Application Report
CC3x00 Production Line Guide

![TEXAS INSTRUMENTS]

Jesus Pintado

ABSTRACT

Texas Instruments provides a number of resources to assist manufacturers using CC3x00 devices produce products quickly and efficiently. This application report focuses on various methods of programming in a production environment as well as RF testing.

Table of Contents

List of Figures

List of Tables

Trademarks

SimpleLink™ is a trademark of Texas Instruments.

All trademarks are the property of their respective owners.

1 Production Line Introduction

Texas Instruments provides a number of resources to assist manufacturers using CC3100 devices produce products quickly and efficiently. From the beginning phases of designing products using these devices, reference design collateral and application notes are available to assist in schematic level design. This includes information about any considerations manufacturers should make in their designs to facilitate efficient manufacturing. During PCB layout development, layout guidelines are given as well. Software and hardware tools have been developed for programming and testing CC3100 devices in the production line. In addition, Over-The-Air programming functionality allows for products to have their software updated periodically even after they have been deployed.
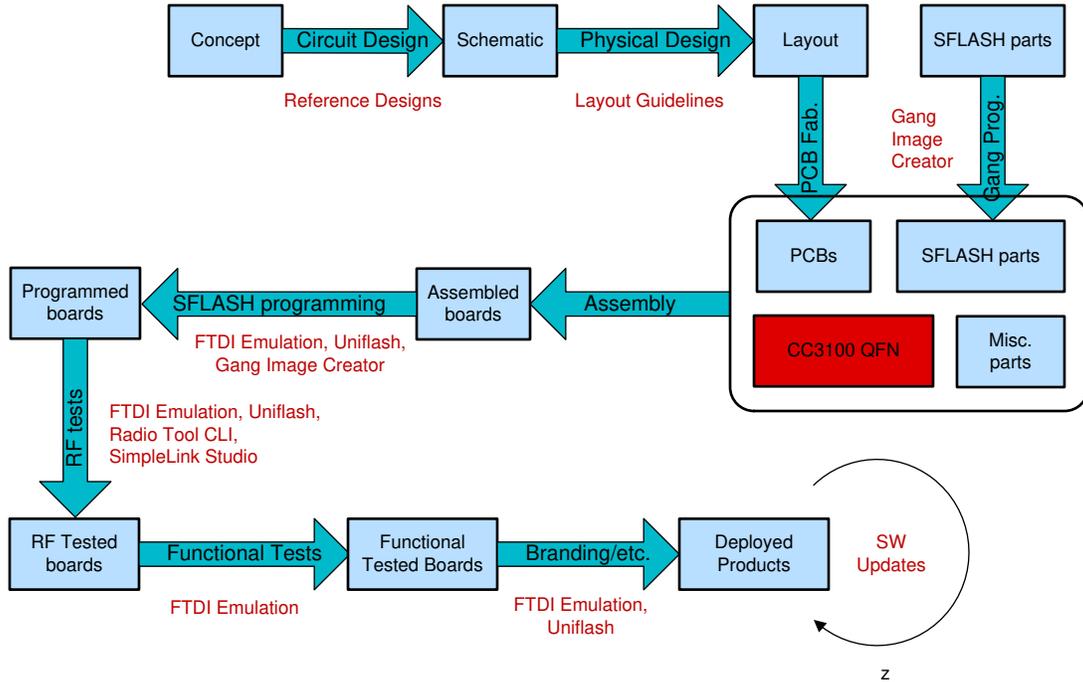

Figure 1-1. Production Flow

Copyright © 2021 Texas Instruments Incorporated

2 Programming the CC3100 QFN in the Production Line

Production with CC3100 devices requires files to be written to the attached serial flash device for proper operation. At the minimum, this includes the service pack that contains necessary software updates and additional features. Configuration files may also be written, which provide an initial configuration for the device upon startup. Security certificates and other content such as web pages, images, scripts, and so forth. can be included as well. Although most of this content is usually written during production, all content including the service pack can be continuously updated over the lifetime of the product. There are three basic methods of loading content onto the CC3100 serial flash:

- Uniflash – A PC based utility can be used for programming the serial flash
- Gang image – Use a special binary image that is written directly to the serial flash, that the CC3100 can use to create the desired contents of the serial flash.
- Host programming – Programming the serial flash through the host processor.
- Over-the-Air Programming (OTA) – Serial flash content can be downloaded through a network connection.

All CC3100 QFNs must have their attached serial flash devices formatted with the SimpleLink™ file system before files can be written. There are a few ways to accomplish this:

- Uniflash - Make a PC connection to the CC3100 UART, and use the Uniflash utility to send a format command to the device
- Host programming - Make a UART connection from the host to the CC3100, and use host code to send a format command to the device.
- Gang image - Flash a gang image to the serial flash. Have the serial flash vendor pre-program the serial flash parts with the gang image. The gang image can include a command that instructs the CC3100 to format the serial flash upon startup.

After the serial flash has been formatted there are a few options to load files onto the serial flash:

- Use Uniflash to load files via the CC3100 UART interface. The CC3100 reads the files sent over universal asynchronous receiver/transmitter (UART), and loads them onto the serial flash.
- If a gang image has been flashed, the CC3100 converts the files in the image and saves them into the file system. Therefore, an additional step for loading files is not necessary.

Because the host can load files on to the serial flash through the host interface as well, it is possible to use the host to programmatically add files using the SimpleLink file system API. After the service pack has been updated, Over-the-Air programming can be used to download content from the internet or from a local connection.

Given the above options for formatting and loading content to the serial flash there are a number of production line flows that are possible. Figure 2-1 shows some possible production line flows ranked roughly in order of speed.

Regardless of which programming methods used, it is required to bring out the CC3100 UART pins to provide a reliable backup flashing mechanism. If using headers to make the required connections to the programming interface, the manufacturer should consider using a single header assembly to accommodate the entire programming and test sequence as a cost saving measure. This would include the power supply, GND, and the lines necessary for flashing the serial flash. The lines necessary for programming the host processor should be included as well.

CC3100 QFN

Using pre-flashed gang image on SF	Programming host, then using host to format and program SF	Flashing host, Formatting SF Using host then using OTA to program SD	Flashing SF with, Uniflash, programming host with OTA program, then using OTA to program SF	Using Uniflash to format and program SF, then programming host

Pre assmbly

Flash gang image

Start

Flash host

Done

Start

Flash host

↓

Format and load files with host

Done

Start

Flash host

↓

Format with host, download SP with host

↓

OTA

Done

Start

Format with host, download SP with host

↓

Flash host

↓

OTA

Done

Start

Format and program SF With Uniflash
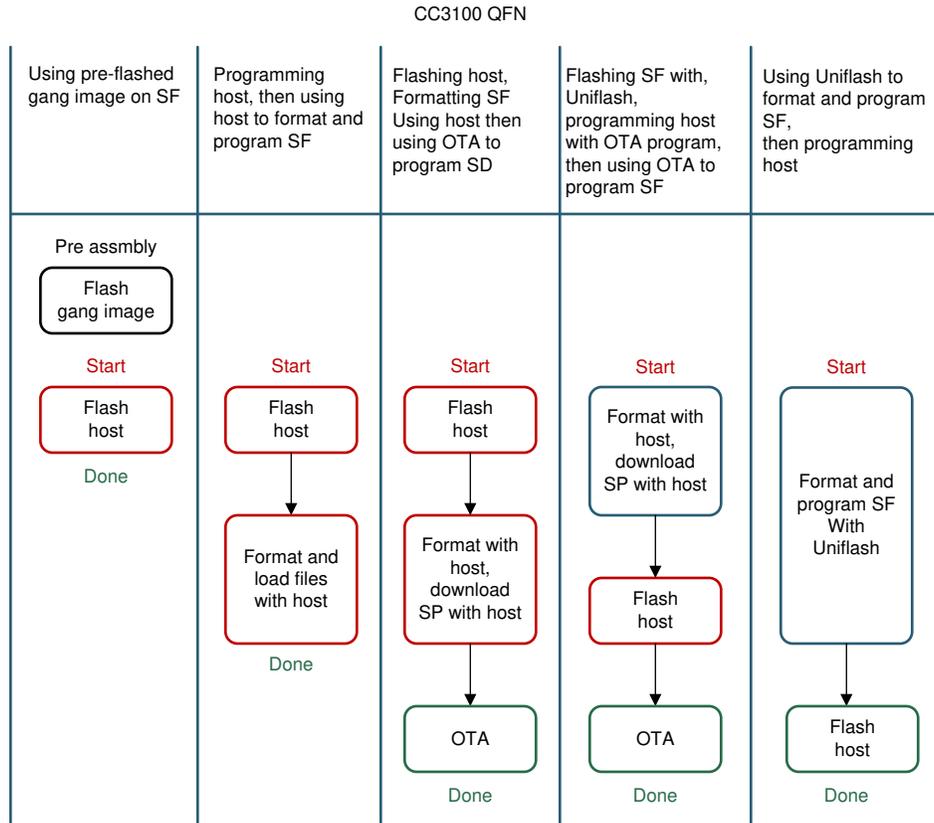
↓

Flash host

Done

Figure 2-1. CC3100 Programming Options

3 Programming the CC3100 Module in the Production Line

Production with CC3100 modules requires files to be written to the internal serial flash for proper operation. At the minimum, this includes the service pack that contains necessary software updates and additional features. Configuration files may also be written, which provide an initial configuration for the device upon startup. Security certificates and other content such as web pages, images, scripts, and so forth, can be included as well. Although most of this content is usually written during production, all content including the service pack can be continuously updated over the lifetime of the product. There are three basic methods of loading content onto the CC3100 module serial flash:

*   Uniflash - A PC based utility can be used for programming the serial flash
*   Host programming - Programming the serial flash through the host processor.
*   Over-the-Air Programming (OTA) - Serial flash content can be downloaded through a network connection.

All CC3100 Modules must have their internal serial flash devices formatted with the SimpleLink file system before files can be written. If the serial flash needs to be reformatted for any reason, this can be accomplished by the following:

*   Uniflash - Make a PC connection to the CC3100 module UART, and use the Uniflash utility to send a format command to the device
*   Host programming - Make a UART connection from the host to the CC3100 module, and use host code to send a format command to the device.

After the serial flash has been formatted, there is the option to use Uniflash to load files on to the serial flash via the CC3100 UART interface. In this case, the CC3100 module reads the files sent over UART, and loads them onto the serial flash. Because the host can load files on to the serial flash through the host interface as well, there are additional options to load files to the serial flash after the host has been programmed:

*   Use the host code to programmatically add files using the SimpleLink file system API.
*   Over-the-Air programming can be incorporated into the host code. This will enable the downloading of content from the internet or from a local connection.

Given the above options for loading content to the serial flash there are a number of production line flows that are possible. Figure 3-1 shows some possible production line flows ranked roughly in order of speed.
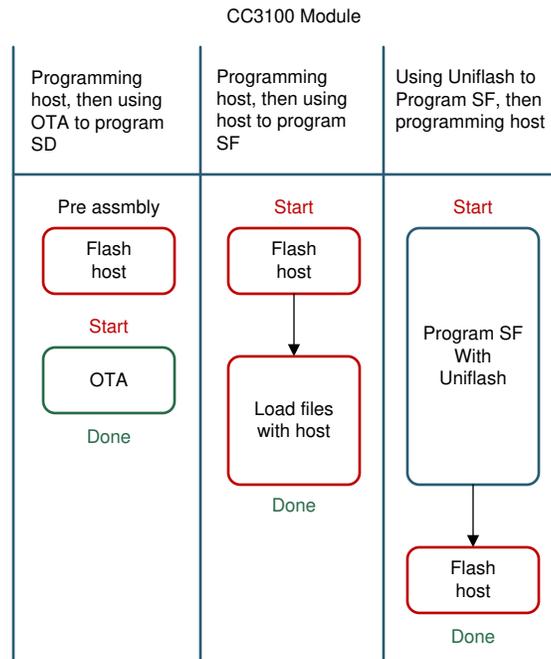

Figure 3-1. CC3100MOD Programming Options

Regardless of which programming methods used, it is required to bring out the CC3100 module UART pins to provide a reliable backup flashing mechanism. If using headers to make the required connections to the programming interface, the manufacturer should consider using a single header assembly to accommodate the entire programming and test sequence as a cost saving measure. This would include the power supply, GND, and the lines necessary for flashing the serial flash. The lines necessary for programming the host processor should be included as well.

4 Programming the CC3200 QFN in the Production Line

Production with CC3200 devices requires files to be written to the attached serial flash device for proper operation. At the minimum, this includes the service pack that contains necessary software updates and additional features. The host program that runs on the internal Cortex M4 is stored on the serial flash also. Configuration files may also be written, which provide an initial configuration for the device upon start up. Security certificates and other content such as web pages, images, scripts, and so forth, can be included as well. Although most of this content is usually written during production, all content including the host program and the service pack can be continuously updated over the lifetime of the product. There are three basic methods of loading content onto the CC3200 serial flash:

- Uniflash - A PC based utility can be used for programming the serial flash.
- Gang image – Use a special binary image that is written directly to the serial flash, that the CC3200 can use to create the desired contents of the serial flash.
- Over-the-Air Programming (OTA) - Serial flash content can be downloaded through a network connection.

All CC3200 QFNs must have their attached serial flash devices formatted with the SimpleLink file system before files can be written. There are a number of ways to accomplish this:

- All CC3200 QFNs must have their attached serial flash devices formatted with the SimpleLink file system before files can be written. There are a number of ways to accomplish this:
- Gang image - Flash a gang image to the serial flash. Have the serial flash vendor pre-program the serial flash parts with the gang image. The gang image can include a command that instructs the CC3200 to format the serial flash upon the next boot of the device.

After the serial flash has been formatted there are a couple of options to load files onto the serial flash:

- Use Uniflash to load files via the CC3200 UART interface. The CC3200 reads the files sent over UART, and loads them onto the serial flash.
- If a gang image has been flashed, the CC3200 converts the files in the image and saves them into the file system. Therefore, an additional step for loading files is not necessary.

After the service pack has been updated, Over-the-Air programming can be used to download content from the internet or from a local connection. If the CC3200 is flashed with a program that can do OTA programming, the entire serial flash contents can be reprogrammed using OTA programming. This will enable the downloading of content from the internet or from a local connection. This may be advantageous if using a small OTA program to load the final contents onto the serial flash.

Given the available options for formatting and loading content to the serial flash there are a couple of production line flows that are possible. Ranked roughly in order of speed are some possible production line flows shown in Figure 4-1.

CC3200 QFN

Using pre-flashed gang image on SF	Using Uniflash to format and program SF with OTA program, then using OTA to program SF	Using Uniflash to format and program SF

Flash gang image

Start

Use Uniflash to add SN

Done

Start

Format, Download SP, flash OTA code And SN with Uniflash

OTA

Done

Start

Format and program SF With Uniflash

Done

Figure 4-1. CC3200 Programming Options

It is required to bring out the UART pins to provide a reliable backup flashing mechanism. If using headers to make the required connections to the programming interface, the manufacturer should consider using a single header assembly to accommodate the entire programming and test sequence as a cost saving measure. This would include the power supply, GND, and the lines necessary for flashing the serial flash.

5 Programming the CC3200 Module in the Production Line

Production with CC3200 modules requires files to be written to the internal serial flash for proper operation. At the minimum, this includes the service pack that contains necessary software updates and additional features. The host program that runs on the internal Cortex M4 is stored on the serial flash also. Configuration files may also be written, which provide an initial configuration for the device upon startup. Security certificates and other content such as web pages, images, scripts, and so forth. can be included as well. Although most of this content is usually written during production, all content including the host program and the service pack can be continuously updated over the lifetime of the product. There are two basic methods of loading content onto the CC3200 module serial flash:

• Uniflash - A PC based utility can be used for programming the serial flash
• Over-the-Air Programming (OTA) - Serial flash content can be downloaded through a network connection.

All CC3200 modules must have their attached serial flash devices formatted with the SimpleLink file system before files can be written. All CC3200 modules should come pre-formatted. However, if the serial flash needs to be reformatted for any reason, Uniflash can be used to perform this operation. This requires a PC connection to the CC3200 UART. After the serial flash has been formatted, Uniflash can be used to load files via the CC3200 UART interface. The CC3200 reads the files sent over UART, and loads them onto the serial flash. If the CC3200 module is flashed with a program that can do OTA programming, the entire serial flash contents can be reprogrammed using OTA programming. This will enable the downloading of content from the internet or from a local connection. This may be advantageous if using a small OTA program to load the final contents onto the serial flash. Given the possible options for formatting and loading content to the serial flash there are a couple of production line flows that are possible.
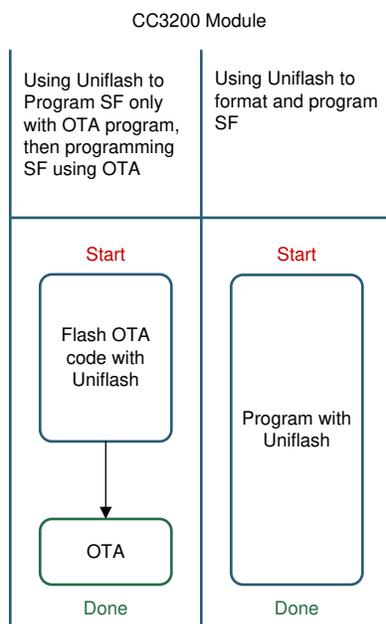
Figure 5-1. CC3200MOD Programming Options

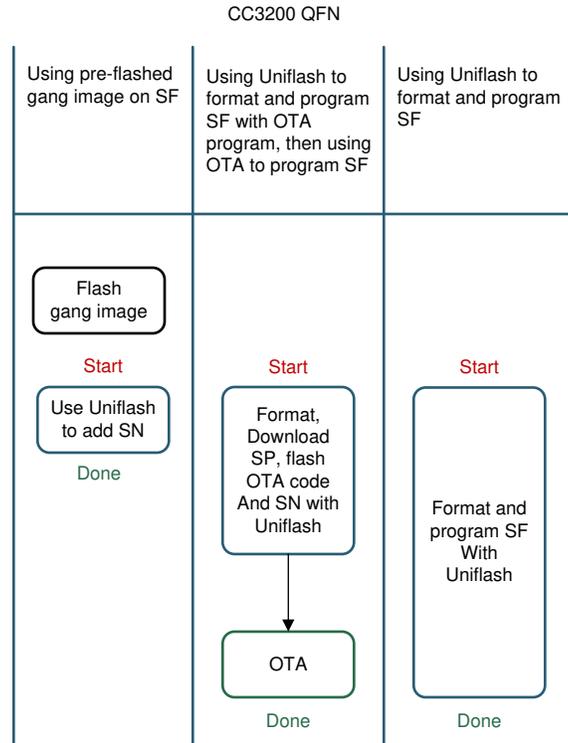Regardless of which programming methods used, it is required to bring out the UART pins to provide a reliable backup flashing mechanism. If using headers to make the required connections to the programming interface, the manufacturer should consider using a single header assembly to accommodate the entire programming and test sequence as a cost saving measure. This would include the power supply, GND, and the lines necessary for flashing the serial flash.

6 Programming Over UART CC3100

Formatting and/or writing files to the serial flash device through the CC3100 UART interface requires the use of the following CC3100 pins:

- 55 - UART1 TX
- 57 - UART1 RX
- 2 - nHIB or 32 - nRESET (either can be used)

The UART TX and RX pins are used for data transfer. RTS and CTS signals are not used. The nHIB and nRESET pins are used to reset the device. The UART data transfer occurs at 921600 bps. Because of this, other methods of writing files to the serial flash may take less time.

The UART configuration is as follows:

- Baud rate: 921600
- Data bits: 8 bits
- Flow control: None
- Parity: None
- Stop bits: 1
- Polarity: Positive

The CMOS logic level specifications for the UART can be found in the *Electrical Characteristics* section of the *CC3100 SimpleLink™ Wi-Fi® Network Processor, Internet-of-Things Solution for MCU Applications Data Sheet*.

7 Programming Over UART CC3100MOD

Formatting and/or writing files to the serial flash device through the CC3100 UART interface requires the use of the following CC3100 module pins:

- 46 - UART1 TX
- 47 - UART1 RX
- 4 - nHIB

The UART TX and RX pins are used for data transfer. RTS and CTS signals are not used. The nHIB and nRESET pins are used to reset the device. The UART data transfer occurs at 921600 bps. Because of this, other methods of writing files to the serial flash may takes less time.

The UART configuration is as follows:

- Baud rate: 921600
- Data bits: 8 bits
- Flow control: None
- Parity: None
- Stop bits: 1
- Polarity: Positive

The CMOS logic level specifications for the UART can be found in the *Electrical Characteristics* section of the *CC3100MOD SimpleLink™ Certified Wi-Fi® Network Processor Internet-of-Things Module Solution for MCU Applications Data Sheet*.

8 Programming Over UART CC3200

Formatting and/or writing files to the serial flash device through the CC3200 UART interface requires the use of the following CC3200 pins:

- 55 - UART1 TX
- 57 - UART1 RX
- 32 - nRESET
- 21 - SOP2

The UART TX and RX pins are used for data transfer. RTS and CTS signals are not used. The nRESET pin is used to reset the device. The UART data transfer occurs at 921600 bps. Because of this, other methods of writing files to the serial flash may takes less time.

The UART configuration is as follows:

- Baud rate: 921600
- Data bits: 8 bits
- Flow control: None
- Parity: None
- Stop bits: 1
- Polarity: Positive

The CMOS logic level specifications for the UART can be found in the *Electrical Characteristics* section of the *CC3100 SimpleLink™ Wi-Fi® Network Processor, Internet-of-Things Solution for MCU Applications Data Sheet*.

9 Programming Over UART CC3200 MOD

Formatting and/or writing files to the serial flash device through the CC3200 module UART interface requires the use of the following CC3200 module pins:

- 46 - UART1 TX
- 47 - UART1 RX
- 35 - nRESET
- 23 - SOP2

The UART TX and RX pins are used for data transfer. RTS and CTS signals are not used. The nRESET pin is used to reset the device. The UART data transfer occurs at 921600 bps.

The UART configuration is as follows:

- Baud rate: 921600
- Data bits: 8 bits
- Flow control: None
- Parity: None
- Stop bits: 1
- Polarity: Positive

The CMOS logic level specifications for the UART can be found in the *Electrical Characteristics* section of the *CC3200MOD SimpleLink™ Wi-Fi® and Internet-of-Things Module Solution, a Single-Chip Wireless MCU Data Sheet*.

10 Programming Using a Gang Image CC3100

Using this method can result in significant time savings in the case where a large amount of information must be written to the serial flash. The gang image is generally programmed on the serial flash device by the serial flash manufacturer, and therefore before it is assembled on the board. This will result in the fastest programming time out of any other method. To program the data to the serial flash via a gang programmer, it is necessary to first create a gang image that can be programmed on the target SFLASH devices. This creation process is generally performed once per product release in order to create the target image for the gang programmer. Upon power up, the CC3100 detects the presence of a gang image, and converts it to the target file system of the device. This conversion process is performed exclusively by the SimpleLink device and does not require any inputs from external interfaces. It does however extend the duration of the first power-up. This extra time should be considered when deciding what method to use in the initial programming of the serial flash. The amount of time required for the first boot after gang programming is primarily dependent on how long it takes to erase this image after the conversion is complete. This can be estimated by considering the size of the gang image, and the amount of time it takes for the serial flash to erase data.

Uniflash is the utility used for creating gang images, and it also has the ability to flash a gang image to the serial flash through UART. For instructions for creating and using gang images with Uniflash, see the *SimpleLink™ Wi-Fi® CC3100, CC3200 UniFlash User's Guide*.

The gang image may be programmed to the serial flash device after assembly on the board provided some considerations are taken:

- The serial flash SPI interface pins must be brought out for physical contact with the programmer (male headers, test pads)
- The SPI lines must not be driven by any other source while programming.
- The CC3100 will be held in hibernate during programming to prevent contention.

11 Programming Using a Gang Image CC3200

Using this method can result in significant time savings in the case where a large amount of information must be written to the serial flash. The gang image is generally programmed on the serial flash device by the serial flash manufacturer, and therefore before it is assembled on the board. This will result in the fastest programming time out of any other method. To program the data to the serial flash via a gang programmer, it is necessary to first create a gang image that can be programmed on the target SFLASH devices. This creation process is generally performed once per product release in order to create the target image for the gang programmer. Upon power up, the CC3200 detects the presence of a gang image, and converts it to the target file system of the device. This conversion process is performed exclusively by the SimpleLink device and does not require any inputs from external interfaces. It does however extend the duration of the first power-up. This extra time should be considered when deciding what method to use in the initial programming of the serial flash. The amount of time required for the first boot after gang programming is primarily dependent on how long it takes to erase this image after the conversion is complete. This can be estimated by considering the size of the gang image, and the amount of time it takes for the serial flash to erase data.

Uniflash is the utility used for creating gang images, and it also has the ability to flash a gang image to the serial flash through UART. The instructions for creating and using gang images with Uniflash, see the *SimpleLink™ Wi-Fi® CC3100, CC3200 UniFlash User's Guide*

The gang image may be programmed to the serial flash device after assembly on the board provided some considerations are taken:

- The serial flash SPI interface pins must be brought out for physical contact with the programmer (male headers, test pads)
- The SPI lines must not be driven by any other source while programming.
- The CC3200 will be held in reset during programming to prevent contention.

12 Programming Using Uniflash CC3100

12.1 Using the Uniflash CLI

The Uniflash utility is PC software that can perform the following operations on the CC3100:

- Format the serial flash
- Add or remove files on the serial flash
- Update the service pack

These operations are accomplished by communicating with the CC3100 via UART. Therefore, it is necessary to supply Uniflash with a COM port number for the USB to CC3100 UART connection. See the chapter Programming over UART for details on the UART connection. Uniflash contains a command line interface that can be used in batch files/scripts for the purposes of programming the CC3100 devices in the production line. With a single line command, Uniflash can format the serial flash, update the service pack, and add any number of files. For example:

- uniflashCLI.bat -config "C:\ti\uniflash_3.2\cc3xxx\sessions\uniflash_template\oob.ucf" -setOptions com=50 secure=false spPath="C:\servicepack.bin" -operations format servicePackUpdate program

When Uniflash begins, it will prompt the user to restart the device. This is so it can synchronize with the CC3100 and begin the flashing process. At this point the CC3100 must be reset by pulling down either the nHIB or nRESET lines. When the nHIB or nRESET line resumes a logic high state, the flashing procedure will begin. In the production line, this reset process can be realized by using a button or other mechanism that temporarily connects the line to GND. For complete information on how to use Uniflash, see the *SimpleLink™ Wi-Fi® CC3100, CC3200 UniFlash User's Guide*.

12.2 UART Hardware Connection Using the FTDI Emulation Board

The CC31XXEMUBOOST can provide the required USB to UART interface for formatting and programming the serial flash via UART. The PC drivers for this board are included in the CC3100 SDK, and they should be installed during installation of the SDK. The CC31XXEMUBOOST is connected via USB from socket J6 to the PC. The jumpers on the CC31XXEMUBOOST should be connected as shown below.
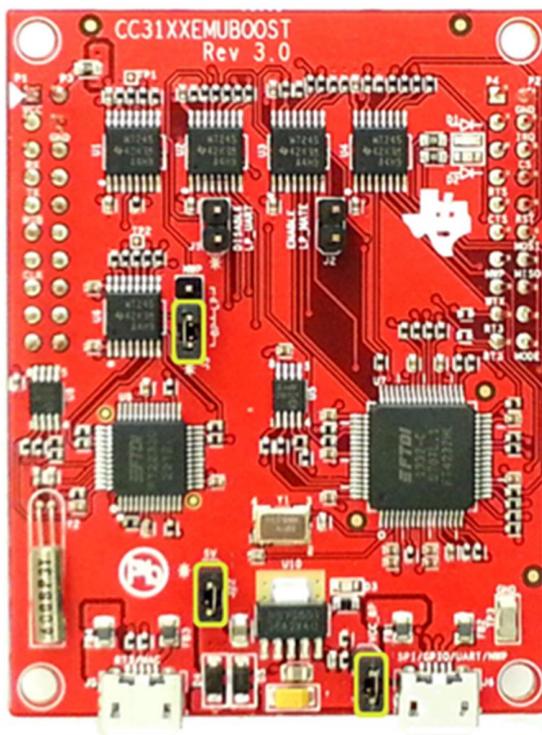

Figure 12-1. CC31XXEMU Jumper Configuration

The CC31XXEMUBOOST uses a logic level of 3.3 V by default, but there are level shifters on the CC31XXEMUBOOST and it can be powered in-dependently with a different IO voltage. This can be accomplished by removing the jumper from J4, and applying the external power to pin 1 of jumper J4.

On the product being programmed, the relevant CC3100 pins must be brought out for physical contact with the programmer (male headers, test pads), and must be driven by no other source while programming. The nHIB or nRESET pin must be able to be temporarily pulled to GND during a reset, which occurs at the beginning of Uniflash flashing procedure. The pulling down of this line can be enabled by the addition of a button or something similar to the test jig.
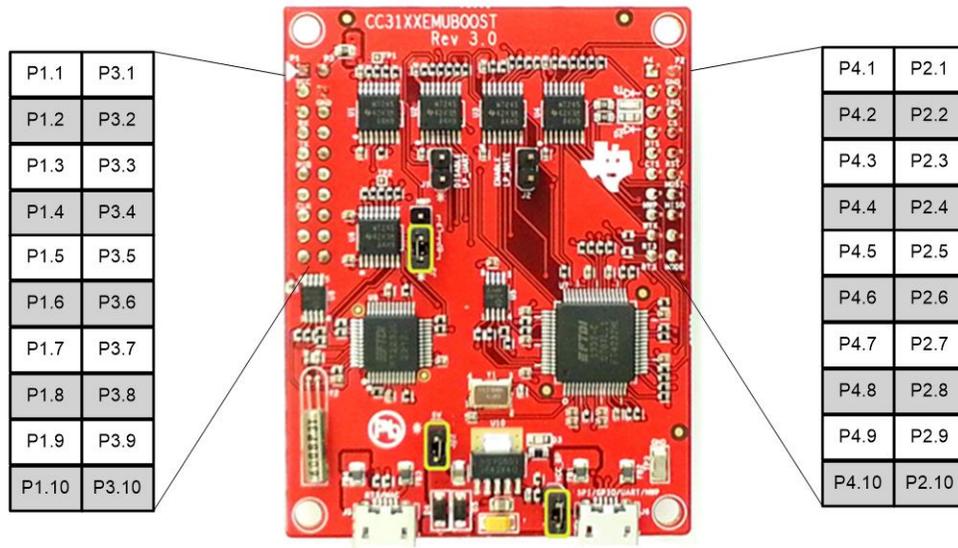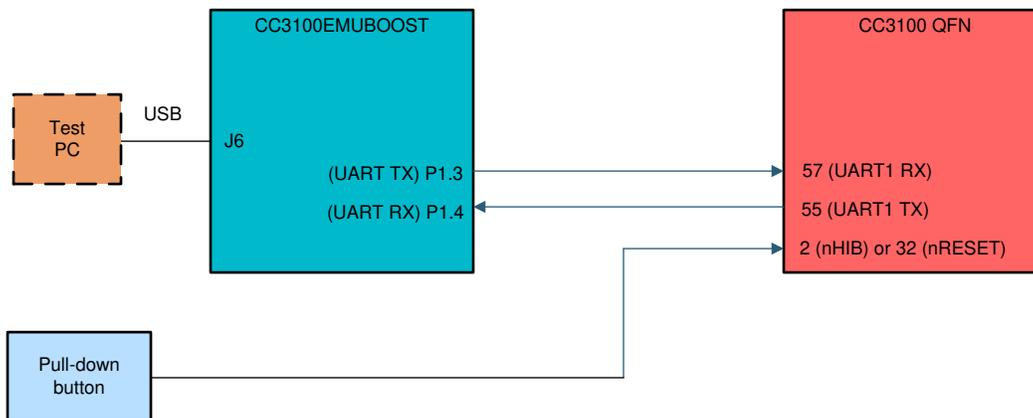

Figure 12-2. CC31XXEMU Pins

Figure 12-3. CC3100 UART Setup

13 Programming Using Uniflash CC3100 Mod

13.1 Using the Uniflash CLI

The Uniflash utility is PC software that can perform the following operations on the CC3100 module:

- Format the serial flash
- Add or remove files on the serial flash
- Update the service pack

These operations are accomplished by communicating with the CC3100 module via UART. Therefore, it is necessary to supply Uniflash with a COM port number for the USB to CC3100 module UART connection. See the chapter Programming over UART for details on the UART connection. Uniflash contains a command line interface that can be used in batch files/scripts for the purposes of programming the CC3100 module devices in the production line. With a single line command, Uniflash can format the serial flash, update the service pack, and add any number of files. For example:

- uniflashCLI.bat -config "C:\ti\uniflash_3.2\cc3xxx\sessions\uniflash_template\oob.ucf" -setOptions com=50 secure=false spPath="C:\servicepack.bin" -operations format servicePackUpdate program

When Uniflash begins, it will prompt the user to restart the device. This is so it can synchronize with the CC3100 module and begin the flashing process. At this point the CC3100 module must be reset by pulling down the nHIB line. When the nHIB line resumes a logic high state, the flashing procedure will begin. In the production line, this reset process can be realized by using a button or other mechanism that temporarily connects the line to GND. For complete information on how to use Uniflash, see the *SimpleLink™ Wi-Fi® CC3100, CC3200 UniFlash User's Guide*.

13.2 UART Hardware Connection Using the FTDI Emulation Board

The CC31XXEMUBOOST can provide the required USB to UART interface for programming the serial flash via UART. The PC drivers for this board are included in the CC3100 SDK, and they should be installed during installation of the SDK. The CC31XXEMUBOOST is connected via USB from socket J6 to the PC. The jumpers on the CC31XXEMUBOOST should be connected as shown below.
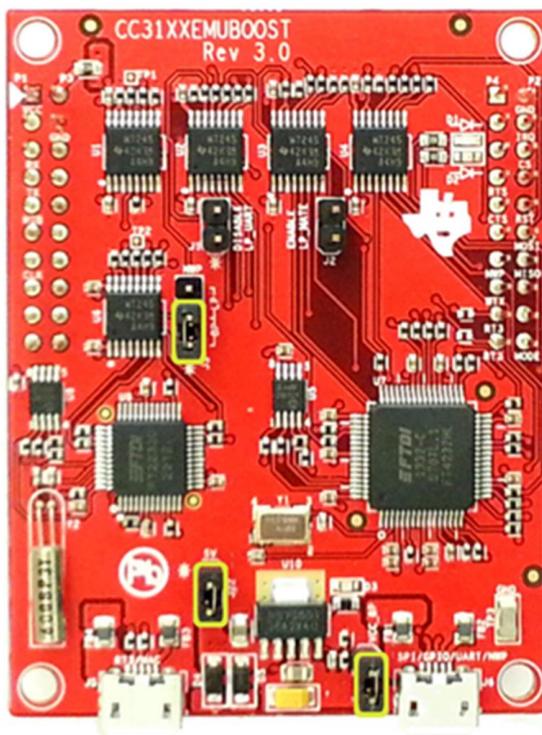

Figure 13-1. CC31XXEMU Jumper Configuration

The CC31XXEMUBOOST uses a logic level of 3.3 V by default, but there are level shifters on the CC31XXEMUBOOST and it can be powered in-dependently with a different IO voltage. This can be accomplished by removing the jumper from J4, and applying the external power to pin 1 of jumper J4. On the product being programmed, the relevant CC3100 module pins must be brought out for physical contact with the programmer (male headers, test pads), and must be driven by no other source while programming. The nHIB pin must be able to be temporarily pulled to GND during a reset, which occurs at the beginning of Uniflash flashing procedure. The pulling down of this line can be enabled by the addition of a button or something similar to the test jig.
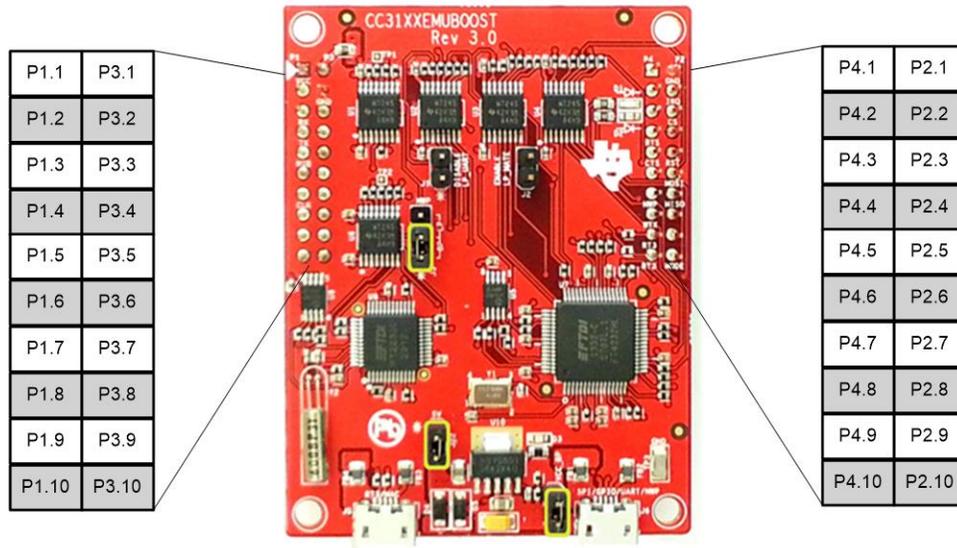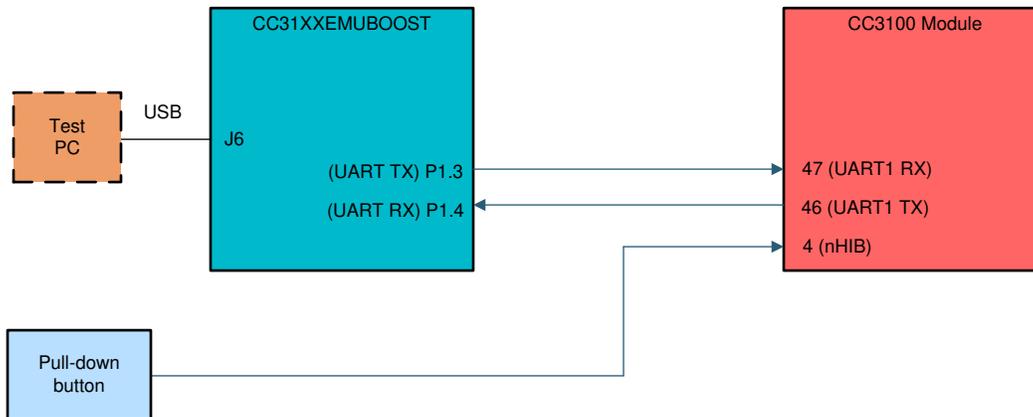

Figure 13-2. CC31XXEMU Pins

Figure 13-3. CC3100MOD UART Setup

14 Programming Using Uniflash CC3200

14.1 Using the Uniflash CLI

The Uniflash utility is PC software that can perform the following operations on the CC3200:

- Format the serial flash
- Add or remove files on the serial flash
- Update the service pack

These operations are accomplished by communicating with the CC3200 via UART. Therefore, it is necessary to supply Uniflash with a COM port number for the USB to CC3200 UART connection. See the chapter Programming over UART for details on the UART connection. Uniflash contains a command line interface that can be used in batch files/scripts for the purposes of programming the CC3200 devices in the production line. With a single line command, Uniflash can format the serial flash, update the service pack, and add any number of files. For example:

- uniflashCLI.bat -config "C:\ti\uniflash_3.2\cc3xxx\sessions\uniflash_template\oob.ucf" -setOptions com=50 secure=false spPath="C:\servicepack.bin" -operations format servicePackUpdate program

When Uniflash begins, it will prompt the user to restart the device. This is so it can synchronize with the CC3200 and begin the flashing process. At this point the CC3200 must be reset by pulling down the nRESET line. When the nRESET line resumes a logic high state, the flashing procedure will begin. In the production line, this reset process can be realized by using a button or other mechanism that temporarily connects the line to GND. The SOP2 pin must be pulled up during the reset. For complete information on how to use Uniflash, see the *SimpleLink™ Wi-Fi® CC3100, CC3200 UniFlash User's Guide*.

14.2 UART Hardware Connection Using the FTDI Emulation Board

The CC31XXEMUBOOST can provide the required USB to UART interface for formatting and programming the serial flash via UART. The PC drivers for this board are included in the CC3100 SDK, and they should be installed during installation of the SDK. The CC31XXEMUBOOST is connected via USB from socket J6 to the PC. The jumpers on the CC31XXEMUBOOST should be connected as shown below.
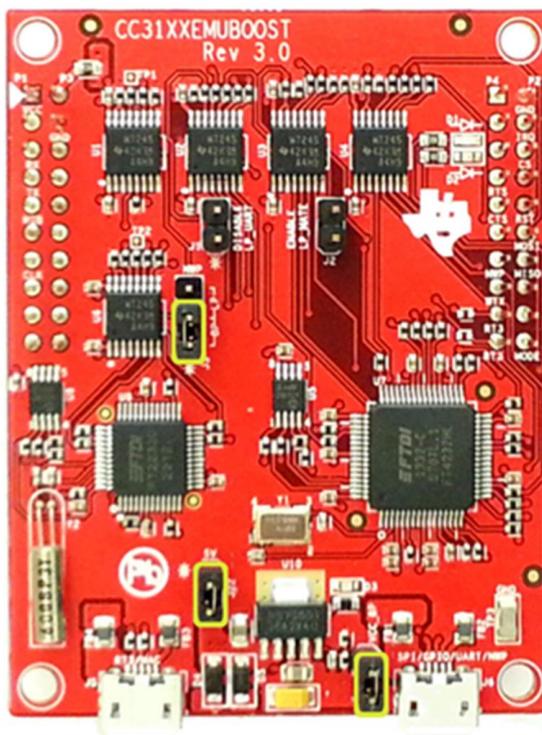

Figure 14-1. CC31XXEMU Jumper Configuration

The CC31XXEMUBOOST uses a logic level of 3.3 V by default, but there are level shifters on the CC31XXEMUBOOST and it can be powered in-dependently with a different IO voltage. This can be accomplished by removing the jumper from J4, and applying the external power to pin 1 of jumper J4. On the product being programmed, the relevant CC3200 pins must be brought out for physical contact with the programmer (male headers, test pads), and must be driven by no other source while programming. The nRESET pin must be able to be temporarily pulled to GND during a reset, which occurs at the beginning of Uniflash flashing procedure. The pulling down of this line can be enabled by the addition of a button or something similar to the test jig. Similarly, the SOP2 pin must be able to be pulled up during the flashing procedure. The SOP2 pin must be pulled up during the device reset in order for the CC3200 to enter a mode for communicating with Uniflash.

P1.1	P3.1			P4.1	P2.1
P1.2	P3.2			P4.2	P2.2
P1.3	P3.3			P4.3	P2.3
P1.4	P3.4			P4.4	P2.4
P1.5	P3.5			P4.5	P2.5
P1.6	P3.6			P4.6	P2.6
P1.7	P3.7			P4.7	P2.7
P1.8	P3.8			P4.8	P2.8
P1.9	P3.9			P4.9	P2.9
P1.10	P3.10			P4.10	P2.10
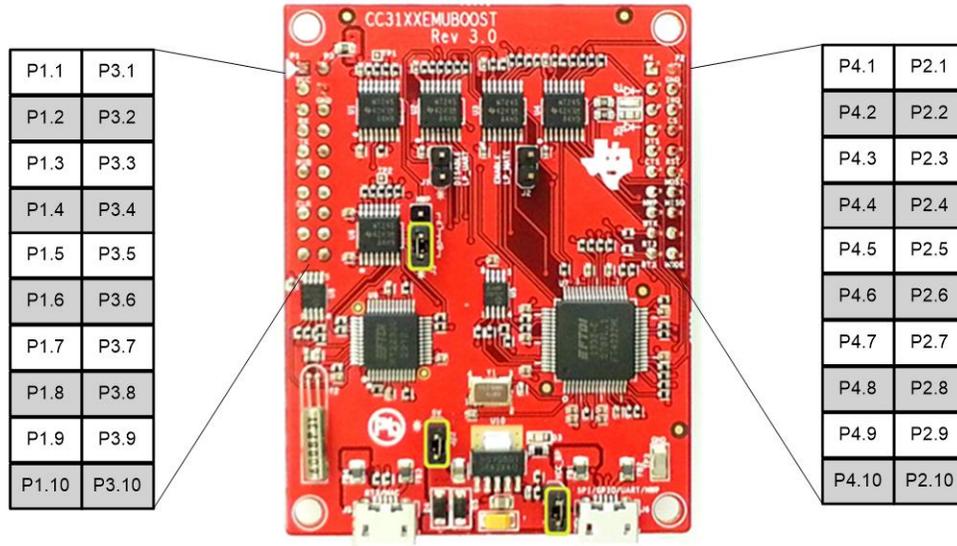
Figure 14-2. CC31XXEMU Pins

Figure 14-3. CC3200 UART Setup

Copyright © 2021 Texas Instruments Incorporated
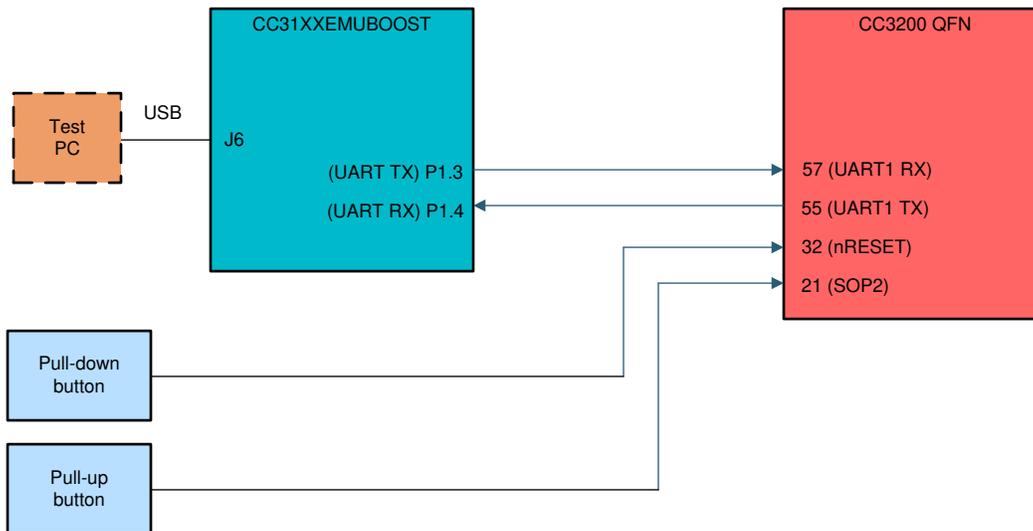
15 Programming Using Uniflash CC3200 Mod

15.1 Using the Uniflash CLI

The Uniflash utility is PC software that can perform the following operations on the CC3200:

- Format the serial flash
- Add or remove files on the serial flash
- Update the service pack

These operations are accomplished by communicating with the CC3200 module via UART. Therefore, it is necessary to supply Uniflash with a COM port number for the USB to CC3200 UART connection. See the chapter Programming over UART for details on the UART connection. Uniflash contains a command line interface that can be used in batch files/scripts for the purposes of programming the CC3200 modules in the production line. With a single line command, Uniflash can format the serial flash, update the service pack, and add any number of files. For example:

- uniflashCLI.bat -config "C:\ti\uniflash_3.2\cc3xxx\sessions\uniflash_template\oob.ucf" -setOptions com=50 secure=false spPath="C:\servicepack.bin" -operations format servicePackUpdate program

When Uniflash begins, it will prompt the user to restart the device. This is so it can synchronize with the CC3200 module and begin the flashing process. At this point the CC3200 module must be reset by pulling down the nRESET line. When the nRESET line resumes a logic high state, the flashing procedure will begin. In the production line, this reset process can be realized by using a button or other mechanism that temporarily connects the line to GND. The SOP2 pin must be pulled up during the reset. For complete information on how to use Uniflash, see the *SimpleLink™ Wi-Fi® CC3100, CC3200 UniFlash User's Guide*.

15.2 UART Hardware Connection Using the FTDI Emulation Board

The CC31XXEMUBOOST can provide the required USB to UART interface for formatting and programming the serial flash via UART. The PC drivers for this board are included in the CC3100 SDK, and they should be installed during installation of the SDK. The CC31XXEMUBOOST is connected via USB from socket J6 to the PC. The jumpers on the CC31XXEMUBOOST should be connected as shown in Figure 15-1.
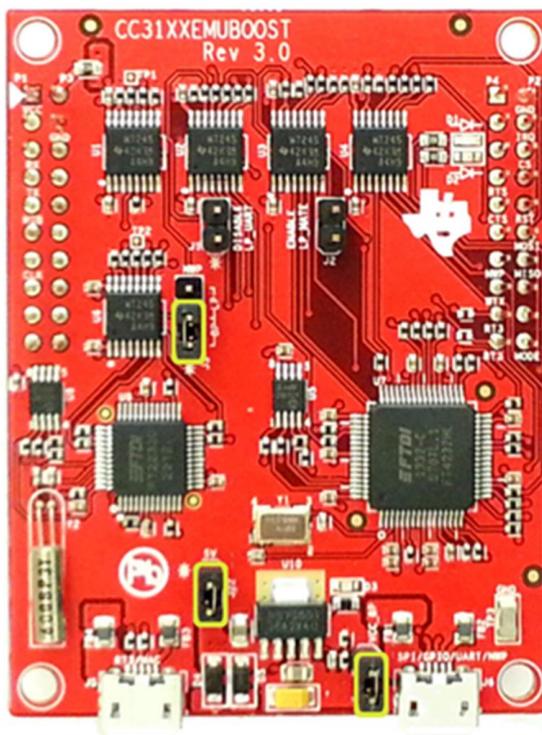

Figure 15-1. CC31XXEMU Jumper Configuration

The CC31XXEMUBOOST uses a logic level of 3.3 V by default, but there are level shifters on the CC31XXEMUBOOST and it can be powered in-dependently with a different IO voltage. This can be accomplished by removing the jumper from J4, and applying the external power to pin 1 of jumper J4. On the product being programmed, the relevant CC3200 module pins must be brought out for physical contact with the programmer (male headers, test pads), and must be driven by no other source while programming. The nRESET pin must be able to be temporarily pulled to GND during a reset that occurs at the beginning of Uniflash flashing procedure. The pulling down of this line can be enabled by the addition of a button or something similar to the test jig. Similarly, the SOP2 pin must be able to be pulled up during the flashing procedure. The SOP2 pin must be pulled up during the device reset in order for the CC3200 module to enter a mode for communicating with Uniflash.
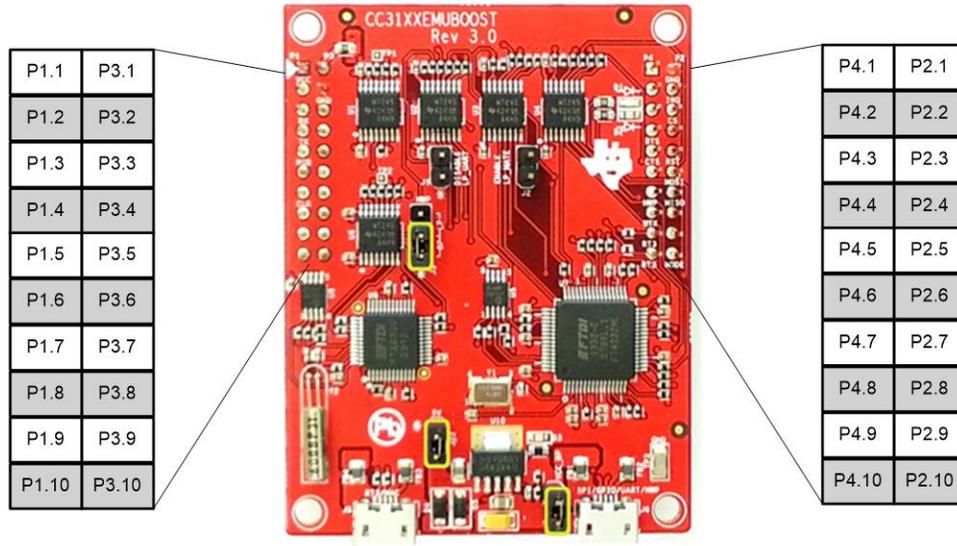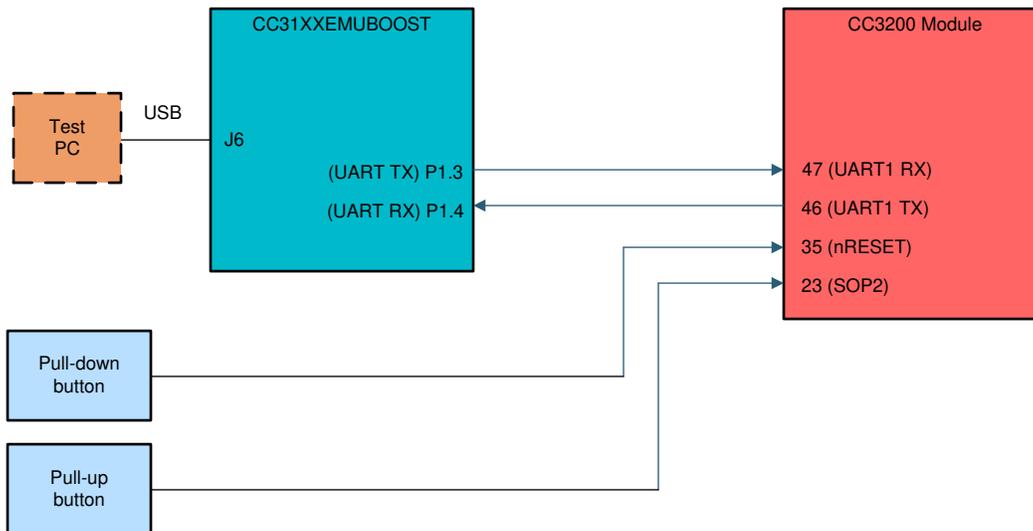

Figure 15-2. CC31XXEMU Pins

Figure 15-3. CC3200MOD UART Setup

16 Programming With the Host

16.1 Formatting Using Host UART Connection

The CC3100 QFN may have its serial flash formatted by the host processor if the host has the required UART connection with the CC3100. A reset of the CC3100 device with the transmission of a specific command sequence over UART will trigger the formatting process. The serial flash of the CC3100 device cannot be formatted by using SimpleLink API functions. Formatting the CC3100 serial flash using the host processor instead of a PC will reduce the number of connections made between the product and the PC. If this is the case, files will generally be written to the CC3100 serial flash either by SimpleLink API, or by using OTA.

The procedure to format the device is as follows:

1. Send a break signal (sending continuous Spacing values (no Start or Stop bits)) on the CC3100 UART RX line. The CC3100 device needs to sense this break signal during powering on.
2. Power on the CC3100 device (or reset if already up and running).
3. The CC3100 device should send an acknowledgment indication on the CC3100 UART TX line.
4. The acknowledgment indication is two bytes long: 0x00 0xCC.

Note

At this point, the CC3100 is ready to receive commands such as format. There is a 5 second timeout for the MCU to send the format command. Failing to do so before the timeout expires would result in CC3100 device initializing normally.

---

5. Upon reception of the acknowledgment indication from the CC3100 device, the MCU should stop sending the break signal and flush the UART lines.
6. MCU should send the string of characters on the CC3100 UART RX line according to the capacity of the serial flash. The supported capacities are described Table 16-1 along with their matching strings.
7. Upon completion of the format operation, the CC3100 device should acknowledge on the CC3100 UART TX line.
8. The acknowledgment indication is two bytes long: 0x00 0xCC.
9. Invoke the sl_Start() function. The CC3100 device should initialize with the new Service Pack.

Table 16-1. CC3100 Flash Formatting String

Flash Capacity [MB]	Formatting String (bytes hexadecimal)
0.5	0x00,0x17,0xAC,0x28,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x02
1	0x00,0x17,0x2D,0x28,0x00,0x00,0x00,0x02,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x02
2	0x00,0x17,0x2E,0x28,0x00,0x00,0x00,0x02,0x00,0x00,0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x02
4	0x00,0x17,0x30,0x28,0x00,0x00,0x00,0x02,0x00,0x00,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0 x00,0x00,0x02
8	0x00,0x17,0x34,0x28,0x00,0x00,0x00,0x02,0x00,0x00,0x08,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0 x00,0x00,0x02
16	0x00,0x17,0x3C,0x28,0x00,0x00,0x00,0x02,0x00,0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x02

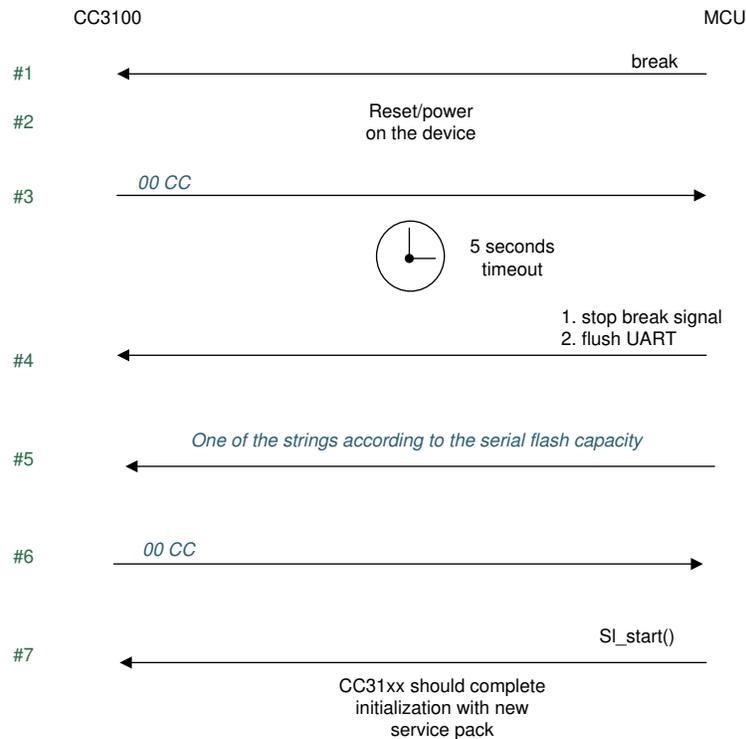To summarize the procedure, see Figure 16-1.

Figure 16-1. CC3100 Format Sequence

16.2 Writing Files to the Serial Flash From the Host

Files may be written to the CC3100 serial flash through the host interface by using the SimpleLink file system API, after the serial flash has been formatted. Data for the files could come from a number of sources:

- Files can be stored as arrays in host code. This method is simple to implement and is most practical for small amounts of data.
- Data can be read in to host memory by a serial peripheral of the host such as SPI, SDIO, or JTAG. This is more practical for larger amounts of data.
- The host can also retrieve data using a Wi-Fi connection from a remote or local server.

If the data rate of the connection from the host to the outside data source is significantly faster than that of the Uniflash UART connection (921600 bps), a time savings over programming using Uniflash can be realized. There are a small number of functions in the SimpleLink API for performing file operations:

- sl_FsOpen() - Open a file for read, write, or create
- sl_FsClose() - Close a file
- sl_FsRead() - Read block of data from a file
- sl_FsWrite() - Write block of data to a file
- sl_FsGetInfo() - Get information for a file
- sl_FsDel() - Delete a file

The complete documentation for the file system API can be found inside the CC3100 SDK folder at docs \simplelink_api\programmers_guide.html. For details on the operation of the file system, see the *Serial Flash File System* section in the *CC3100/CC3200 SimpleLink™ Wi-Fi® Interneton- a-Chip User's Guide*.

Given that all files can be written to the serial flash by using the SimpleLink API, the service pack can be updated by this method as well. The service pack binary image contains a set of patches on top of the ROM version. Each service pack must be authenticated by the CC3100 device as part of its secured file system. Thus a service pack includes two files: one is the binary image itself and the other is the image digital signature. The service pack can be downloaded from: http://www.ti.com/tool/cc3100sdk.

This package contains two formats of binaries:

- Firmware binary used by Uniflash tool
- OTA binary along with its signature file. The OTA version can be used by the host MCU and can be programmed into the serial flash using the file system API. These OTA binaries are used for over the air updates but can also be used for host programming.

After successfully installing the package (in this case for service pack v1.0.0.1.2), the following two files can be found:

- OTA_1.0.0.1.2.ucf.ucf – this is the service pack binary
- OTA_1.0.0.1.2.ucf.signed.bin – this is the signature of the service pack

The procedure to program the service pack is as follows:

1. Make sure the service pack image and its 256 byte signature are converted to C-arrays and stored on your host MCU memory.
2. Open the service pack file with the following configuration:
   a. Open to write/create (if not exists)
   b. Maximum size should be set to 128KB
   c. Flags:
      i. Commit
      ii. Secured
      iii. Public write
      iv. Write the service pack image

Note

It is mandatory to write the files in 16 byte aligned chunks.

---

3. Close the file. Upon closing a file, its signature must be provided as well. The certificate should be left NULL.

The following is a code snippet illustrating the procedure:

```
 /* create/open the servicepack file for 128KB with rollback, secured and public write */
  retVal = sl_FsOpen("/sys/servicepack.ucf", FS_MODE_OPEN_CREATE(131072,_FS_FILE_OPEN_FLAG_SECURE|
 _FS_FILE_OPEN_FLAG_COMMIT|_FS_FILE_PUBLIC_WRITE), &Token, &fileHandle);

  if(retVal < 0)
  {
      /* cannot open Service Pack file */
      return -1;
  }
  /* program the service pack */
  remainingLen = sizeof(servicePackImage);
  movingOffset = 0;
  chunkLen = (_u32)find_min(1024 /*CHUNK_LEN*/, remainingLen);
  /* Flashing is done in 1024 bytes chunks because of a bug resolved in later patches */

  do
  {
  retVal = sl_FsWrite(fileHandle, movingOffset, (_u8 *)&servicePackImage[movingOffset], chunkLen);

      if (retVal < 0)
      {
          /* cannot program ServicePack file */
          return -1;
      }

      remainingLen -= chunkLen;
      movingOffset += chunkLen;
      chunkLen = (_u32)find_min(1024 /*CHUNK_LEN*/, remainingLen);

  }while (chunkLen > 0);

  /* close the servicepack file */
  retVal = sl_FsClose(fileHandle, 0, (_u8 *)servicePackImageSig, sizeof(servicePackImageSig));

  if (retVal < 0)
```

```
{
    /* cannot close Service Pack file */
    return -1;
}
```

An example for programming the service pack via host is available in the CC3100 SDK: http://www.ti.com/tool/cc3100sdk.

17 Programming With the Host CC3100 MOD
17.1 Formatting Using Host UART Connection

The CC3100 module should come pre-formatted. If another format of the internal serial flash is required for any reason, this can be done by the host processor if the host has the required UART connection with the CC3100 module. A reset of the CC3100 module with the transmission of a specific command sequence over UART will trigger the formatting process. Formatting the CC3100 module using the host processor instead of a PC will reduce the number of connections made between the product and the PC. The serial flash of the CC3100 module cannot be formatted by using SimpleLink API functions.

The procedure to format the device is as follows:

1. Send a break signal (sending continuous Spacing values (no Start or Stop bits)) on the CC3100 UART RX line. The CC3100 device needs to sense this break signal during powering on.
2. Power on the CC3100 device (or reset if already up and running).
3. The CC3100 device should send an acknowledgment indication on the CC3100 UART TX line.

   The acknowledgment indication is two bytes long: 0x00 0xCC.

Note

At this point, the CC3100 is ready to receive commands such as format. There is a 5 second timeout for the MCU to send the format command. Failing to do so before the timeout expires would result in CC3100 device initializing normally.

---

4. Upon reception of the acknowledgment indication from the CC3100 device, the MCU should stop sending the break signal and flush the UART lines.
5. MCU should send the string of characters on the CC3100 UART RX line according to the capacity of the serial flash. The supported capacities are described in Table 17-1 along with their matching strings.
6. Upon completion of the format operation, the CC3100 device should acknowledge on the CC3100 UART TX line.

   The acknowledgment indication is two bytes long: 0x00 0xCC.
7. Invoke the sl_Start() function. The CC3100 device should initialize with the new Service Pack.

Table 17-1. CC3100MOD Flash Formatting String

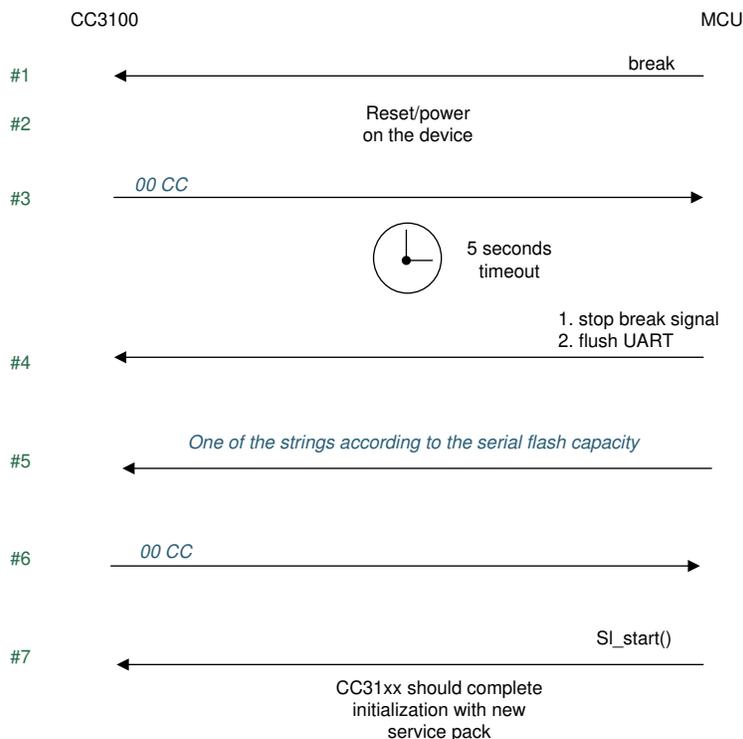| Flash Capacity [MB]	Formatting String (bytes hexadecimal)
0.5	0x00,0x17,0xAC,0x28,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x02
1	0x00,0x17,0x2D,0x28,0x00,0x00,0x00,0x02,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x02
2	0x00,0x17,0x2E,0x28,0x00,0x00,0x00,0x02,0x00,0x00,0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x02
4	0x00,0x17,0x30,0x28,0x00,0x00,0x00,0x02,0x00,0x00,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0 x00,0x00,0x02
8	0x00,0x17,0x34,0x28,0x00,0x00,0x00,0x02,0x00,0x00,0x08,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0 x00,0x00,0x02
16	0x00,0x17,0x3C,0x28,0x00,0x00,0x00,0x02,0x00,0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x02

To summarize the procedure, see Figure 17-1.

Figure 17-1. CC3100MOD Format Sequence

17.2 Writing Files to the Serial Flash From the Host

Files may be written to the CC3100 module through the host interface by using the SimpleLink file system API. Data for the files could come from a number of sources:

- Files can be stored as arrays in host code. This method is simple to implement and is most practical for small amounts of data.
- Data can be read in to host memory by a serial peripheral of the host such as SPI, SDIO, or JTAG. This is more practical for larger amounts of data.
- The host can also retrieve data using a Wi-Fi connection from a remote or local server.

If the data rate of the connection from the host to the outside data source is significantly faster than that of the Uniflash UART connection (921600 bps), a time savings over programming using Uniflash can be realized. There are a small number of functions in the SimpleLink API for performing file operations:

- sl_FsOpen() - Open a file for read, write, or create
- sl_FsClose() - Close a file
- sl_FsRead() - Read block of data from a file
- sl_FsWrite() - Write block of data to a file
- sl_FsGetInfo() - Get information for a file
- sl_FsDel() - Delete a file

The complete documentation for the file system API can be found inside the CC3100 SDK folder at docs \simplelink_api\programmers_guide.html wiki. For details on the operation of the file system, see the *Serial Flash File System* section in the *CC3100/CC3200 SimpleLink Wi-Fi Internet-on-a-Chip User's Guide*.

Given that all files can be written to the serial flash by using the SimpleLink API, the service pack can be updated by this method as well. The service pack binary image contains a set of patches on top of the ROM version. Each service pack must be authenticated by the CC3100 device as part of its secured file system. Thus, a service pack includes two files: one is the binary image itself and the other is the image digital signature. The service pack can be downloaded from: http://www.ti.com/tool/cc3100sdk.

This package contains 2 formats of binaries:

- Firmware binary used by Uniflash tool
- OTA binary along with its signature file. The OTA version can be used by the host MCU and can be programmed into the serial flash using the file system API. These OTA binaries are used for over the air updates but can also be used for host programming.

After successfully installing the package (in this case for service pack v1.0.0.1.2), the following two files can be found:

- OTA_1.0.0.1.2.ucf.ucf - this is the service pack binary
- OTA_1.0.0.1.2.ucf.signed.bin - this is the signature of the service pack

The procedure to program the service pack is as follows:

- Make sure the service pack image and its 256 byte signature are converted to C-arrays and stored on your host MCU memory

Open the service pack file with the following configuration:

- Open to write/create (if not exists)
- Maximum size should be set to 128KB

Flags:

- Commit
- Secured
- Public write
- Write the service pack image

Note

It is mandatory to write the files in 16 byte aligned chunks.

- Close the file. Upon closing a file, its signature must be provided as well. The certificate should be left NULL.

The following is a code snippet illustrating the procedure:

```
/* create/open the servicepack file for 128KB with rollback, secured and public write */

retVal = sl_FsOpen("/sys/servicepack.ucf", FS_MODE_OPEN_CREATE(131072,_FS_FILE_OPEN_FLAG_SECURE|
_FS_FILE_OPEN_FLAG_COMMIT|_FS_FILE_PUBLIC_WRITE), &Token, &fileHandle);


if(retVal < 0)
{
    /* cannot open Service Pack file */
    return -1;
}


/* program the service pack */

remainingLen = sizeof(servicePackImage);
movingOffset = 0;
chunkLen = (_u32)find_min(1024 /*CHUNK_LEN*/, remainingLen);


/* Flashing is done in 1024 bytes chunks because of a bug resolved in later patches */

do
{
retVal = sl_FsWrite(fileHandle, movingOffset, (_u8 *)&servicePackImage[movingOffset], chunkLen);

    if (retVal < 0)
    {
        /* cannot program ServicePack file */
        return -1;
    }
```

```
        remainingLen -= chunkLen;
        movingOffset += chunkLen;
        chunkLen = (_u32)find_min(1024 /*CHUNK_LEN*/, remainingLen);

    }while (chunkLen > 0);


    /* close the servicepack file */
    retVal = sl_FsClose(fileHandle, 0, (_u8 *)servicePackImageSig, sizeof(servicePackImageSig));

    if (retVal < 0)
    {
        /* cannot close Service Pack file */
        return -1;
    }
```

An example for programming the service pack via host is available in the CC3100 SDK: http://www.ti.com/tool/cc3100sdk.

18 Over the Air Programming CC3100

The CC3100 has the capability for Over-the-Air programming that allows for files to be written and updated over a network connection. An OTA programming library is available in the CC3100 SDK. The OTA update scheme relies on a subroutine that periodically checks with a remote OTA server to see if there are any available file updates. The OTA server will respond to such requests with a list of resources to update, and will also respond to requests for locations of those resources on the internet. The OTA client that runs on the MCU will download the listed resources from the locations specified by the OTA server. In the field, the resources will typically reside on a Content Delivery Network (CDN). The MCU can test the validity of the files before committing them. The committing of the new version of the file causes it to be used in place of the old one already stored in serial flash.

Using OTA in the production line can enable faster data transfer over other methods. For using OTA in the production line, a PC on the local network can run the OTA server, and can also run the content delivery server for the resources as well. The service pack should be updated before doing OTA or using any wireless functionality. For the fastest transfer of data using OTA, it is recommended to minimize RF congestion in the production environment. The OTA User's Guide can be found in the CC3100 SDK folder at docs/CC3100 Simplelink OTA Extlib API User's Guide.chm.

19 Over the Air Programming CC3100 MOD

The CC3100 module has the capability for Over-the-Air programming that allows for files to be written and updated over a network connection. An OTA programming library is available in the CC3100 SDK. The OTA update scheme relies on a subroutine that periodically checks with a remote OTA server to see if there are any available file updates. The OTA server will respond to such requests with a list of resources to update, and will also respond to requests for locations of those resources on the internet. The OTA client that runs on the MCU will download the listed resources from the locations specified by the OTA server. In the field, the resources will typically reside on a Content Delivery Network (CDN). The MCU can test the validity of the files before committing them. The committing of the new version of the file causes it to be used in place of the old one already stored in serial flash.

Using OTA in the production line can enable faster data transfer over other methods. For using OTA in the production line, a PC on the local network can run the OTA server, and can also run the content delivery server for the resources as well. For the fastest transfer of data using OTA, it is recommended to minimize RF congestion in the production environment. The OTA User's Guide can be found in the CC3100 SDK folder at docs/CC3200 Simplelink OTA Extlib API User's Guide.chm.

20 Over the Air Programming CC3200

The CC3200 has the capability for Over-the-Air programming that allows for files to be written and updated over a network connection. An OTA programming library is available in the CC3200 SDK. The OTA update scheme relies on a subroutine that periodically checks with a remote OTA server to see if there are any available file updates. The OTA server will respond to such requests with a list of resources to update, and will also respond to requests for locations of those resources on the internet. The OTA client that runs on the MCU will download the listed resources from the locations specified by the OTA server. In the field, the resources will typically reside on a Content Delivery Network (CDN). The MCU can test the validity of the files before committing them. The committing of the new version of the file causes it to be used in place of the old one already stored in serial flash.

Using OTA in the production line can enable faster data transfer over other methods. For using OTA in the production line, a PC on the local network can run the OTA server, and can also run the content delivery server for the resources as well. The service pack should be updated before doing OTA or using any wireless functionality. For the fastest transfer of data using OTA, it is recommended to minimize RF congestion in the production environment. The OTA User's Guide can be found in the CC3200 SDK folder at docs/CC3200 Simplelink OTA Extlib API User's Guide.chm.

21 Over the Air Programming CC3200MOD

The CC3200 module has the capability for Over-the-Air programming that allows for files to be written and updated over a network connection. An OTA programming library is available in the CC3200 SDK. The OTA update scheme relies on a subroutine that periodically checks with a remote OTA server to see if there are any available file updates. The OTA server will respond to such requests with a list of resources to update, and will also respond to requests for locations of those resources on the internet. The OTA client that runs on the MCU will download the listed resources from the locations specified by the OTA server. In the field, the resources will typically reside on a Content Delivery Network (CDN). The MCU can test the validity of the files before committing them. The committing of the new version of the file causes it to be used in place of the old one already stored in serial flash.

Using OTA in the production line can enable faster data transfer over other methods. For using OTA in the production line, a PC on the local network can run the OTA server, and can also run the content delivery server for the resources as well. For the fastest transfer of data using OTA, it is recommended to minimize RF congestion in the production environment. The OTA User's Guide can be found in the CC3200 SDK folder at *docs/CC3200 Simplelink OTA Extlib API User's Guide.chm*.

22 Production Line RF Testing CC3100

Testing of hardware and software functionality is highly specific to each product, but there are some tools Texas Instruments has made available to assist with testing RF performance. The CC3100 can be instructed to perform RF testing operations in a number of ways:

- The host code may have a built in subroutine that is dedicated to RF testing. This could be run once upon first power-up, or could be triggered using a special external command.
- A SimpleLink Studio program or a script using the Radio Tool CLI could control the CC3100 from a PC. This would require the CC3100 to be temporarily disconnected from the host, and connected to the PC through a UART to USB connection.
- The CC3100 could be controlled by interfacing with a dedicated RF tester. This would require the CC3100 to be temporarily disconnected from the host.

22.1 Testing Software Options

22.1.1 MCU Controlled RF Testing

SimpleLink API functions are available that can put the CC3100 device into modes used for RF testing. This allows for:

- Transmission of packets at specified channels, modulations, and so forth.
- Receipt of packets while gathering statistics for RSSI, modulation, and so forth.
- Carrier wave transmission

For information on how to use these features, see the wiki. For comprehensive information about the SimpleLink API, see the Programmer's Guide in the CC3100 SDK at: docs\simplelink_api\programmers_guide.html.

The Radio Tool library provides a set of convenience functions for RF testing, and can be ported to any MCU:

- RadioToolOpen() - Initializes the device in preparation for RF testing
- RadioToolClose() - Stops the device
- RadioStartTX() - Transmit packets
- RadioStopTX() - Stop transmission
- RadioStartRX() - Start receiver
- RadioStopRX() - Stop receiver
- RadioGetStats() - Get statistics regarding received packets

- RadioGetMacAddr() - Retrieve MAC address of device
- RadioGetDeviceVersion() - Retrieve device Firmware version numbers

For Radio Tool library source code and for information on how to incorporate RF testing functionality into an MCU application, see the *SimpleLink™ Wi-Fi® CC3x20, CC3x3x Radio Tool User Guide*.

22.2 PC Controlled RF Testing

22.2.1 SimpleLink Studio

SimpleLink Studio can be used to run a PC application to perform any function with the CC3100. This will require access to the CC3100 host interface, and thus it will need to be disconnected from the host to prevent contention. Usually this is combined with the CC31XXEMUBOOST to assist with development of software independently of hardware, but it may be used for testing purposes as well. It may be useful especially in cases where the host code cannot implement testing routines itself. SimpleLink Studio can be run on Microsoft Visual Studio, or Eclipse environments. For more information, see the CC3100 SimpleLink Studio wiki page.

22.2.2 Radio Tool CLI

The Radio Tool command line interface is a PC based tool that can be used to perform RF tests on CC3200/ CC3100 devices. Generally, it will be incorporated into in scripts or batch files used for production line testing. When used with the CC3100, the Radio Tool CLI communicates directly through the host interface of the CC3100. Because of this, the host interface will need to be temporarily disconnected from the host to prevent contention. This tool requires use of the CC31XXEMUBOOST for communication between the CC3100 and the PC. For more information, see the *SimpleLink™ Wi-Fi® CC3x20, CC3x3x Radio Tool User Guide*.

22.3 Testing With an Access Point

A straightforward method of checking for acceptable RF performance is to put the device being tested through a trial run in an RF environment with worst case conditions. Such a trial run would begin with the device under test connecting to an access point, and then communicating with either a PC on the local network or with a remote cloud server. The communication between the device under test and its peer can be monitored for reliability and speed. In order to get consistent and relevant results for all devices being tested, some actions may be taken with respect to the controlling RF environment for this type of testing:

- Minimize unintentional RF congestion in the test area. This can be accomplished by turning off other nearby 2.4 GHz band devices, and/or performing the testing in an RF shielded enclosure.
- Introduce controlled RF congestion. This can involve something such as having another device connected to the same access point, which transmits a steady stream of packets to the access point.
- Introduce attenuation in the antenna path for the access point, or place at a distance from the device being tested.
- Set the access point to communicate only on a specific channel, modulation, and so forth.

22.4 Board to Board RF Testing

For performing RF tests, it is possible to use one CC3100/CC3200 device to test the RF performance of another. One CC3100/CC3200 device in this case will be designated as a "golden device": where the device is proven to have good RF characteristics, and will be used to measure the performance of devices under test. These devices would generally put into modes allowing for direct control of TX parameters, and measurement of RX statistics. This can be accomplished through using the Radio Tool CLI, the Radio Tool library, or the SimpleLink API directly. Due to limitations in the FTDI driver, two computers must be used to do board to board testing if using the Radio Tool for both the DUT and the golden device. One computer will control the golden device, and the other will control the DUT.

Synchronization between the DUT and the golden device may be achieved by having them communicate through a different channel of communication (other than Wi-Fi), or by using an algorithm similar to the following DUT algorithm:

1. Send packets to Golden board for 4 seconds
2. Start receiver
3. Wait 4 seconds
4. Measure PER and/or RSSI of received packets

5. Stop receiver
6. If PER and/or RSSI is acceptable
   a. Return success
      else
      Return failure

Golden board algorithm:

1. Start receiver
2. Wait 2 seconds
3. Measure PER and/or RSSI of received packets
4. Stop receiver
5. If >=1000 packets received, and PER and/or RSSI is acceptable
   a. Send packets to DUT for 4 seconds
6. Go to 1

Note that for this algorithm, the Golden board continues its routine indefinitely so that no intervention is required for it to function with a continuous supply of DUTs. The basis of this algorithm is that receipt of packets of an acceptable quality by the Golden board will result in a reply by the Golden board with its own packets, which will then be judged for quality by the DUT.

Please note that the DUT and the golden device must be shielded from outside RF signals. This is necessary to ensure one device is reporting statistics only for packets sent from the other. This means that if performing radiated tests, both devices should be in a single shielded enclosure. If performing conducted tests, the DUT and the golden device should be in separate shielded enclosures with an attenuation system connecting them. This is because any PER measurements will be invalidated by RF leakage from the surface of one board to the other. If the DUT and the golden device synchronize with each other through a separate communication channel, steps should be taken to ensure the physical path for that channel does not carry any RF energy that would interfere with the measurements.

22.5 Dedicated Wireless Test Equipment

Using dedicated wireless testing equipment can be used if RF performance measurements with a high level of accuracy are required. Typically this will also require the use of an RF probe connector (U.FL, Murata UMC) for conducted tests, but in a carefully controlled environment radiated tests could be performed as well.

22.5.1 Testing With Litepoint

In collaboration with Litepoint, Texas Instruments provides support of testing CC3100/CC3200 devices with Litepoint testers. The computer that is performing the testing would be connected to the Litepoint tester via an Ethernet connection and to the DUT using a UART connection. Litepoint's console based test tool would run on the PC, and be incorporated into the production line test script. For more information, see www.litepoint.com.

23 References

- Texas Instruments: *CC3100 SimpleLink™ Wi-Fi® Network Processor, Internet-of-Things Solution for MCU Applications Data Sheet*
- Texas Instruments: *CC3100MOD SimpleLink™ Certified Wi-Fi® Network Processor Internet-of-Things Module Solution for MCU Applications Data Sheet*
- Texas Instruments: *CC3200MOD SimpleLink™ Wi-Fi® and Internet-of-Things Module Solution, a Single-Chip Wireless MCU Data Sheet*
- Texas Instruments: *CC3100/CC3200 SimpleLink™ Wi-Fi® Interneton- a-Chip User's Guide*
- Texas Instruments: *SimpleLink™ Wi-Fi® CC3100, CC3200 UniFlash*
- Texas Instruments: *SimpleLink™ Wi-Fi® CC3x20, CC3x3x Radio Tool User Guide*

IMPORTANT NOTICE AND DISCLAIMER