AN4221
应用笔记

STM32 自举程序中使用的 I2C 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 I2C 协议。它详细说明了每个支持的指令。

若需器件自举程序 I2C 硬件资源和要求的更多信息，请参考应用笔记 AN2606 “STM32 微控制器系统存储器自举模式 ”。

.

表 1. 适用产品

类型	料号或产品系列
微控制器	STM32F0 系列： – STM32F042xx、 STM32F072xx STM32F3 系列： – STM32F318xx、 STM32F328xx – STM32F334xx – STM32F358xx、 STM32F378xx – STM32F303x4(6/8) STM32F4 系列： – STM32F401xx、 STM32F411xx – STM32F405xx、 STM32F407xx – STM32F415xx、 STM32F417xx – STM32F429xx、 STM32F439xx

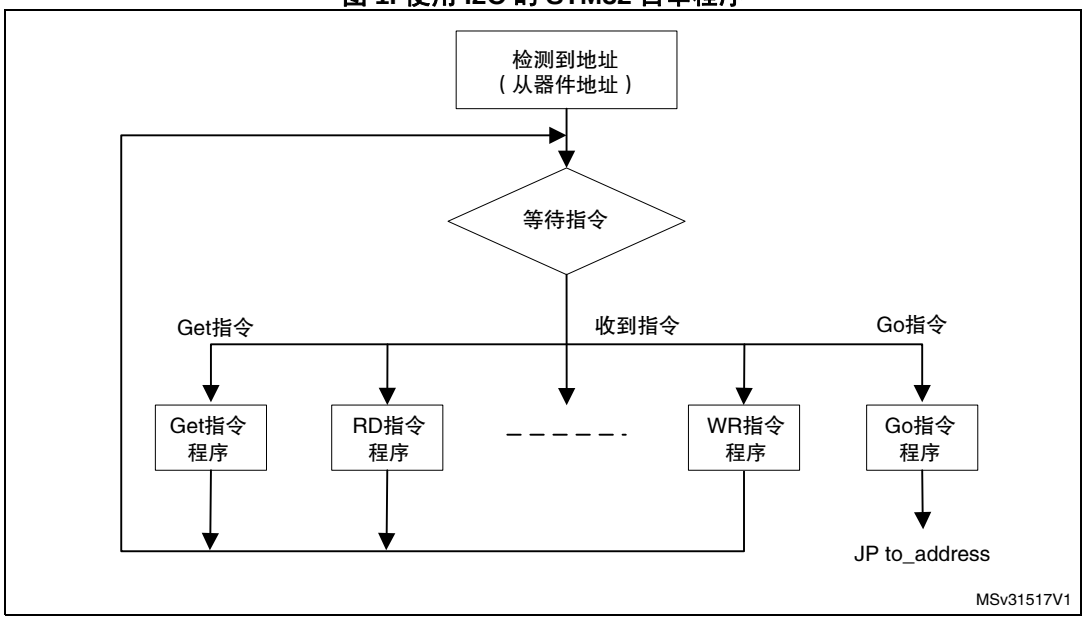
目录

表格索引

图片索引

1 I2C 自举程序代码序列

图 1. 使用 I2C 的 STM32 自举程序



注： *在 AN2606 中规定了每个产品自举程序的 I2C 从地址。*

进入系统存储器自举模式后，并且此时 STM32 微控制器已配置好（若需更详细信息，请参考您的 STM32 系统存储器自举模式应用笔记），自举程序代码开始扫描 I2C_SDA 线引脚，等待检测它自身在总线上的地址。检测到之后， I2C 自举程序固件开始接收主机指令。

2 自举程序指令集

从 V1.1 协议版本开始支持 "No Stretch" 指令，当自举程序完成操作之前主机必须长时间等待时，它可以支持更好的进行指令管理。

只要可能，强烈建议使用 "No Stretch" 指令而不使用相应的普通指令。

支持的指令列于*表 2* 中。

表 2. I2C 自举程序指令

指令(1)	指令代码	指令说明
Get(2)	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(2)	0x01	获取自举程序版本
Get ID(2)	0x02	获取芯片 ID
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(3)	0x21	跳转到内部 Flash 中的用户应用代码
Write Memory(3)	0x31	向存储器写入最多 256 字节，由应用指定起始地址
No-Stretch Write Memory(3)(4)	0x32	向存储器写入最多 256 字节，由应用指定起始地址。当操作正在进行时返回忙状态
Erase	0x44	使用双字节寻址模式擦除一个到所有的 Flash 页面或扇区
No-Stretch Erase(3)(4)	0x45	使用双字节寻址模式擦除一个到所有的 Flash 页面或扇区，当操作正在进行时返回忙状态
Write Protect	0x63	对一些扇区使能写保护
No-Stretch Write Protect(4)	0x64	对一些扇区使能写保护，当操作正在进行时返回忙状态
Write Unprotect	0x73	对所有 Flash 扇区禁用写保护
No-Stretch Write Unprotect(4)	0x74	对所有 Flash 扇区禁用写保护，当操作正在进行时返回忙状态
Readout Protect	0x82	使能读保护
No-Stretch Readout Protect(4)	0x83	使能读保护，当操作正在进行时返回忙状态
Readout Unprotect(2)	0x92	禁用读保护
No-Stretch Readout Unprotect(2)(4)	0x93	禁用读保护，当操作正在进行时返回忙状态

1. 若收到了拒绝指令，或指令执行期间发生了错误，则自举程序会发送 NACK 字节，然后返回到检查指令状态。
2. 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。
3. 若需了解哪些存储器空间可执行这些指令，请参考 STM32 产品数据手册和 AN2606：STM32 微控制器系统存储器自举模式。
4. 仅 V1.1 的 I2C 协议才支持 No-Stretch 指令。

No-Stretch 指令

当自举程序执行操作时， No-Stretch 指令可执行 Write、 Erase、 Write Protect、 Write Unprotect、 Read Protect、 Read Unprotect 操作而不延长 I2C 线。当自举程序执行的操作需要等待时间时，这些指令允许与总线上的其它器件通信。

这些指令与标准指令的不同之处在于指令结束：当主机在指令结束要求 ACK/NACK 时，自举程序不会延长 I2C 线，而是使用第三种状态 ——Busy （0x76）来响应。当主机收到 Busy 状态时，它会再对状态轮询，读取一个字节，直到收到 ACK 或 NACK 响应。

通信安全

从编程主机到器件的所有通信都经过校验和验证。接收的数据字节块都经过异或计算。所有字节异或计算后算出一个字节，加到每次通信的末尾 （校验和字节）。对所有收到的字节 —— 数据 + 校验和 —— 做异或计算，最后结果必须为 0x00。

对每个指令，主机会发送一个字节及其补码 （异或 = 0x00）。

每个包或接受 （ACK 应答）或丢弃 （NACK 应答）：

- *ACK = 0x79*
- *NACK = 0x1F*

对于 No-Stretch 指令，当操作正在进行时，会发送 Busy 状态而不是 ACK 或 NACK：

- *BUSY= 0x76*

注： *主机的帧可为下列之一：*

- *发送指令帧：主机作为主发送端发起通信，向器件发送两字节：命令代码 + XOR。*
- *等待 ACK/NACK 帧：主机作为主接收端发起 I2C 通信，从器件接收一个字节：ACK 或 NACK 或 BUSY。*
- *接收数据帧：主机作为主接收端发起 I2C 通信，从器件收到响应。收到的字节数取决于指令。*
- *发送数据帧：主机作为主发送端发起 I2C 通信，向器件发送需要的字节。发送的字节数取决于指令。*

小心： I2C 通信实现了超时机制，这是自举程序指令正确执行所必需要考虑的。此超时在同一指令的两个 I2C 帧间实现。例如，对于 Write memory 指令，在指令发送帧和地址存储器发送帧之间有超时机制。 此外也将在同一 I2C 帧中的两个连续数据接收或发送实例之间插入同一超时周期。如果超时周期已过，则生成系统复位以避免自举程序崩溃。有关每种 STM32 产品的 I2C 超时值，请参考 AN2606， “I2C 自举程序时序特性 ” 一节。

2.1 Get 指令

Get 指令可帮您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，如*图 2* 中所示。

图 2. Get 指令：主机端

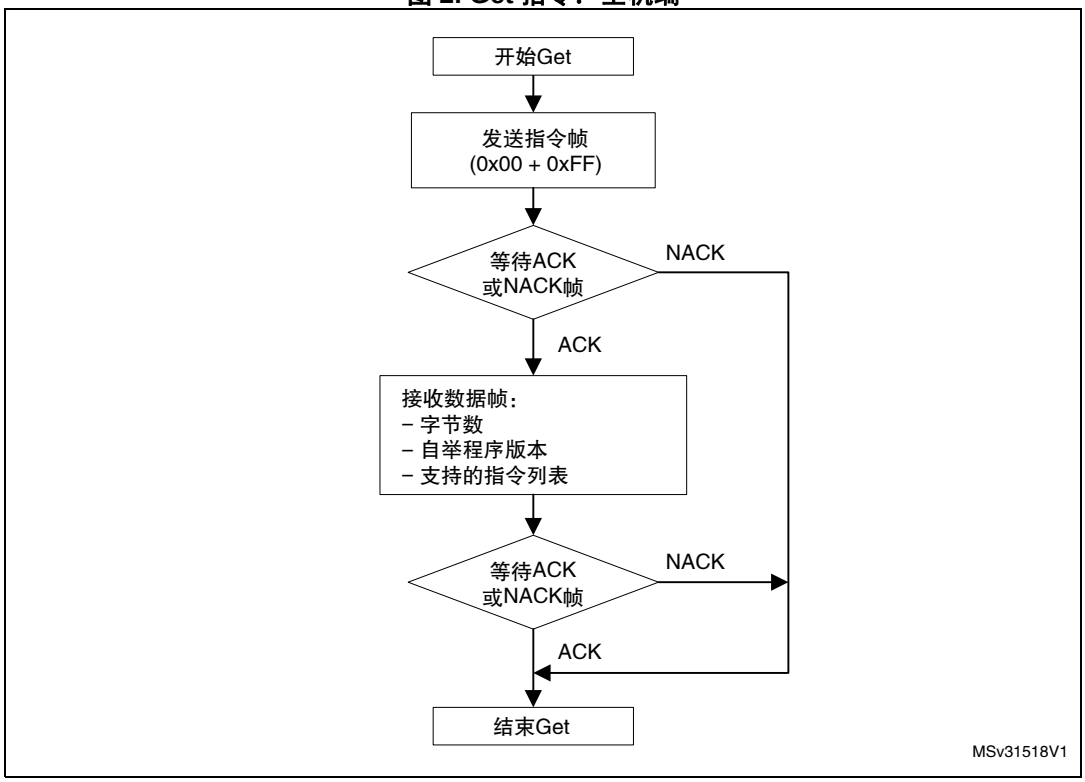
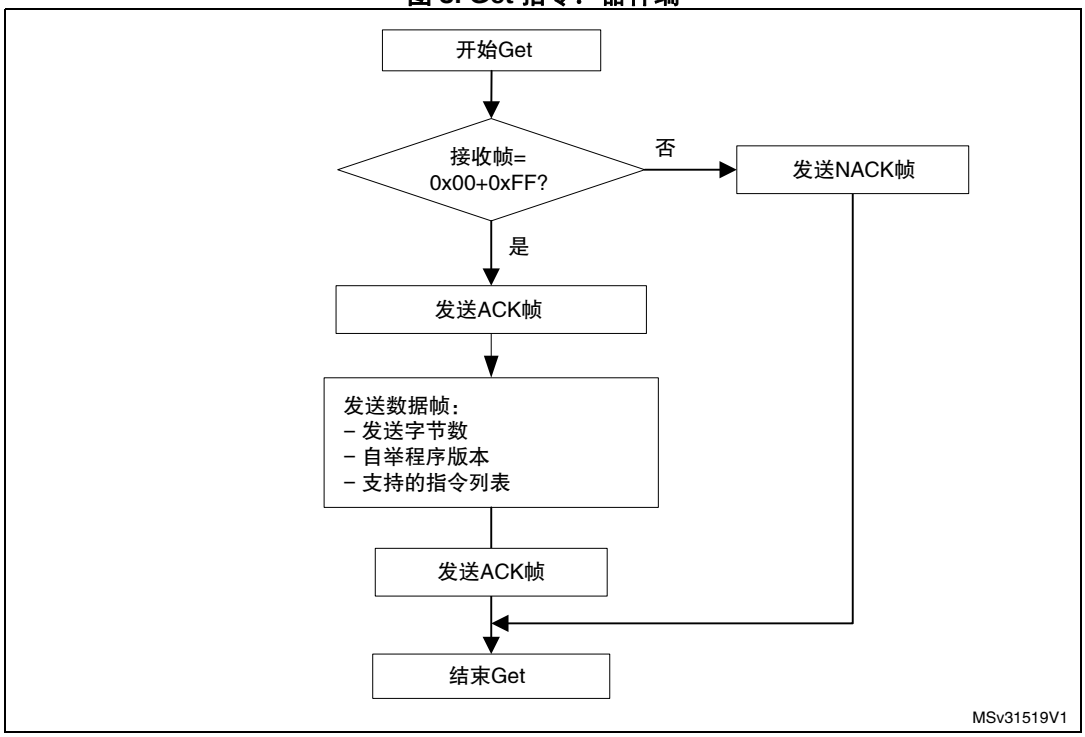


图 3. Get 指令：器件端



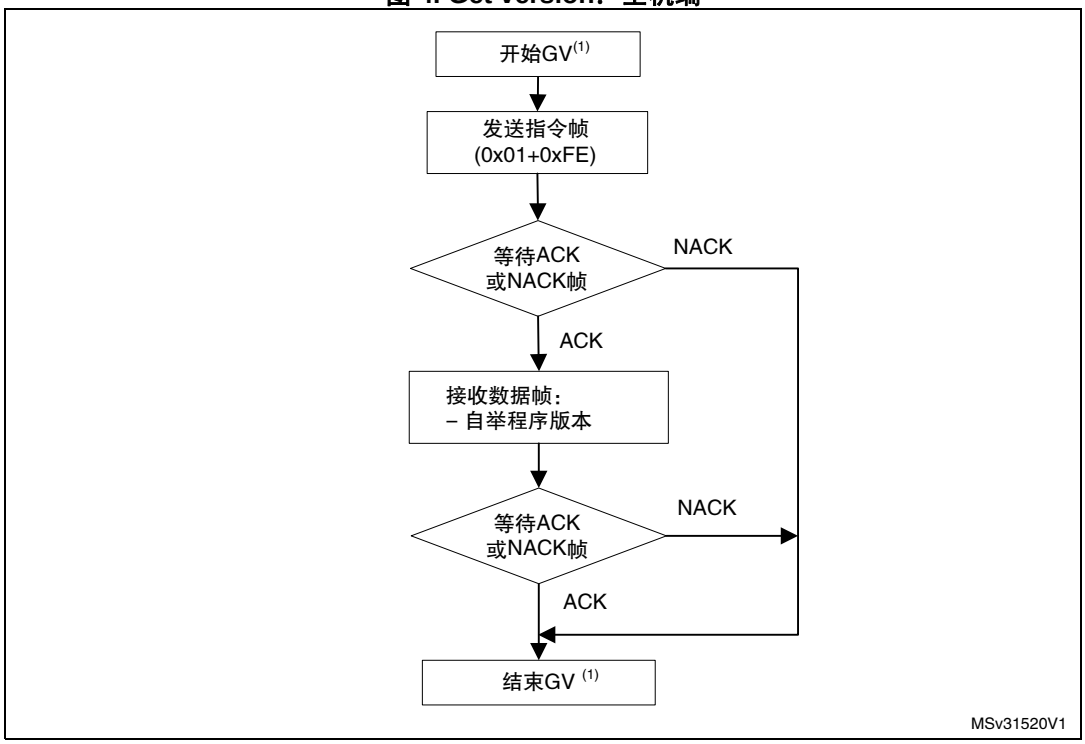
STM32 发送的字节如下：

- *对于 I2C 协议 V1.0：*
  - 字节 1：ACK
  - 字节 2：N = 11 = 后续字节数 - 1，不包括当前字节和 ACK。
  - 字节 3：自举程序版本 0x10 = 1.0 版本
  - 字节 4：0x00 - Get 指令
  - 字节 5：0x01 - Get Version
  - 字节 6：0x02 - Get ID
  - 字节 7：0x11 - Read Memory 指令
  - 字节 8：0x21 - Go 指令
  - 字节 9：0x31 - Write Memory 指令
  - 字节 10：0x44 - Erase 指令
  - 字节 11：0x63 - Write Protect 指令
  - 字节 12：0x73 - Write Unprotect 指令
  - 字节 13：0x82 - Readout Protect 指令
  - 字节 14：0x92 - Readout Unprotect 指令
  - 字节 15：ACK
- *For I2C protocol V1.1:*
  - 字节 1：ACK
  - 字节 2：N = 17 = 后续字节数 - 1，不包括当前字节和 ACK。
  - 字节 3：自举程序版本 0x11 = 1.1 版本
  - 字节 4：0x00 - Get 指令
  - 字节 5：0x01 - Get Version
  - 字节 6：0x02 - Get ID
  - 字节 7：0x11 - Read Memory 指令
  - 字节 8：0x21 - Go 指令
  - 字节 9：0x31 - Write Memory 指令
  - 字节 10：0x44 - Erase 指令
  - 字节 11：0x63 - Write Protect 指令
  - 字节 12：0x73 - Write Unprotect 指令
  - 字节 13：0x82 - Readout Protect 指令
  - 字节 14：0x92 - Readout Unprotect 指令
  - 字节 15：0x32 - No-Stretch Write Memory 指令
  - 字节 16：0x45 - No-Stretch Erase 指令
  - 字节 17：0x64 - No-Stretch Write Protect 指令
  - 字节 18：0x74 - No-Stretch Write Unprotect 指令
  - 字节 19：0x83 - No-Stretch Readout Protect 指令
  - 字节 20：0x93 - No-Stretch Readout Unprotect 指令
  - 字节 21：ACK

2.2 Get version 指令

Get Version 指令用于获取 I2C 自举程序版本。当自举程序收到该指令时，它会向主机发送如下信息 （自举程序版本）。

图 4. Get version：主机端

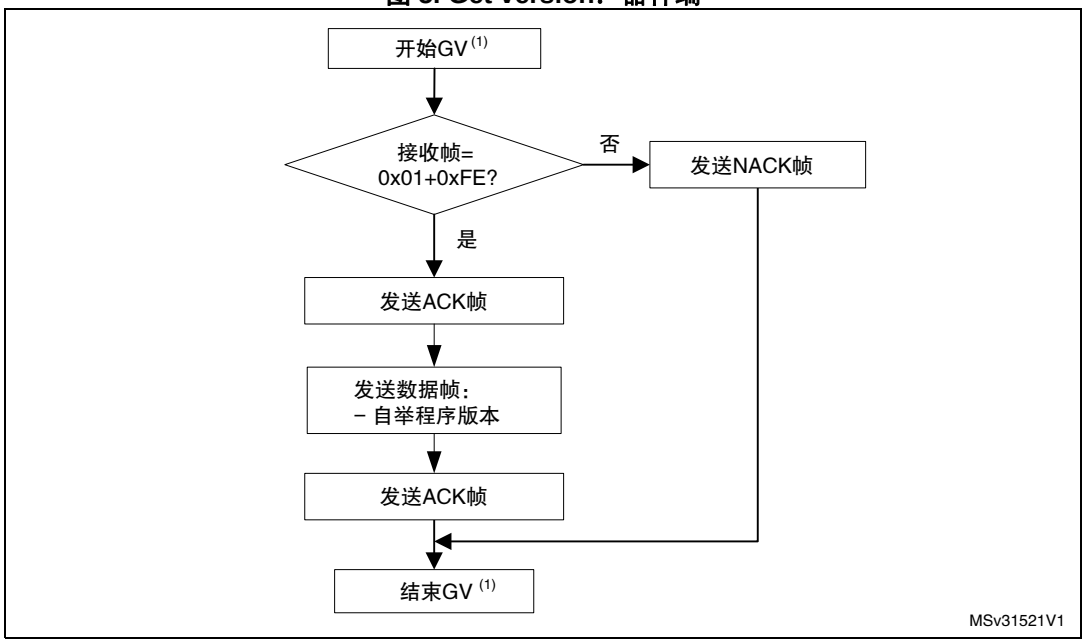


1. GV = Get Version.

STM32 发送的字节如下：

- *字节1：ACK*
- *字节2：自举程序版本 （0 < 版本 ≤ 255）（例如， 0x10 = 1.0 版本）*
- *字节3：ACK*

图 5. Get version：器件端



1. GV = Get Version

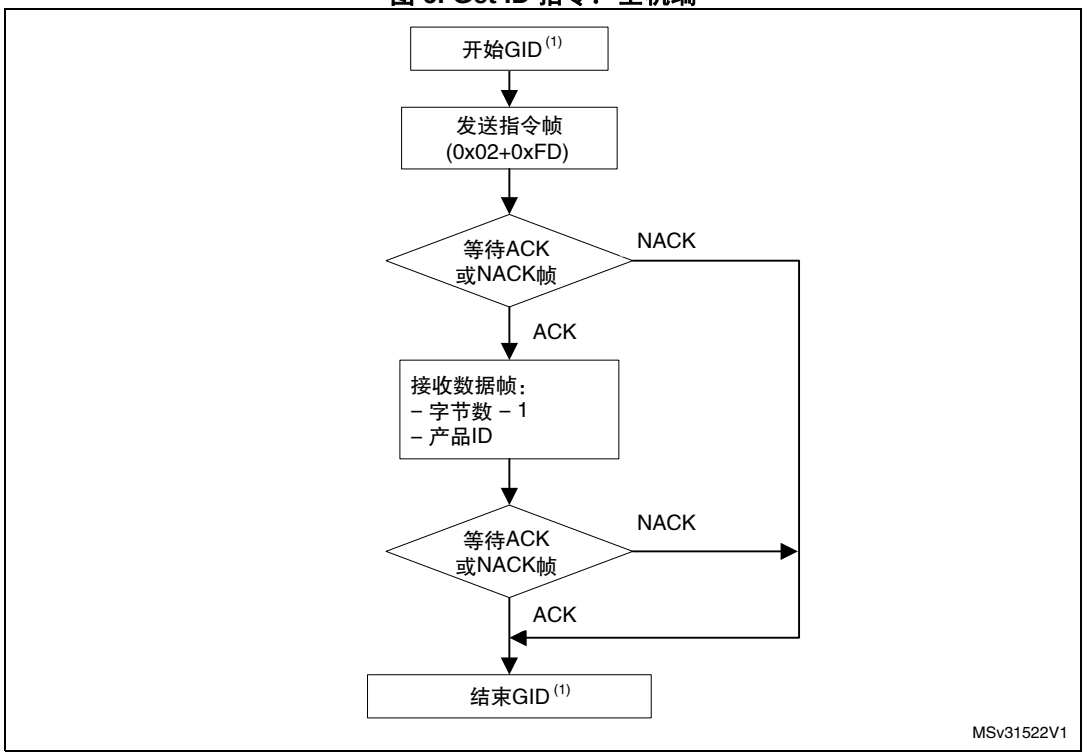
2.3 Get ID 指令

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。

STM32 器件发送的字节如下：

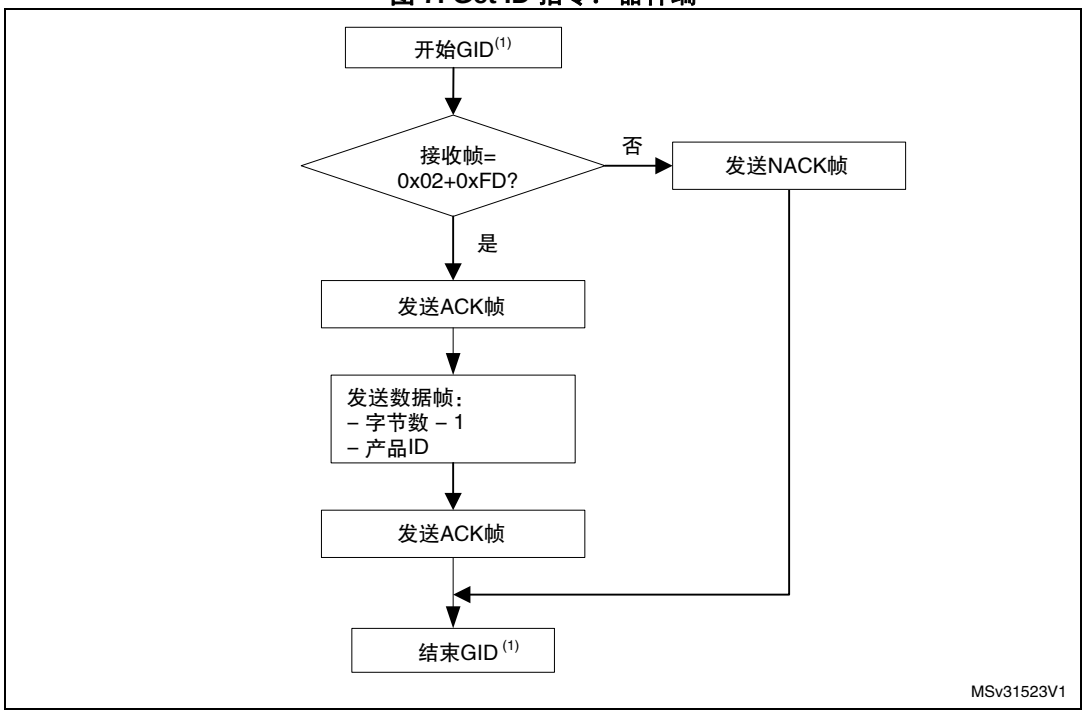
- *字节 1：ACK*
- *字节 2：N = 字节数 - 1 （对于 STM32， N = 1），不包括当前字节和 ACK。*
- *字节 3-4：PID （产品 ID）*
  - 字节 3 = MSB
  - 字节 4 = LSB
- *字节 5：ACK*

图 6. Get ID 指令：主机端



1. GID = Get ID.

图 7. Get ID 指令：器件端



1. GID = Get ID.

2.4 Read memory 指令

Read Memory 指令用于从任何有效存储器地址读取数据。

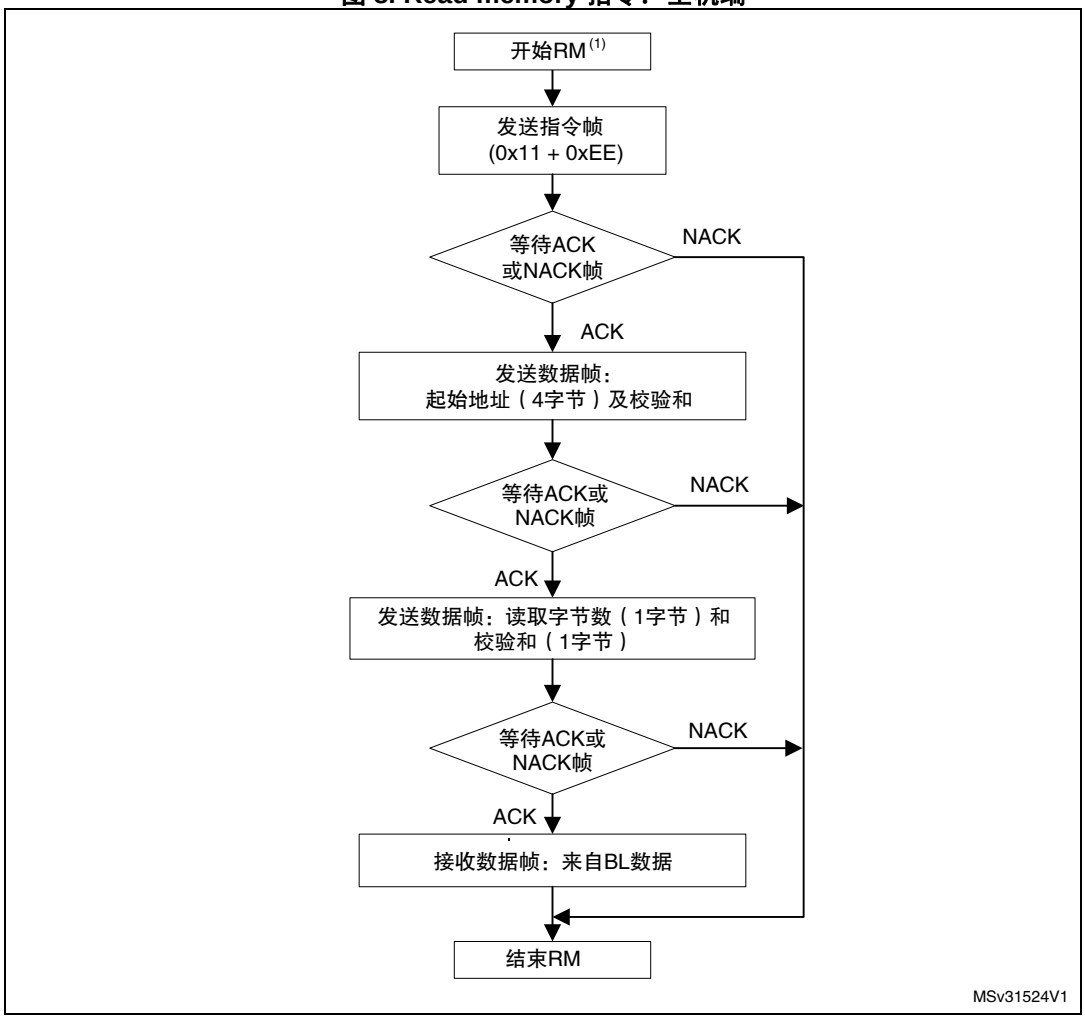
当自举程序收到 Read Memory 指令时，它会向应用程序发送 ACK 字节。然后，自举程序等待一个 4 字节的地址 （字节 1 为地址 MSB，字节 4 为 LSB）及校验和字节，之后它会检查收到的地址。若地址有效且校验和正确，则自举程序发送 ACK 字节；否则它发送 NACK 字节并终止该指令。

若地址有效且校验和正确，则自举程序等待要发送的字节数 （N 字节）及其补码字节 （校验和） 。若校验和正确，则自举程序从收到的地址开始向应用发送需要的数据。若校验和不正确，则它会在终止指令之前发送 NACK。

主机向 STM32 发送的字节如下：

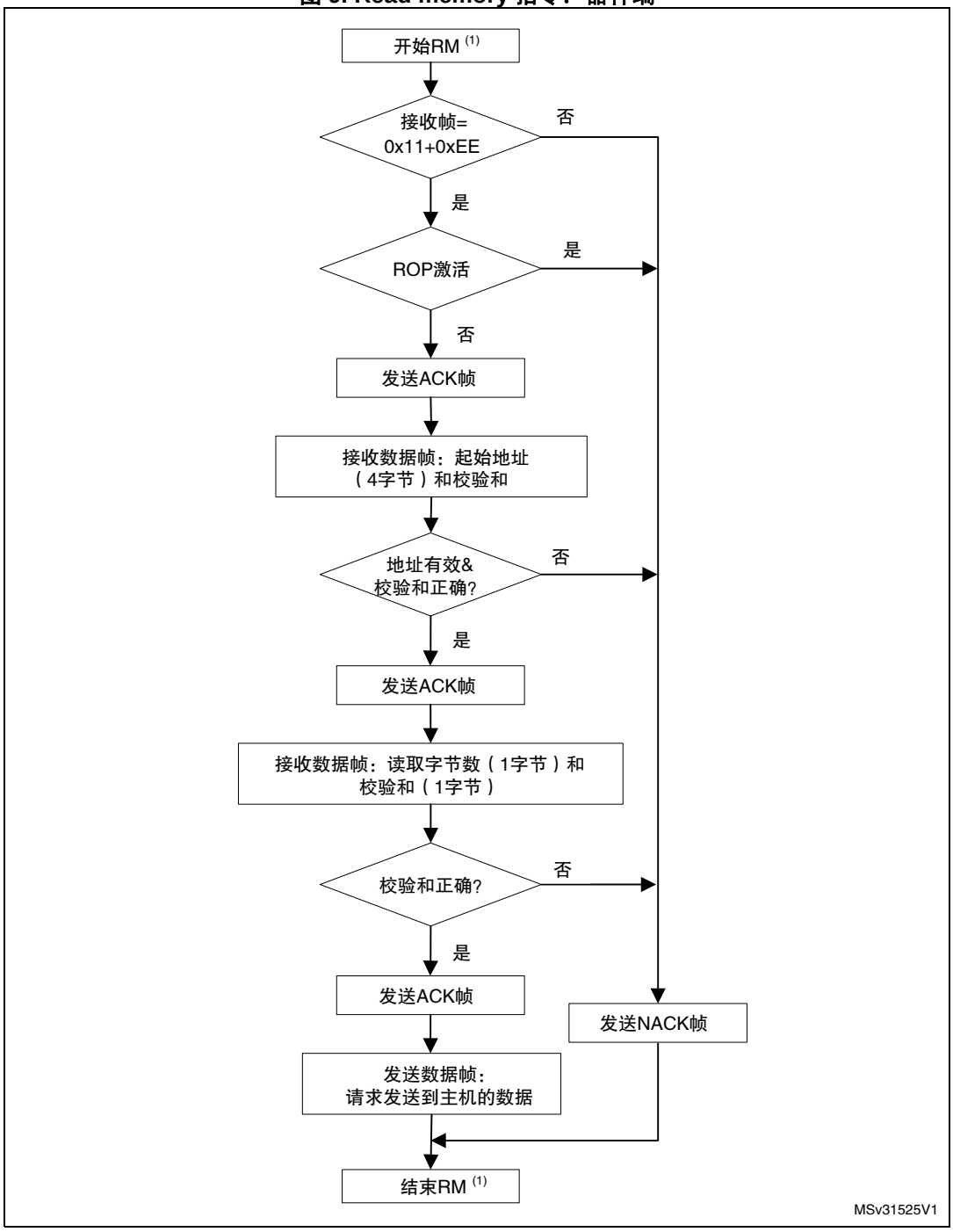
1. 字节 1-2：0x11+0xEE
2. 等待 ACK
3. 字节 3-6：起始地址 （字节 3：MSB，字节 6：LSB）
4. 字节 7：校验和：XOR （字节 3、字节 4、字节 5、字节 6）
5. 等待 ACK
6. 字节 8：要读取的字节数 - 1 （0 < N ≤ 255）
7. 字节 9：校验和：XOR 字节 8 （字节 8 的补码）

图 8. Read memory 指令：主机端



1. RM = Read Memory.

图 9. Read memory 指令：器件端



1. RM = Read Memory.

2.5 Go 指令

Go 指令用于从应用程序指定的地址开始执行已下载的代码或任何其它代码。当自举程序收到 Go 指令时，它会向应用发送 ACK 字节。然后，自举程序等待一个 4 字节的地址 （字节 1 为地址 MSB，字节 4 为 LSB）及校验和字节，之后检查收到的地址。若地址有效且校验和正确，则自举程序发送 ACK 字节；否则它发送 NACK 字节并终止该指令。

若地址有效且校验和正确，则自举程序固件会执行如下操作：

1. 将自举程序所用外设的寄存器初始化至其默认复位值。
2. 初始化用户应用的主堆栈指针
3. 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 0x08000000，则自举程序跳转至编程为 0x08000004 地址的存储器位置。

总之，主机发送基址，应用编程跳转。

注： *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。*

主机向 STM32 发送的字节如下：

1. 字节 1：0x21
2. 字节 2：0xDE
3. 等待 ACK
4. 字节 3 至字节 6：起始地址
   - 字节 3：MSB
   - 字节 6：LSB
5. 字节 7：校验和：XOR （字节 3、字节 4、字节 5、字节 6）
6. 等待 ACK

图 10. Go 指令：主机端

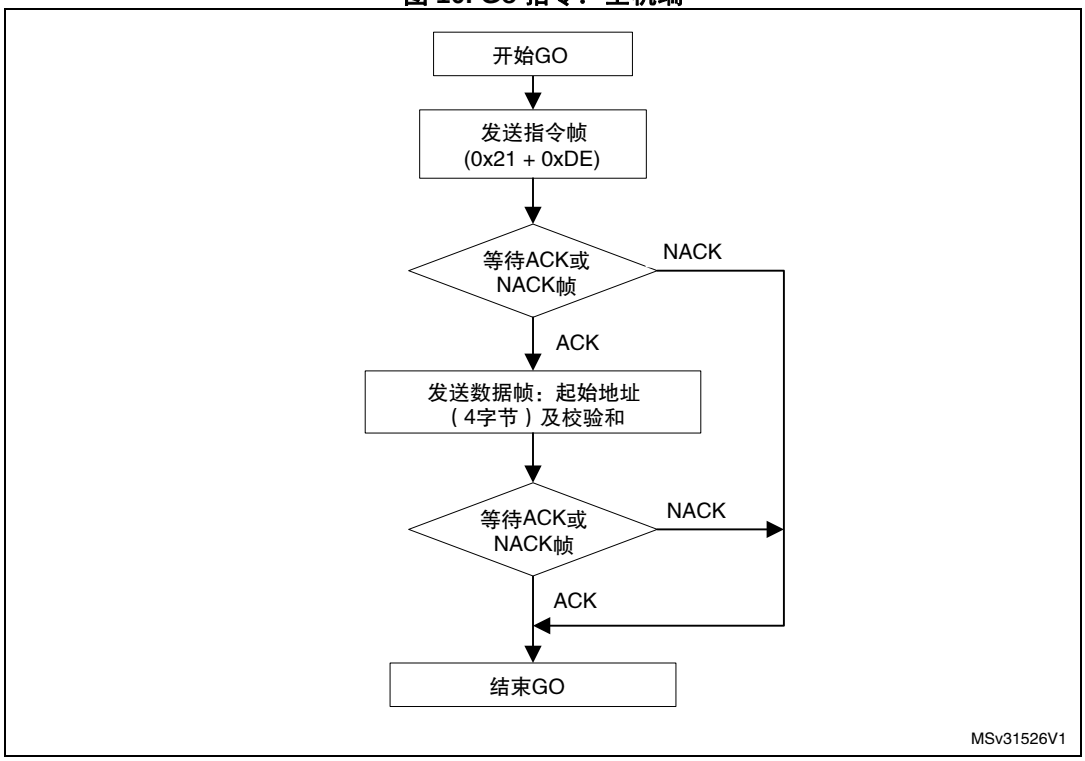
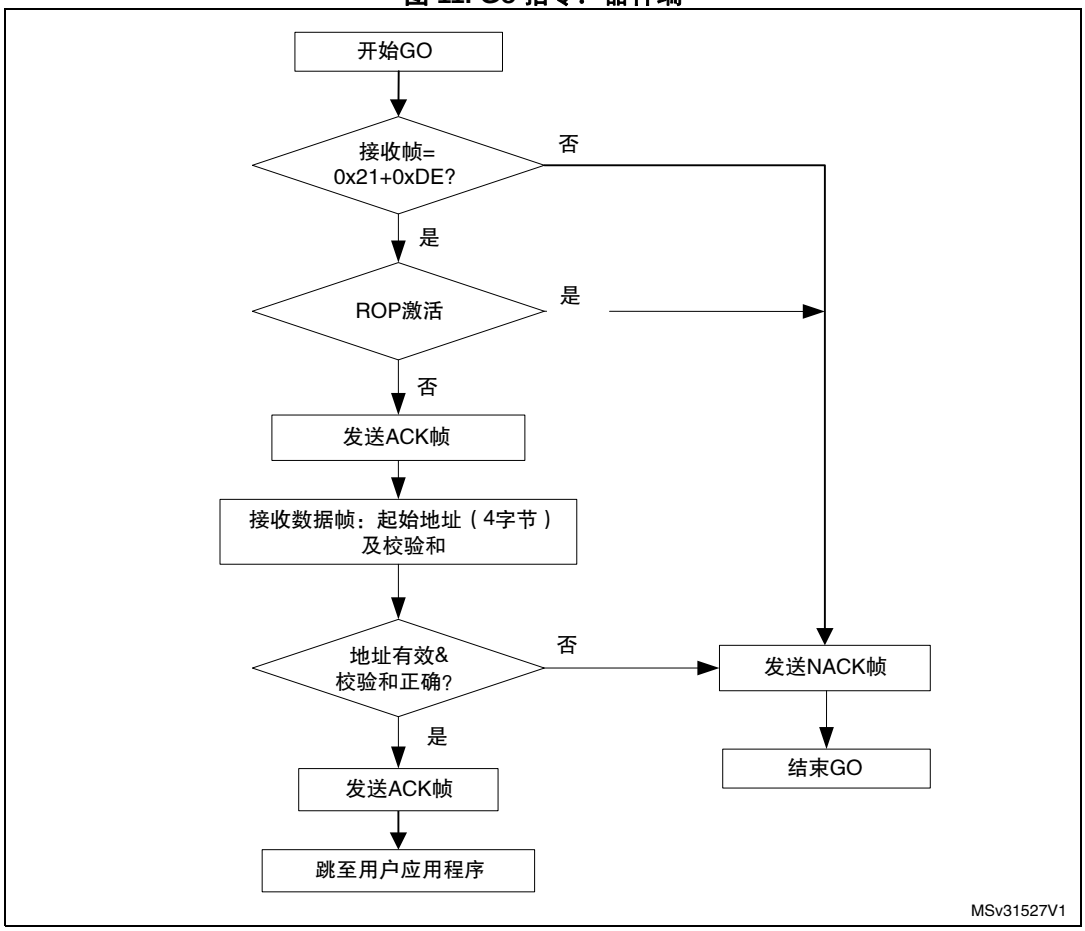

开始GO
发送指令帧
(0x21 + 0xDE)
等待ACK或
NACK帧
NACK
ACK
发送数据帧：起始地址
（4字节）及校验和
等待ACK或
NACK帧
NACK
ACK
结束GO
MSv31526V1

图 11. Go 指令：器件端


开始GO
接收帧=
0x21+0xDE?
否
是
ROP激活
是
否
发送ACK帧
接收数据帧：起始地址（4字节）
及校验和
地址有效&
校验和正确?
否
发送NACK帧
是
结束GO
发送ACK帧
跳至用户应用程序
MSv31527V1

2.6 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 （见下面的 *注：*）写入数据。

当自举程序收到 Write Memory 指令时，它会向应用发送 ACK 字节。之后自举程序等待一个 4 字节的地址 （字节 1 为地址 MSB，字节 4 为 LSB）及校验和字节，然后校验收到的地址。

若收到的地址有效且校验和正确，则自举程序发送 ACK 字节；否则它发送 NACK 字节并终止该指令。若地址有效且校验和正确，则自举程序会：

1. 得到一个字节 N，它包含要接收的数据字节数
2. 接收 （(N + 1) 字节）用户数据及其校验和 （N 和所有数据字节的异或）
3. 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 字节；否则它发送 NACK 字节并终止指令。

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。在该指令结束时，自举程序会生成系统复位，以使选项字节的新配置生效。

写入选项字节的最大块长度取决于 STM32 产品，且从主机收到的地址必须为选项字节区域的起始地址。若需选项字节的更多信息，请参考 STM32 产品参考手册。

注： *写入 RAM 或 Flash 的最大块长度为 256 字节。*

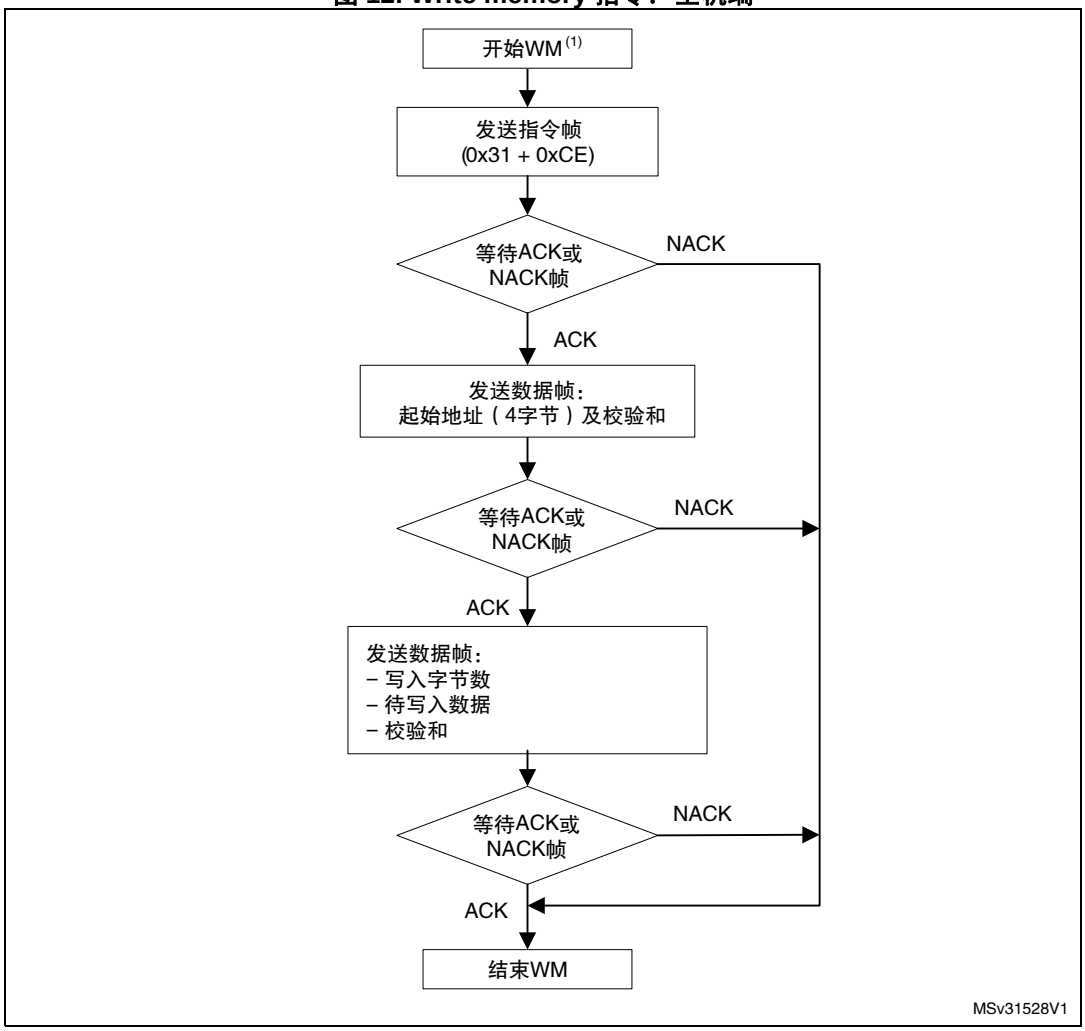
当写入 RAM 时，请注意不要与自举程序固件使用的第一个 RAM 存储器重叠。

当向写保护的扇区执行写操作时，不会返回错误。

主机向 STM32 发送的字节如下：

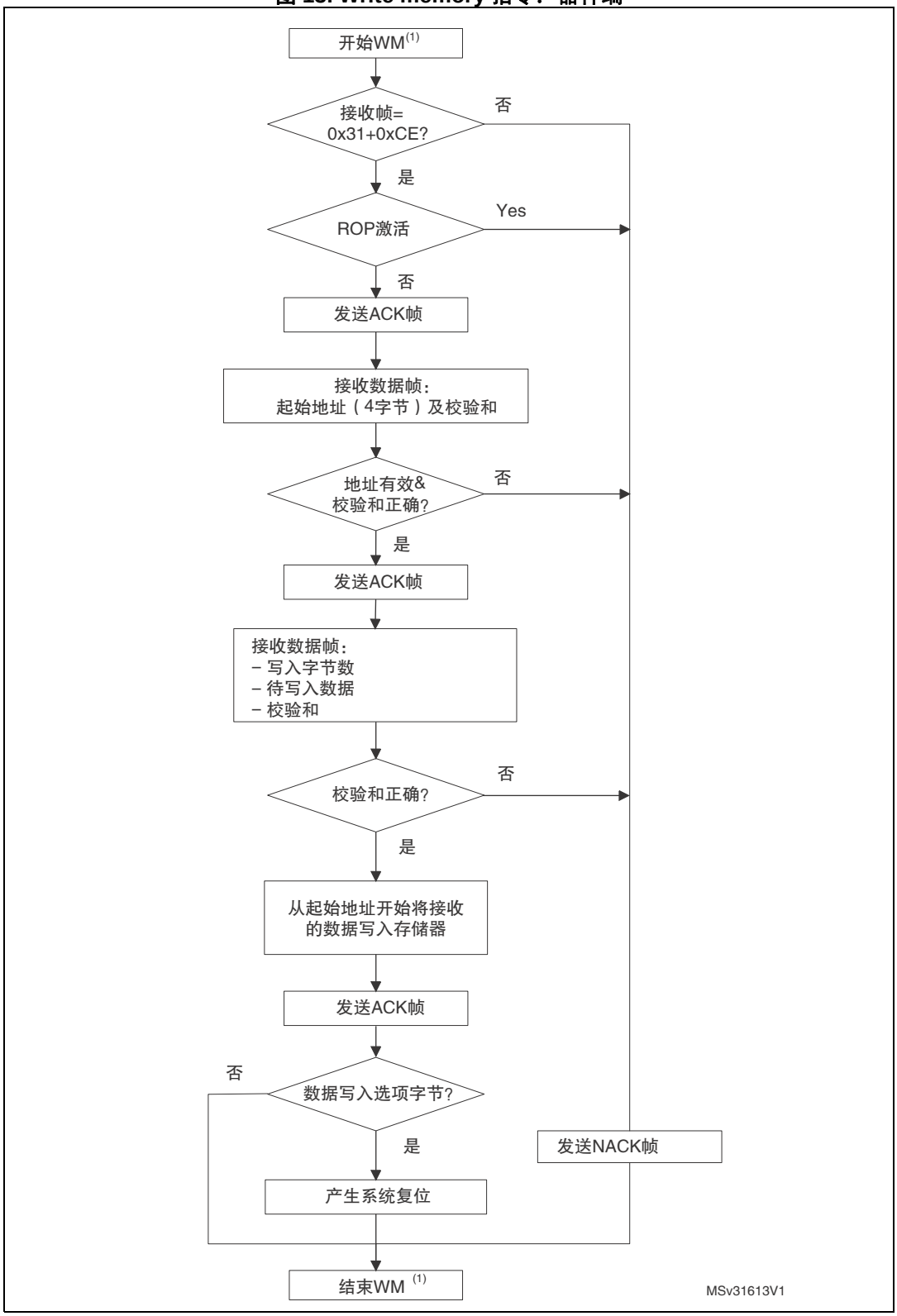
1. 字节 1：0x31
2. 字节 2：0xCE
3. 等待 ACK
4. 字节 3 至字节 6：起始地址
   - 字节 3：MSB
   - 字节 6：LSB
5. 字节 7：校验和：XOR （Byte3、 Byte4、 Byte5、 Byte6）
6. 等待 ACK
7. 字节 8：要接收的字节数 - 1 （0 < N ≤ 255）
8. N +1 数据字节：（最大 256 字节）
9. 校验和字节：XOR （N， N+1 数据字节）
10. 等待 ACK

图 12. Write memory 指令：主机端



1. WM = Write Memory.

图 13. Write memory 指令：器件端



1. WM = Write Memory.

2.7 Erase memory 指令

Erase Memory 指令可使主机用双字节寻址模式擦除 Flash 页面或扇区。当自举程序收到 Erase Memory 指令时，它会向主机发送 ACK 字节。然后自举程序接收两个字节 （要擦除的页面或扇区数）、 Flash 页面或扇区码 （每个都以双字节编码， MSB 在前）、校验和字节（所发送字节的 XOR）。若校验和正确，则自举程序擦除存储器并向主机发送 ACK 字节；否则它向主机发送 NACK 字节，指令终止。

Erase Memory 指令规范

自举程序收到半个字 （两个字节），它包含 N—— 要擦除的页面或扇区数。对于 N = 0xFFFY （其中 Y 为 0 到 F），会执行特殊擦除 （0xFFFF 为全局批量擦除， 0xFFFE 和 0xFFFD 分别为批量擦除 bank1 和 bank2）。

注： *一些产品不支持批量擦除特性，在这种情况下您发送的擦除指令可使用所有页面或扇区的数目。*

注： *代码 0xFFFC 到 0xFFF0 为保留值。*

对于 0 ≤ N < 页面或扇区最大数的其它值，会擦除 N + 1 个页面或扇区。

自举程序接收：

- *对于特殊擦除的情况，一个字节：之前字节的校验和*
- *0x00 对应 0xFFFF，全局擦除*

若擦除 N+1 个页面或扇区，则自举程序接收 （2 x (N + 1)）个字节，其中的每半个字都包含了一个双字节编码的页面或扇区号， MSB 在前。这时所有前面字节的校验和都在一个字节接收。

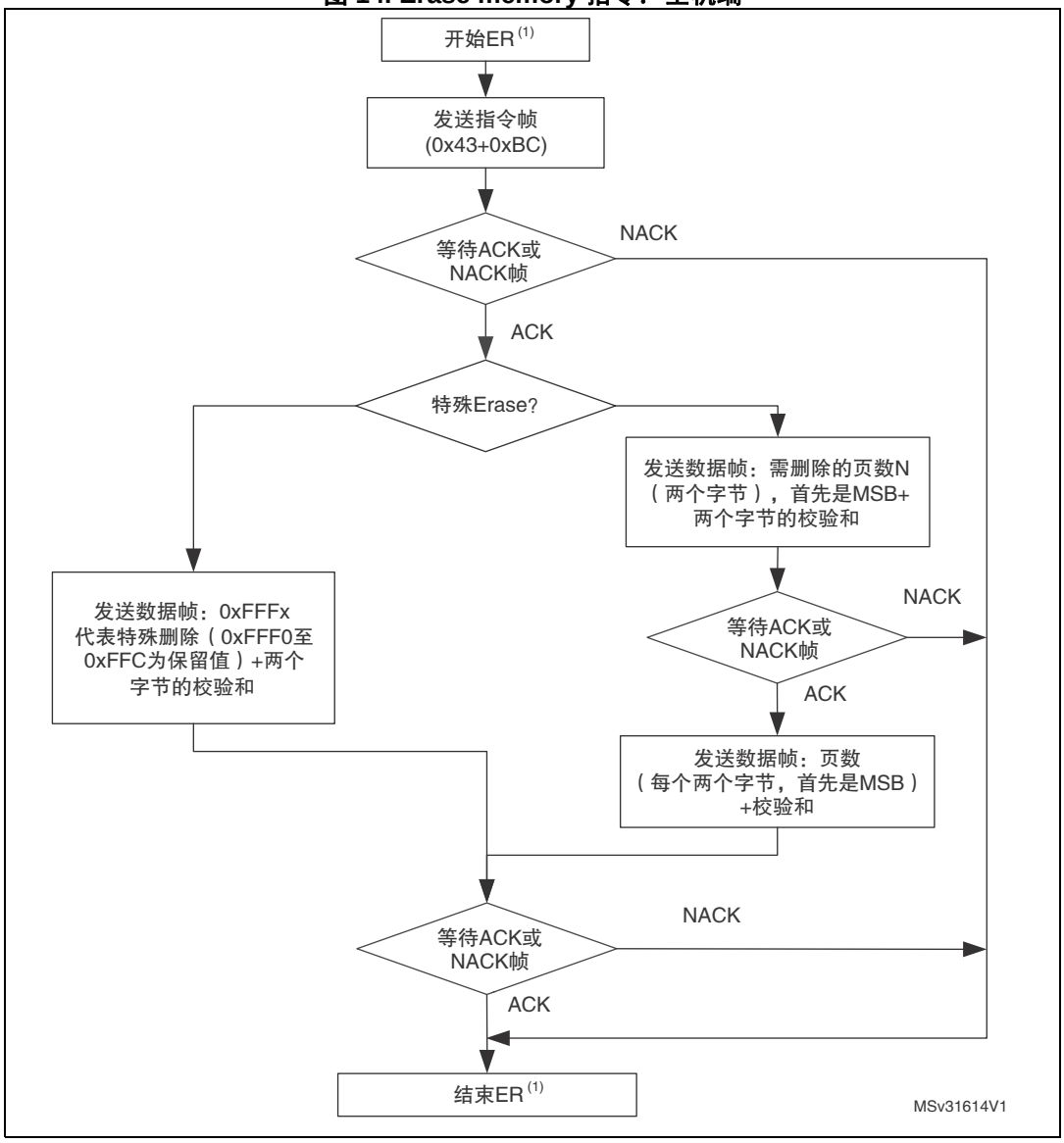
注： *当向写保护的扇区执行擦除操作时，不会返回错误。*

页面或扇区的最大数目与产品有关，因此应提起注意。

主机向 STM32 发送的字节如下：

1. 字节 1：0x44
2. 字节 2：0xBB
3. 等待 ACK
4. 字节 3-4：
   - 特殊擦除 （0xFFFx），或者
   - 要擦除的页面或扇区数 （N+1，其中：0 ≤ N < 页面或扇区的最大数目）
5. 等待 ACK （如未要求特殊擦除）
6. 其余字节：
   - 对于特殊擦除 （0x00），为字节 3-4 的校验和，或者
   - （2 x (N + 1)）字节 （双字节编码的页面或扇区号， MSB 在前），然后是字节 3-4 与其后所有字节的校验和）。
7. 等待 ACK

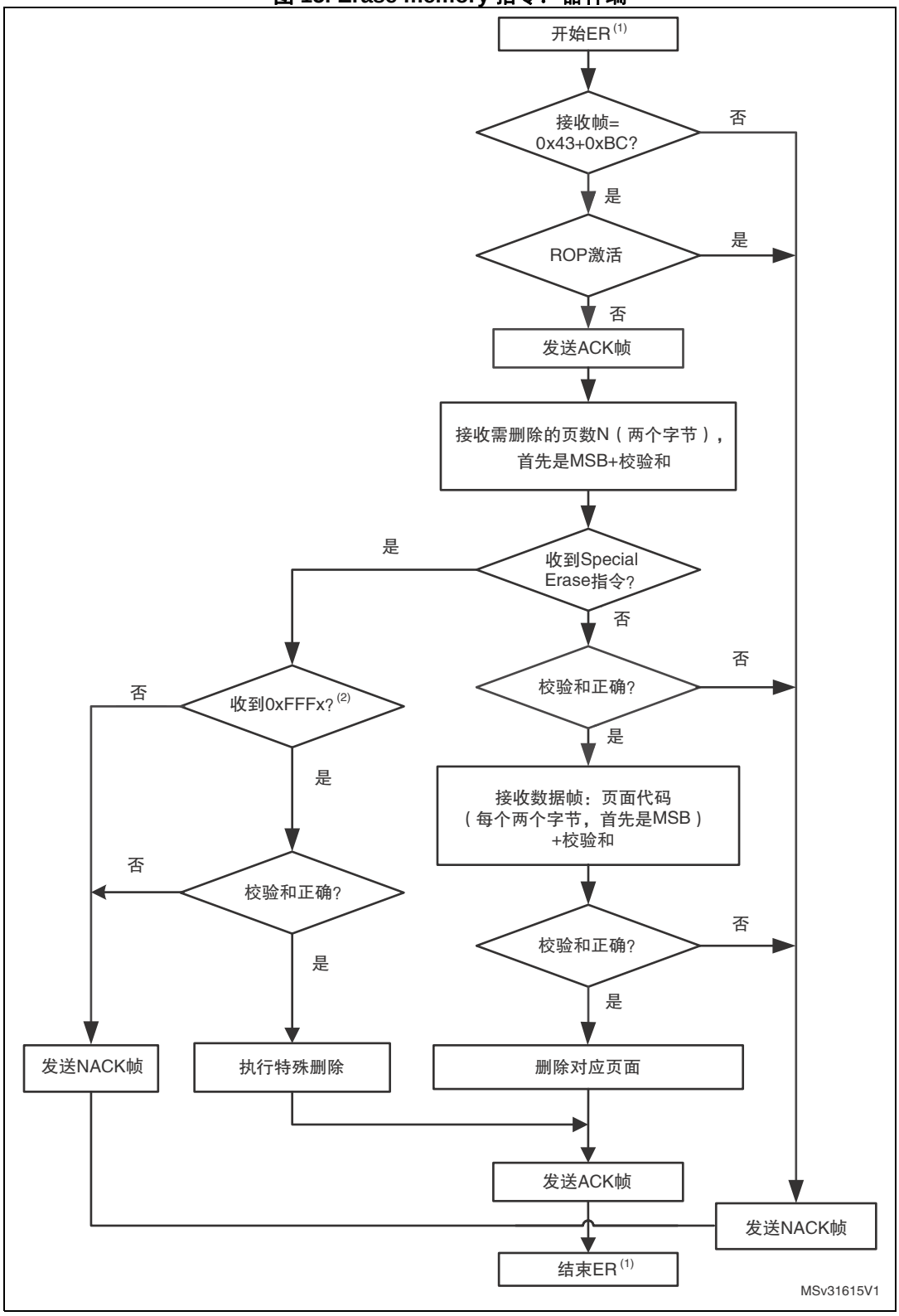
图 14. Erase memory 指令：主机端



1. ER = Erase Memory.

注： *一些产品不支持特殊擦除特性。对于这些产品，此命令将被 NACK。*

图 15. Erase memory 指令：器件端



1. ER = Erase Memory.
2. 若 STM32 产品不支持特殊擦除指令，则请求的特殊擦除指令将被 NACK。

2.8 Write protect 指令

Write Protect 指令用于对部分或全部 Flash 扇区启用写保护。当自举程序收到 Write Protect 指令时，它会向主机发送 ACK 字节。之后自举程序等待要接收的字节数 （要保护的扇区），然后从应用收到 Flash 扇区码。

在 Write Protect 指令结束时，自举程序会发送 ACK 字节并生成系统复位，以使选项字节的新配置生效。

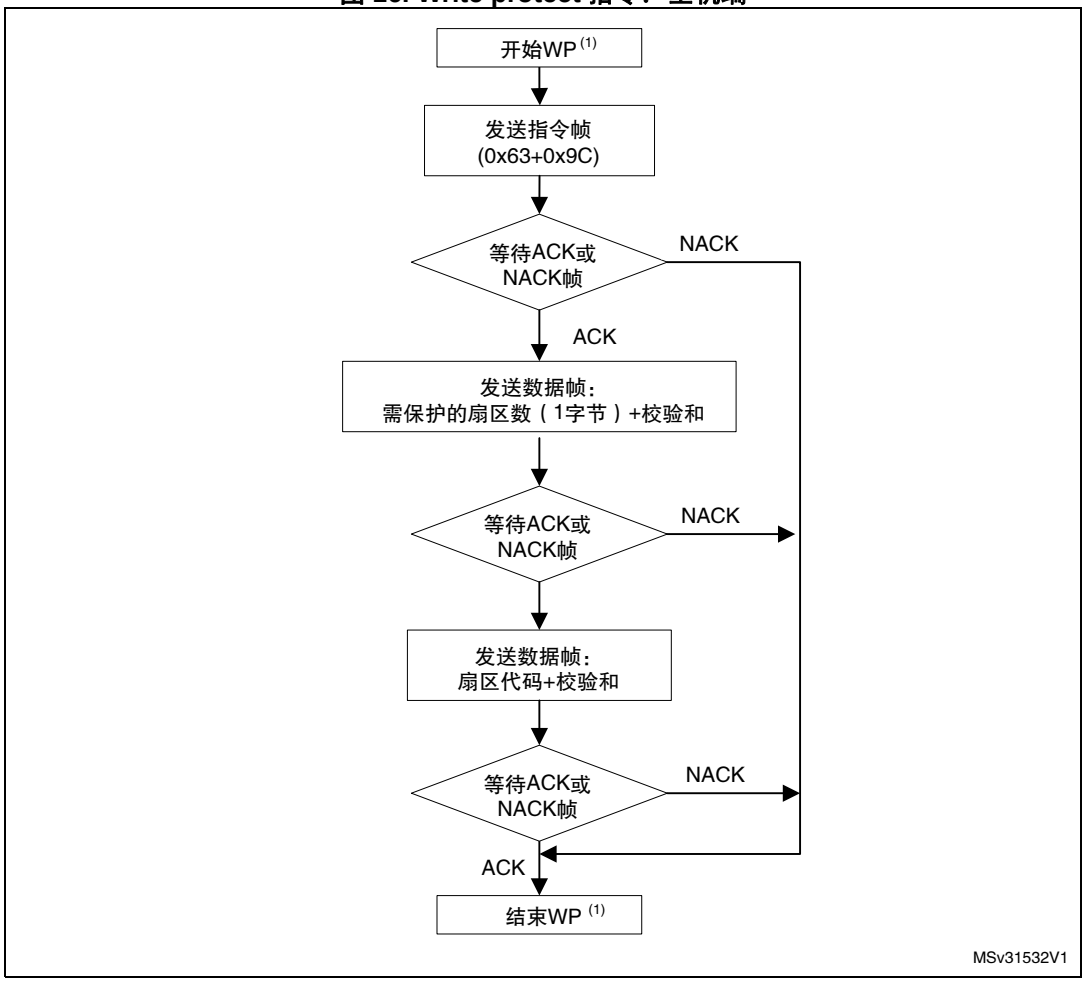
Write Protect 指令流程如下：

- *自举程序收到一个字节，它包含 N—— 要写保护的扇区数 - 1 （0 ≤ N ≤ 255）。*
- *自举程序收到 （N + 1）字节，其中每个字节都包含扇区码。*

注： *扇区总数和要保护的扇区号都不会被校验。这意味着即使指令的保护扇区数错误，或扇区号错误，都不会有错误返回。*

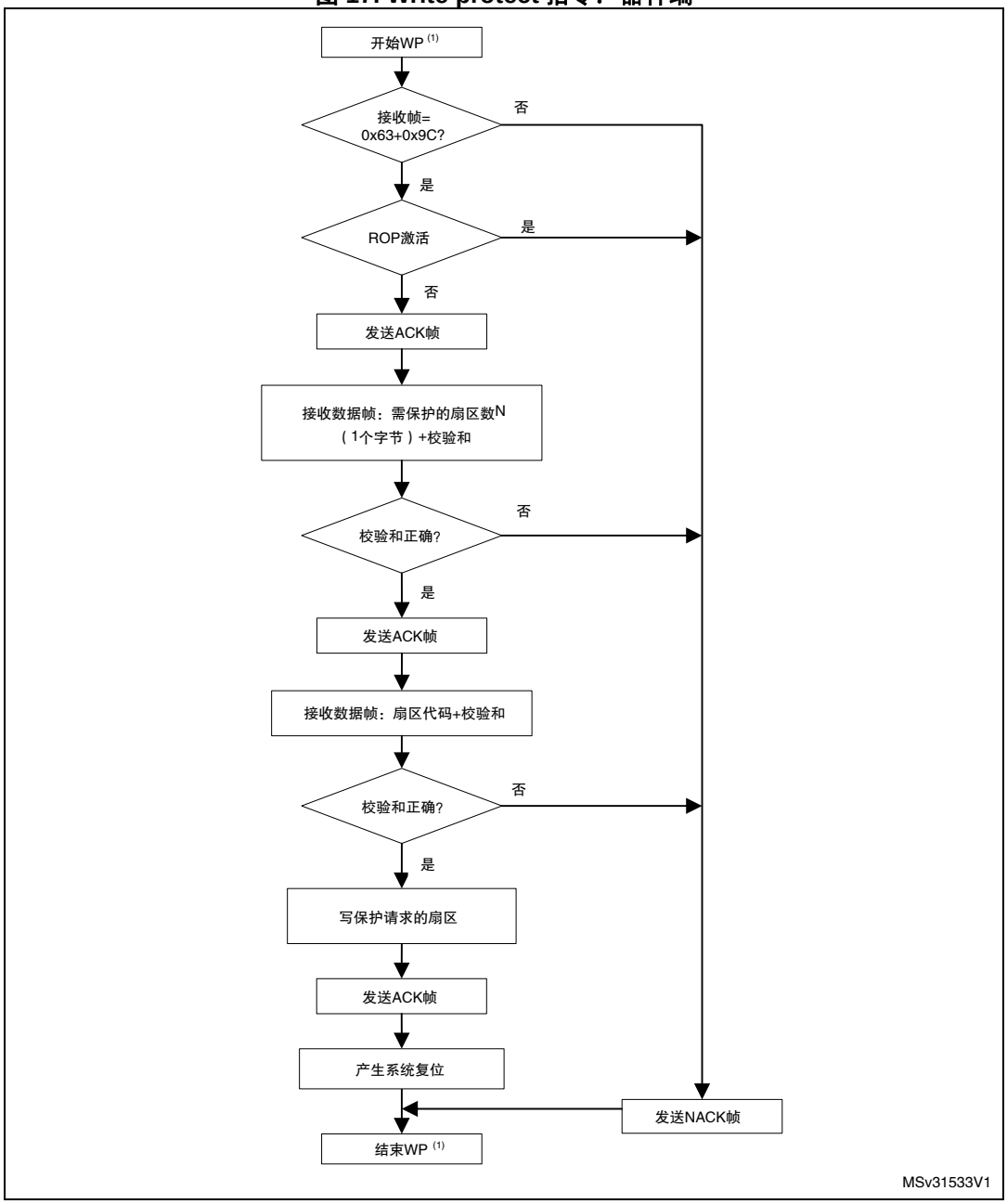
若执行了第二个 Write Protect 指令，则第一个指令已经保护的 Flash 扇区会被解除保护，只有第二个 Write Protect 指令内的扇区才会被保护。

图 16. Write protect 指令：主机端



1. WP = Write Protect.

图 17. Write protect 指令：器件端



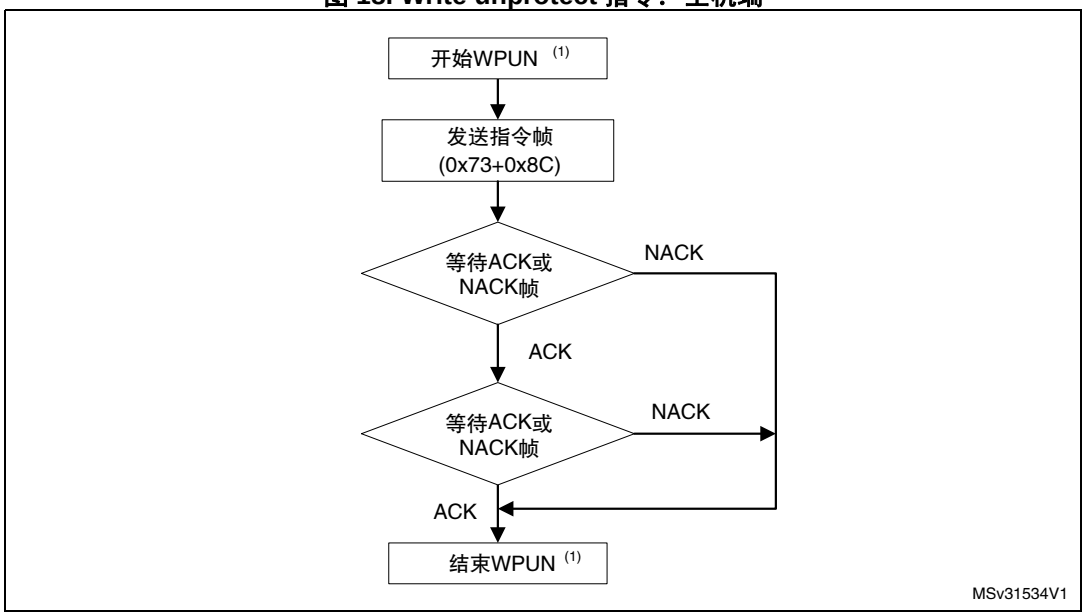
1. WP = Write Protect.

2.9 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，它会向主机发送 ACK 字节。之后自举程序对全部 Flash 扇区禁用写保护，然后发送 ACK 字节。

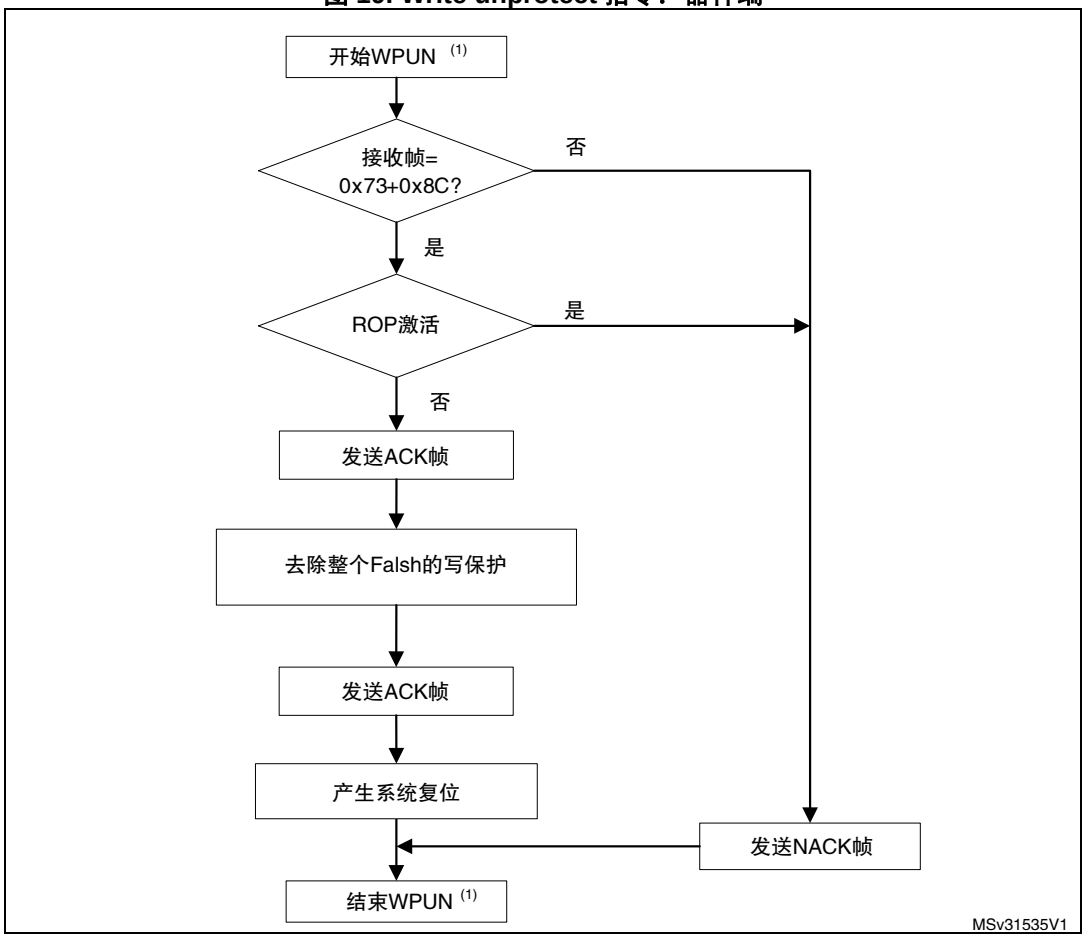
会生成系统复位以使选项字节的新配置生效。

图 18. Write unprotect 指令：主机端



1. WPUN = Write Unprotect.

图 19. Write unprotect 指令：器件端



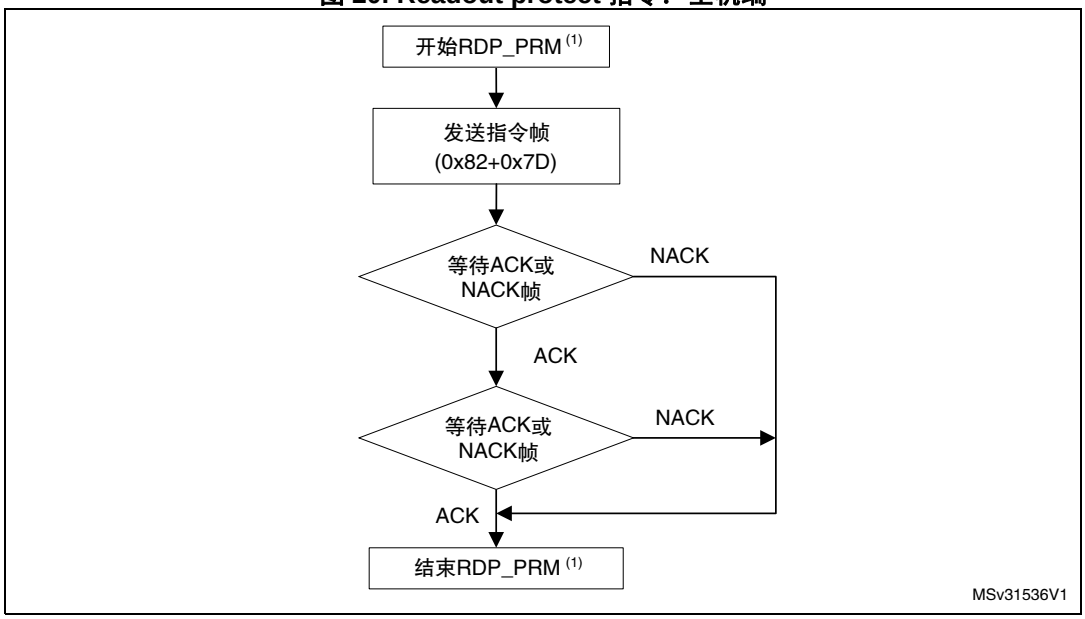
1. WPUN = Write Unprotect.

2.10 Readout protect 指令

Readout Protect 指令用于启用 Flash 的读保护。当自举程序收到 Readout Protect 指令时，它会向主机发送 ACK 字节并对 Flash 启用读保护。

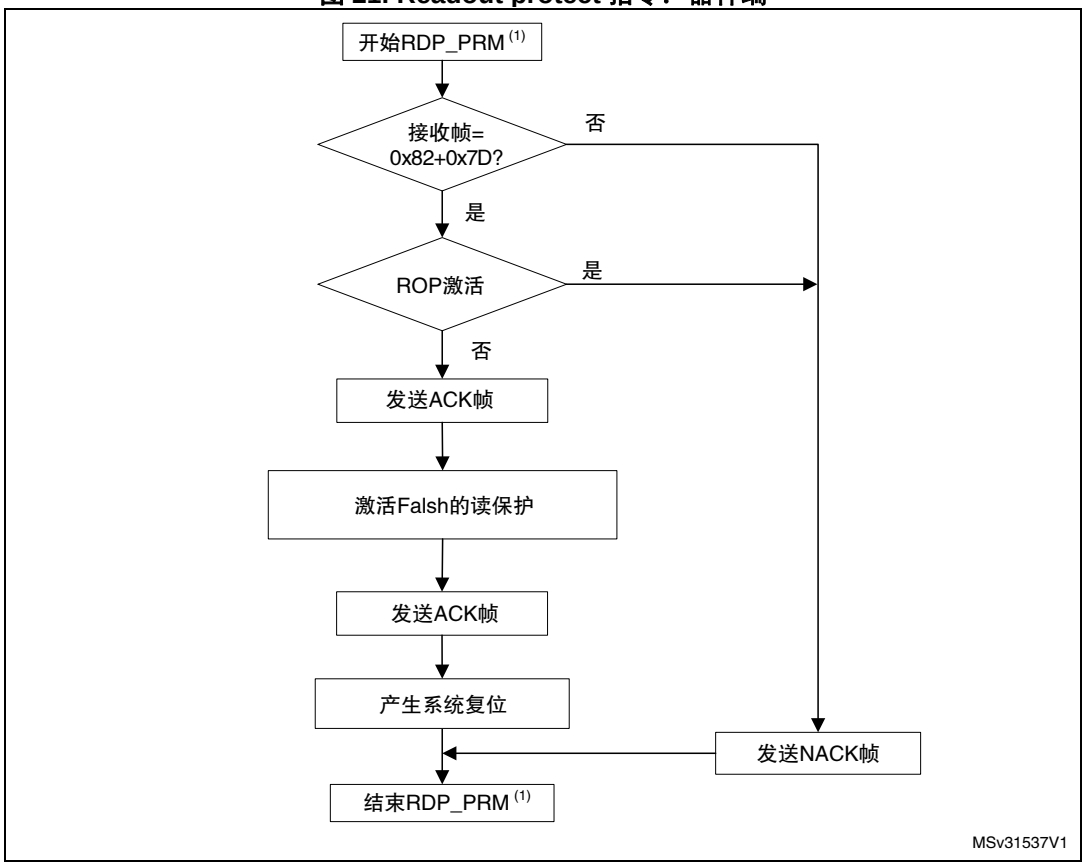
在 Readout Protect 指令结束时，自举程序会发送 ACK 字节并生成系统复位，以使选项字节的新配置生效。

图 20. Readout protect 指令：主机端



1. RDP_PRM = Readout Protect.

图 21. Readout protect 指令：器件端



1. RDP_PRM = Readout Protect.

2.11 Readout unprotect 指令

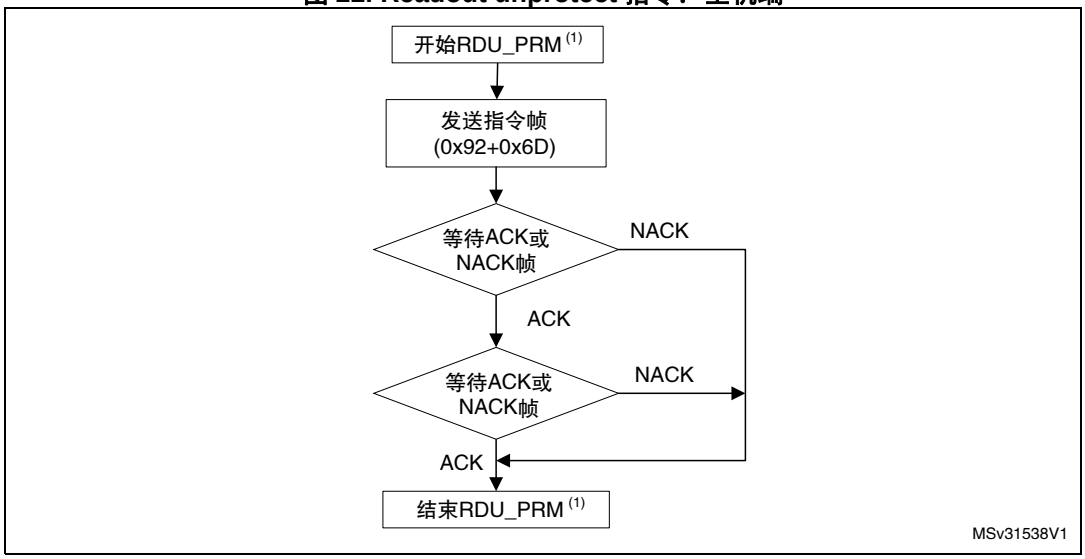
Readout Unprotect 指令用于禁用 Flash 的读保护。当自举程序收到 Readout Unprotect 指令时，它会向主机发送 ACK 字节。

之后自举程序会对全部 Flash 禁用读保护，这会导致擦除全部 Flash。若该操作不成功，则自举程序会发送 NACK，读保护仍然有效。

注： *此操作与擦除所有 Flash 页面或扇区（或者说，若产品支持，执行批量擦除）所花费的时间相同，因此主机应等待，直到此操作结束。关于 Flash 擦除的时序，请参考产品数据手册。*

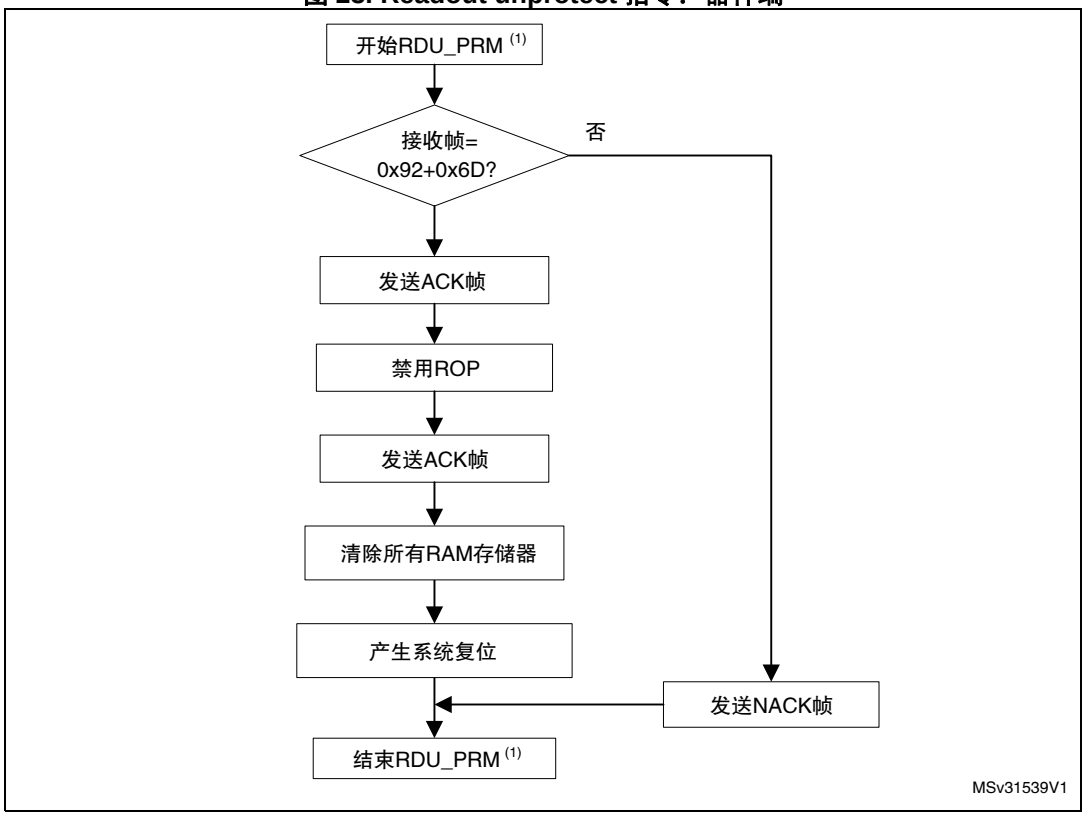
在 Readout Unprotect 指令结束时，自举程序会发送 ACK 并生成系统复位，以使选项字节的新配置生效。

图 22. Readout unprotect 指令：主机端



1. RDU_PRM = Readout Unprotect.

图 23. Readout unprotect 指令：器件端



1. RDU_PRM = Readout Unprotect.

2.12 No-Stretch Write memory 指令

No-Stretch Write Memory 指令用于向任何有效存储器区域写入数据。

当自举程序收到 No-Stretch Write Memory 指令时，它会向应用发送 ACK 字节。之后自举程序等待一个 4 字节的地址 （字节 1 为地址 MSB，字节 4 为 LSB）及校验和字节，然后校验收到的地址。

若收到的地址有效且校验和正确，则自举程序发送 ACK 字节；否则它发送 NACK 字节并终止该指令。若地址有效且校验和正确，则自举程序会：

1. 得到一个字节 N，它包含要接收的数据字节数
2. 接收 （(N + 1) 字节）用户数据及其校验和 （N 和所有数据字节的异或）
3. 从收到的地址开始将用户数据编程至存储器
4. 当操作正在进行时，返回忙状态 （0x76）

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 字节；否则它发送 NACK 字节并终止指令。

注： *若 No-Stretch Write Memory 指令用于选项字节区域，则自举程序会生成系统复位，以使选项字节的新配置生效。*

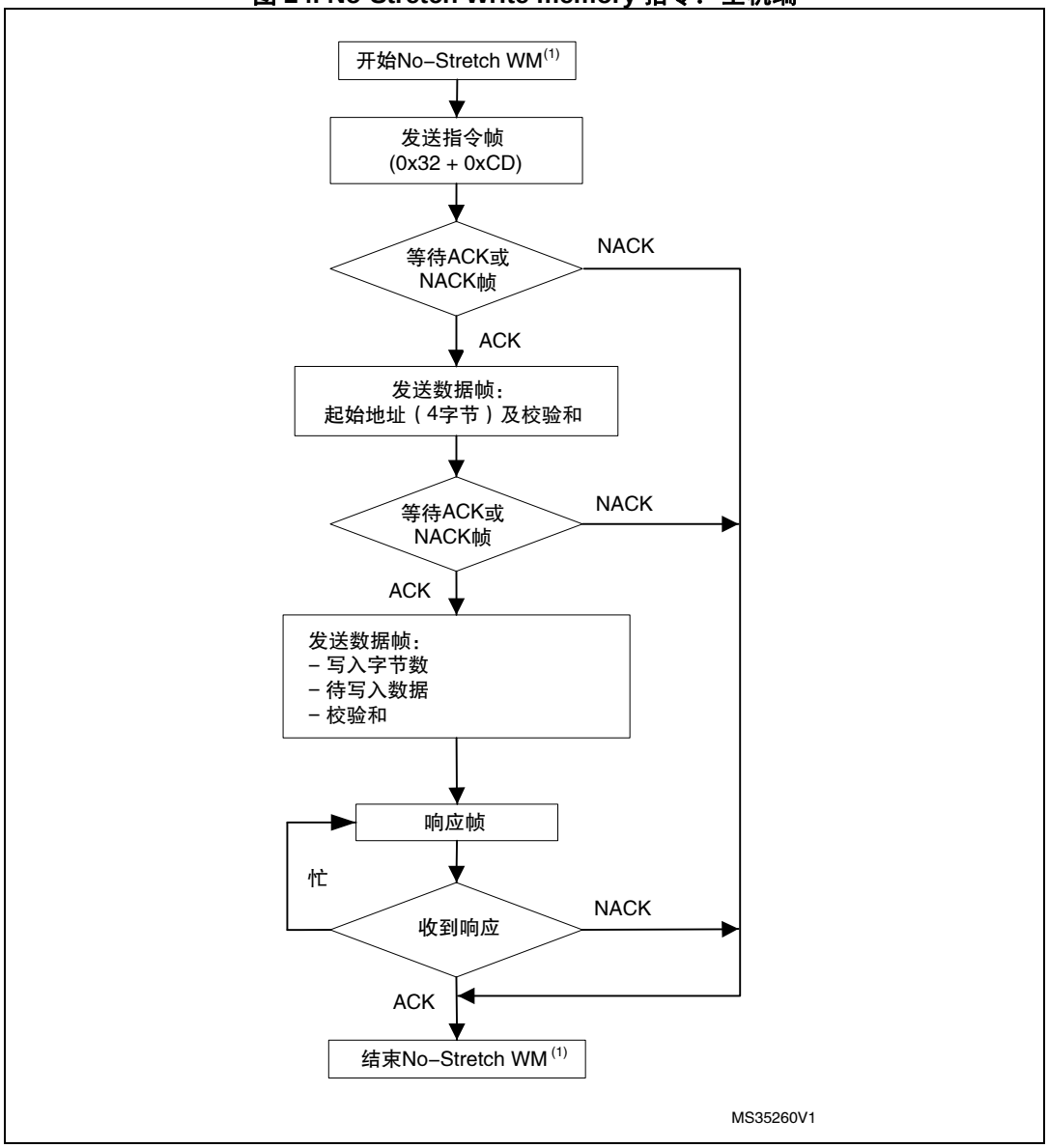
写入存储器的最大块长度为 256 字节，但对于选项字节的情况，最大长度取决于 STM32 产品，且从主机收到的地址必须为选项字节区域的起始地址。若需更多信息，请参考 STM32 产品参考手册。

当向写保护的扇区执行写操作时，不会返回错误。

主机向 STM32 发送的字节如下：

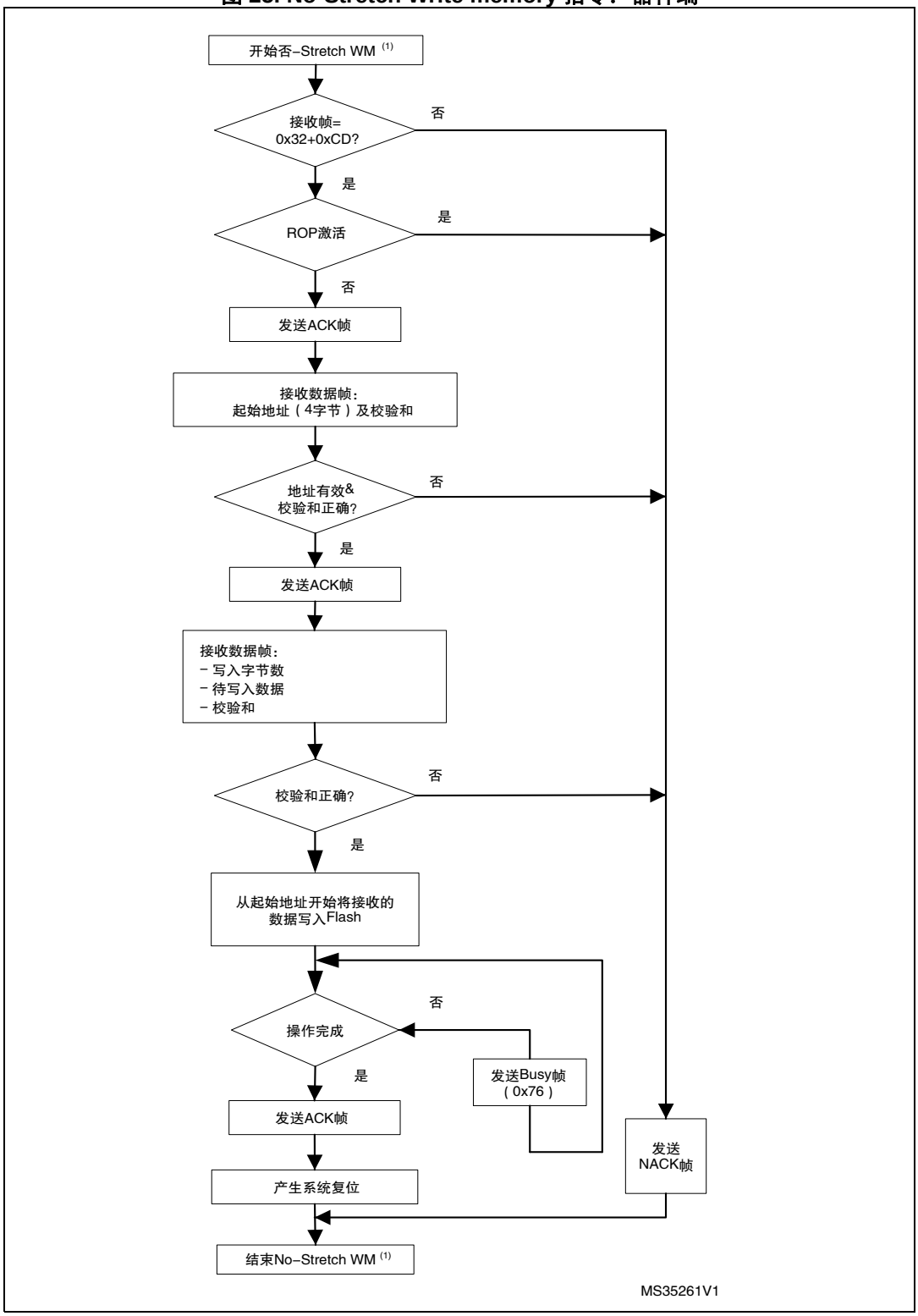
1. 字节 1：0x32
2. 字节 2：0xCD
3. 等待 ACK
4. 字节 3 至字节 6：起始地址
   - 字节 3：MSB
   - 字节 6：LSB
5. 字节 7：校验和：XOR （Byte3、 Byte4、 Byte5、 Byte6）
6. 等待 ACK
7. 字节 8：要接收的字节数 - 1 （0 < N ≤ 255）
8. N +1 数据字节：（最大 256 字节）
9. 校验和字节：XOR （N， N+1 数据字节）
10. 等待 ACK （若忙则继续轮询 ACK/NACK）

图 24. No-Stretch Write memory 指令：主机端



1. WM = Write Memory.

图 25. No-Stretch Write memory 指令：器件端



1. WM = Write Memory.

2.13 No-Stretch Erase memory 指令

No-Stretch Erase Memory 指令可使主机用双字节寻址模式擦除 Flash 页面或扇区。当自举程序收到 Erase Memory 指令时，它会向主机发送 ACK 字节。然后自举程序接收两个字节（要擦除的页面或扇区数）、 Flash 页面或扇区码 （每个都以双字节编码， MSB 在前）、校验和字节 （所发送字节的 XOR）。若校验和正确，则自举程序擦除存储器 （当操作正在进行时，返回忙状态 （0x76）），并向主机发送 ACK 字节；否则它向主机发送 NACK 字节，指令终止。

No-Stretch Erase Memory 指令规范

自举程序收到半个字 （两个字节），它包含 N—— 要擦除的页面或扇区数。对于 N = 0xFFFY （其中 Y 为 0 到 F），会执行特殊擦除 （0xFFFF 为全局批量擦除， 0xFFFE 和 0xFFFD 分别为批量擦除 bank1 和 bank2）。

注： *一些产品不支持批量擦除特性，在这种情况下您发送的擦除指令可使用所有页面或扇区的数目。*

注： *代码 0xFFFC 到 0xFFF0 为保留值。*

对于 0 ≤ N < 页面或扇区最大数的其它值，会擦除 N + 1 个页面或扇区。

自举程序接收：

- *对于特殊擦除的情况，一个字节：之前字节的校验和*
- *0x00 对应 0xFFFF，全局擦除*

若擦除 N+1 个页面或扇区，则自举程序接收 （2 x (N + 1)）个字节，其中的每半个字都包含了一个双字节编码的页面或扇区号， MSB 在前。这时所有前面字节的校验和都在一个字节接收。

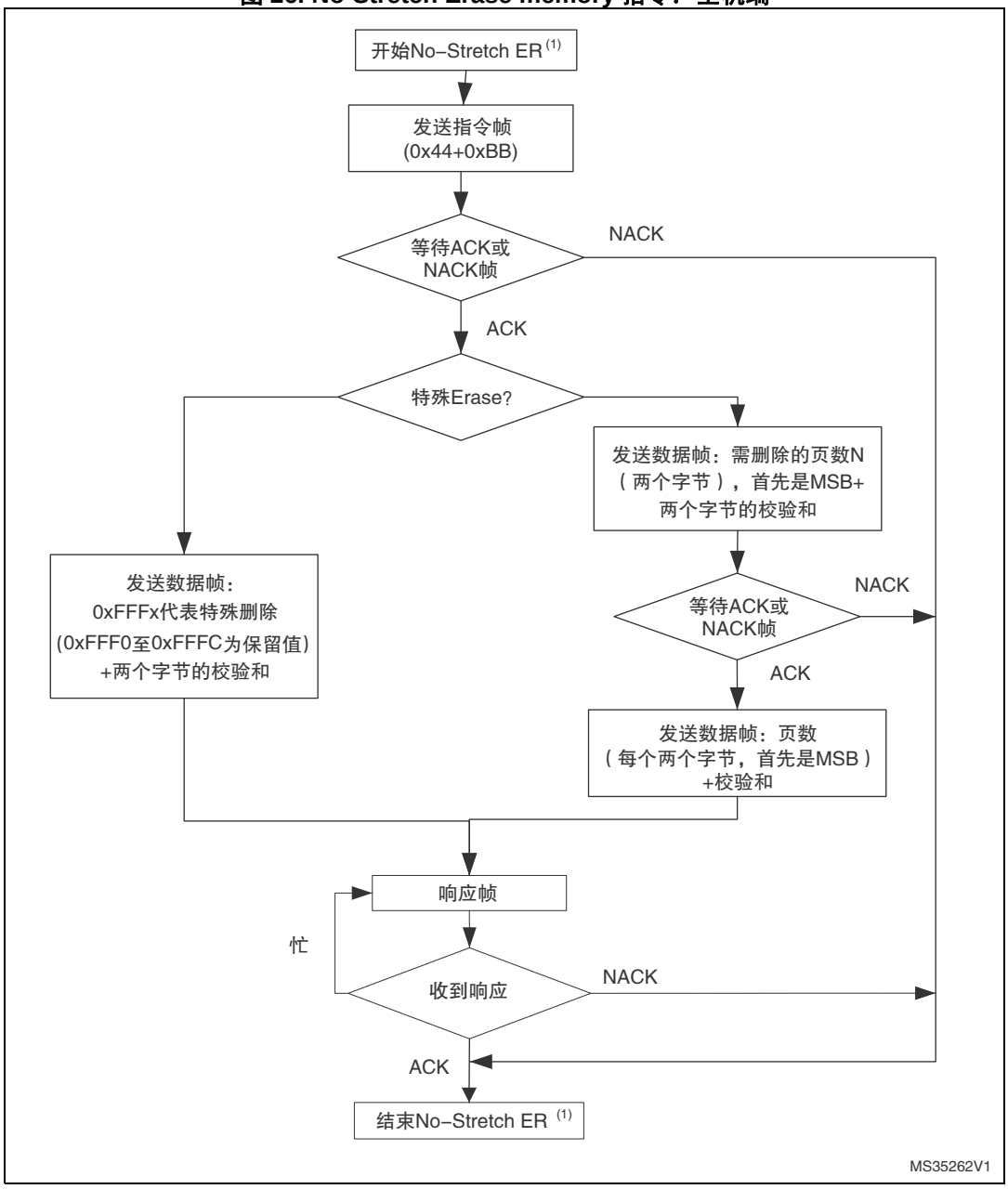
注： *当向写保护的扇区执行擦除操作时，不会返回错误。*

页面或扇区的最大数目与产品有关，因此应提起注意。

主机向 STM32 发送的字节如下：

1. 字节 1：0x44
2. 字节 2：0xBB
3. 等待 ACK
4. 字节 3-4：
   - 特殊擦除 （0xFFFx），或者
   - 要擦除的页面或扇区数 （N+1，其中：0 ≤ N < 页面或扇区的最大数目）
5. 等待 ACK （如未要求特殊擦除）
6. 其余字节：
   - 对于特殊擦除 （0x00），为字节 3-4 的校验和，或者
   - （2 x (N + 1)）字节 （双字节编码的页面或扇区号， MSB 在前），然后是字节 3-4 与其后所有字节的校验和）。
7. 等待 ACK （若忙则继续轮询 ACK/NACK）

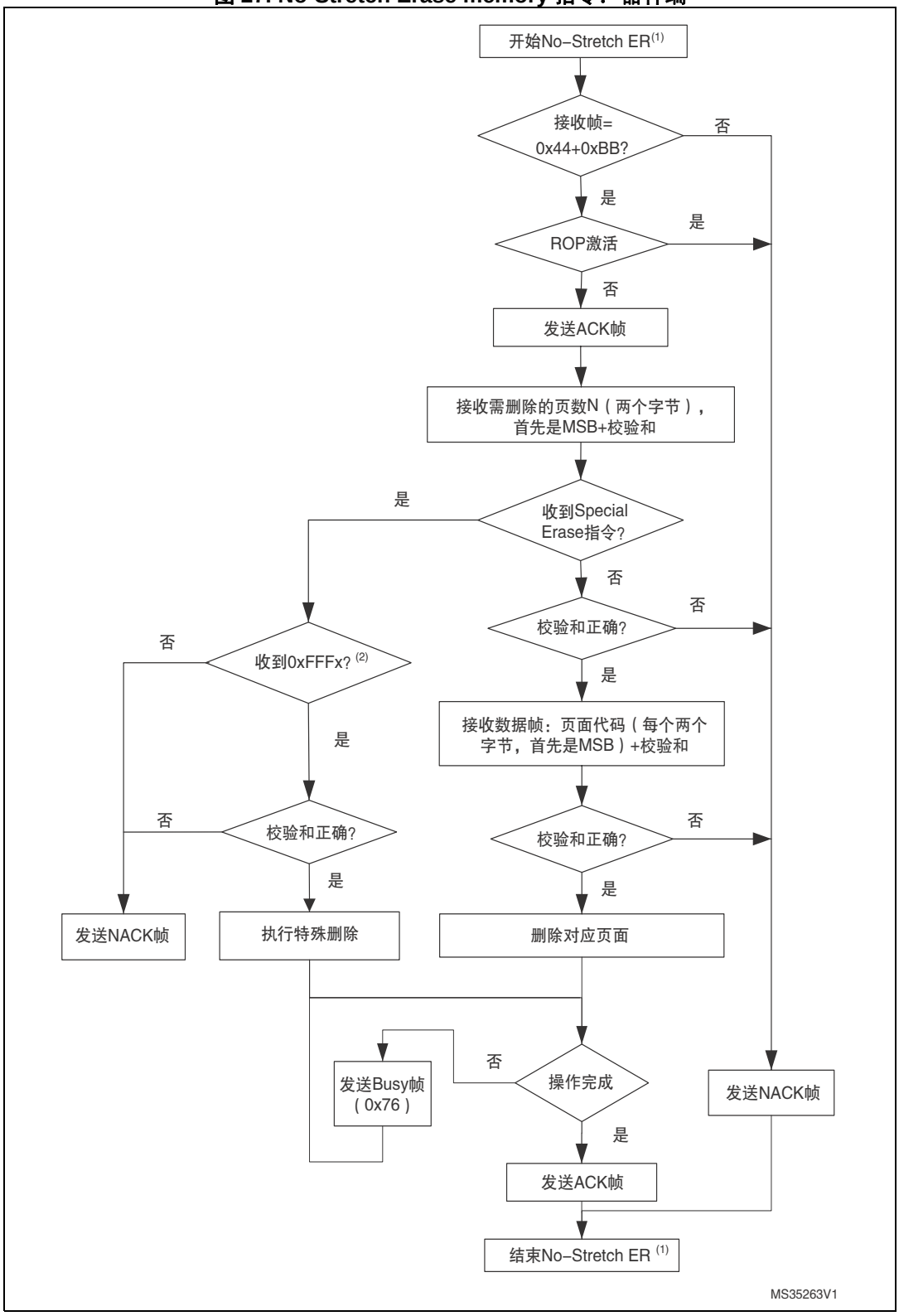
图 26. No-Stretch Erase memory 指令：主机端



1. ER = Erase Memory.

注： *一些产品不支持特殊擦除特性。对于这些产品，此命令将被 NACK。*

图 27. No-Stretch Erase memory 指令：器件端



1. ER = Erase Memory.
2. 若 STM32 产品不支持特殊擦除指令，则请求的特殊擦除指令将被 NACK。

2.14 No-Stretch Write protect 指令

No-Stretch Write Protect 指令用于对部分或全部 Flash 扇区启用写保护。当自举程序收到 Write Protect 指令时，它会向主机发送 ACK 字节。之后自举程序等待要接收的字节数 （要保护的扇区），然后从应用收到 Flash 扇区码。当操作正在进行时，返回忙状态 （0x76）。

在 No-Stretch Write Protect 指令结束时，自举程序会发送 ACK 字节并生成系统复位，以使选项字节的新配置生效。

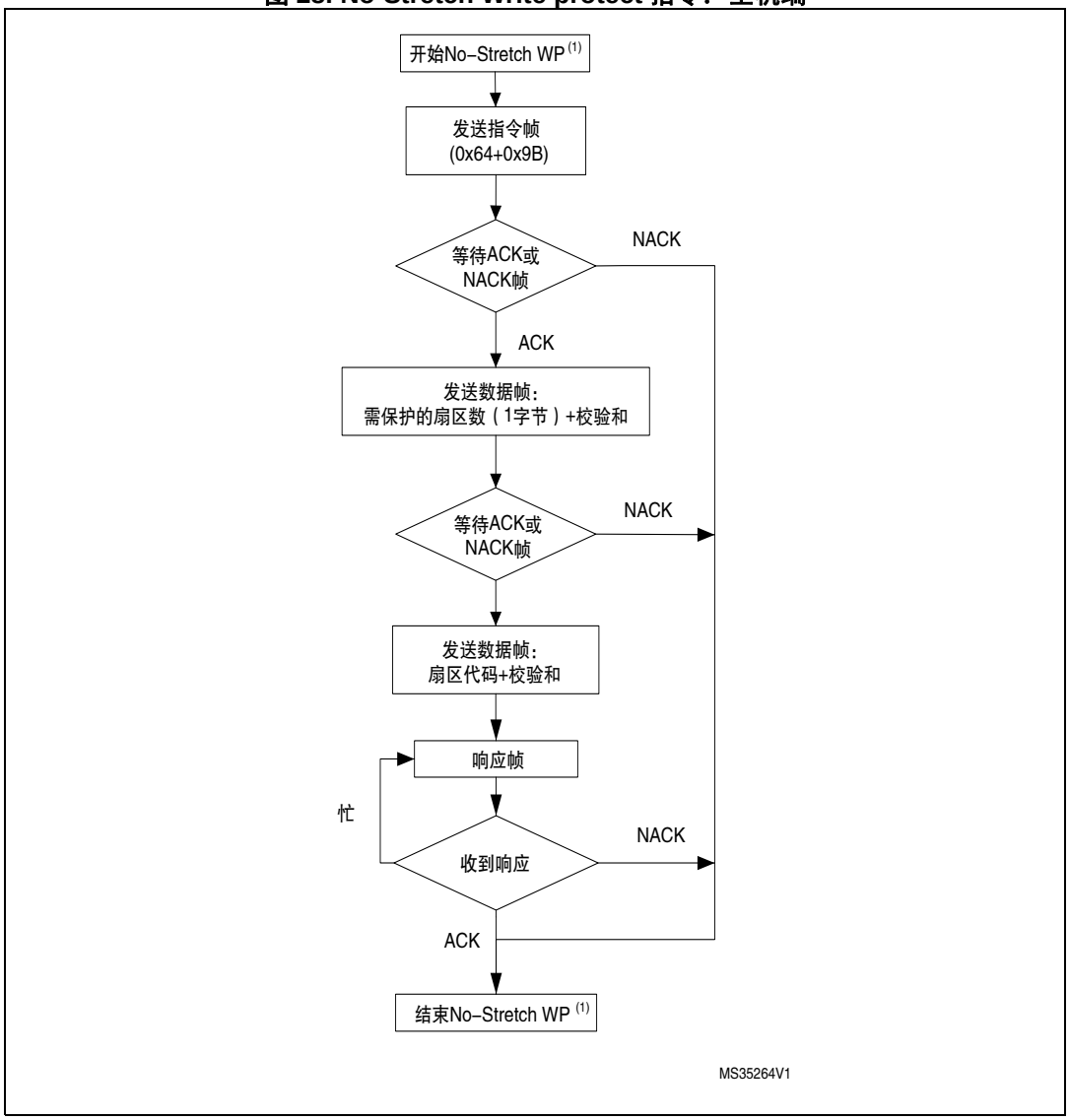
Write Protect 指令流程如下：

- *自举程序收到一个字节，它包含 N—— 要写保护的扇区数 - 1 （0 ≤ N ≤ 255）。*
- *自举程序接收 （N + 1）字节，其中每个字节都包含扇区码。*

注： *扇区总数和要保护的扇区号都不会被校验。这意味着即使指令的保护扇区数错误，或扇区号错误，都不会有错误返回。*

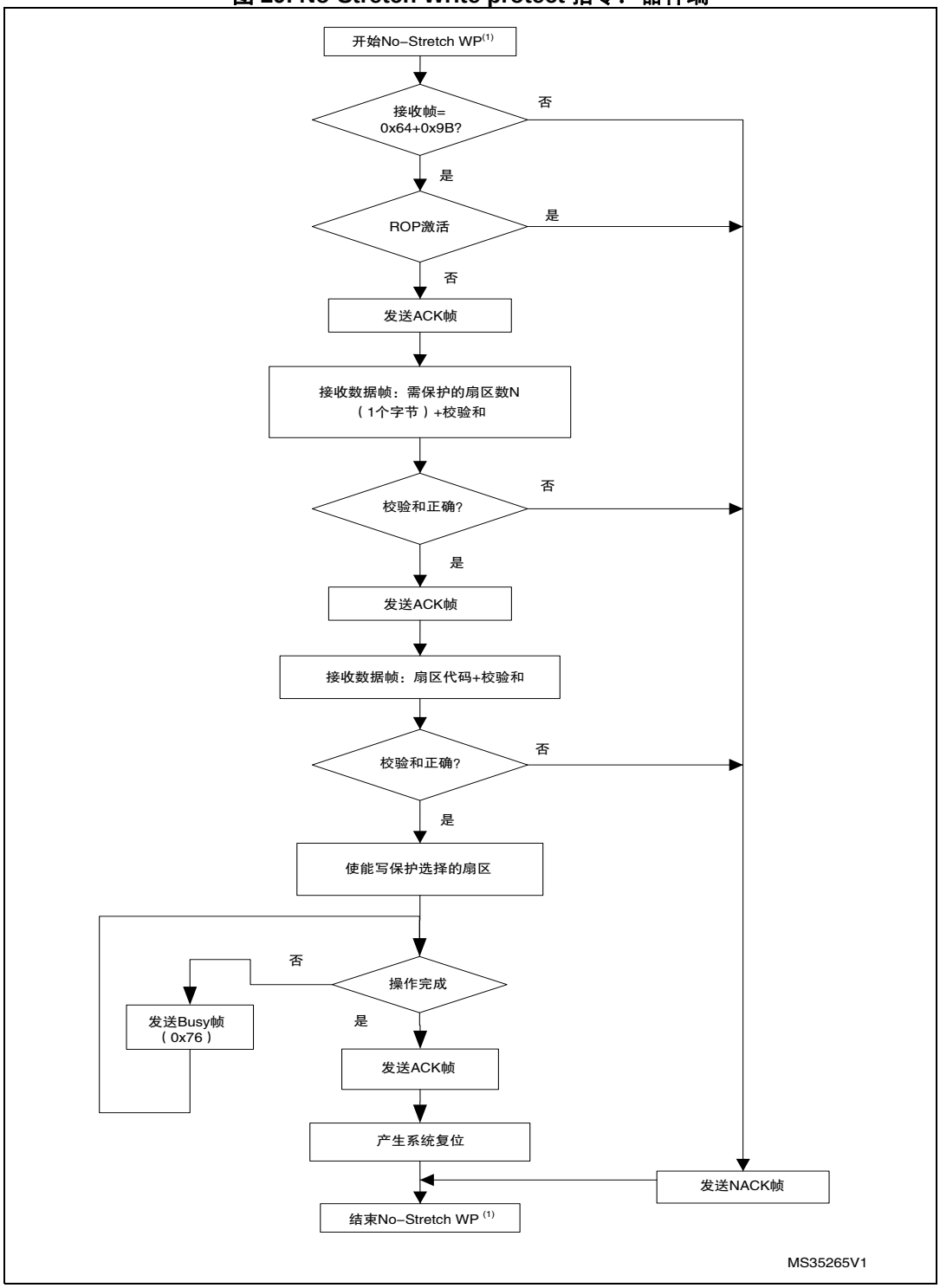
若执行了第二个 Write Protect 指令，则第一个指令已经保护的 Flash 扇区会被解除保护，只有第二个 Write Protect 指令内的扇区才会被保护。

图 28. No-Stretch Write protect 指令：主机端



1. WP = Write Protect.

图 29. No-Stretch Write protect 指令：器件端



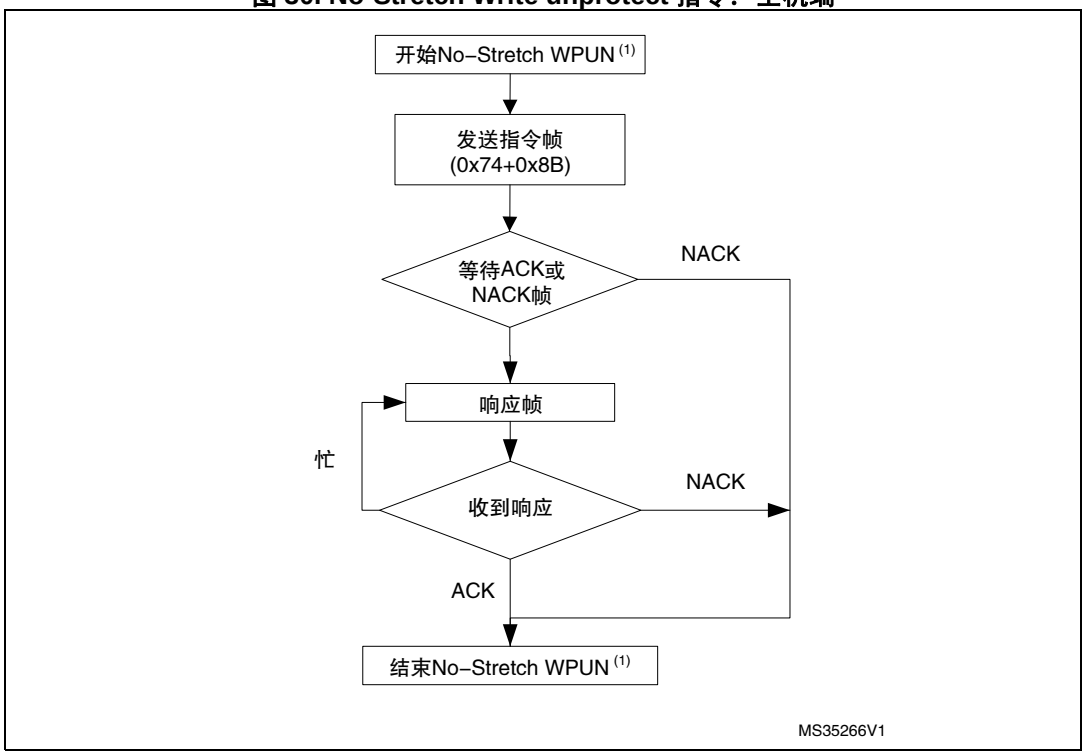
1. WP = Write Protect.

2.15 No-Stretch Write unprotect 指令

No-Stretch 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，它会向主机发送 ACK 字节。之后自举程序对全部 Flash 扇区禁用写保护，当操作正在进行时，返回忙状态 （0x76）。最后它发送 ACK 字节。

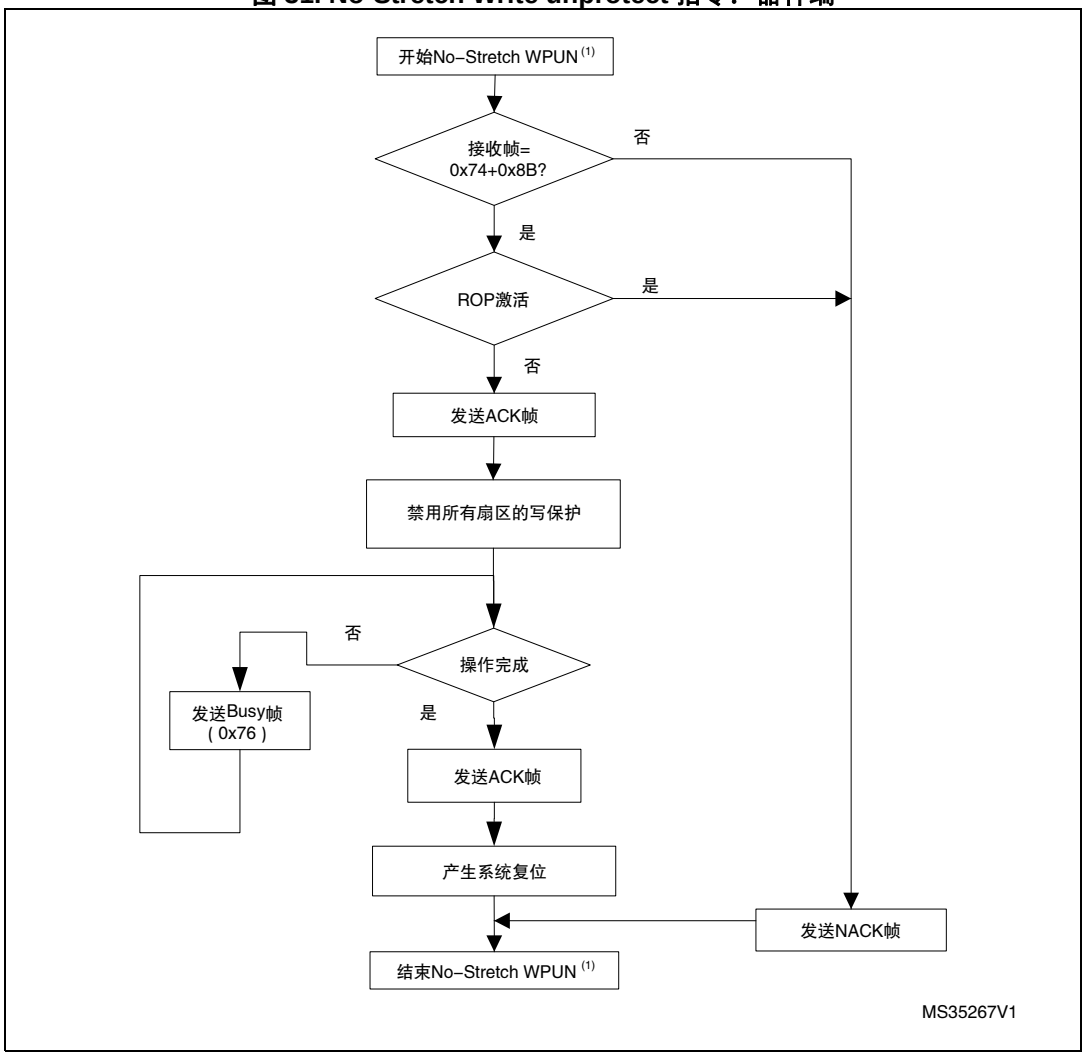
会生成系统复位以使选项字节的新配置生效。

图 30. No-Stretch Write unprotect 指令：主机端



1. WPUN = Write Unprotect.

图 31. No-Stretch Write unprotect 指令：器件端



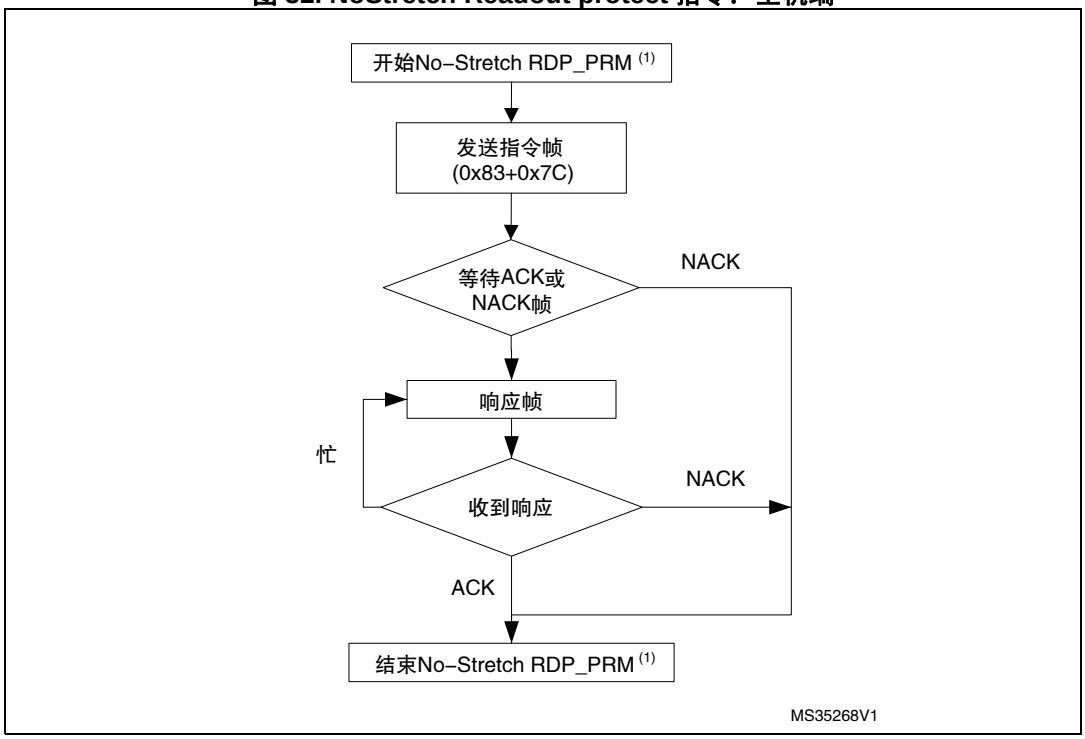
1. WPUN = Write Unprotect.

2.16 No-Stretch Readout protect 指令

No-Stretch Readout Protect 指令用于启用 Flash 的读保护。当自举程序收到 Readout Protect 指令时，它会向主机发送 ACK 字节并对 Flash 启用读保护。当操作正在进行时，返回忙状态 （0x76）。

在 No-Stretch Readout Protect 指令结束时，自举程序会发送 ACK 字节并生成系统复位，以使选项字节的新配置生效。

图 32. NoStretch Readout protect 指令：主机端



1. RDP_PRM = Readout Protect.

图 33. No-Stretch Readout protect 指令：器件端

1. RDP_PRM = Readout Protect.

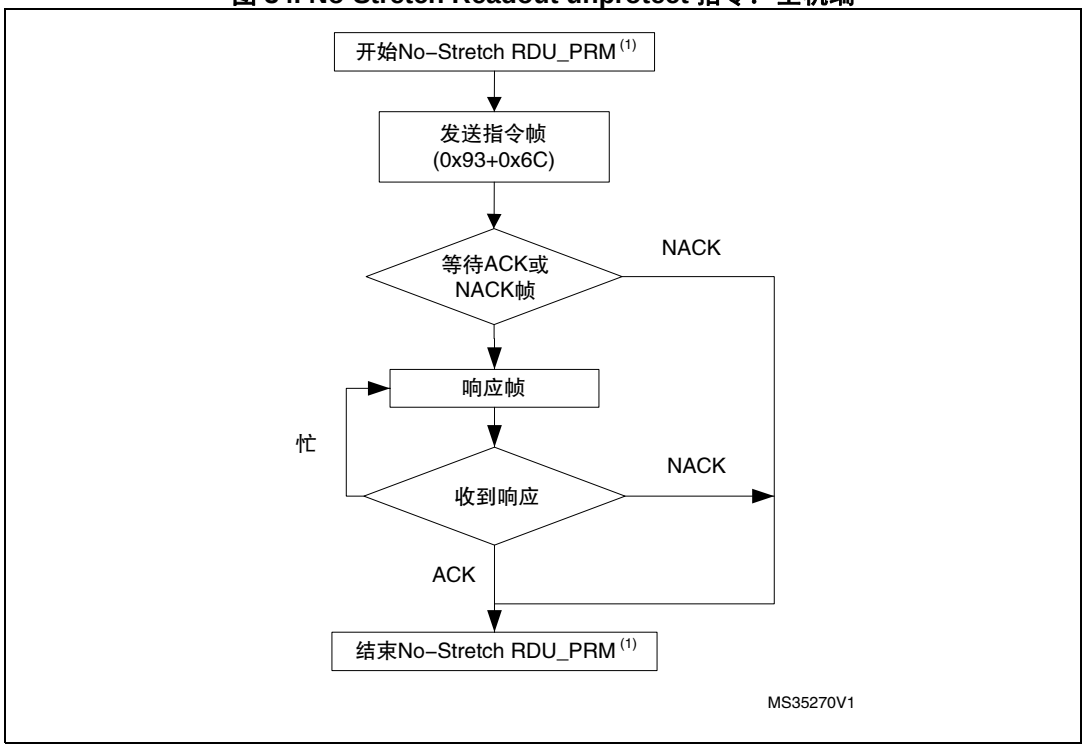
2.17 No-Stretch Readout unprotect 指令

No-Stretch 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 字节。

之后自举程序会对全部 Flash 禁用读保护，这会导致擦除全部 Flash。当操作正在进行时，返回忙状态 （0x76）。若该操作不成功，则自举程序会发送 NACK，读保护仍然有效。

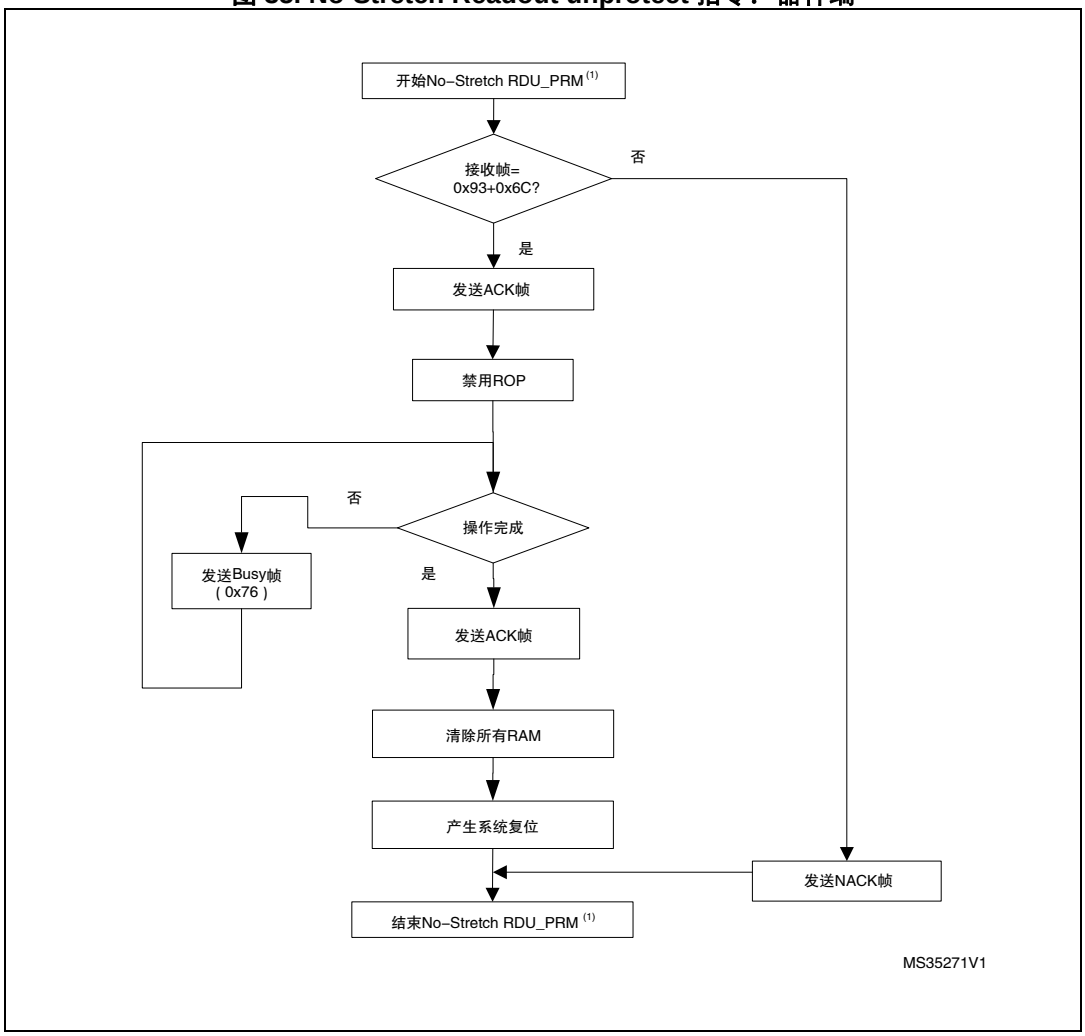
在 No-Stretch Readout Unprotect 指令结束时，自举程序会发送 ACK 并生成系统复位，以使选项字节的新配置生效。

图 34. No-Stretch Readout unprotect 指令：主机端



1. RDU_PRM = Readout Unprotect.

图 35. No-Stretch Readout unprotect 指令：器件端



1. RDU_PRM = Readout Unprotect.

3 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
V1.0	初始协议版本。
V1.1	此版本实现了新的 I2C 指令：No-Stretch Write Memory、No-Stretch Erase Memory、 No-Stretch Write Prtotect、 No-Stretch Write Unprotect、 No-Stretch ReadOut Protect、 No-Stretch ReadOut Unprotect。

4 修订历史

表 4. 文档修订历史

日期	修订	变更
2013 年 1 月 18 日	1	初始版本。
2014 年 5 月 2 日	2	更新了*表 1* 中的适用产品列表。 更新了*表 2* 中的指令集。 更新了*第 2 章节：自举程序指令集*。 增加了*第 2.12 章节*，*第 2.13 章节*，*第 2.14 章节*，*第 2.15 章节*、*第 2.16 章节*和*第 2.17 章节*。 在*表 3* 中增加了新的协议版本。

请仔细阅读：

中文翻译仅为方便阅读之目的。该翻译也许不是对本文档最新版本的翻译，如有任何不同，以最新版本的英文原版文档为准。

本文档中信息的提供仅与 ST 产品有关。意法半导体公司及其子公司 （“ST”）保留随时对本文档及本文所述产品与服务进行变更、更正、修改或改进的权利，恕不另行通知。

所有 ST 产品均根据 ST 的销售条款出售。

买方自行负责对本文所述 ST 产品和服务的选择和使用，ST 概不承担与选择或使用本文所述 ST 产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为 ST 授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在 ST 的销售条款中另有说明，否则，ST 对 ST 产品的使用和 / 或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途 （及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

意法半导体的产品不得应用于武器。此外，意法半导体产品也不是为下列用途而设计并不得应用于下列用途：（A）对安全性有特别要求的应用，例如，生命支持、主动植入设备或对产品功能安全有要求的系统； （B）航空应用； （C）汽车应用或汽车环境，且 / 或 （D）航天应用或航天环境。如果意法半导体产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向意法半导体发出了书面通知，采购商仍将独自承担因此而导致的任何风险，意法半导体的产品设计规格明确指定的汽车、汽车安全或医疗工业领域专用产品除外。根据相关政府主管部门的规定，ESCC、QML 或 JAN 正式认证产品适用于航天应用。

经销的 ST 产品如有不同于本文档中提出的声明和 / 或技术特点的规定，将立即导致 ST 针对本文所述 ST 产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大 ST 的任何责任。

本文档中的信息取代之前提供的所有信息。

意法半导体集团公司

澳大利亚 - 比利时 - 巴西 - 加拿大 - 中国 - 捷克共和国 - 芬兰 - 法国 - 德国 - 中国香港 - 印度 - 以色列 - 意大利 - 日本 - 马来西亚 - 马耳他 - 摩洛哥 - 菲律宾 - 新加坡 - 西班牙 - 瑞典 - 瑞士 - 英国 - 美国

www.st.com