CC3100 Host Programming Application

Introduction

Return to CC31xx & CC32xx Home Page

The SimpleLink™ CC31xx is a Wi-Fi and networking device provides a comprehensive networking solution for low-cost and low-power micro-controllers using a thin driver and simple APIs set.

Each product that has an embedded CC31xx device on board must also have a serial flash device connected.

There are two common schemes for programming the serial flash, one where the serial flash is already assembled on the board and second where the serial flash is programmed first and only then assembled on the board.

Host Programming addresses the 1st scheme and thus intended for customers who would like to have the ability to program the device from the on-board MCU. The main gap is when the serial flash is assembled for the first time on Production Line and thus may be either unformatted or formatted but not programmed.

Following paragraphs describe the setup, procedure and usage of the Host Programming add-on that needs to be followed in order to program (and optionally format) the on-board assembled serial flash from the host.

Prerequisites

Hardware

1. CC3100 Booster pack with PG1.33 device.
2. MSP430F5529 Launchpad

Software

1. CC3100 SDK installed on the PC
2. CC3100 ServicePack installed on PC
3. CCS v6 installed

Setup

In case TI Evaluation boards are used, there is no need for special HW connections besides connecting the CC3100 BoosterPack to the MSP430F5529 Launchpad.

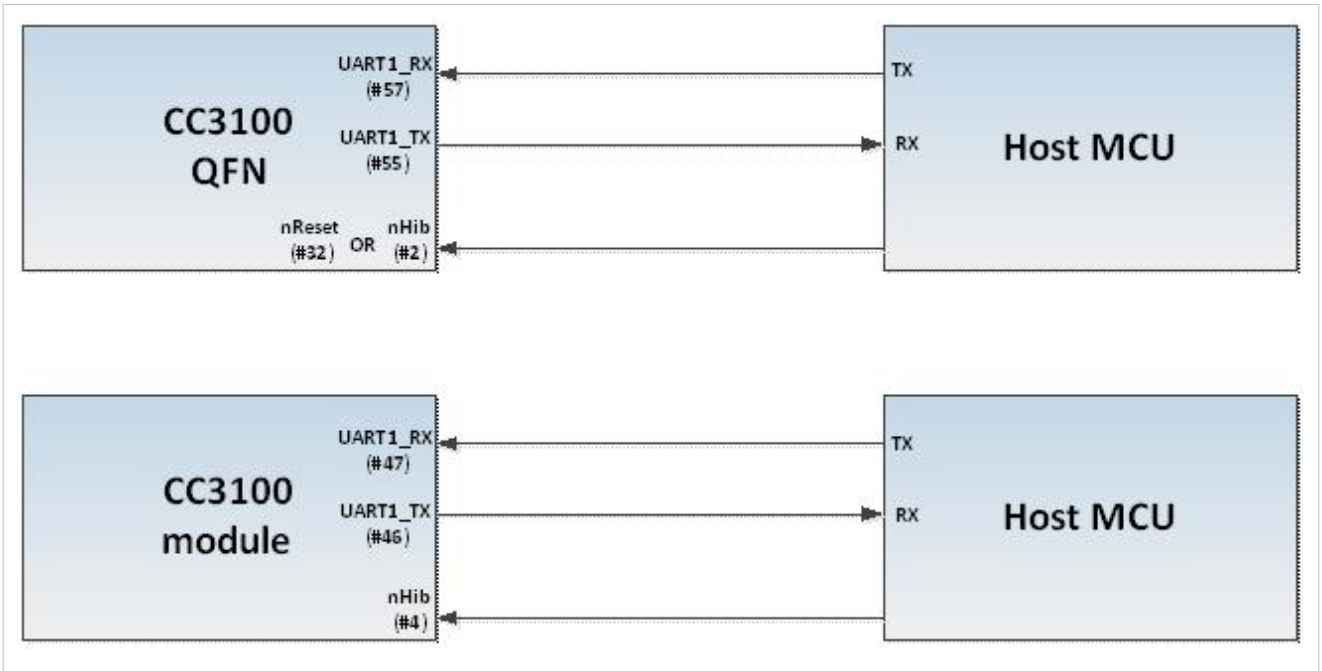
In case custom board is desired, and if serial flash formatting is also required, UART interface must be exposed and routed. Thus, the setup dictates that a UART interface must be connected between the host MCU and CC31xx device. Only two pins are required, UART TX and UART RX. Flow control is not required.

UART configuration depends on the chipset used.

The common configuration that applies to all chipsets is: baud rate of 921600bps, 8 bits, no parity, 1 stop bit.

CC3100 QFN package requires the following lines (can be found on the schematics): UART1_TX (pin #55), UART1_RX (pin #57) and either nHib (pin #2) or nReset (pin #32).

CC3100 module package requires the following lines (can be found on the schematics): UART1_TX (pin #46), UART1_RX (pin #47) and nHib (pin #4).



Procedure

The procedure can be divided into two main steps, formatting the device (optional) and programming the serial flash with the service pack.

Device Formatting

Device formatting is applied via the UART interface.

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 CC31xx UART RX line. CC31xx device needs to sense this break signal during powering on.
2. Power on CC31xx device (or reset it if already up and running)
3. CC31xx device should send an acknowledgement indication on CC31xx UART TX line.

   The acknowledgement indication is two bytes long: ***00 CC***

```
Note:  At this point, CC31xx is ready to receive commands such as format. There are 5 seconds for the MCU to send the format command.

       Failing to do so before the timeout expires would result in CC31xx device initializing normally.
```

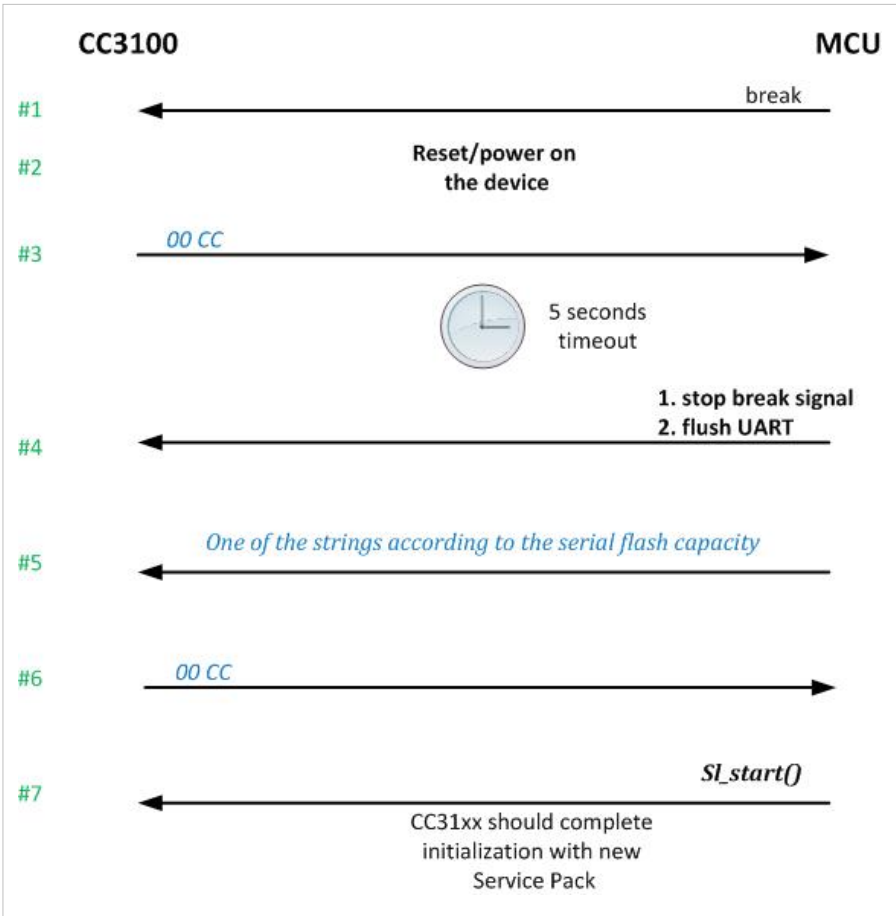
4. Upon reception of the acknowledgement indication from CC31xx device, the MCU should stop sending the break signal and flush the UART lines.
5. MCU should send the string of characters on CC31xx UART RX line according to the capacity of the serial flash. The supported capacities are described in the table below along with their matching strings.
6. Upon completion of format operation, CC31xx device should acknowledge on CC31xx UART TX line.

   The acknowledgement indication is two bytes long: ***00 CC***

7. Invoke ***sl_start()*** API. CC31xx device should initialize with the new Service Pack.

Flash capacity [MB]	Formatting string
0.5	***00 17 AC 28 00 00 00 02 00 00 00 80 00 00 00 00 00 00 00 00 00 00 00 02***
1	***00 17 2D 28 00 00 00 02 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 02***
2	***00 17 2E 28 00 00 00 02 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00 02***
4	***00 17 30 28 00 00 00 02 00 00 04 00 00 00 00 00 00 00 00 00 00 00 00 02***
8	***00 17 34 28 00 00 00 02 00 00 08 00 00 00 00 00 00 00 00 00 00 00 00 02***
16	***00 17 3C 28 00 00 00 02 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 02***

To summarize the procedure, please find below a graphic illustration:



Service pack programming

Note: host programming is supporting Service Pack update beginning from version 1.0.0.10.0 onward.

Service pack binary image contains a set of patches on top of the ROM version. Each service pack must be authenticated by CC3100/CC3200 device as part of its secured file system. Service Pack can be downloaded from: http://www.ti.com/tool/cc3100sdk

This package contains the firmware in 3 different formats:

1. Firmware binary used by Uniflash tool. Syntax is *servicepack_<ver>.bin*

2. OTA binary along with its signature file. The OTA version can be used by host MCU and be programmed into serial flash using the file system API. These OTA binaries are used for over the air update.

   The binary syntax is *OTA_<ver>.ucf.ucf*.

The signature syntax is *OTA_<ver>.ucf.signed.bin*.

3. Host Programming C-array. These are C-array format of the OTA binaries used mainly for host programming via host.

The ucf syntax is *host_programming_<ver>.h*.

The signature syntax is *host_programming_<ver>_signed.h*.

The procedure to program the service pack is as follows:

1. Open the service pack file with the following configuration:

a. Opens to write/create (if not exists)

b. Maximum size should be set to 128KB

c. Flags:

i. Commit

ii. Secured

iii. Public write

2. Write the service pack image

Note: it is mandatory to write the files in 16 bytes aligned chunks

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

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;
}
```

Usage

Basic setup

1. Install the Host Programming add-on under the CC3100 SDK installation folder

   a. Host programming application files should reside under

   ./cc3100-sdk/examples/host_programming

   b. Host programming platform specific files should reside under

   ./cc3100-sdk/platform/msp430f5529lp/example_project_ccs

2. Copy the 2 header files from the ServicePack installation directory to the host programming application directory.

   host_programming_<ver>.h and *host_programming_<ver>_signed.h* should be copied to *./cc3100-sdk/examples/host_programming*

3. Make sure that the versions of the C-array ServicePack,

   host_programming _<ver>.h and *host_programming_<ver>_signed.h* matches the #include in *main.c*.

4. Open CCS and import the host_programming project into your workspace.

5. In case format is desired prior to ServicePack programming, definition of *FORMAT_ENABLE* flag needs to be added to the project.

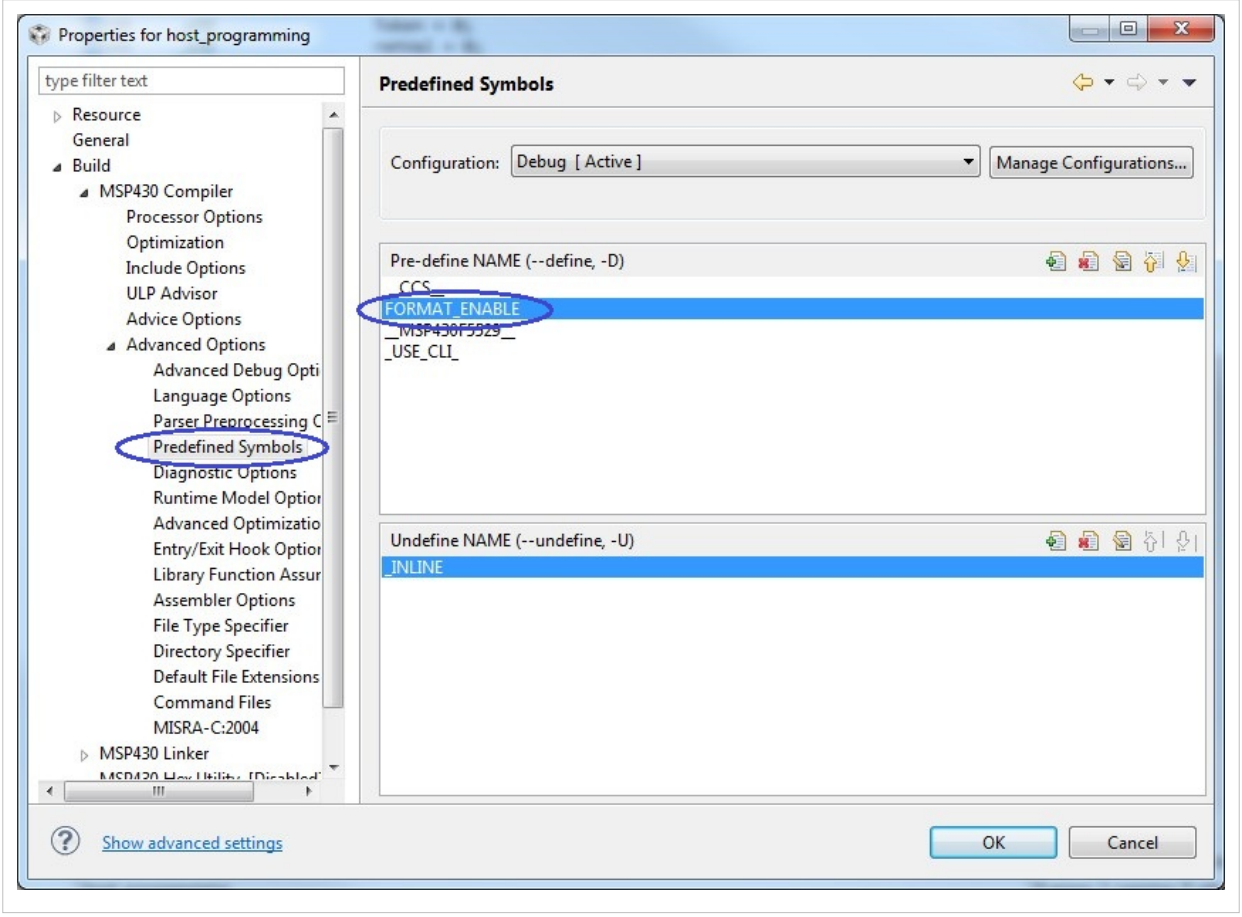
6. Build and launch the project.

Serial Flash formatting

In case formatting is required, 2 steps need to be completed:

1. Define needs to be added to the project. The definition is *FORMAT_ENABLE*

The figure below illustrates where the define needs to be added:



2. Since Boot Loader UART is using the same UART handler as the host UART interface, in order not to get a "redefine error", the following steps needs to be completed:

   a. Open the file *board.c* under ***./platform/msp430f5529lp***

   b. Locate the UART handler, void CC3100_UART_ISR(void). Should be at the end of the file

   c. Wrap it with *#ifndef FORMAT_ENABLE*

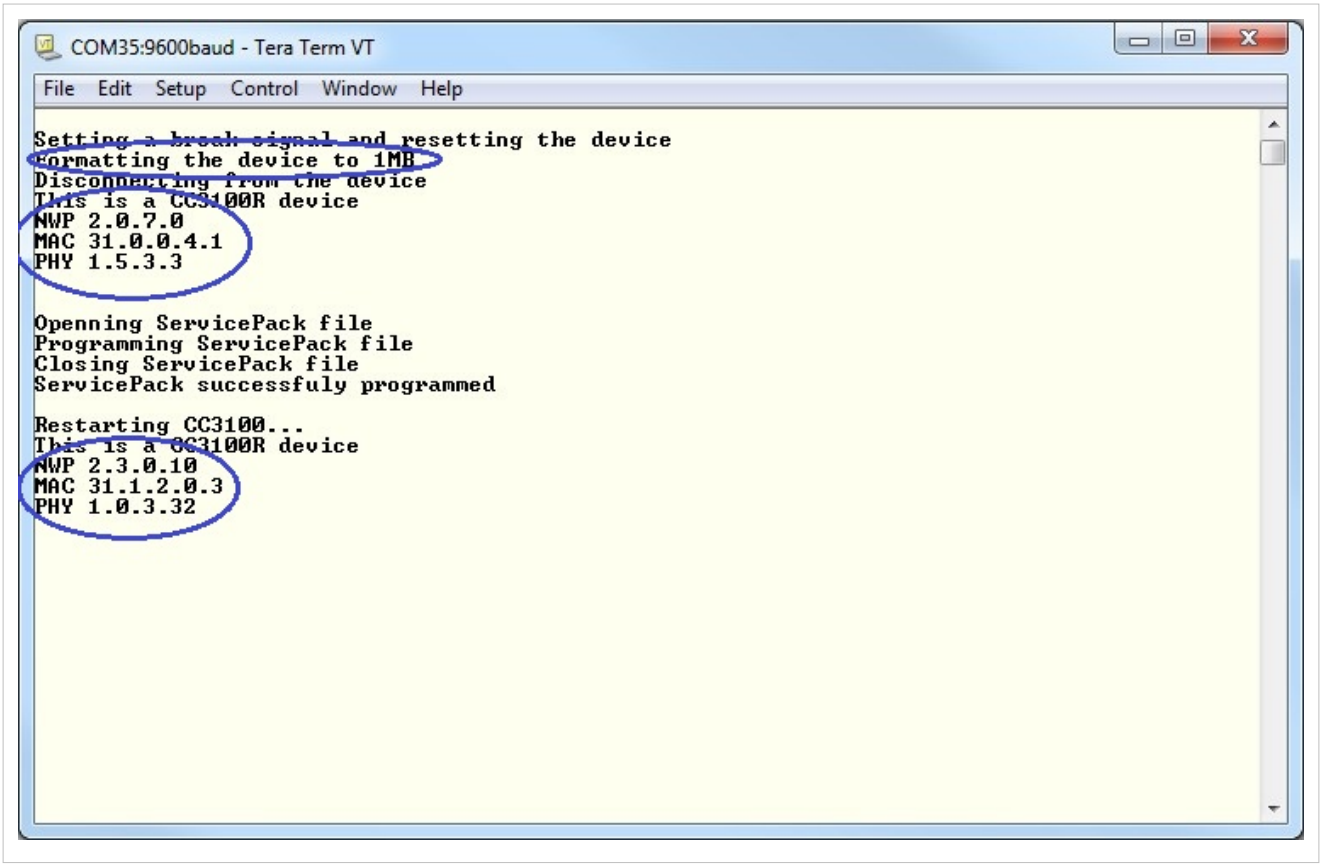
```
For reference, the following code snippet should be preceded by this ''#ifndef'':<br />
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=USCI_A0_VECTOR
__interrupt
#elif defined(__GNUC__)
__attribute__((interrupt(USCI_A0_VECTOR)))
#endif

void CC3100_UART_ISR(void)
```

Running the example

1. Mount the CC3100 Booster pack on MSP430F5529 Launchpad.
2. Connect to COM port via Tera-term or Hyper Terminal with following configuration:
    a. Baud rate: 9600
    b. Data: 8 bit
    c. Parity: None
    d. Stop: 1 bit
    e. Flow control: None
3. Run the example.
4. The application will apply the following steps:
    a. Format the serial flash if request
    b. Read and present the current device version
    c. Program the new ServicePack
    d. Restart the device
    e. Read and present the current device version. The new version should be aligned with the version of the ServicePack.

An example of this process, including format, can be shown in the following figure:



Limitations/Known issues

The capacity of the serial flash is assumed to be one of a fixed list as described in the table.

Article Sources and Contributors

CC3100 Host Programming Application *Source*: http://processors.wiki.ti.com/index.php?oldid=227242 *Contributors*: A0387625, SarahP

Image Sources, Licenses and Contributors

File:Cc31xx_cc32xx_return_home.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Cc31xx_cc32xx_return_home.png *License*: unknown *Contributors*: A0221015
Image:CC3100 QFN setup.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:CC3100_QFN_setup.jpg *License*: unknown *Contributors*: A0387625
Image:Host Programming Format.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Host_Programming_Format.jpg *License*: unknown *Contributors*: A0387625
Image:FORMAT ENABLE macro.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:FORMAT_ENABLE_macro.jpg *License*: unknown *Contributors*: A0387625
Image:host programming terminal output.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Host_programming_terminal_output.jpg *License*: unknown *Contributors*: A0387625