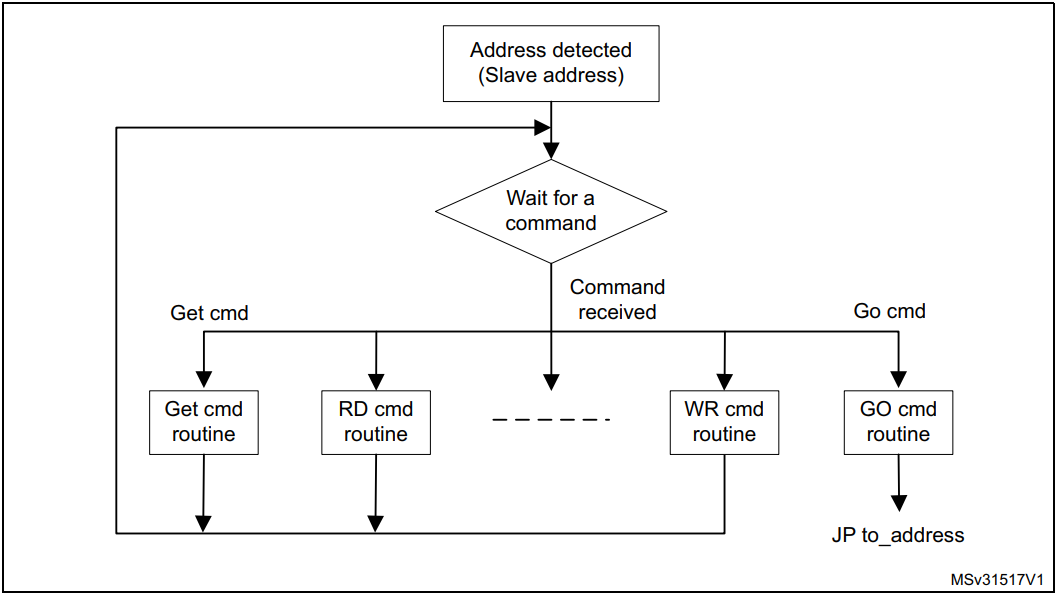
**1 Последовательность работы I2C загрузчика**

Последовательность работы кода загрузчика I2C для микроконтроллеров STM32, основанных на ядре Arm®(a), показана на рисунке 1.

**Рисунок 1. I2C загрузчик для STM32**   


***Примечание****:* Адрес подчиненного устройства I2C для каждого загрузчика продукта указан в AN2606.

После входа в режим загрузки системной памяти и настройки микроконтроллера STM32 (более подробную информацию см. в примечаниях к применению режима загрузки системной памяти STM32), код загрузчика начинает сканировать вывод линии I2C\_SDA, ожидая обнаружения собственного адреса на шине. После обнаружения микропрограмма загрузчика I2C начинает получать команды хоста.

**2 Набор команд загрузчика**

Команды без растяжения (No-Stretch) поддерживаются, начиная с версии протокола V1.1, и позволяют лучше управлять командами, когда хосту приходится ждать значительное время, прежде чем загрузчик завершит операцию.  
 По возможности рекомендуется использовать команды «Без растяжения» вместо эквивалентных обычных команд.  
 Поддерживаемые команды перечислены в таблице 2.

**Таблица 2. Команды загрузчика I2C**

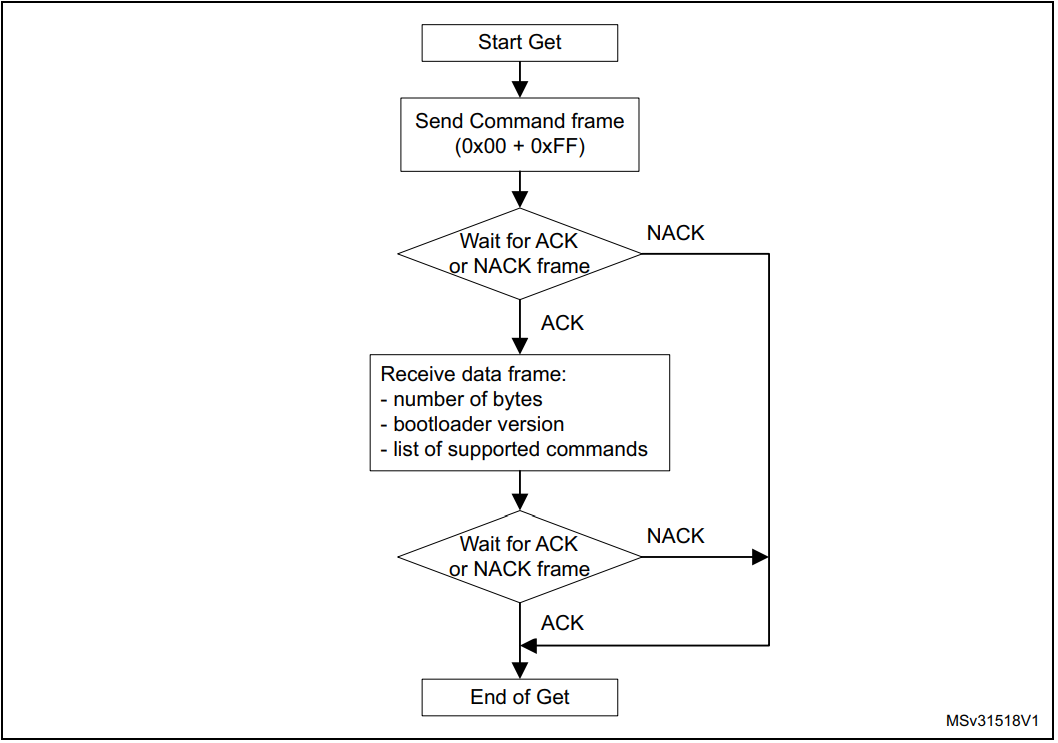
|  |  |  |
| --- | --- | --- |
| **Команда (1)** | **Код команды** | **Описание** |
| **Get (2)** | 0x00 | Получает версию и разрешенные команды, поддерживаемые текущей версией загрузчика. |
| **Get Version (2)** | 0x01 | Получает версию загрузчика. |
| **Get ID (2)** | 0x02 | Получает идентификатор чипа |
| **Read Memory** | 0x11 | Читает до 256 байт памяти, начиная с адреса, указанного приложением. |
| **Go (3)** | 0x21 | Переходит к коду пользовательского приложения, расположенному во внутренней флэш-памяти. |
| **Write Memory (3)** | 0x31 | Записывает в память до 256 байт, начиная с адреса, указанного приложением. |
| **No-Stretch Write Memory (3)(4)** | 0x32 | Записывает в память до 256 байт, начиная с адреса, указанного приложением, и возвращает состояние занятости во время выполнения операции. |
| **Erase** | 0x44 | Стирает от одной до всех страниц или секторов флэш-памяти, используя режим двухбайтовой адресации. |
| **No-Stretch Erase (3)(4)** | 0x45 | Стирает от одной до всех страниц или секторов флэш-памяти, используя режим двухбайтовой адресации, и возвращает состояние занятости во время выполнения операции. |
| **Special** | 0x50 | Общая команда, которая позволяет добавлять новые функции в зависимости от ограничений продукта, не добавляя новую команду для каждой функции. |
| **Extended Special** | 0x51 | Общая команда, которая позволяет пользователю отправлять больше данных по сравнению со специальной командой. |
| **Write Protect** | 0x63 | Включает защиту от записи для некоторых секторов. |
| **No-Stretch Write Protect (4)** | 0x64 | Включает защиту от записи для некоторых секторов и возвращает состояние занятости во время выполнения операции. |
| **Write Unprotect** | 0x73 | Отключает защиту от записи для всех секторов флэш-памяти. |
| **No-Stretch Write Unprotect (4)** | 0x74 | Отключает защиту от записи для всех секторов флэш-памяти и возвращает состояние занятости во время выполнения операции. |
| **Readout Protect** | 0x82 | Включает защиту от чтения. |
| **No-Stretch Readout Protect (4)** | 0x83 | Включает защиту от чтения и возвращает состояние занятости во время выполнения операции. |
| **Readout Unprotect (2)** | 0x92 | Отключает защиту от чтения. |
| **No-Stretch Readout Unprotect (2)(4)** | 0x93 | Отключает защиту от чтения и возвращает состояние занятости во время выполнения операции. |
| **No-Stretch Get Memory Checksum (2)** | 0xA1 | Получает значение контрольной суммы CRC для области памяти на основе ее смещения и длины. |

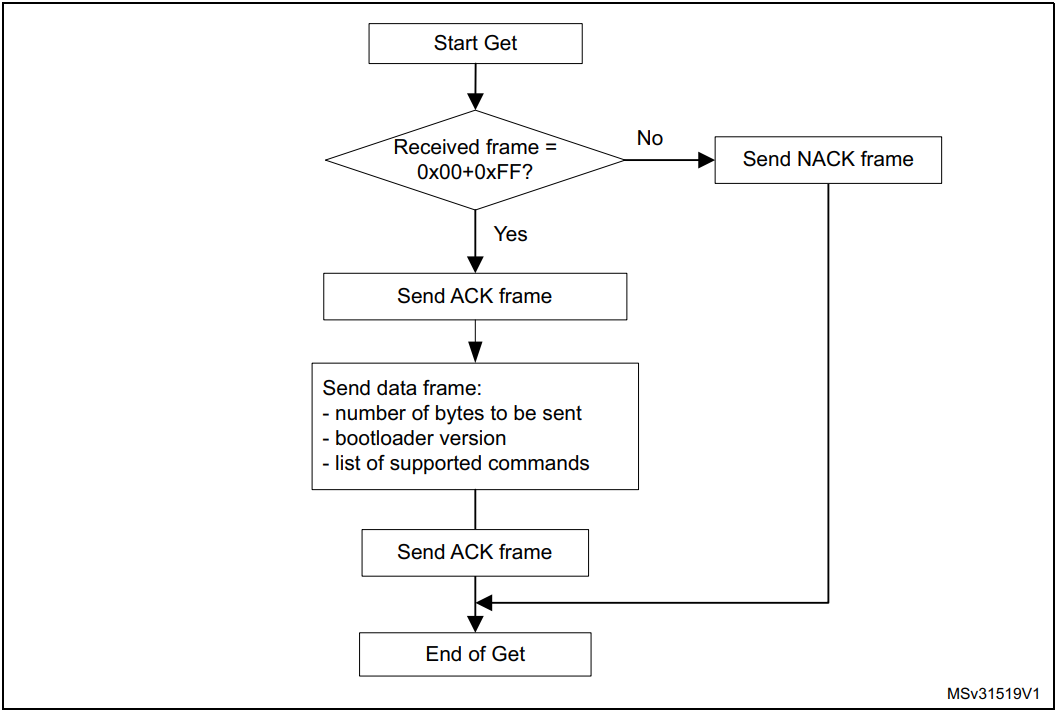
1. Если получена запрещенная команда или если во время выполнения команды возникает ошибка, загрузчик отправляет байт NACK и возвращается к проверке команды.  
 2. Защита от чтения. Когда параметр RDP (защита от чтения) активен, доступно только это ограниченное подмножество команд. Все остальные команды получают NACK и не влияют на устройство. После удаления RDP другие команды становятся активными.  
 3. Обратитесь к техническому описанию продукта STM32 и к AN2606, чтобы узнать области памяти, допустимые для этих команд.  
 4. Команды No-Stretch доступны только с протоколом I2C версии 1.1.

**Команды без растяжения** (**No-Stretch commands**)  
 Команды No-Stretch позволяют выполнять операции Write, Erase, Write Protect, Write Unprotect, Read Protect и Read Unprotect без растягивания линии I2C, пока загрузчик выполняет операцию. Возможна связь с другими устройствами на шине, пока загрузчик выполняет операции, требующие времени ожидания.  
 Отличие этих команд от стандартных заключается в конце команды: когда Хост запрашивает ACK/NACK в конце команды, вместо растягивания линии I2C загрузчик отвечает третьим состоянием — Занято (0x76). Когда хост получает состояние «Занято», он снова опрашивает состояние и считывает один байт, пока не получит пока не получит ответ ACK или NACK.

**Безопасность связи (Communication safety)** Все сообщения от хоста программирования к устройству проверяются по контрольной сумме. Полученные блоки байтов данных подвергаются операции XOR. Байт, содержащий вычисленное XOR всех предыдущих байтов, добавляется в конец каждого сообщения (байт контрольной суммы). При выполнении XOR всех полученных байтов, данных и контрольной суммы результат в конце пакета должен быть 0x00.  
 Для каждой команды хост отправляет байт и его дополнение (XOR = 0x00).  
 Каждый пакет либо принимается (ответ ACK), либо отбрасывается (ответ NACK):  
- ACK = 0x79  
- NACK = 0x1F

С командами без растяжения (No-Stretch commands) состояние занятости BUSY отправляется вместо ACK или NACK, когда операция выполняется:  
- BUSY = 0x76  
  
***Примечание***: Главный фрейм может быть одним из следующих:  
 **-Отправить кадр команды**: Хост инициирует связь в качестве главного передатчика и отправляет на устройство два байта: код команды + XOR.  
 **-Ожидание кадра ACK/NACK**: Хост инициирует связь I2C в качестве главного получателя и получает от устройства один байт: ACK, NACK или BUSY.  
 **-Получение кадра данных**: Хост инициирует связь I2C в качестве главного приемника и получает ответ от устройства. Количество полученных байтов зависит от команды.  
 **-Кадр отправки данных**: Хост инициирует связь I2C в качестве главного передатчика и отправляет необходимые байты на устройство. Количество передаваемых байтов зависит от команды.  
  
**Внимание!** Для команд Write, Erase и Read Unprotect хост должен соблюдать соответствующие временные интервалы (т. е. запись страницы, стирание сектора), указанные в технических описаниях продуктов. Например, при запуске команды Erase хост должен ждать (до последнего ACK команды) в течение времени, эквивалентного максимальному времени стирания сектора/страницы, указанному в таблице данных (или, по крайней мере, типичному времени стирания сектора/страницы).  
  
**Внимание!** Для связи I2C реализован механизм тайм-аута, его необходимо соблюдать для корректного выполнения команд загрузчика. Этот тайм-аут реализуется между двумя кадрами I2C в одной команде. Например, для команды записи в память между кадром отправки команды и кадром отправки адресной памяти вставляется тайм-аут. Кроме того, один и тот же период тайм-аута вставляется между двумя последовательными моментами приема или передачи данных в одном и том же кадре I2C. Если время ожидания истекло, генерируется сброс системы, чтобы избежать сбоя загрузчика. Обратитесь к разделу, посвященному времени подключения I2C AN2606, чтобы получить значение тайм-аута I2C для каждого продукта STM32.  
  
**2.1 Команда Get** Команда Get позволяет пользователю получить версию загрузчика и поддерживаемые команды. Когда загрузчик получает команду Get, он передает хосту версию загрузчика и поддерживаемые коды команд, как показано на рисунке 2.

**Рисунок 2. Команда Get: сторона хоста**

**Рисунок 3. Команда Get: сторона устройства**

STM32 отправляет байты следующим образом:  
для протокола V1.0:  
– Byte 1: ACK  
– Byte 2: N = 11 = Number of bytes to follow - 1, except current and ACKs  
– Byte 3: Bootloader version 0x10 = Version 1.0  
– Byte 4: 0x00 - Get command   
– Byte 5: 0x01 - Get Version   
– Byte 6: 0x02 - Get ID   
– Byte 7: 0x11 - Read Memory command   
– Byte 8: 0x21 - Go command   
– Byte 9: 0x31 - Write Memory command   
– Byte 10: 0x44 - Erase command   
– Byte 11: 0x63 - Write Protect command   
– Byte 12: 0x73 - Write Unprotect command   
– Byte 13: 0x82 - Readout Protect command   
– Byte 14: 0x92 - Readout Unprotect command   
– Byte 15: ACK  
для протокола V1.1:  
– Byte 1: ACK

– Byte 2: N = 17 = Number of bytes to follow - 1, except current and ACKs

– Byte 3: Bootloader version 0x11 = Version 1.1

– Byte 4: 0x00 - Get command

– Byte 5: 0x01 - Get Version

– Byte 6: 0x02 - Get ID

– Byte 7: 0x11 - Read Memory command

– Byte 8: 0x21 - Go command

– Byte 9: 0x31 - Write Memory command

– Byte 10: 0x44 - Erase command

– Byte 11: 0x63 - Write Protect command

– Byte 12: 0x73 - Write Unprotect command

– Byte 13: 0x82 - Readout Protect command

– Byte 14: 0x92 - Readout Unprotect command

– Byte 15: 0x32 - No-Stretch Write Memory command

– Byte 16: 0x45 - No-Stretch Erase command

– Byte 17: 0x64 - No-Stretch Write Protect command

– Byte 18: 0x74 - No-Stretch Write Unprotect command

– Byte 19: 0x83 - No-Stretch Readout Protect command

– Byte 20: 0x93 - No-Stretch Readout Unprotect command

– Byte 21: ACK

для протокола V1.2:  
– Byte 1: ACK

– Byte 2: N = 18 = Number of bytes to follow - 1, except current and ACKs

– Byte 3: Bootloader version 0x12 = Version 1.2

– Byte 4: 0x00 - Get command

– Byte 5: 0x01 - Get Version

– Byte 6: 0x02 - Get ID

– Byte 7: 0x11 - Read Memory command

– Byte 8: 0x21 - Go command

– Byte 9: 0x31 - Write Memory command

– Byte 10: 0x44 - Erase command

– Byte 11: 0x63 - Write Protect command

– Byte 12: 0x73 - Write Unprotect command

– Byte 13: 0x82 - Readout Protect command

– Byte 14: 0x92 - Readout Unprotect command

– Byte 15: 0x32 - No-Stretch Write Memory command

– Byte 16: 0x45 - No-Stretch Erase command

– Byte 17: 0x64 - No-Stretch Write Protect command

– Byte 18: 0x74 - No-Stretch Write Unprotect command

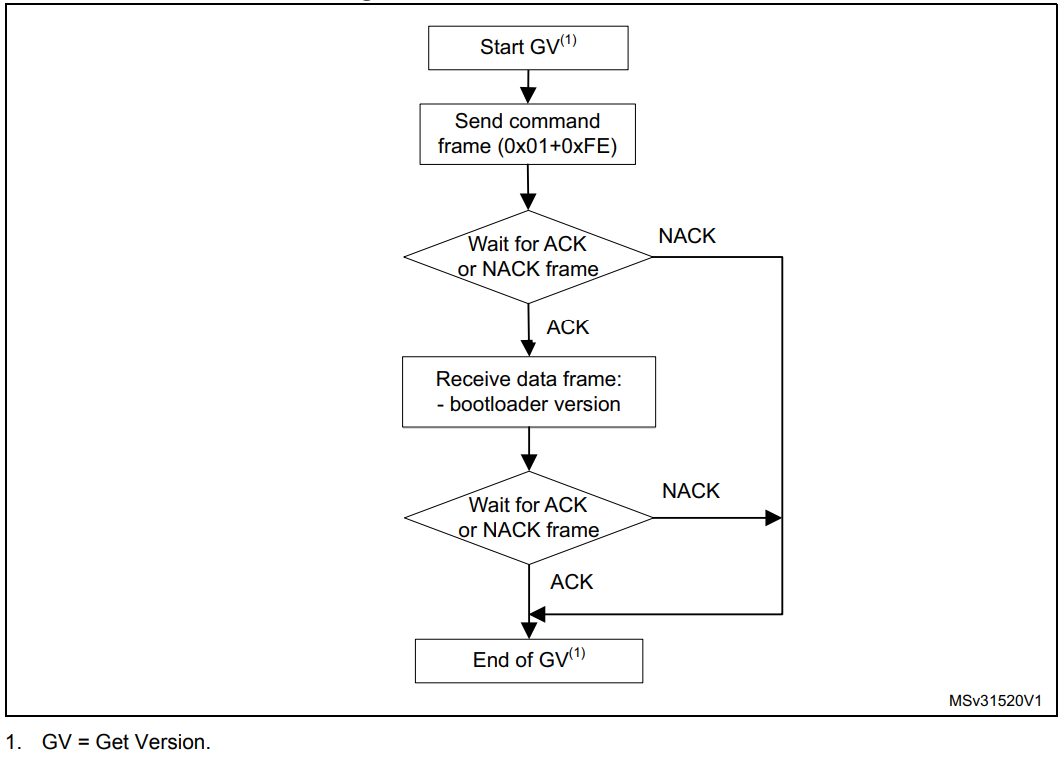
– Byte 19: 0x83 - No-Stretch Readout Protect command

– Byte 20: 0x93 - No-Stretch Readout Unprotect command

– Byte 21: 0xA1 - No-Stretch Get Memory Checksum command

– Byte 22: ACK

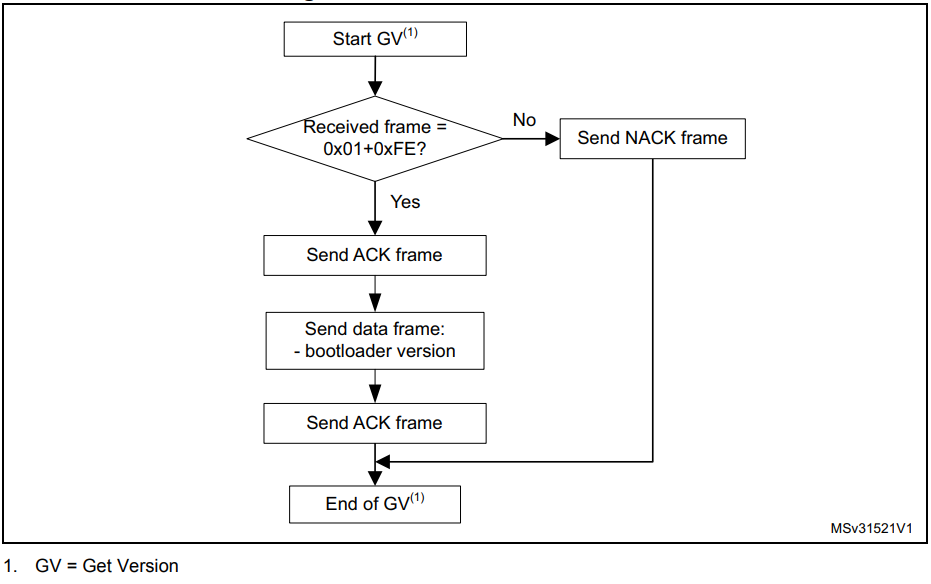
**2.2 Команда Get Version**

Команда Get Version используется для получения версии загрузчика I2C. Когда загрузчик получает команду, он передает хосту описанную ниже информацию (версия загрузчика).  
**Рисунок 4. Команда Get Version: сторона Хоста**

STM32 отправляет байты следующим образом:  
- Byte 1: ACK

- Byte 2: Bootloader version (0 < Version <= 255) (for example, 0x10 = Version 1.0)

- Byte 3: ACK

**Рисунок 5. Команда Get Version: сторона устройства**

**2.3 Команда Get ID** Команда Get ID используется для получения версии идентификатора чипа. Когда загрузчик получает эту команду, он передает идентификатор продукта хосту.

Устройство STM32 отправляет байты следующим образом:  
- Byte 1: ACK

- Byte 2: N = the number of bytes-1 (for STM32, N = 1), except for current byte and ACKs

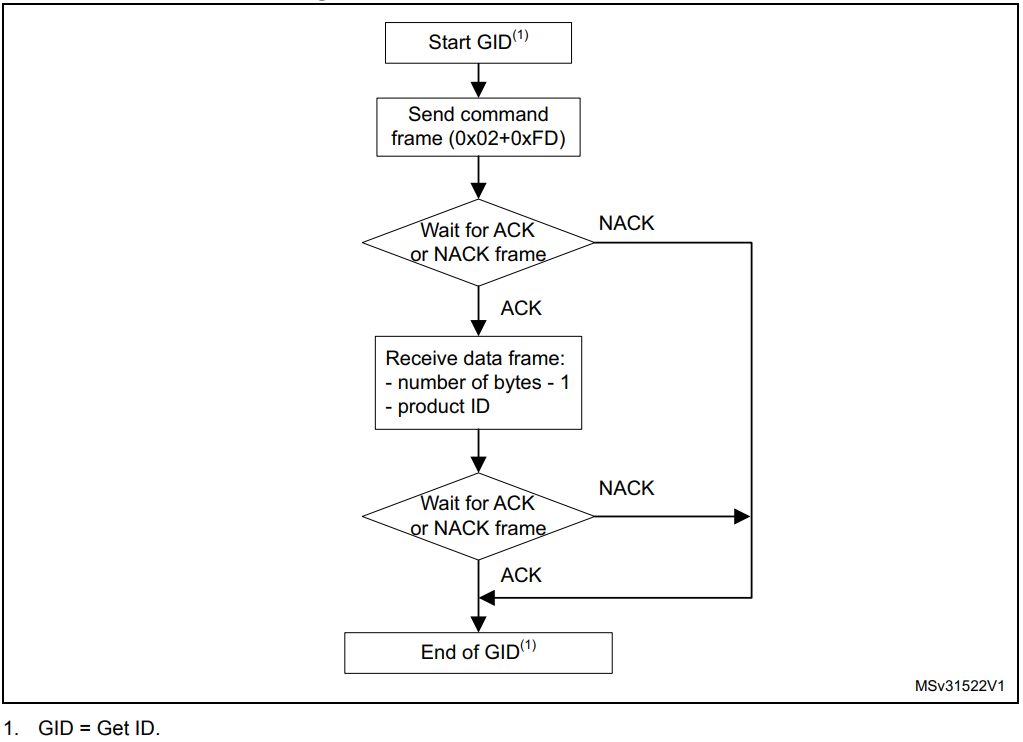
- Bytes 3-4: PID (product ID)

- Byte 3 = MSB

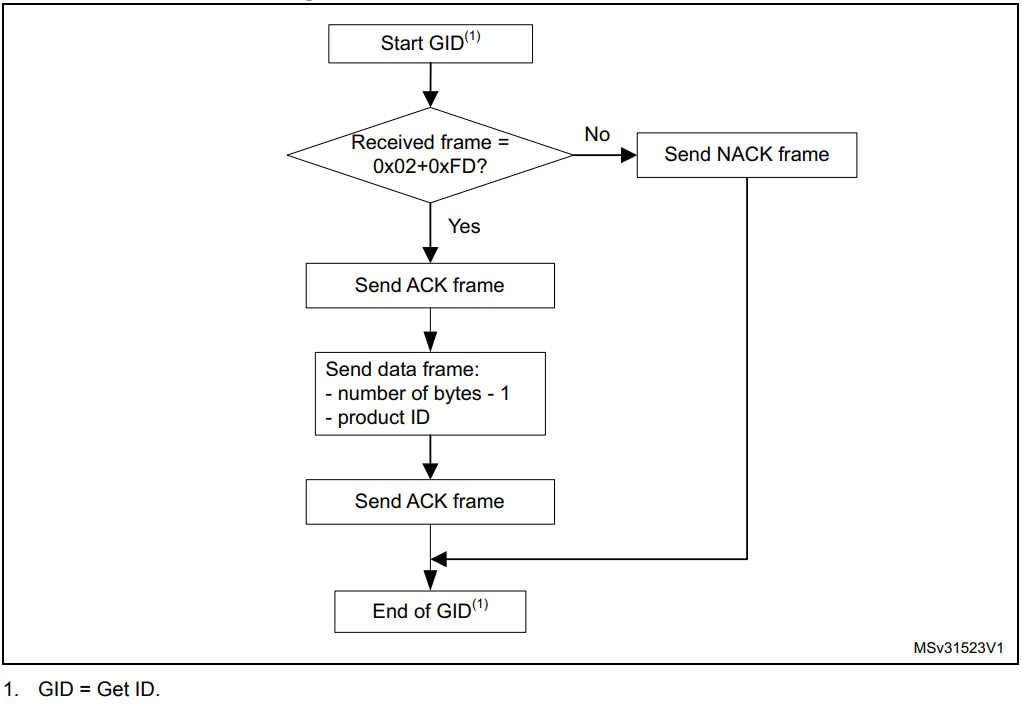
- Byte 4 = LSB

- Byte 5: ACK

**Рисунок 6. Команда Get ID: сторона хоста**



**Рисунок 7. Команда Get ID: сторона устройства**



**2.4 Команда Read Memory** Команда Read Memory используется для чтения данных с любого допустимого адреса памяти.

Когда загрузчик получает команду Read Memory, он передает приложению байт ACK. Затем загрузчик ожидает 4-байтовый адрес (1-й байт - MSB, 4-й байт - LSB) и байт контрольной суммы, после чего проверяет полученный адрес. Если адрес действителен и контрольная сумма верна, загрузчик передает байт ACK, в противном случае он передает байт NACK и прерывает команду.  
 Если адрес действителен и контрольная сумма верна, загрузчик ожидает количества передаваемых байтов (N байтов) и его дополненного байта (контрольная сумма). Если контрольная сумма верна, загрузчик передает необходимые данные приложению, начиная с полученного адреса. Если контрольная сумма неверна, он отправляет NACK перед прерыванием команды.  
  
 Хост отправляет байты на STM32 следующим образом:  
- Bytes 1-2: 0x11 + 0xEE

Wait for ACK

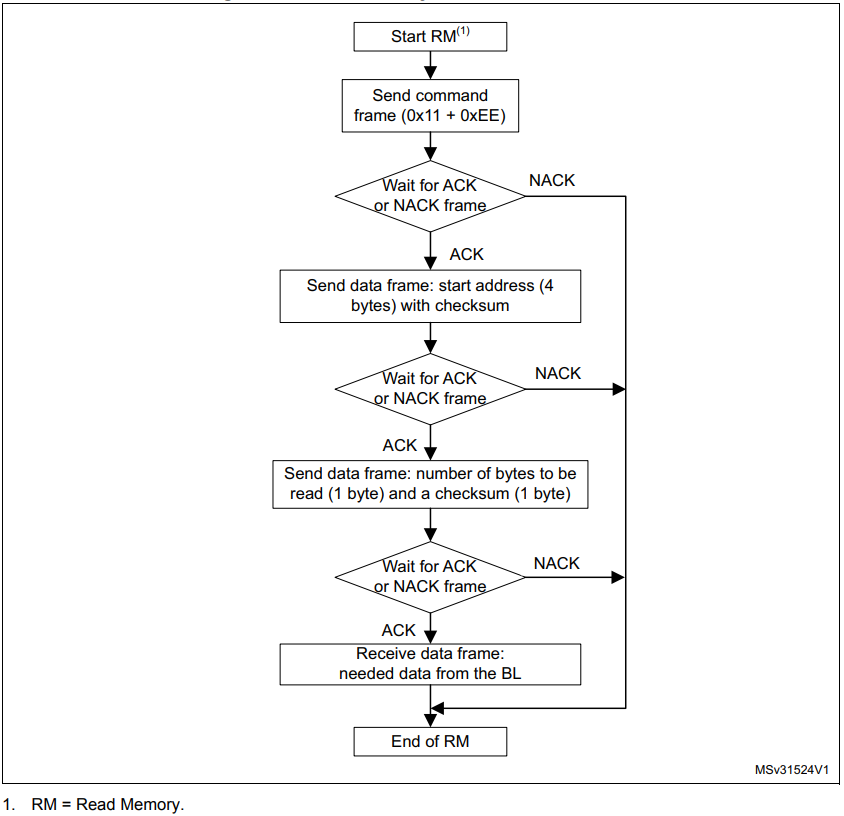
- Bytes 3-6: Start address (byte 3: MSB, byte 6: LSB)

- Byte 7: Checksum: XOR (byte 3, byte 4, byte 5, byte 6)

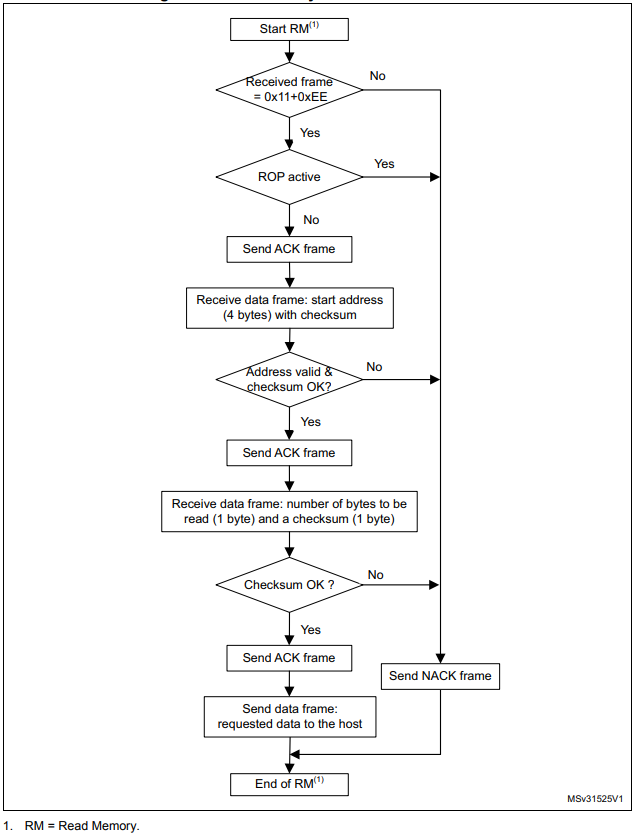
Wait for ACK

- Byte 8: The number of bytes to be read - 1 (0 < N <= 255)

- Byte 9: Checksum: XOR byte 8 (complement of byte 8)

**Рисунок 8. Команда Read Memory: сторона хоста**

**Рисунок 9. Команда Read Memory: сторона устройства**



**2.5 Команда Go** Команда Go используется для выполнения загруженного кода или любого другого кода путем ветвления по адресу, указанному приложением.   
 Когда загрузчик получает команду Go, он передает приложению байт ACK. Затем загрузчик ожидает 4x-байтовый адрес (1-й байт - MSB, 4-й байт - LSB) и байт контрольной суммы, после чего проверяет полученный адрес. Если адрес действителен и контрольная сумма верна, загрузчик передает байт ACK, в противном случае он передает байт NACK и прерывает команду.  
 Когда адрес действителен и контрольная сумма верна, загрузчик выполняет следующие операции:

1. Инициализирует регистры периферийных устройств, используемых загрузчиком, до значений сброса по умолчанию.

2. Инициализирует указатель основного стека пользовательского приложения.

3. Переходит к ячейке памяти, запрограммированной в полученном 'адресе + 4' (соответствует адресу обработчика сброса приложения). Например, если получен адрес 0x08000000, загрузчик переходит к ячейке памяти, запрограммированной по адресу 0x08000004.  
  
 Как правило, хост отправляет адрес для перехода на запрограммированное приложение.  
  
***Примечание***: Переход к приложению работает только в том случае, если пользовательское приложение правильно устанавливает таблицу векторов так, чтобы она указывала на адрес приложения.  
  
Хост отправляет байты на STM32 следующим образом:  
- Byte 1: 0x21

- Byte 2: 0xDE

Wait for ACK

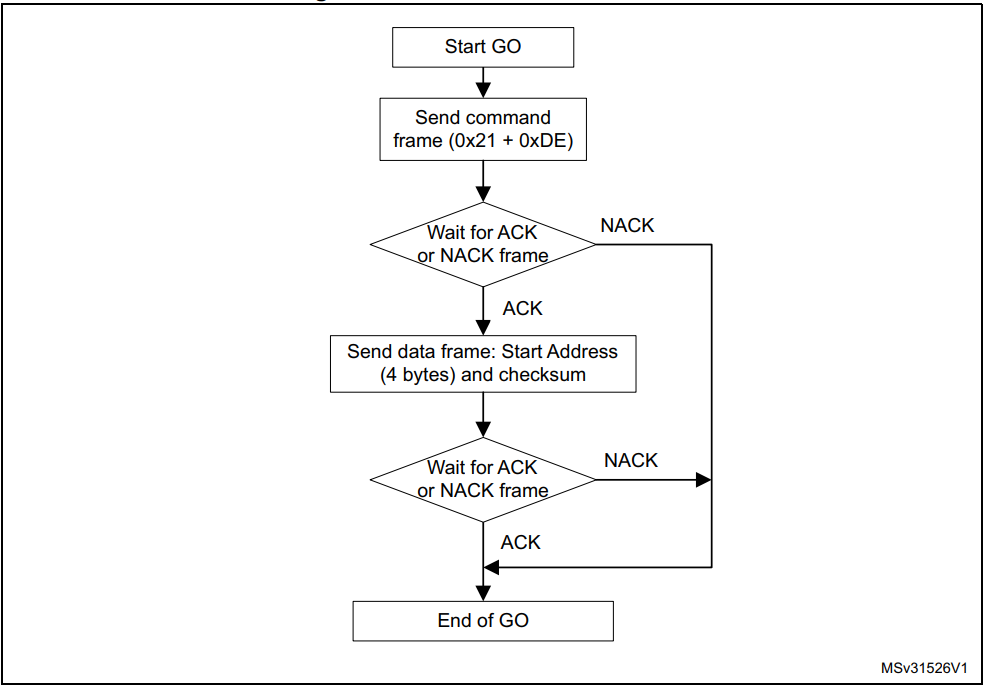
- Byte 3 to byte 6: start address

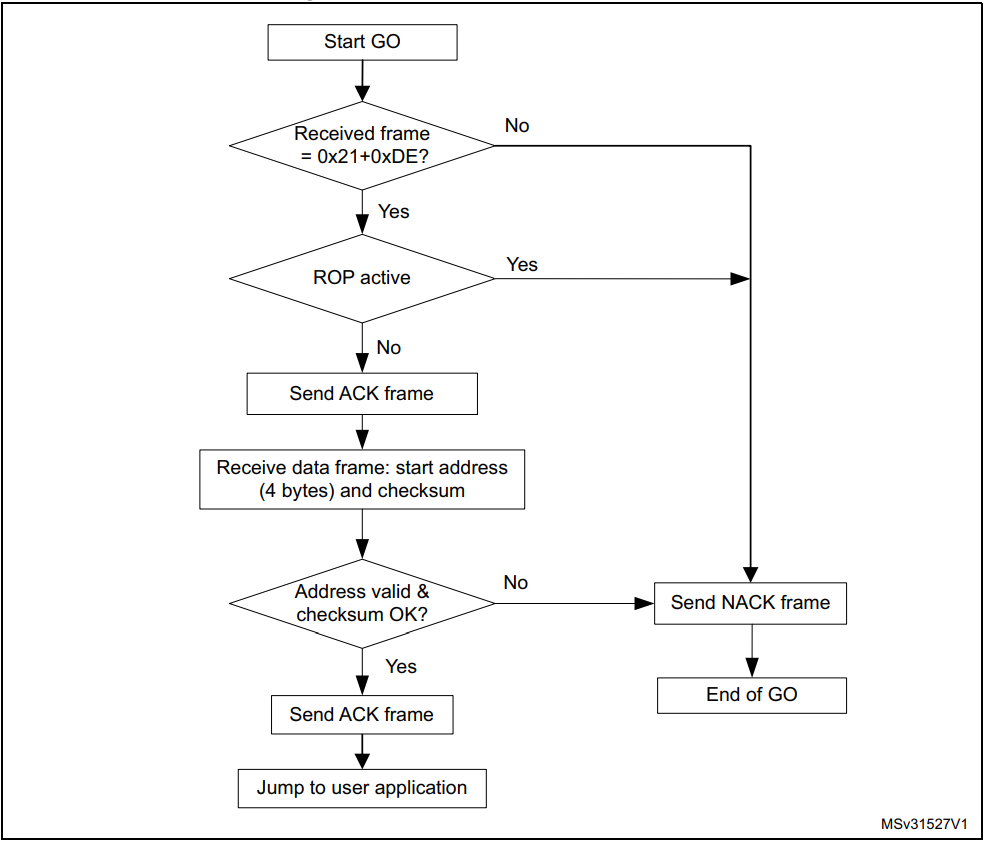
- Byte 3: MSB

- Byte 6: LSB

- Byte 7: checksum: XOR (byte 3, byte 4, byte 5, byte 6)

Wait for ACK

**Рисунок 10. Команда Go: сторона хоста**

**Рисунок 11. Команда Go: сторона устройства**

**2.6 Команда Write Memory** Команда Write Memory используется для записи данных в любой допустимый адрес памяти (см. Примечание: ниже) ОЗУ, флэш-памяти или в область байтов управления (the option byte area).  
 Когда загрузчик получает команду Write Memory, он передает приложению байт ACK. Затем загрузчик ожидает 4х-байтовый адрес (1-й байт - MSB, 4-й байт - LSB) и байт контрольной суммы, а затем проверяет полученный адрес.  
 Если полученный адрес валидный(доступный) и контрольная сумма верна, загрузчик передает байт ACK, в противном случае он передает байт NACK и прерывает команду.   
 Когда адрес валиден(доступен) и контрольная сумма верна, загрузчик:

1. получает байт N, который содержит количество байтов данных, которые необходимо получить

2. получает пользовательские данные (N + 1) байт и контрольную сумму (исключающее ИЛИ со всеми байтами, начиная с байта N)

3. заносит данные пользователя в память, начиная с полученного 4х-байтового адреса.

В конце команды, если операция записи прошла успешно, загрузчик передает байт ACK, в противном случае он передает приложению байт NACK и прерывает выполнение команды.  
 Если команда Write Memory производит запись в область байтов управления (the option byte area), все опции стираются перед записью новых значений. В конце команды загрузчик генерирует сброс системы, чтобы применилась записанная конфигурация.  
 Максимальная длина блока, записываемого в область байтов управления, зависит от продукта STM32, и адрес, полученный от хоста, должен быть начальным адресом области байтов управления. Дополнительные сведения о область байтов управления см. в справочном руководстве по продукту STM32.

***Примечание***: Максимальная длина блока для записи в ОЗУ или флэш-память составляет 256 байт. При записи в ОЗУ следите за тем, чтобы не перекрывать первое ОЗУ, используемое прошивкой загрузчика.

При выполнении операций записи в защищенные от записи сектора ошибка не возвращается.

Хост отправляет байты на STM32 следующим образом:  
- Byte 1: 0x31

- Byte 2: 0xCE

Wait for ACK

- Byte 3 to byte 6: Start address (Byte 3: MSB, Byte 6: LSB)

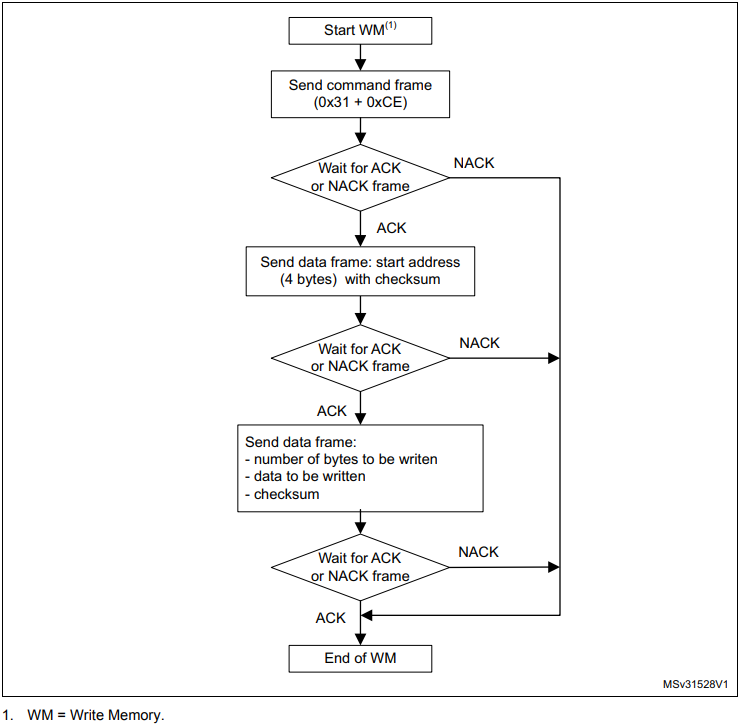
- Byte 7: Checksum: XOR (byte 3, byte 4, byte 5, byte 6)  
Wait for ACK

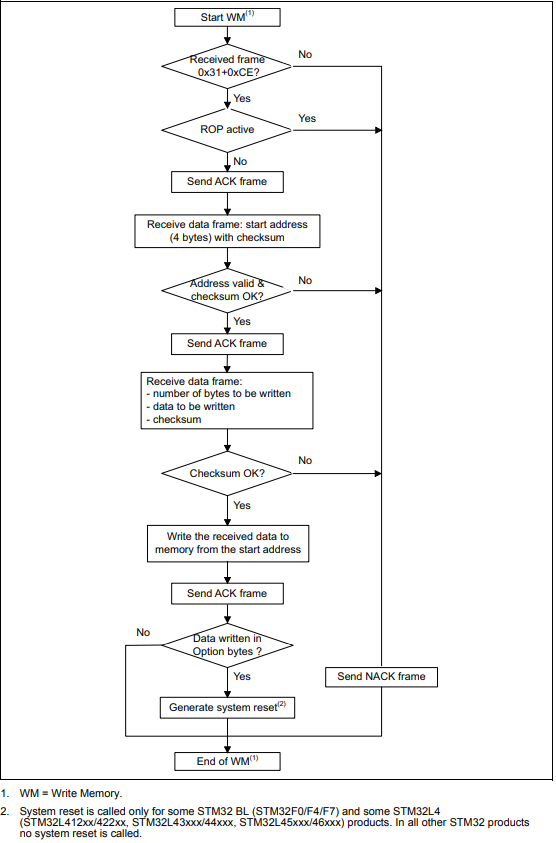
- Byte 8: Number of bytes to be received (0 < N <= 255)

- N+1 data bytes: (max 256 bytes)

- Checksum byte: XOR (N, N+1 data bytes)

Wait for ACK

**Рисунок 12. Команда Write Memory: сторона хоста**  


**Рисунок 13. Команда Write Memory: сторона устройства**

**2.7 Команда Erase Memory** Команда Erase Memory позволяет хосту стирать страницы или сектора флэш-памяти, используя режим двухбайтовой адресации. Когда загрузчик получает команду Erase Memory, он передает хосту байт ACK. Затем загрузчик получает два байта (количество страниц или секторов, которые необходимо стереть), коды страниц или секторов флэш-памяти (каждый закодирован двумя байтами, сначала старший байт) и байт контрольной суммы (исключающее ИЛИ отправленных байтов). Если контрольная сумма верна, загрузчик стирает память и отправляет хосту байт ACK, в противном случае он отправляет байт NACK хосту, и команда прерывается.

**Спецификации команды стирания памяти** Загрузчик получает одно полуслово (два байта), содержащее количество страниц или секторов, которые необходимо стереть, уменьшенное на 1. Если получено 0xFFFY (где Y от 0 до F), выполняется специальное стирание (0xFFFF для глобальной очистки, 0xFFFE и 0xFFFD, соответственно, для массового стирания банка 1 и банка 2).

Загрузчик получает:

- в случае специального стирания один байт: контрольная сумма предыдущих байтов (например, 0x00 для 0xFFFF)

- в случае стирания N страниц или секторов загрузчик получает 2 x N байтов, каждое полуслово которых содержит номер страницы или сектора, закодированный двумя байтами, причем старший бит идет первым. Затем все предыдущие байтовые контрольные суммы принимаются в одном байте.

***Примечание***: Некоторые продукты не поддерживают функцию массового стирания, в этом случае используйте команду стирания для стирания всех страниц или секторов.

Максимальное количество страниц или секторов связано с продуктом и должно соблюдаться. Максимальное количество страниц или секторов, которые можно стереть одной командой, равно 512.

Коды от 0xFFFC до 0xFFF0 зарезервированы.

Ошибка не возвращается при выполнении операций стирания в защищенных от записи секторах.

Хост отправляет байты на STM32 следующим образом:  
- Byte 1: 0x44

- Byte 2: 0xBB

Wait for ACK

For Special erase:

- Bytes 3-4: Special erase (0xFFFx)

- Bytes 5: Checksum of Bytes 3-4

Wait for ACK

For Page erase:

- Bytes 3-4: Number of pages or sectors to be erased - 1

- Bytes 5: Checksum of Bytes 3-4

Wait for ACK

- (2 x N) bytes (page numbers or sectors coded on two bytes MSB first) and then the checksum for these bytes.

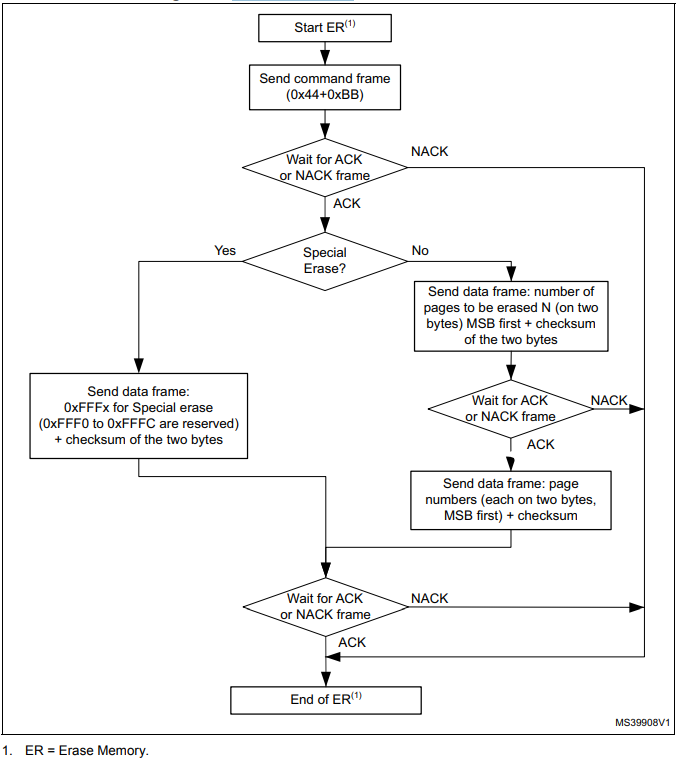
Wait for ACK

**Example of I2C frame**:

– erase page 1:   
0x44 0xBB Wait\_ACK 0x00 0x00 0x00 Wait\_ACK 0x00 0x01 0x01 Wait\_ACK

– erase page 1 and page 2:

0x44 0xBB Wait\_ACK 0x00 0x01 0x01 Wait\_ACK 0x00 0x01 0x00 0x02 0x03 Wait\_ACK

**Рисунок 14. Команда Erase Memory: сторона хоста**

***Примечание***: Некоторые продукты не поддерживают функцию специального стирания. Для этих продуктов эта команда возвращает NACK.

**Рисунок 15. Команда Erase Memory: сторона устройства**