = Readout Protect

STM32发送的消息如下：

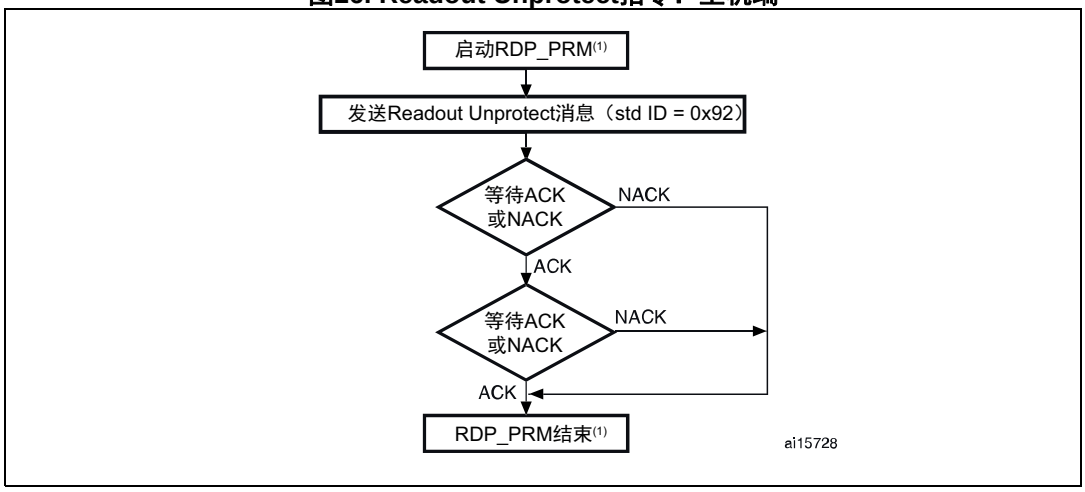
ACK消息：Std ID = 0x82，DLC = 1，data[0] = ACK：如果指令内容正确且ROP未激活；否则data[0] = NACK。

3.12 Readout Unprotect指令

Readout Unprotect指令用于禁用Flash的读保护。当自举程序收到Readout Unprotect指令时，它会向主机发送ACK消息。发送完ACK消息之后，自举程序擦除所有Flash扇区，对整个Flash禁用读保护。若擦除操作成功，则自举程序取消激活RDP。

在Readout Unprotect指令末尾，自举程序会发送ACK消息并生成系统复位，以使选项字节的新配置生效。

图26. Readout Unprotect指令：主机端

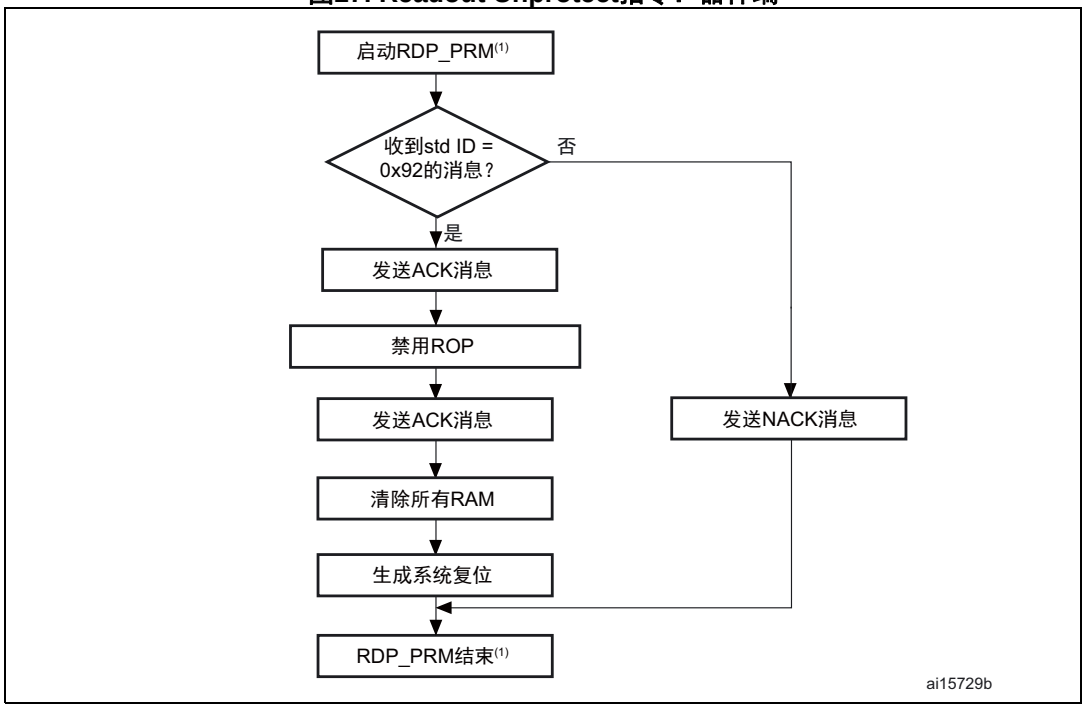


1. RDU_PRM = Readout Unprotect.

主机发送的消息如下

指令消息：Std ID = 0x92，DLC = 0x01，数据 = 00。

图27. Readout Unprotect指令：器件端



1. RDU_PRM = Readout Unprotect.

STM32发送的消息如下：

ACK消息：Std ID = 0x92，DLC = 1，data[0] = ACK：如果指令内容正确且ROP未激活；否则data[0] = NACK。

4 自举程序协议版本演进

*表 3*列出了自举程序的版本。

表3. 自举程序协议版本

版本	说明
V2.0	初始自举程序版本。

5 版本历史

表4. 文档版本历史

日期	版本	变更
2010年3月9日	1	初始版本。
2011年4月15日	2	更新了*图 2：检查HSE频率*并增加了*注 1*。
2011年4月22日	3	更新了*第 3.4节：Speed指令*中消息2的Std ID。
2012 年 10 月 24 日	4	更新了 *第 3.2节： Get Version & Read Protection Status指令* *图 6：Get Version & Read Protection Status指令：主机端* *第 3.7节： Write Memory指令* *第 3.8节： Erase Memory指令* *图 19：Erase Memory指令：器件端* *第 3.9节： Write Protect指令* *图 25：Readout Protect指令：器件端*
2014年5月2日	5	更新了*表 1：适用产品*和*表 2：CAN自举程序指令*。 删除了专门介绍器件相关自举程序参数的章节。 更新了*第 3.5节：Read Memory指令*和*第 3.7节：Write Memory指令*。
2016年10月21日	6	更新了*前言*、*第 1节：自举程序代码序列*和*第 3.1节：Get指令*。 更新了*表 1：适用产品*。

表5. 中文文档版本历史

日期	版本	变更
2018年5月11日	1	中文初始版本。

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。本文档的中文版本为英文版本的翻译件，仅供参考之用；若中文版本与英文版本有任何冲突或不一致，则以英文版本为准。