User's Guide
SWPA230A–May 2017–Revised June 2018

CC3120 and CC3220 SimpleLink™ Wi-Fi® Embedded Programming

The CC3120 and CC3220 devices are part of the SimpleLink™ microcontroller (MCU) platform, which consists of Wi-Fi®, *Bluetooth*® low energy, Sub-1 GHz, and host MCUs, which all share a common, easy-to- use development environment with a single core software development kit (SDK) and rich tool set. A one-time integration of the SimpleLink platform lets you add any combination of devices of the portfolio into your design, allowing 100 percent code reuse when your design requirements change. For more information, visit www.ti.com/SimpleLink.

Contents

1 Introduction 2
2 Embedded Programming Schemes 3
3 Setup 3
4 Bootloader Protocol 4
4.1 Overview 4
4.2 General Message Format 4
4.3 Commands 5
4.4 Responses 9
5 Embedded Programming Procedure 11
5.1 Overview 11
5.2 High-Level Flow Diagram 12
5.3 Image Programming in Detail 13

List of Figures

1 CC3120 and CC3220 QFN Programming Setup 3
2 High-Level Programming Flow 12
3 Get Storage List 13
4 Get Version Info 13
5 Switch UART Lines to APPS MCU Command 14
6 Switch UART to APPS MCU Procedure 14
7 Get Storage Info 14
8 Raw Storage Erase to SRAM 15
9 Raw Storage Write to SRAM (Zoomed Out) 15
10 Raw Storage Write to SRAM (Zoomed In) 15
11 Get SFLASH Storage Info 16
12 Raw Storage Erase to SFLASH 16
13 Raw Storage Write to SFLASH (Zoomed Out) 17
14 Raw Storage Write to SFLASH (Zoomed In) 17
15 FS Programming (Zoomed Out) 18
16 FS Programming of Unencrypted Image (Zoom In) 18
17 FS Programming of Encrypted Image (Zoomed In) 18

List of Tables

1 General Message Format 4

Copyright © 2017–2018, Texas Instruments Incorporated

2 Bootloader Commands and Responses 4
3 Get Status 5
4 Get Storage List 5
5 Raw Storage Write 5
6 Get Version Info 6
7 Raw Storage Erase 6
8 Get Storage Info 6
9 Execute from RAM 7
10 Switch UART to APPS MCU 7
11 FS Programming 8
12 Ack 9
13 Nack 9
14 Last Status 9
15 Storage List 10
16 Storage Info 10
17 Version Info 10

Trademarks

SimpleLink is a trademark of Texas Instruments.
Stellaris is a registered trademark of Texas Instruments.
ARM, Cortex are registered trademarks of ARM Limited.
Wi-Fi is a registered trademark of Wi-Fi Alliance.
All other trademarks are the property of their respective owners.

1 Introduction

The SimpleLink™ CC3120 and CC3220 devices are Wi-Fi® and networking devices that provide a comprehensive networking solution for low-cost and low-power microcontrollers (MCU) using a thin driver and simple API set.

Each product with an embedded CC3120 or CC3220 device onboard must also have a serial flash device connected. The serial flash must be formatted, and at a minimum programmed with the Service Pack, which contains necessary software updates and additional features. For the CC3220, a binary image running on the internal MCU processor must also be programmed.

There are several options for serial flash programming, as follows:

- UniFlash – a PC-based utility offering image creation and programming. Content is programmed using the UART.
- Industrial flash programmer – flashes a complete image prepared with UniFlash directly to the serial flash. Can be applied when no SimpleLink device is attached to the serial flash. Content is programmed using the serial flash SPI lines.
- Over-the-air programming – the serial flash must be formatted in advance. Content is delivered through a network connection.

This application note describes in details additional options that leverage all the features UniFlash has to offer, but without the necessary connected PC. This option is referred to as *Embedded Programming*. To achieve embedded programming, bootloader protocol implemented over UART is described in detail.

The following sections describe the setup, bootloader protocol, and procedure of the embedded programming feature.

Copyright © 2017–2018, Texas Instruments Incorporated

2 Embedded Programming Schemes

Several schemes can leverage full image programming over the UART, as follows.

- Embedded programming in production line – there are setups on the production line that do not include the PC. Instead, programmable devices such as the MCU, DSP, or FPGA are used.
- Main external processor (other than the CC3220 MCU) – in many cases, CC3120 and CC3220 devices are just another component in the device enabling network communication. Such devices have main processors that usually control and schedule everything in the system. These cores must have the ability to upgrade and program CC3120 or CC3220 peripherals.

3 Setup

The UART interface must be connected between the CC3120 or CC3220 device and the main processor. Only two pins are required, UART TX and UART RX. Flow control is not required.

The common configuration that applies to all chipsets follows:

- Baud rate of 921600bps
- 8 bits
- No parity
- 1 stop bit

In addition, the nHib/nReset pin is required and it must be have the ability to be temporarily pulled to GND during a reset to make the device go into bootloader mode.

For the CC3220 device, an additional SOP1 pin must be pulled up during the device reset to make the device go into bootloader mode.



Figure 1. CC3120 and CC3220 QFN Programming Setup

Copyright © 2017–2018, Texas Instruments Incorporated

4 Bootloader Protocol

4.1 Overview

The CC3120 and CC3220 bootloader protocol is a simple command-response protocol. The protocol is based on the protocol used by the Stellaris® LM Flash Programmer. The commands are serially executed and there are no unsolicited events during the bootloader phase.

4.2 General Message Format

Table 1 provides the general message format.

Table 1. General Message Format

Length (Big Endian)	Checksum	Opcode	Data [optional]
2 bytes	1 byte	1 byte	n bytes

The length includes all fields except the checksum (including the Length field itself), so in general:

Length = Length(Length) + Length(Opcode) + Length(Data) = 3 + Length(Data) (1)

Checksum is a simple hexadecimal addition of the Opcode and Data fields. Checksum is then clipped to occupy the least significant byte only.

Table 2 lists all the commands and responses of the bootloader protocol.

Table 2. Bootloader Commands and Responses

Item	Command or Response	Link
Get Status	Command	Section 4.3.1
Get Storage List	Command	Section 4.3.2
Raw Storage Write	Command	Section 4.3.3
Get Version Information	Command	Section 4.3.4
Raw Storage Erase	Command	Section 4.3.5
Get Storage Info	Command	Section 4.3.6
Execute from RAM	Command	Section 4.3.7
Switch UART to APPS MCU	Command	Section 4.3.8
FS Programming	Command	Section 4.3.9
Ack	Response	Section 4.4.1
Nack	Response	Section 4.4.2
Last Status	Response	Section 4.4.3
Storage List	Response	Section 4.4.4
Storage Info	Response	Section 4.4.5
Version Info	Response	Section 4.4.6

Copyright © 2017–2018, Texas Instruments Incorporated

4.3 Commands

4.3.1 Get Status

Table 3 lists the Get Status command.

Table 3. Get Status

	Description
Brief	This command returns the status of the last command executed. Typically, this command must be invoked after every command sent to ensure that the previous command was successful or, if unsuccessful, to properly respond upon failure. The bootloader responds by sending a 1-byte packet containing the current status code.
Opcode	0x23
Direction	Host to target
Response	Ack + Last Status response
Format	[USHORT] Length (exclude checksum) [BYTE] Checksum (exclude length) [BYTE] Opcode
Comments	Host responds back with Ack (Ack + Last Status response).

4.3.2 Get Storage List

Table 4 lists the Get Storage List command.

Table 4. Get Storage List

	Description
Brief	This command is used to fetch the list of existing storages.
Opcode	0x27
Direction	Host to target
Response	Ack + Storage List response
Format	[USHORT] Length (exclude checksum) [BYTE] Checksum (exclude length) [BYTE] Opcode
Comments	—

4.3.3 Raw Storage Write

Table 5 lists the Raw Storage Write command.

Table 5. Raw Storage Write

	Description
Brief	This command triggers sending a chunk of raw storage data specified by StorageID, starting from a position specified by Offset and with a size specified by Length.
Opcode	0x2D
Direction	Host to target
Response	Ack
Format	[USHORT] Length (exclude checksum) [BYTE] Checksum (exclude length) [BYTE] Opcode [UINT32] StorageID [UINT32] Offset (starting offset relative to the start of storage, in bytes) [UINT32] Length (amount of bytes to write) [BYTE Stream] Data
Comments	The chunk is 4096 bytes and the Length must be smaller than (chunk_size-16). The SRAM storage ID is 0x0. The serial flash storage ID is 0x2.

Copyright © 2017–2018, Texas Instruments Incorporated

4.3.4 Get Version Info

Table 6 lists the Get Version Info command.

Table 6. Get Version Info

	Description
Brief	This command is used to fetch version information of all cores and chip types.
Opcode	0x2F
Direction	Host to target
Response	Ack + Version info response
Format	[USHORT] Length (exclude checksum) [BYTE] Checksum (exclude length) [BYTE] Opcode
Comments	—

4.3.5 Raw Storage Erase

Table 7 lists the Raw Storage Erase command.

Table 7. Raw Storage Erase

	Description
Brief	This command erases the specified blocks (making them writable) from storage specified by StorageID. NumOfBlocks specifies the amount of blocks to erase and Offset specifies the offset of the first blocks from the start of the device.
Opcode	0x30
Direction	Host to target
Response	Ack
Format	[USHORT] Length (exclude checksum) [BYTE] Checksum (exclude length) [BYTE] Opcode [UINT32] StorageID [UINT32] Offset (in blocks, relative to start of device) [UINT32] NumOfBlocks (number of blocks to erase)
Comments	The SRAM storage ID is 0x0. The serial flash storage ID is 0x2.

4.3.6 Get Storage Info

Table 8 lists the Get Storage Info command.

Table 8. Get Storage Info

	Description
Brief	This command is used to fetch storage information of a requested storage ID.
Opcode	0x31
Direction	Host to target
Response	Ack + Storage Info response
Format	[USHORT] Length (exclude checksum) [BYTE] Checksum (exclude length) [BYTE] Opcode [UINT32] StorageID
Comments	The SRAM storage ID is 0x0. The serial flash storage ID is 0x2. Host responds back with Ack on the (Ack + Storage Info) response.

Copyright © 2017–2018, Texas Instruments Incorporated

4.3.7 Execute from RAM

Table 9 lists the Execute from RAM command.

Table 9. Execute from RAM

	Description
Brief	This command is used to execute a preloaded program in the SRAM.
Opcode	0x32
Direction	Host to target
Response	Ack followed by another Ack
Format	[USHORT] Length (exclude checksum) [BYTE] Checksum (exclude length) [BYTE] Opcode
Comments	First Ack indicates a complete command and the second Ack indicates that initialization has completed.

4.3.8 Switch UART to APPS MCU

Table 10 lists the Switch UART to APPS MCU command.

Table 10. Switch UART to APPS MCU

	Description
Brief	This command instructs UART lines to get MUX from the CC3220 MCU to the CC3120 network processor.
Opcode	0x33
Direction	Host to target
Response	Ack
Format	[USHORT] Length (exclude checksum) [BYTE] Checksum (exclude length) [BYTE] Opcode [UINT32] Delay [in ticks]
Comments	Applies only to CC3220. One second is required, hence, the value 26666667 must be used.

Copyright © 2017–2018, Texas Instruments Incorporated

4.3.9 FS Programming

Table 11 lists the FS Programming command.

Table 11. FS Programming

	Description
Brief	This command is used for programming a single chunk of a complete image.
Opcode	0x34
Direction	Host to target
Response	Ack + 4 bytes status code
Format	[USHORT] Length (exclude checksum) [BYTE] Checksum (exclude length) [BYTE] Opcode [UINT16] key size [UINT16] chunk size [UINT32] flags [key buffer in bytes] key buffer [chunk buffer in bytes] chunk buffer
Comments	The status code indicates the accumulated number of bytes received. The status for the last chunk must be 0 to indicate successful programming, otherwise, a negative status is returned. Chunks must be serially activated as the NWP assumes orderliness. Chunk size must be the minimum between 4096 and the residue left from the complete image. Flags are for future use and must be 0. If an image is encrypted, a 16-byte key is required.

Copyright © 2017–2018, Texas Instruments Incorporated

4.4 Responses

4.4.1 Ack

Table 12 lists the Ack response.

Table 12. Ack

	Description
Brief	This message is sent to acknowledge a packet.
Opcode	0xCC
Direction	Target to host
Response	None
Format	[BYTE] 0 (zero) [BYTE] Opcode
Comments	—

4.4.2 Nack

Table 13 lists the Nack response.

Table 13. Nack

	Description
Brief	This message is sent to indicate an acknowledge error.
Opcode	0x33
Direction	Target to host
Response	None
Format	[BYTE] 0 (zero) [BYTE] Opcode
Comments	—

4.4.3 Last Status

Table 14 lists the Last Status response.

Table 14. Last Status

	Description
Brief	This message is sent to indicate last operation status.
Opcode	N/A
Direction	Target to host
Response	Ack from host
Format	[4 BYTES] Status code
Comments	The data includes 2 bytes for length, 1 byte of checksum, and 1 byte for status.

Copyright © 2017–2018, Texas Instruments Incorporated

4.4.4 Storage List

Table 15 lists the Storage List response.

Table 15. Storage List

	Description
Brief	This message is sent as a response to the Get Storage List command.
Opcode	N/A
Direction	Target to host
Response	None
Format	[BYTE] Storage list bitmap: • FLASH_DEV_BIT (0x02) • SFLASH_DEV_BIT (0x04) • SRAM_DEV_BIT (0x80)
Comments	—

4.4.5 Storage Info

Table 16 lists the Storage Information response.

Table 16. Storage Info

	Description
Brief	This message is sent as a response to the Get Storage Info command.
Opcode	N/A
Direction	Target to host
Response	Ack from Host
Format	[2 BYTES] block size [2 BYTES] number of blocks [4 BYTES] reserved
Comments	—

4.4.6 Version Info

Table 17 lists the Version Information response.

Table 17. Version Info

	Description
Brief	This message is sent as a response to the Get Version Info command.
Opcode	N/A
Direction	Target to host
Response	Ack
Format	[USHORT] Length (exclude checksum) [BYTE] Checksum (exclude length) [4 BYTES] Bootloader version (x.x.x.x) [4 BYTES] NWP Version [4 BYTES] MAC Version [4 BYTES] PHY Version [4 BYTES] Chip type [4 BYTES] Reserved [4 BYTES] Reserved
Comments	—

Copyright © 2017–2018, Texas Instruments Incorporated

5 Embedded Programming Procedure

5.1 Overview

The following sections describe the bootloader commands and responses, as well as the ordered steps during image programming.

Because content and timing are important during the procedure, all steps were captured for reference. Saleae is the logic used. This utility is free for use and can be downloaded from https://www.saleae.com/downloads.

The image must first be created using UniFlash. Detailed instructions on how to create an image are listed in the *UniFlash CC3120 and CC3220 SimpleLink™ Wi-Fi® and IoC Solution ImageCreator and Programming Tool User's Guide* under *Image Creation and Programming Using UniFlash*.

Copyright © 2017–2018, Texas Instruments Incorporated

5.2 High-Level Flow Diagram

Figure 2 shows the programming flow in high level. For a low-level description, see the following subsections.

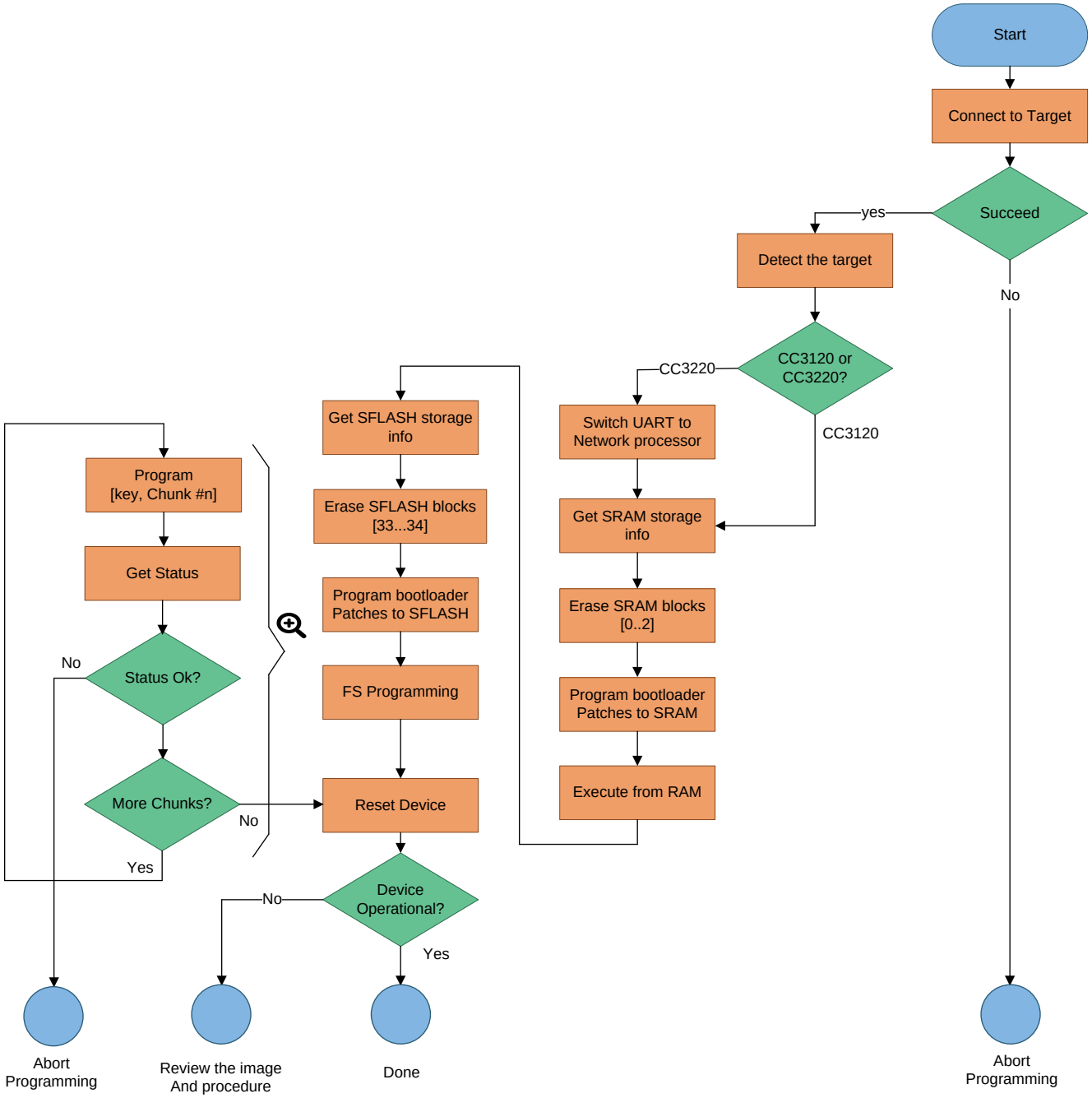


Figure 2. High-Level Programming Flow

Copyright © 2017–2018, Texas Instruments Incorporated

5.3 Image Programming in Detail

5.3.1 Step 1: Target Connection

Before commands can be sent to the device, it must go into bootloader mode. The procedure for connecting to the target in bootloader phase follows:

1. Flush the UART RX line (CC3120 or CC3220 UART TX line).
2. Send a break signal (sending continuous spacing values, no start or stop bits) on the CC3120 or CC3220 UART RX line. The CC3120 or CC3220 device must sense this break signal during power up.
3. Power on the device (or reset it if it is already up and running).
4. The CC3120 or CC3220 device sends an acknowledgment indication. The acknowledgment indication is described in Ack.
5. On receiving the acknowledgment indication from the CC3120 or CC3220 device, the main processor stops sending the break signal and flushes the UART lines.

NOTE: At this point, the CC3120 or CC3220 device is ready to receive any command. The main processor has 5 seconds to send any command. Failing to do so before the time-out expires results in the CC3120 or CC3220 device initializing normally (aborting bootloader mode).

6. The main processor sends the Get Storage List command.
7. The CC3120 or CC3220 device responds with an Ack followed by a 1-byte storage list bitmap.

Figure 3 shows the Get Storage List command.



Figure 3. Get Storage List

5.3.2 Step 2: Target Detection

It is essential to determine whether the connected device is a CC3120 or CC3220. It is important because additional steps are required for the CC3220. The provided information indicates whether the device is a CC3220 (nonsecured), CC3220S, or CC3220SF (flash device).

The procedure for detecting the target follows:

1. The main processor sends the Get Version Info command.
2. The CC3120 or CC3220 device responds with an Ack, followed by the Version Info response. The first byte of the Chip type field must be tested.

```
If (chip type & 0x10)    //the    device is CC3220 flavor//
If (chip type == 0x10)    //the    device is CC3220 (nonsecured)//
If (chip type == 0x18)    //the    device is CC3220S (secured ROM)//
If (chip type == 0x19)    //the    device is CC3220SF (secured flash)//
else    //the    device is CC3220 flavor//
```

3. The main processor responds with an Ack.

Figure 4 shows the Get Version Info command and the Version Info response.



Figure 4. Get Version Info

Copyright © 2017–2018, Texas Instruments Incorporated

5.3.3 Step 3: MUX UART to the Network Processor

If the device is a CC3220, it is required to MUX the UART lines from the internal application processor (ARM® Cortex®-M4) to the network processor.

The procedure for switching UART lines in CC3220 case follows:

1. The main processor sends the Switch UART to APPS MCU command. The user must provide the delay until the network processor is ready. One second is sufficient and should be used. During this time, the UART lines are switched, and the network processor is rebooted into bootloader mode. The reset is internal so the user does not need to externally apply it.
2. The CC3220 device responds with an Ack. This is the command response.
3. The main processor should send a break signal (sending continuous Spacing values [no Start or Stop bits]) on the CC3220 UART RX line. The network processor must sense this break signal during power up.
4. Because there are cases in which the break signal is missed or the Ack packet is being missed, TI recommends sending the break signal up to four times. Follow the following pseudocode for details.
5. The network processor responds with an Ack. This response is an indication that the network processor sensed the break signal and entered bootloader mode.
6. The main processor deasserts the break signal.

```
For i=1..4
set BREAK sleep 100mSec read ACK
unset BREAK
If successful break
Else continue
```

Figure 5 shows the Switch UART to APPS MCU command and the Ack command response.

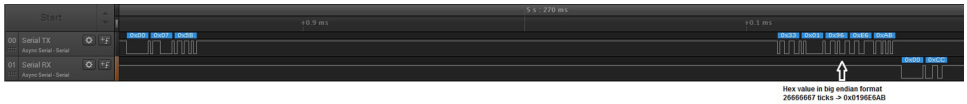


Figure 5. Switch UART Lines to APPS MCU Command

Figure 6 shows the BREAK assertion and deassertion followed by an Ack.



Figure 6. Switch UART to APPS MCU Procedure

5.3.4 Step 4: Get SRAM Storage Info

To introduce some fixes to the ROM bootloader, it is necessary to download patches to the SRAM. The procedure for getting information of SRAM storage follows:

1. The main processor sends the Get Storage Info command. The user must provide the storage ID for the SRAM.
2. The CC3120 or CC3220 device responds with an Ack followed by the Storage Info response. The response includes the block size and number of blocks for the SRAM.
3. The main processor responds with an Ack.

Figure 7 shows the Get Storage Info command followed by an Ack and Storage Info Response.



Figure 7. Get Storage Info

Copyright © 2017–2018, Texas Instruments Incorporated

5.3.5 Step 5: Raw Storage Erase – SRAM

Before downloading the patches to SRAM, it is essential to erase the relevant memory location in the SRAM. The procedure for erasing the SRAM follows:

1. The main processor sends the Raw Storage Erase command. The user must provide the storage ID for the SRAM, offset in blocks, and number of blocks to erase. In this case, erase three blocks starting from offset 0.
2. The CC3120 or CC3220 device responds with an Ack.
3. The main processor responds with an Ack.
4. The main processor sends the Get Status command.
5. The CC3120 or CC3220 device responds with an Ack followed by the Last Status response. Only the fourth byte should be inspected, 0x40 means success whereas other values indicate error.
6. The main processor sends an Ack response.

Figure 8 shows the Raw Storage Erase command.



Figure 8. Raw Storage Erase to SRAM

5.3.6 Step 6: Raw Storage Write – SRAM

Programming the patches to the SRAM is applied in chunks. The chunk size is 4096 bytes but the maximal size that can be programmed is 4080 bytes. Thus, the buffer is programmed in actual chunks of 4080 bytes. The procedure for programming the SRAM follows:

1. The main processor sends the Raw Storage Write command in chunks of 4080 bytes. The user should provide the storage ID for SRAM, offset in bytes and number of bytes to program. The offset in SRAM is 0 (beginning of SRAM).
2. The CC3120 or CC3220 device responds with an Ack.
3. The main processor sends the Get Status command.
4. The CC3120 or CC3220 device responds with an Ack followed by the Last Status response. Only the fourth byte should be inspected; 0x40 means success whereas other values indicate error.
5. The main processor sends an Ack response.
6. Steps 1-5 repeat if the data is larger than chunk size.

Figure 9 shows the SRAM programming procedure in a zoomed-out pane. As observed, the procedure is divided into three separate programming instances.

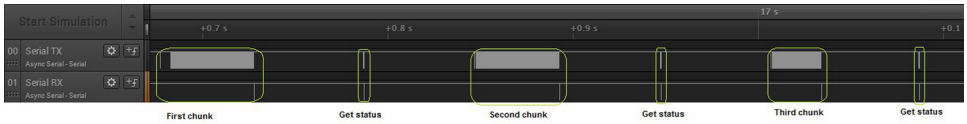


Figure 9. Raw Storage Write to SRAM (Zoomed Out)

Figure 10 shows the SRAM programming procedure in a zoomed-in pane to the second chunk. As observed, the chunk is 4080 bytes in offset 4080.



Figure 10. Raw Storage Write to SRAM (Zoomed In)

Copyright © 2017–2018, Texas Instruments Incorporated

5.3.7 Step 7: Execute from RAM

After the bootloader patches are programmed to the SRAM, it is required to execute the code. The ROM bootloader ROM is executed together with the programmed patches.

The procedure for executing the bootloader from the RAM follows:

1. The main processor sends the Execute from RAM command.
2. The CC3120 or CC3220 device responds with an Ack. This first Ack is to indicate a command is complete.
3. The CC3120 or CC3220 device sends a second Ack to indicate that initialization is completed.

5.3.8 Step 8: Get SFLASH Storage Info

The same ROM bootloader fixes that were downloaded to the SRAM must also be programmed to the SFLASH. The main purpose is to have all fixes during the return-to-factory-default feature. The procedure for getting information from SFLASH storage follows:

1. The main processor sends the Get Storage Info command. The user must provide the storage ID for the SFLASH.
2. The CC3120 or CC3220 device responds with an Ack followed by the Storage Info response. The response includes the block size and number of blocks for the SFLASH.
3. The main processor responds with an Ack.

Figure 11 shows the Get SFLASH Storage Info command followed by an Ack and Storage Info response.



Figure 11. Get SFLASH Storage Info

5.3.9 Step 9: Raw Storage Erase – SFLASH

Before downloading the patches to the SFLASH, it is essential to erase the relevant memory location in the SFLASH. The procedure for erasing the SFLASH follows:

1. The main processor sends the Raw Storage Erase command. The user must provide the storage ID for the SFLASH, offset in blocks, and number of blocks to erase. In this case, erase two blocks starting from offset 33.
2. The CC3120 or CC3220 device responds with an Ack.
3. The main processor responds with an Ack.
4. The main processor sends the Get Status command.
5. The CC3120 or CC3220 device responds with an Ack followed by the Last Status response. Only the fourth byte should be inspected; 0x40 means success whereas other values indicate error.
6. The main processor sends an Ack response.

Figure 12 shows the Raw Storage Erase command.



Figure 12. Raw Storage Erase to SFLASH

Copyright © 2017–2018, Texas Instruments Incorporated

5.3.10 Step 10: Raw Storage Write – SFLASH

Programming the patches to the SFLASH is applied in chunks. The chunk size is 4096 bytes but the maximal size that can be programmed is 4080 bytes. Thus, the buffer is programmed in actual chunks of 4080 bytes. The procedure for programming the SFLASH follows:

1. The main processor sends the Raw Storage Write command in chunks of 4080 bytes. The user must provide the storage ID for the SFLASH, offset in bytes, and number of bytes to program. The content must reside in the 8 bytes offset of block 33, which makes the total offset equal to 33 × 4096 + 8 = 135176 bytes.
2. The CC3120 or CC3220 device responds with an Ack.
3. The main processor sends the Get Status command.
4. The CC3120 or CC3220 device responds with an Ack followed by the Last Status response. Only the fourth byte should be inspected; 0x40 means success whereas other values indicate error.
5. The main processor sends an Ack response.
6. Steps 1 to 5 repeat if the data is larger than chunk size.

Figure 13 shows the SFLASH programming procedure in a zoomed-out pane. As observed, the procedure is divided into three separate programming instances.

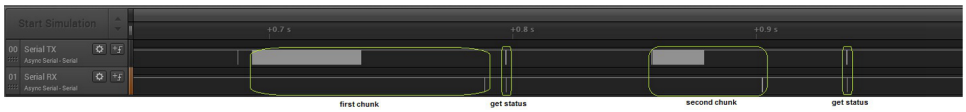


Figure 13. Raw Storage Write to SFLASH (Zoomed Out)

Figure 14 shows the SFLASH programming procedure in a zoomed-in pane to the first chunk. As observed, the chunk is 4080 bytes in offset (33 × 4096 + 8 = 135176 bytes).

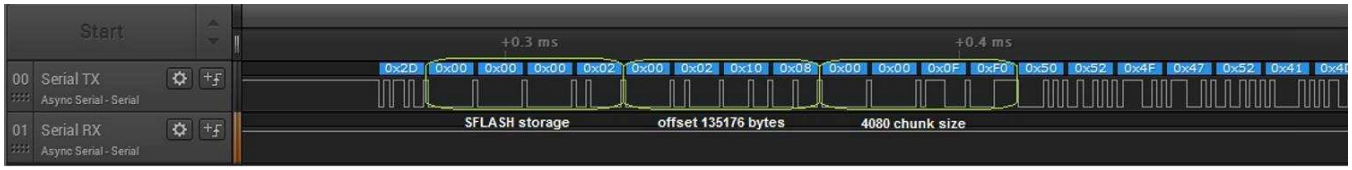


Figure 14. Raw Storage Write to SFLASH (Zoomed In)

Copyright © 2017–2018, Texas Instruments Incorporated

5.3.11 Step 11: FS Programming

The next step is the actual programming of the image to the SFLASH. Programming is applied in chunks of 4096 bytes. The image can be either unencrypted or encrypted with a 16-byte symmetric key.

- If the image is unencrypted, the key size is irrelevant and uses a length of 0.
- If the image is encrypted, the key size must be 16 bytes. The key buffer precedes the data chunk.

In both cases, flags are for future use and must be 0. The procedure for programming the SFLASH follows:

1. The main processor sends the FS Programming command in chunks of 4096 bytes. The user must provide the following elements in order:
    1. Key size (in bytes for an encrypted image) must be 16, otherwise 0.
    2. Chunk size (in bytes) must be 4096, except for the last chunk, which may be smaller according to the residue from the total size.
    3. Flags must be 0.
    4. Key buffer
    5. Data buffer
2. The CC3120 or CC3220 device responds with an Ack followed by a 4-byte response indicating the accumulated number of bytes received. The status for the last chunk must be 0 to indicate successful programming, otherwise, a negative status is returned.
3. Steps 1 and 2 repeat until the entire image is programmed.
4. The image gets extracted and the file system is created.

Figure 15 shows the programming procedure in a zoomed-out pane. The last status code is 0 to indicate a successful programming and is delayed in time to reflect the period of image extraction.



Figure 15. FS Programming (Zoomed Out)

Figure 16 shows the programming procedure in a zoomed-in pane for an unencrypted image.

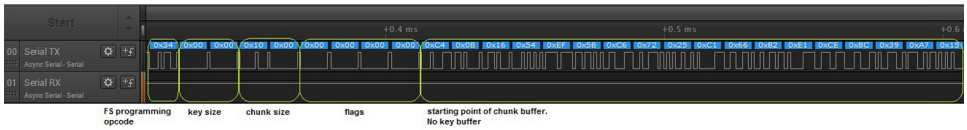


Figure 16. FS Programming of Unencrypted Image (Zoom In)

Figure 17 shows the programming procedure in a zoomed-in pane for an encrypted image.

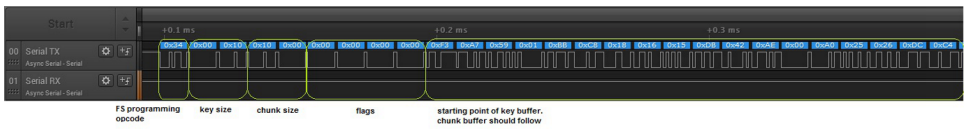


Figure 17. FS Programming of Encrypted Image (Zoomed In)

5.3.12 Step 12: Device Reset

The final step is the device reset.

Copyright © 2017–2018, Texas Instruments Incorporated

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (May 2017) to A Revision	Page
• Updated OpCode field for FS Programming.	8

Copyright © 2017–2018, Texas Instruments Incorporated

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2018, Texas Instruments Incorporated