Application Report
SLAA754–August 2017

MSP430™ Spy-Bi-Wire With SimpleLink™ MCUs

Nima Eskandari *MSP430 Applications*

ABSTRACT

The embedded memory of MSP430™ microcontrollers (MCUs) can be programmed using the on-chip JTAG interface. The MSP430 MCUs support 2-wire JTAG interface. The 2-wire JTAG interface is referred to as Spy-Bi-Wire (SBW). Using Spy-Bi-Wire a host can access the programmable memory (flash memory), the data memory (RAM), and in FRAM devices, the nonvolatile FRAM memory. The host MCU can access the memory of the target MSP430 MCU during the prototyping phase, final production, and in service (field software updates).

This application report uses the SimpleLink™ MSP432P401R, CC3220, and CC2640R2F devices as the hosts for the SBW communication with the target MSP430 MCU. Both flash-based and FRAM-based MSP430 MCUs are used in this report to showcase the differences between the two device families, related to SBW communication. Software examples are provided for each of the SimpleLink devices mentioned. The software examples also make use of the SimpleLink Software Development Kit (SDK), making it easy to port to other SimpleLink devices.

The source code and other files described in this application report can be downloaded from http://www.ti.com/lit/zip/slaa754. The example source code demonstrates how a SimpleLink MCU can access the memory of a target MSP430 MCU (flash or FRAM based) through the SBW communication.

Contents

List of Figures

List of Tables

Trademarks

MSP430, SimpleLink, E2E, LaunchPad, Code Composer Studio are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Introduction

Spy-Bi-Wire (SBW) provides a method for the MSP430 MCUs to be programmed. For the MSP430 MCU to be programmed through SBW, commands must be sent to the dedicated pins of the target MSP430 MCU, using the SBW protocol.

To enable the SBW connection, an entry sequence must be applied to the dedicated SBW pins. After the SBW entry sequence is applied, commands can be sent to the target MSP430 MCU. The SBW commands are different depending on whether the MSP430 MCU is flash or FRAM based.

Software examples are provided for both flash and FRAM based MSP430 MCUs. These software examples use MSP432P401R, CC3220, and CC2640 as the SBW host to the target MSP430 MCU. The SBW protocol in these examples is implemented on top of the SimpleLink SDK, which makes it extremely easy to port to other SimpleLink devices.

1.1 Supplementary Online Information

For more information, visit Replicator for MSP430 MCU. This page contains links to additional SBW and JTAG user's guides and source code for using an MSP430F5437 to program other MSP430 MCUs.

For more information on JTAG and SBW communication, see MSP430™ Programming With the JTAG Interface, which describes how the JTAG and SBW interface can be used to program MSP430 MCUs. It also describes the commands used by SBW protocol.

Additional support is provided by the TI E2E™ Community.

2 Software Example

Example software is available for SimpleLink devices to act as the SBW host to target MSP430 MCUs.

The following LaunchPad™ development kits were used to develop the software examples:

- SimpleLink MSP432P401R LaunchPad Development Kit
- SimpleLink Wi-Fi® CC3220SF Wireless Microcontroller LaunchPad Development Kit
- SimpleLink Bluetooth® low energy CC2640R2F wireless MCU LaunchPad development kit

2.1 Software Example File Descriptions

The example software can be downloaded from http://www.ti.com/lit/zip/slaa754.

Table 1 describes the content of the top-level folder, and the following tables describe the content of the software examples.

Table 1. Top-Level Folder Description

Folder Name	Description
CC2640R2Host_SBW_MSP430	Example SimpleLink SDK project for CC2640R2. CC2640R2 acts as a SBW host for MSP430 flash-based devices. This example must be used with the CC2640R2F wireless MCU LaunchPad development kit. Note: This is a Code Composer Studio™ IDE project and must be imported into the IDE. This project depends on tirtos_builds_CC2640R2_LAUNCHXL_release_ccs, which also must be imported to the IDE.
CC2640R2Host_SBW_MSP430FR	Example SimpleLink SDK project for CC2640R2. CC2640R2 acts as a SBW host for MSP430 FRAM-based devices. This example must be used with the CC2640R2F wireless MCU LaunchPad development kit. Note: This is a Code Composer Studio IDE project and must be imported into the IDE. This project depends on tirtos_builds_CC2640R2_LAUNCHXL_release_ccs, which also must be imported to the IDE.
CC3220Host_SBW_MSP430	Example SimpleLink SDK project for CC3220. CC3220 acts as a SBW host for MSP430 flash-based devices. This example must be used with the CC3220 wireless MCU LaunchPad development kit. Note: This is a Code Composer Studio IDE project and must be imported into the IDE. This project depends on tirtos_builds_CC3220SF_LAUNCHXL_release_ccs, which also must be imported to the IDE.
CC3220Host_SBW_MSP430FR	Example SimpleLink SDK project for CC3220. CC3220 acts as a SBW host for MSP430 FRAM based devices. This example must be used with the CC3220 wireless MCU LaunchPad development kit. Note: This is a Code Composer Studio IDE project and must be imported into the IDE. This project depends on tirtos_builds_CC3220SF_LAUNCHXL_release_ccs, which also must be imported to the IDE.
MSP432Host_SBW_MSP430	Example SimpleLink SDK project for MSP432P401R. MSP432P401R acts as a SBW host for MSP430 flash-based devices. This example must be used with the MSP432P401R LaunchPad development kit. Note: This is a Code Composer Studio project and must be imported into the IDE. This project depends on tirtos_builds_MSP_EXP432P401R_release_ccs, which also must be imported to the IDE.
MSP432Host_SBW_MSP430FR	Example SimpleLink SDK project for MSP432P401R. MSP432P401R acts as a SBW host for MSP430 FRAM based devices. This example must be used with the MSP432P401R LaunchPad development kit. Note: This is a Code Composer Studio IDE project and must be imported into the IDE. This project depends on tirtos_builds_MSP_EXP432P401R_release_ccs, which also must be imported to the IDE.
tirtos_builds_CC2640R2_LAUNCHXL_release_ccs	TI RTOS project to be used with CC2640R2 example projects. This project is referenced by all CC2640R2 projects.

Table 1. Top-Level Folder Description (continued)

Folder Name	Description
tirtos_builds_CC3220SF_LAUNCHXL_release_ccs	TI RTOS project to be used with CC3220 example projects. This project is referenced by all CC3220 projects.
tirtos_builds_MSP_EXP432P401R_release_ccs	TI RTOS project to be used with MSP432 example projects. This project is referenced by all MSP432 projects.
Python_Scripts/FLASH_TI_txt_hex_to_image.py and Python_Scripts/FRAM_TI_txt_hex_to_image.py	Converts TI HEX TXT file to the header file format used by the host. These files are python scripts and must be run using python. Example: `python FLASH_TI_txt_hex_to_image.py`

The following tables describe the contents of each project.

Table 2. CC2640R2 Host Example Project Content

Name	Description
SBW (Directory)	Contains the SBW implementation, example firmware image and peripheral drivers.
Board.h	Contains board specific macros.
main_tirtos.c	Entry point for TI RTOS.
CC2640R2_LAUNCHXL.h	CC2640R2_LAUNCHXL board-specific APIs.
CC2640R2_LAUNCHXL.c	CC2640R2 specific for TI RTOS. Responsible for setting up the board specific items for the CC2640R2_LAUNCHXL board.
cc2640r2host_sbw_msp430.c or cc2640r2host_sbw_msp430fr.c	Driver Initialization. Starts the SBW programming with the example firmware image.

Table 3. CC3220 Host Example Project Content

Name	Description
SBW (Directory)	Contains the SBW implementation, example firmware image and peripheral drivers.
Board.h	Contains board specific macros.
main_tirtos.c	Entry point for TI RTOS.
CC3220SF_LAUNCHXL.h	CC3220SF_LAUNCHXL board specific APIs.
CC3220SF_LAUNCHXL.c	CC3220 specific for TI RTOS. Responsible for setting up the board specific items for the CC3220SF_LAUNCHXL board.
cc3220host_sbw_msp430.c or cc3220host_sbw_msp430fr.c	Driver Initialization. Starts the SBW programming with the example firmware image.

Table 4. MSP432 Host Example Project Content

Name	Description
SBW (Directory)	Contains the SBW implementation, example firmware image and peripheral drivers.
Board.h	Contains board specific macros.
main_tirtos.c	Entry point for TI RTOS.
MSP_EXP432P401R.h	MSP_EXP432P401R board specific APIs.
MSP_EXP432P401R.c	MSP432 specific for TI RTOS. Responsible for setting up the board specific items for the MSP_EXP432P401R board.
msp432host_sbw_msp430.c or msp432host_sbw_msp430fr.c	Driver Initialization. Starts the SBW programming with the example firmware image.

The SBW folder in each project contains the drivers for the peripherals used by the SimpleLink host. This folder also contains the SBW commands and an example firmware image for the target MSP430 MCU.

Table 5 describes the content of the SBW folder.

Table 5. SBW Folder Content

Name	Description
image/msp430_image.h or image/msp430fr_image.h	Contains the example firmware image for flash or FRAM target MSP430 MCU. The msp430_image.h contains the blinky example for MSP430G2553 device, while the msp430fr_image.h contains the blinky example for MSP430FR2311 device.
SBW430.c or SBW430FR.c	Implementation of the SBW commands. This file contains the commands used to program an MSP430 MCU through Spy-Bi-Wire.
SBW430.h or SBW430FR.h	Contains the function declarations for the SBW commands.
sbw_main.c	Programs the target device with the specified example firmware image. It resets the device and allows it to run after reset.
sbw_main.h	Contains the function declarations for the sbw_main.c
config.h	Contains configurable options such as number of time to retry when a time-out occurs and whether or not to output software progress to the serial terminal.
debug.c	Initializes the backchannel UART for communication with PC (Serial terminal programs such as Putty)
debug.h	Contains the function declarations for the debug.c
gpio_if.c	Device specific interface file to access the GPIOs. The GPIOs are used to control the dedicated SBW entry sequence pins and send SBW commands.
gpio_if.h	Contains the function declarations for the gpio_if.c
uart_if.c	Interfaces with the UART module on the device to send and receive data through UART protocol. (This file is not used in the example project)
uart_if.h	Contains the function declarations for the uart_if.c
utils.c	Interfaces with the timer modules on the device to generate specified delays. Also contain other utilities for debugging.
utils.h	Contains the function declarations for the utils.c

3 Spy-Bi-Wire (SBW) Connections

The SimpleLink host and target MSP430 MCU must be connected to through the SBW pins and share the GND signal. The SBW pins include the two dedicated pins used for 2-wire JTAG interface.

3.1 *Spy-Bi-Wire Connections for the Target MSP430 MCU*

To access the MSP430 memory through the SBW interface, an entry sequence must be applied to the dedicated pins. In this report, the MSP430G2553 and MSP430FR2311 are used as the target for SBW communication.

The SBW entry sequence must be applied to the TST and RST pins on the target MSP430 MCU to invoke the SBW communication. For more information on the SBW entry sequence, see MSP430™ Programming With the JTAG Interface.

Figure 1 and Figure 2 show the dedicated pins for SBW on the MSP430G2553 and the MSP430FR2311, respectively.

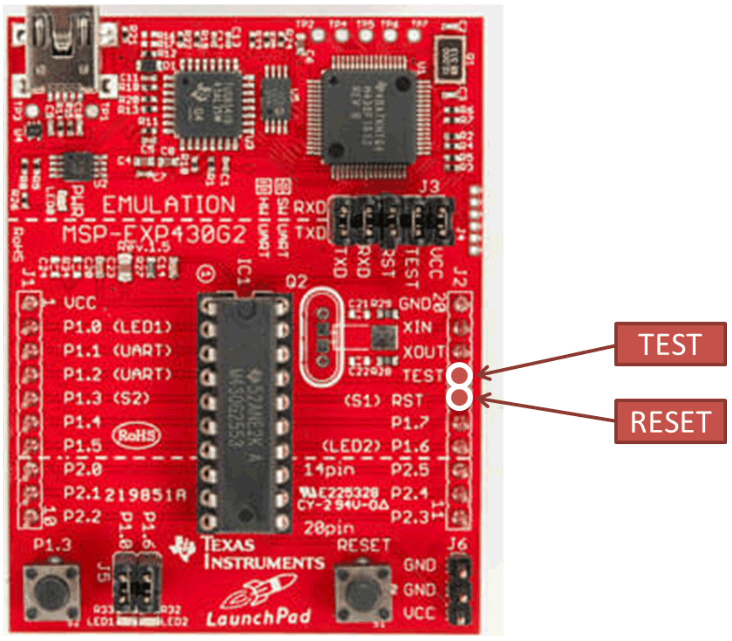


Figure 1. MSP430G2553 SBW Pins

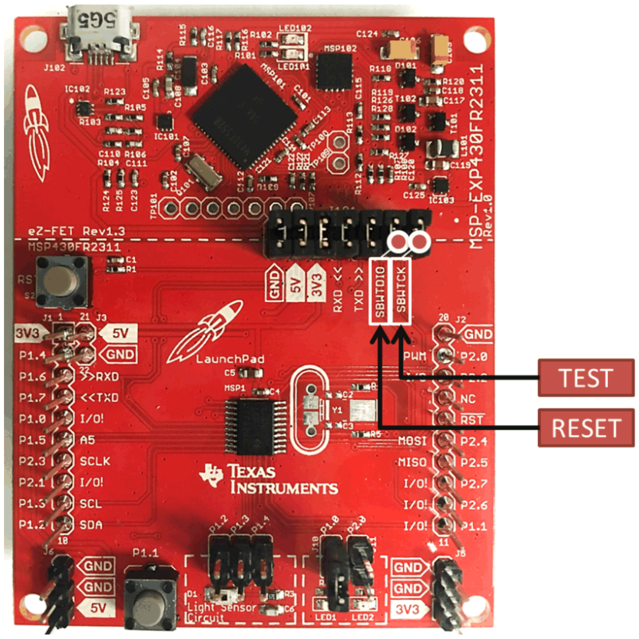


Figure 2. MSP430FR2311 SBW Pins

These pins are connected to the GPIO of the SimpleLink host. After the SBW entry sequence is completed, the host can use these same pins to send and receive SBW commands.

The pins for SBW interface are Reset and Test pins. Some LaunchPad development kits specify these two pins as SBWTDIO and SBWTCK.

- SBWTDIO: Spy-By-Wire Data Input/Output (RESET pin)
- SBWTCK: Spy-By-Wire Clock (TEST pin)

The final connections needed on the target MSP430 MCU are the power connections. All power pins must be connected to the required voltages. In this case, both MSP430FR2311 and MSP430G2553 are connected to a 3.3-V supply through the VCC and GND pins.

3.2 Spy-Bi-Wire (SBW) Connections for the Host SimpleLink Device

The host device uses two GPIO pins to communicate through the SBW interface to the target MSP430 MCU. The host also uses one UART module. The UART module communicates the status of code execution to the PC through the backchannel serial port. These pins are defined in the board-specific file of every project (MSP_EXP432P401R.c, CC3220SF_LAUNCHXL.c, and CC2640R2_LAUNCHXL.c).

The software example provided for the supported LaunchPad development kits uses the following pins on each board:

MSP432P401R LaunchPad development kit:

- Target Reset Pin: P6.0
- Target Test Pin: P6.1
- Backchannel UART to PC:
  - TX: P1.3
  - RX: P1.2

CC3220SF wireless microcontroller LaunchPad development kit:

- Target Reset Pin: PIN 03 (GPIO 12)
- Target Test Pin: PIN 04 (GPIO 13)
- Backchannel UART to PC:
  - TX: PIN 57
  - RX: PIN 55

CC2640R2 wireless microcontroller LaunchPad development kit:

- Target Reset Pin: DIO23
- Target Test Pin: DIO25
- Backchannel UART to PC:
  - TX: DIO03
  - RX: DIO02

Figure 3, Figure 4, and Figure 5 show each LaunchPad development kit and the pins used for the software example.

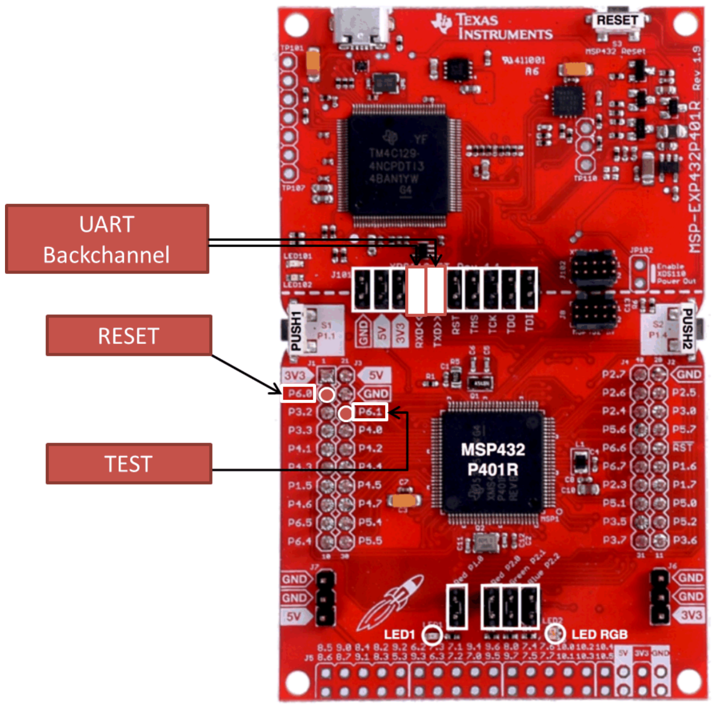


Figure 3. MSP432P401R LaunchPad Development Kit SBW Host Pins

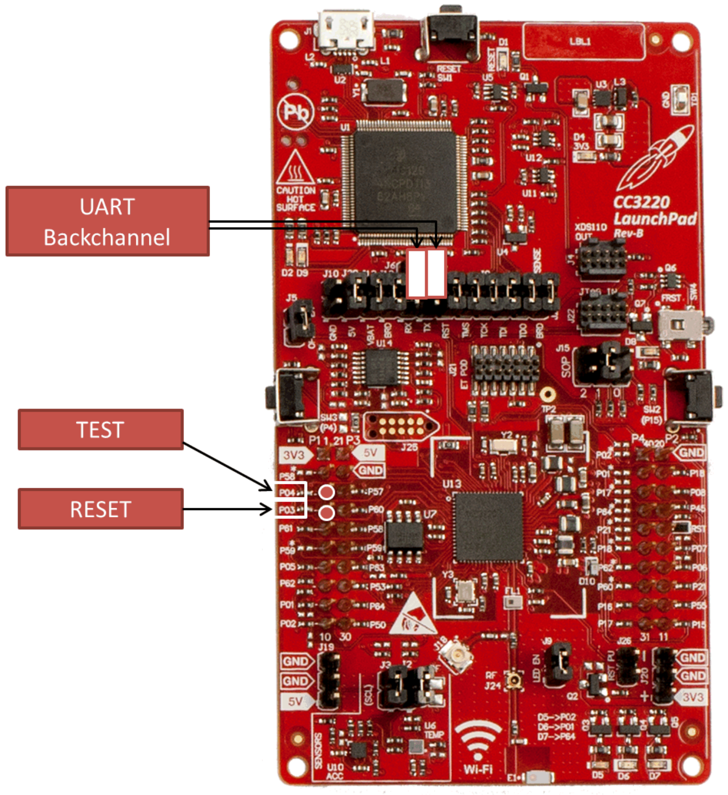


Figure 4. CC3220SF LaunchPad Development Kit SBW Host Pins

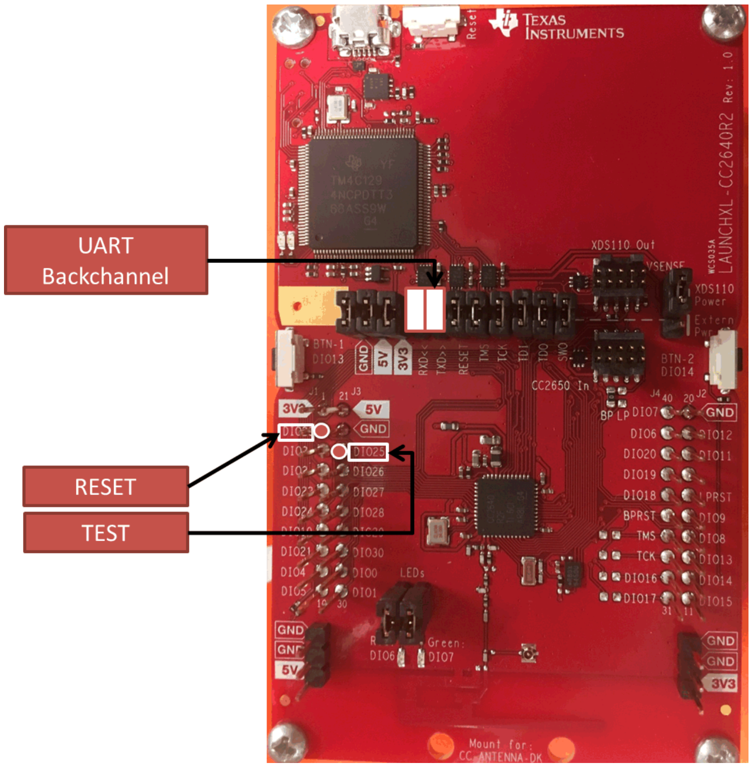


Figure 5. CC2640R2 LaunchPad Development Kit SBW Host Pins

The SimpleLink Host connects to the MSP430 MCU through the Reset (SBW data input/output pin) and Test (SBW clock pin) pins. The SimpleLink host and MSP430 MCU must share the ground signal. Also, the MSP430 MCU must be powered up. Finally, through the backchannel UART, a PC can be used to view the status of the MSP430 firmware update.

4 How to Use the Software Examples

The software examples in this application report used the following SimpleLink SDKs:

- SimpleLink CC32xx SDK v:1.40.00.03
- SimpleLink MSP432 SDK v:1.40.01.00
- SimpleLink CC2640R2 SDK v:1.35.00.33

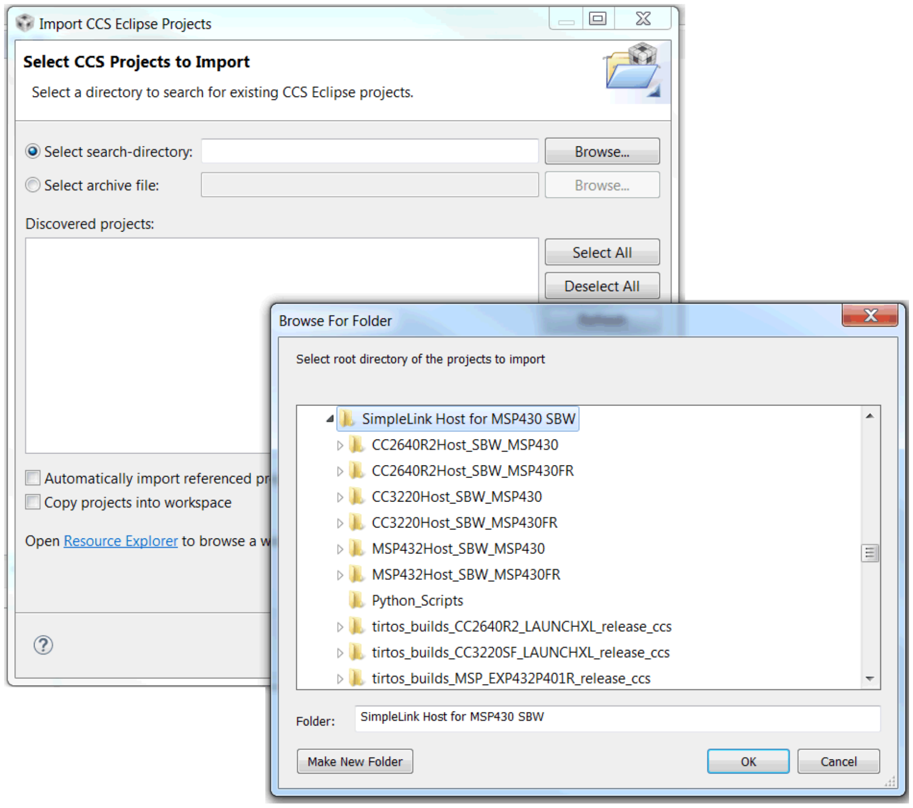
Code Composer Studio (CCS) IDE Version 7.1.0.00016 was used to compile and debug the projects.

Download and extract the zip file containing the software examples.

4.1 Import the Projects to CCS

To import the projects, click Project > Import CCS Projects.

Select Browse and navigate to the extracted file from the software example zip file.


Import CCS Eclipse Projects

Select CCS Projects to Import

Select a directory to search for existing CCS Eclipse projects.

Select search-directory: Browse...

Select archive file: Browse...

Discovered projects:

Select All

Deselect All

Automatically import referenced pr

Copy projects into workspace

Open Resource Explorer to browse a w

Browse For Folder

Select root directory of the projects to import

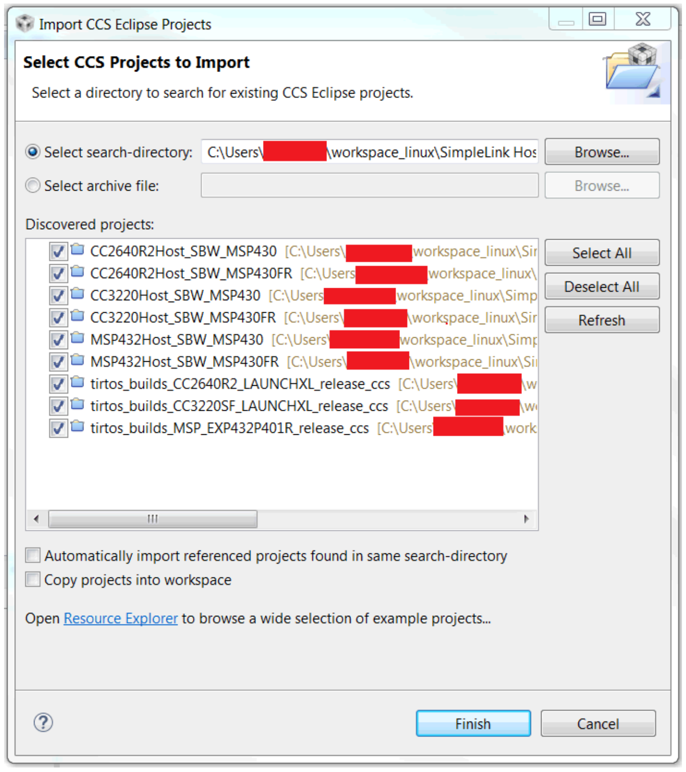
SimpleLink Host for MSP430 SBW
CC2640R2Host_SBW_MSP430
CC2640R2Host_SBW_MSP430FR
CC3220Host_SBW_MSP430
CC3220Host_SBW_MSP430FR
MSP432Host_SBW_MSP430
MSP432Host_SBW_MSP430FR
Python_Scripts
tirtos_builds_CC2640R2_LAUNCHXL_release_ccs
tirtos_builds_CC3220SF_LAUNCHXL_release_ccs
tirtos_builds_MSP_EXP432P401R_release_ccs

Folder: SimpleLink Host for MSP430 SBW

Make New Folder OK Cancel

Figure 6. Import Software Example Projects to CCS

CCS automatically finds all of the projects inside the folder. Select all projects including the TI RTOS build projects and import them to your workspace.


Import CCS Eclipse Projects

Select CCS Projects to Import

Select a directory to search for existing CCS Eclipse projects.

(●) Select search-directory: C:\Users\[redacted]\workspace_linux\SimpleLink Hos [Browse...]

( ) Select archive file: [Browse...]

Discovered projects:

- ✓ CC2640R2Host_SBW_MSP430 [C:\Users\[redacted]workspace_linux\Sir
- ✓ CC2640R2Host_SBW_MSP430FR [C:\Users[redacted]workspace_linux\
- ✓ CC3220Host_SBW_MSP430 [C:\Users[redacted]workspace_linux\Simp
- ✓ CC3220Host_SBW_MSP430FR [C:\Users\[redacted]\workspace_linux\Sir
- ✓ MSP432Host_SBW_MSP430 [C:\Users\[redacted]workspace_linux\Simp
- ✓ MSP432Host_SBW_MSP430FR [C:\Users\[redacted]\workspace_linux\Si
- ✓ tirtos_builds_CC2640R2_LAUNCHXL_release_ccs [C:\Users\[redacted]\w
- ✓ tirtos_builds_CC3220SF_LAUNCHXL_release_ccs [C:\Users\[redacted]\w
- ✓ tirtos_builds_MSP_EXP432P401R_release_ccs [C:\Users[redacted]work

[Select All] [Deselect All] [Refresh]

☐ Automatically import referenced projects found in same search-directory
☐ Copy projects into workspace

Open Resource Explorer to browse a wide selection of example projects...

[Finish] [Cancel]

Figure 7. Import All Projects to CCS

After importing, the projects must be built.

Select all of the imported projects, except for the TI RTOS build projects. Right click and press Rebuild Project.

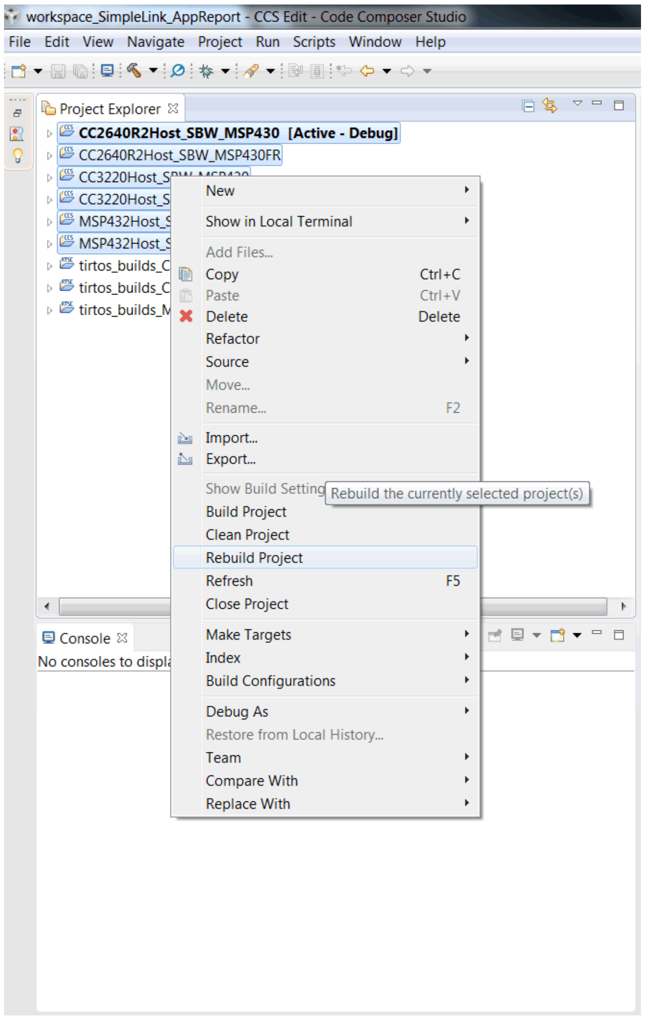
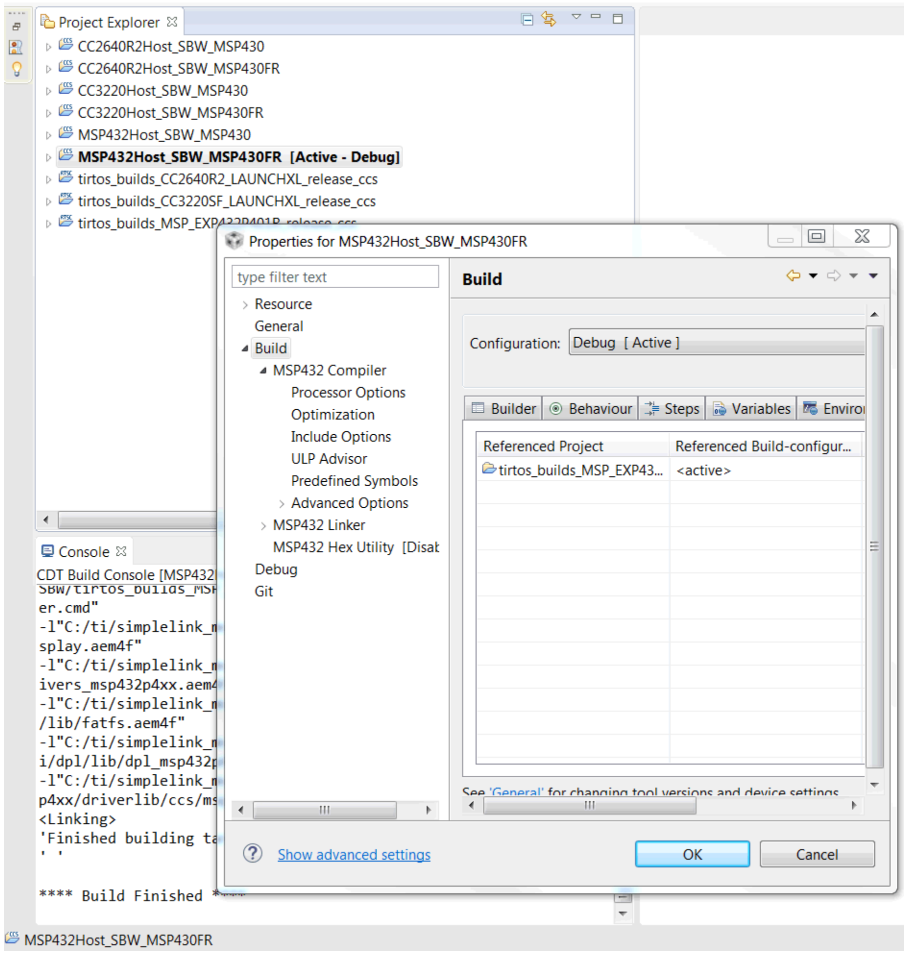


Figure 8. Building Software Example Projects

The tirtos_builds_CC2640R2_LAUNCHXL_release_ccs, tirtos_builds_CC3220SF_LAUNCHXL_release_ccs, and tirtos_builds_MSP_EXP432P401R_release_ccs will be rebuilt automatically. These projects are referenced by the other projects with the same platform.


Project Explorer
CC2640R2Host_SBW_MSP430
CC2640R2Host_SBW_MSP430FR
CC3220Host_SBW_MSP430
CC3220Host_SBW_MSP430FR
MSP432Host_SBW_MSP430
MSP432Host_SBW_MSP430FR [Active - Debug]
tirtos_builds_CC2640R2_LAUNCHXL_release_ccs
tirtos_builds_CC3220SF_LAUNCHXL_release_ccs
tirtos_builds_MSP_EXP432P401R_release_ccs

Properties for MSP432Host_SBW_MSP430FR

Build

Configuration: Debug [Active]

Referenced Project	Referenced Build-configur...
tirtos_builds_MSP_EXP43...	<active>

See 'General' for changing tool versions and device settings

```
CDT Build Console [MSP432
SBW/tirtos_builds_MSP
er.cmd"
-l"C:/ti/simplelink_m
splay.aem4f"
-l"C:/ti/simplelink_m
ivers_msp432p4xx.aem4
-l"C:/ti/simplelink_m
/lib/fatfs.aem4f"
-l"C:/ti/simplelink_m
i/dpl/lib/dpl_msp432p
-l"C:/ti/simplelink_m
p4xx/driverlib/ccs/ms
<Linking>
'Finished building ta
' '

**** Build Finished ****
```

Figure 9. TI RTOS Build Project Dependency

4.2 Run the Software Examples

Connect the SimpleLink host to the MSP430 target. For the first part of this demo, the MSP432P401R LaunchPad and MSP430G2553 are used. As for the second part, MSP430FR2311 is used as the target.

4.2.1 MSP430G2553 SBW Target Example

Connect the two devices as shown in the following figures.

The pin-to-pin connection is also available in the README file.

Figure 10, Figure 11, and Figure 12 show the connections for the MSP432P401R and MSP430G2553 LaunchPad development kits.

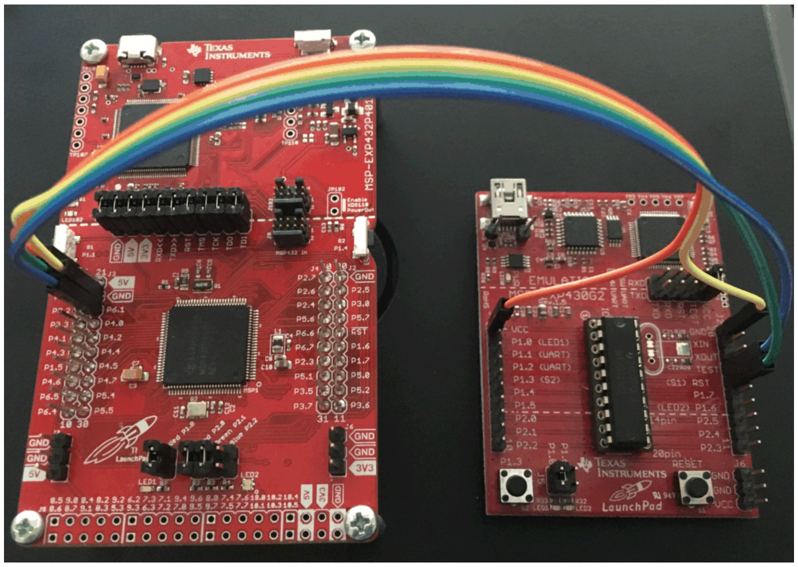


Figure 10. MSP432P401R LaunchPad Development Kit Connected to MSP430G2553

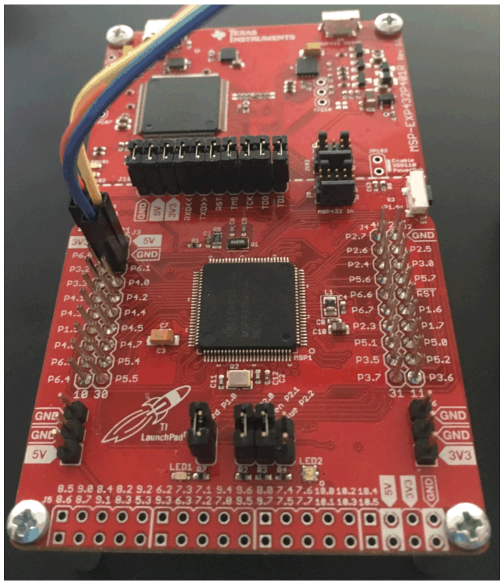


Figure 11. MSP432P401R LaunchPad Development Kit SBW Host Connections

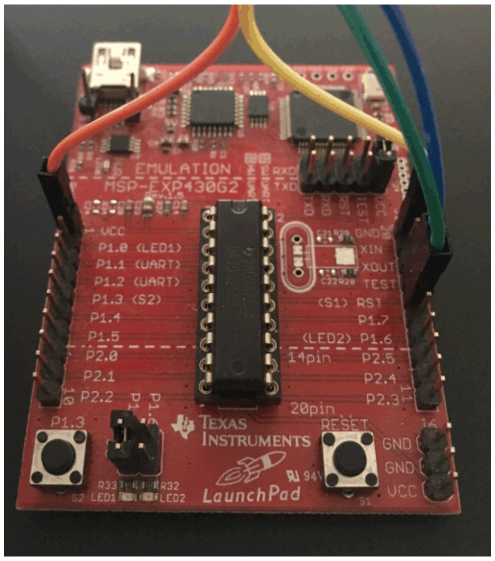
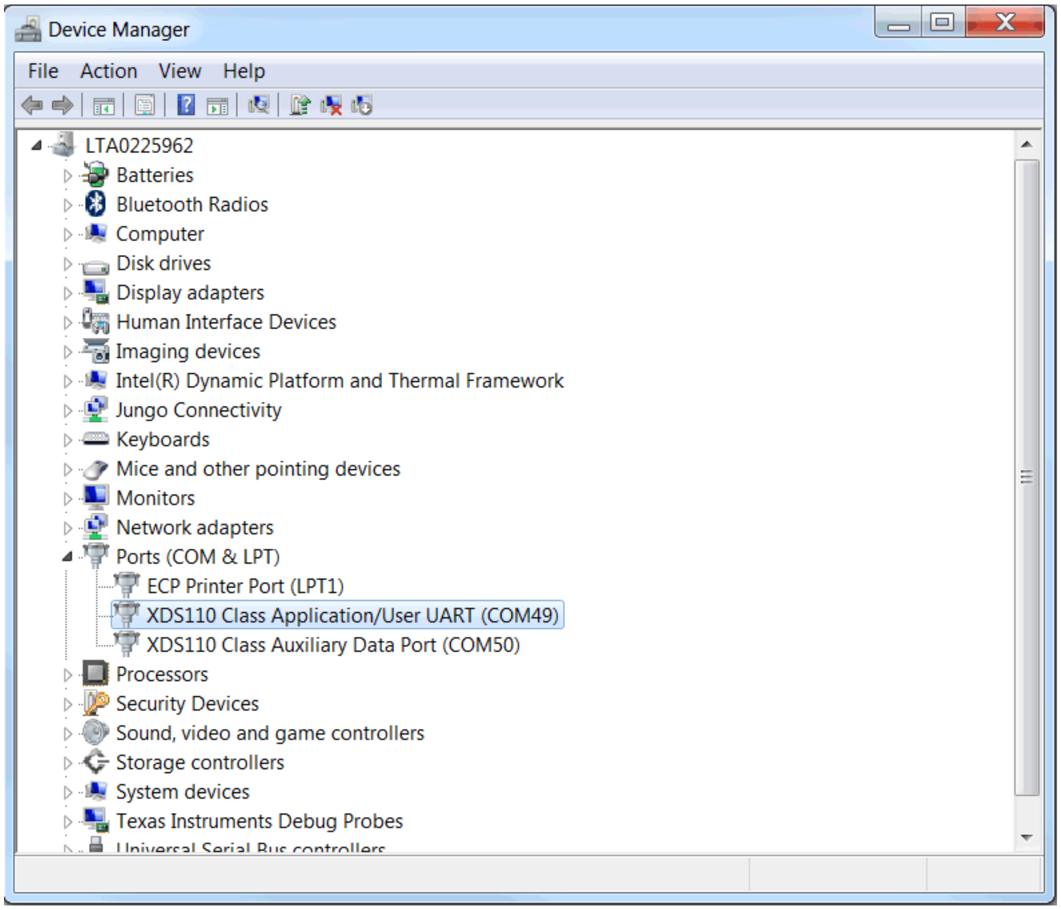


Figure 12. MSP430G2553 LaunchPad Development Kit SBW Connections

In this setup, the MSP430G2553 is powered from the MSP432P401R LaunchPad development kit. The MSP432P401R LaunchPad development kit is powered through the USB connection to PC.

After connecting the MSP432 LaunchPad development kit to the PC, the COM port number for serial communication to the board can be found in Windows Device Manager.



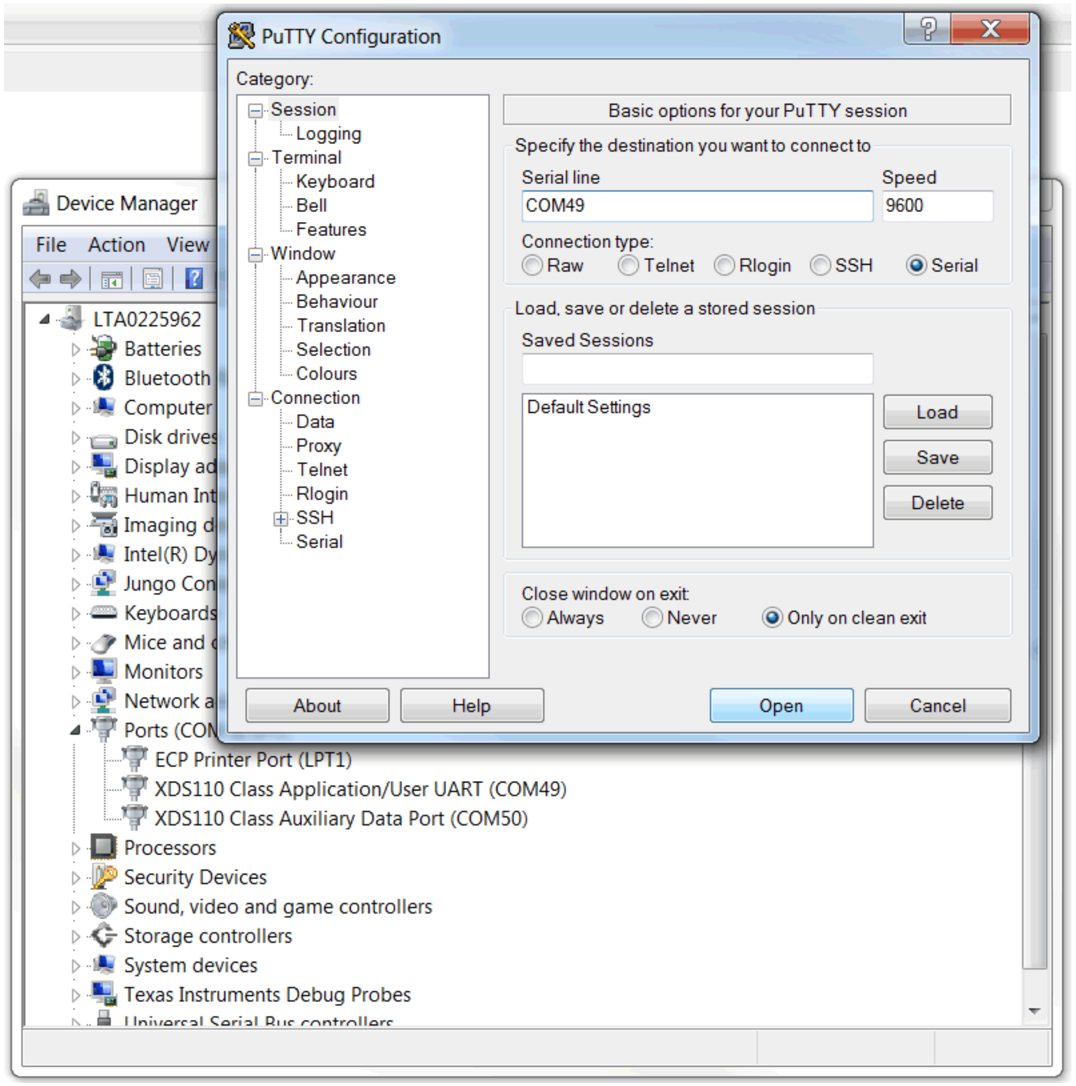
LTA0225962
- Batteries
- Bluetooth Radios
- Computer
- Disk drives
- Display adapters
- Human Interface Devices
- Imaging devices
- Intel(R) Dynamic Platform and Thermal Framework
- Jungo Connectivity
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - ECP Printer Port (LPT1)
  - XDS110 Class Application/User UART (COM49)
  - XDS110 Class Auxiliary Data Port (COM50)
- Processors
- Security Devices
- Sound, video and game controllers
- Storage controllers
- System devices
- Texas Instruments Debug Probes
- Universal Serial Bus controllers

Figure 13. Application UART COM Port Number in Windows Device Manager

A serial communication terminal program such as PuTTY can be used to view the execution of the example software.

Using PuTTY, open a serial communication with the following specification to the COM port found in the Windows Device Manager (see Figure 14).

- Baud rate: 9600
- Data bits: 8
- Stop bits: 1
- Parity: None



Specify the destination you want to connect to

Serial line: COM49

Speed: 9600

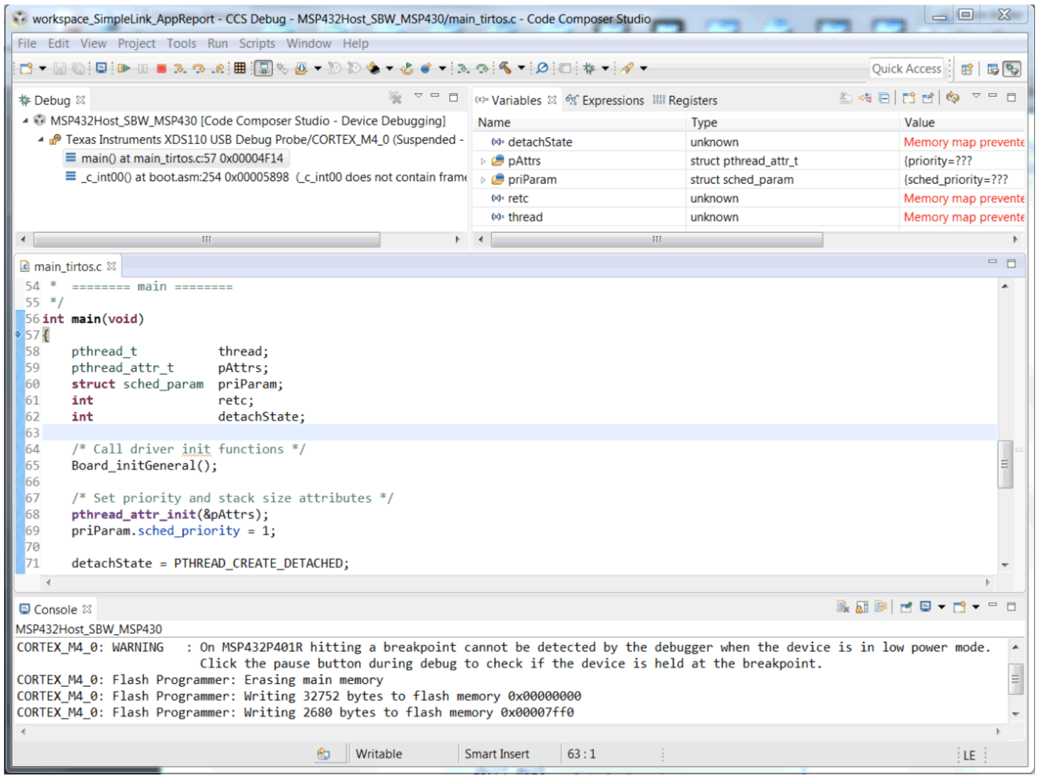
Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Close window on exit: Always Never [✓ Only on clean exit]

Figure 14. PuTTY Serial Communication Settings

Next, debug the project MSP432Host_SBW_MSP430 by clicking Run > Debug.

After the binary firmware image is uploaded to MSP432P401R, run the example by clicking the green play button.



workspace_SimpleLink_AppReport - CCS Debug - MSP432Host_SBW_MSP430/main_tirtos.c - Code Composer Studio

Name	Type	Value
detachState	unknown	Memory map prevente
pAttrs	struct pthread_attr_t	{priority=???
priParam	struct sched_param	{sched_priority=???
retc	unknown	Memory map prevente
thread	unknown	Memory map prevente

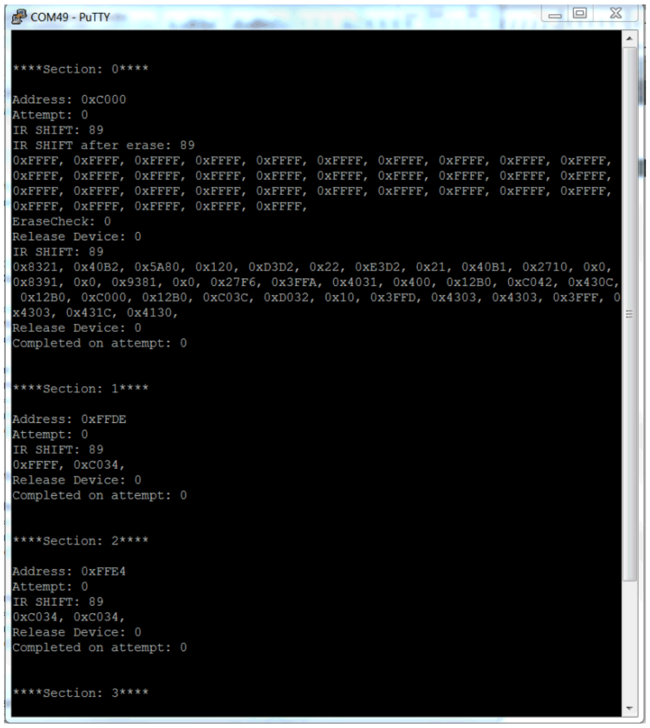
```
MSP432Host_SBW_MSP430 [Code Composer Studio - Device Debugging]
  Texas Instruments XDS110 USB Debug Probe/CORTEX_M4_0 (Suspended -
    main() at main_tirtos.c:57 0x00004F14
    _c_int00() at boot.asm:254 0x00005898  (_c_int00 does not contain frame
```

```
 *  ======== main ========
 */
int main(void)
{
    pthread_t           thread;
    pthread_attr_t      pAttrs;
    struct sched_param  priParam;
    int                 retc;
    int                 detachState;

    /* Call driver init functions */
    Board_initGeneral();

    /* Set priority and stack size attributes */
    pthread_attr_init(&pAttrs);
    priParam.sched_priority = 1;

    detachState = PTHREAD_CREATE_DETACHED;
```

```
MSP432Host_SBW_MSP430
CORTEX_M4_0: WARNING   : On MSP432P401R hitting a breakpoint cannot be detected by the debugger when the device is in low power mode.
                         Click the pause button during debug to check if the device is held at the breakpoint.
CORTEX_M4_0: Flash Programmer: Erasing main memory
CORTEX_M4_0: Flash Programmer: Writing 32752 bytes to flash memory 0x00000000
CORTEX_M4_0: Flash Programmer: Writing 2680 bytes to flash memory 0x00007ff0
```

Figure 15. MSP432Host_SBW_MSP430 Project Debug Mode

The next step is to view the serial output on PuTTY (see Figure 16).

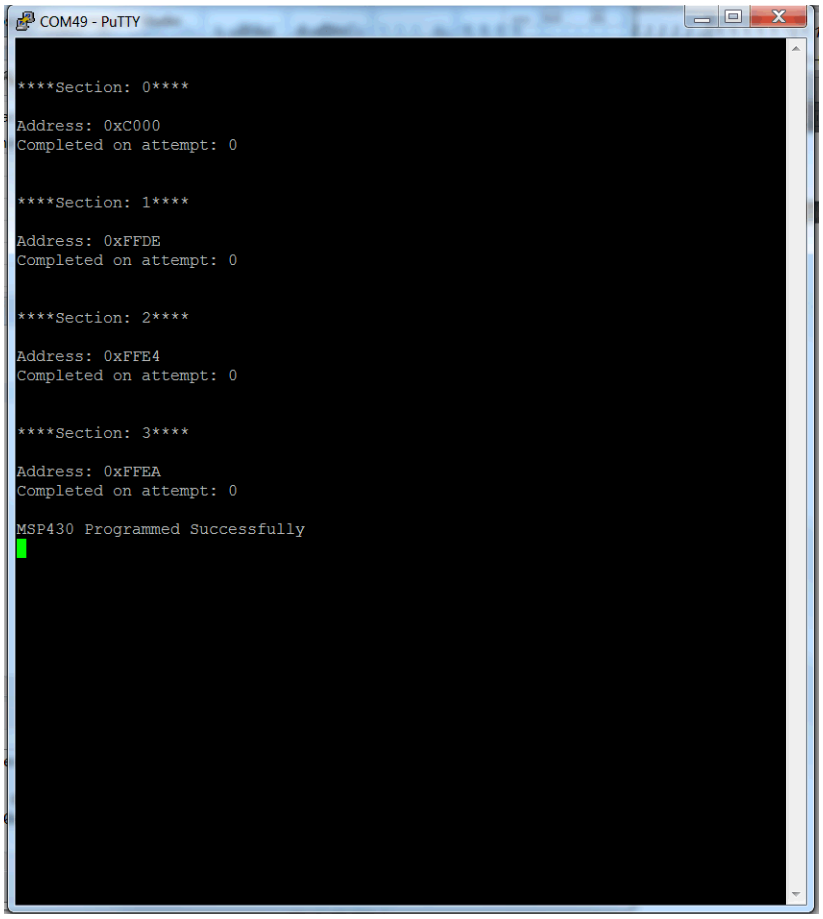


COM49 - PuTTY

```

****Section: 0****

Address: 0xC000
Attempt: 0
IR SHIFT: 89
IR SHIFT after erase: 89
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
EraseCheck: 0
Release Device: 0
IR SHIFT: 89
0x8321, 0x40B2, 0x5A80, 0x120, 0xD3D2, 0x22, 0xE3D2, 0x21, 0x40B1, 0x2710, 0x0,
0x8391, 0x0, 0x9381, 0x0, 0x27F6, 0x3FFA, 0x4031, 0x400, 0x12B0, 0xC042, 0x430C,
 0x12B0, 0xC000, 0x12B0, 0xC03C, 0xD032, 0x10, 0x3FFD, 0x4303, 0x4303, 0x3FFF, 0
x4303, 0x431C, 0x4130,
Release Device: 0
Completed on attempt: 0


****Section: 1****

Address: 0xFFDE
Attempt: 0
IR SHIFT: 89
0xFFFF, 0xC034,
Release Device: 0
Completed on attempt: 0


****Section: 2****

Address: 0xFFE4
Attempt: 0
IR SHIFT: 89
0xC034, 0xC034,
Release Device: 0
Completed on attempt: 0


****Section: 3****
```

Figure 16. MSP432Host_SBW_MSP430 Project Serial Output (DEBUG is 1)

If the DEBUG macro is defined as 1 in the config.h file, the serial output looks similar to Figure 16. In this case, the execution of the code can be viewed in detail. If the DEBUG macro is defined as 0, the serial output looks similar to Figure 17.

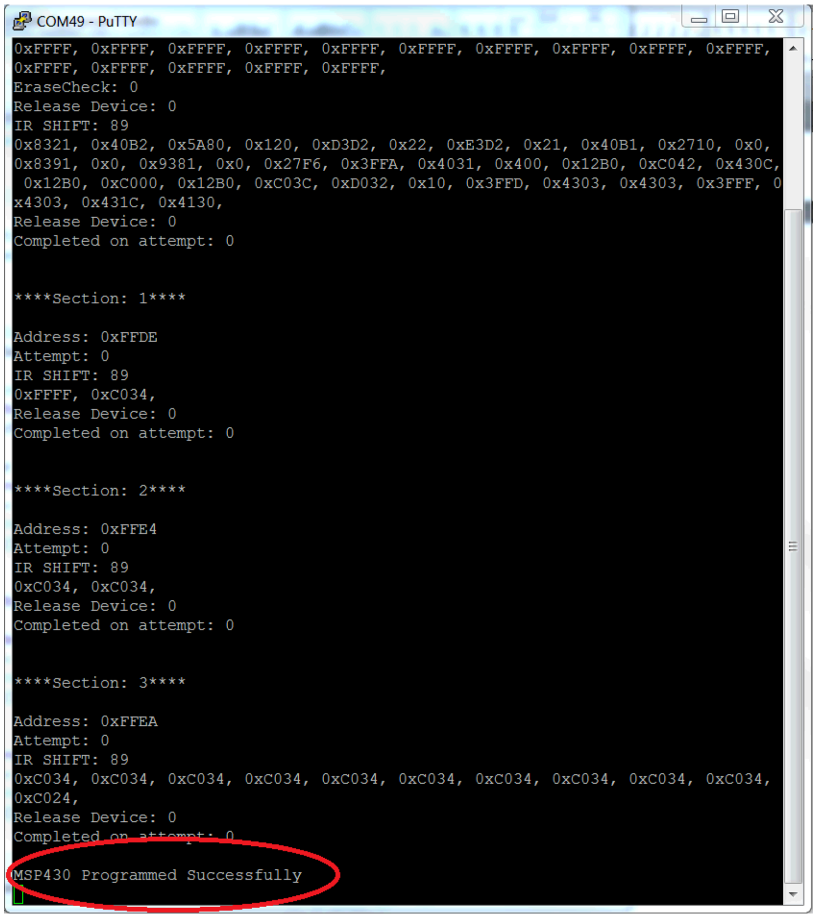


```
****Section: 0****

Address: 0xC000
Completed on attempt: 0


****Section: 1****

Address: 0xFFDE
Completed on attempt: 0


****Section: 2****

Address: 0xFFE4
Completed on attempt: 0


****Section: 3****

Address: 0xFFEA
Completed on attempt: 0

MSP430 Programmed Successfully
```

Figure 17. MSP432Host_SBW_MSP430 Project Serial Output (DEBUG is 0)

Finally the MSP430 MCU is programmed and the message, “MSP430 Programmed Successfully” is shown on the serial terminal.

The example program that is downloaded to the target MSP430 toggles P1.0, which can be seen by the toggling of the LED on the MSP430G2553 LaunchPad development kit.


COM49 - PuTTY

```
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
EraseCheck: 0
Release Device: 0
IR SHIFT: 89
0x8321, 0x40B2, 0x5A80, 0x120, 0xD3D2, 0x22, 0xE3D2, 0x21, 0x40B1, 0x2710, 0x0,
0x8391, 0x0, 0x9381, 0x0, 0x27F6, 0x3FFA, 0x4031, 0x400, 0x12B0, 0xC042, 0x430C,
 0x12B0, 0xC000, 0x12B0, 0xC03C, 0xD032, 0x10, 0x3FFD, 0x4303, 0x4303, 0x3FFF, 0
x4303, 0x431C, 0x4130,
Release Device: 0
Completed on attempt: 0


****Section: 1****

Address: 0xFFDE
Attempt: 0
IR SHIFT: 89
0xFFFF, 0xC034,
Release Device: 0
Completed on attempt: 0


****Section: 2****

Address: 0xFFE4
Attempt: 0
IR SHIFT: 89
0xC034, 0xC034,
Release Device: 0
Completed on attempt: 0


****Section: 3****

Address: 0xFFEA
Attempt: 0
IR SHIFT: 89
0xC034, 0xC034, 0xC034, 0xC034, 0xC034, 0xC034, 0xC034, 0xC034, 0xC034, 0xC034,
0xC024,
Release Device: 0
Completed on attempt: 0

MSP430 Programmed Successfully
```

Figure 18. MSP430G2553 Programmed Successfully

The same steps must be followed to program any flash-based MSP430 MCU using a CC2640R2 or CC3220 device.

4.2.2 MSP430FR2311 SBW Target Example

To program an MSP430FR2311 using the MSP432P401R through SBW, the same steps as in Section 4.2.1 must be followed.

Figure 19 shows the connections between the MSP430FR2311 and the MSP432P401R

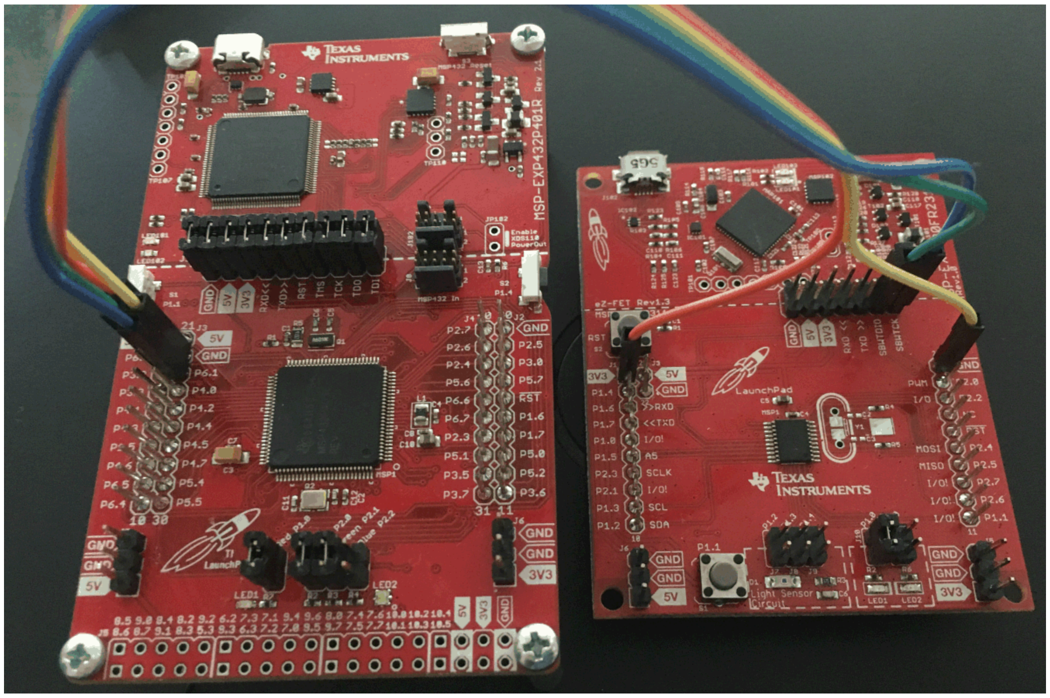


Figure 19. MSP432P401R LaunchPad Development Kit Connected to MSP430FR2311

The MSP432P401R connections are the same as in Section 4.2.1.

Figure 20 shows the MSP430FR2311 connections.

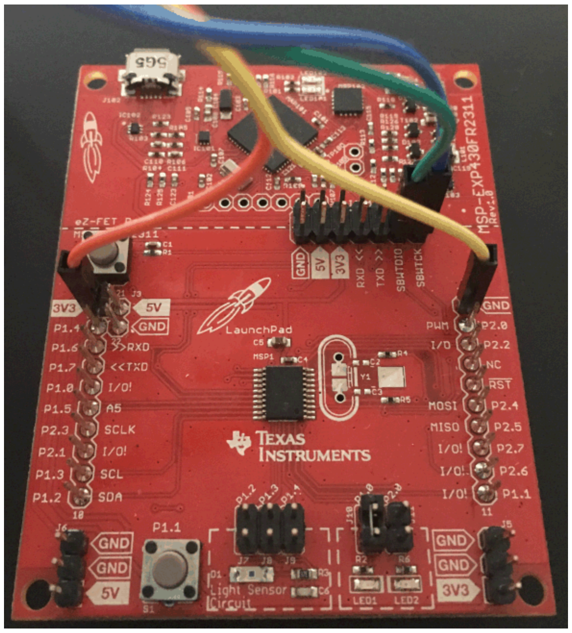
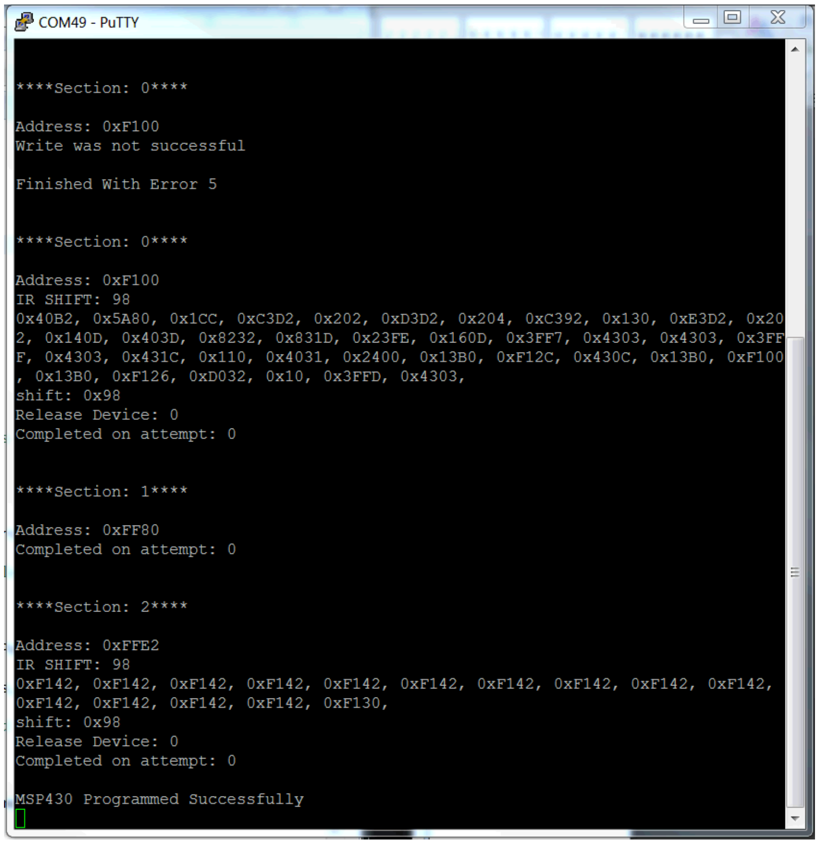


Figure 20. MSP430FR2311 LaunchPad Development Kit SBW Connections

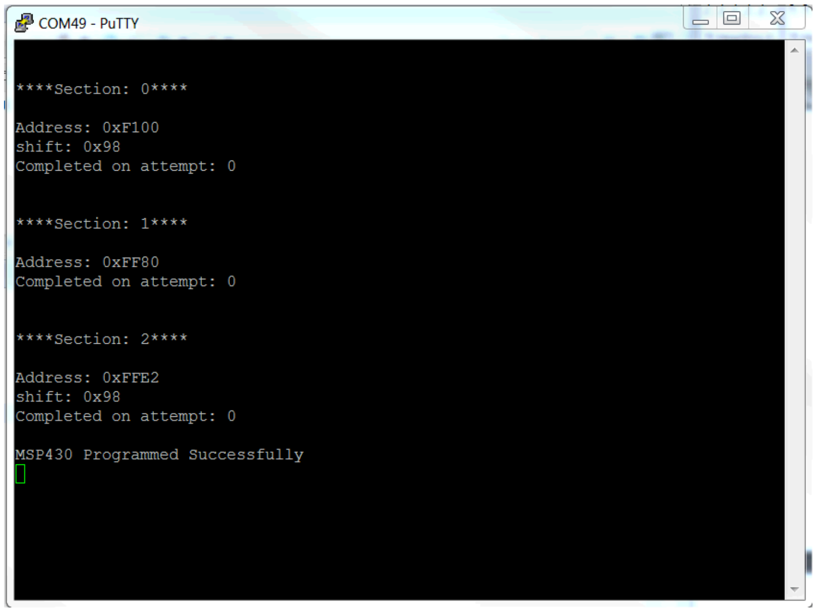
After the connections are made, the example project MSP432Host_SBW_MSP430FR can be debugged and the result can be viewed using PuTTY (see Figure 21).


COM49 - PuTTY

```
****Section: 0****

Address: 0xF100
Write was not successful

Finished With Error 5


****Section: 0****

Address: 0xF100
IR SHIFT: 98
0x40B2, 0x5A80, 0x1CC, 0xC3D2, 0x202, 0xD3D2, 0x204, 0xC392, 0x130, 0xE3D2, 0x20
2, 0x140D, 0x403D, 0x8232, 0x831D, 0x23FE, 0x160D, 0x3FF7, 0x4303, 0x4303, 0x3FF
F, 0x4303, 0x431C, 0x110, 0x4031, 0x2400, 0x13B0, 0xF12C, 0x430C, 0x13B0, 0xF100
, 0x13B0, 0xF126, 0xD032, 0x10, 0x3FFD, 0x4303,
shift: 0x98
Release Device: 0
Completed on attempt: 0


****Section: 1****

Address: 0xFF80
Completed on attempt: 0


****Section: 2****

Address: 0xFFE2
IR SHIFT: 98
0xF142, 0xF142, 0xF142, 0xF142, 0xF142, 0xF142, 0xF142, 0xF142, 0xF142, 0xF142,
0xF142, 0xF142, 0xF142, 0xF142, 0xF130,
shift: 0x98
Release Device: 0
Completed on attempt: 0

MSP430 Programmed Successfully
```

Figure 21. MSP432Host_SBW _MSP430FR Serial Output (DEBUG is 1)

After running the example, the following result is seen in the serial terminal (see Figure 22).


COM49 - PuTTY

```
****Section: 0****

Address: 0xF100
shift: 0x98
Completed on attempt: 0


****Section: 1****

Address: 0xFF80
Completed on attempt: 0


****Section: 2****

Address: 0xFFE2
shift: 0x98
Completed on attempt: 0

MSP430 Programmed Successfully
```

Figure 22. MSP430FR2311 Programmed Successfully

The same steps must be followed to program any FRAM-based MSP430 MCU using a CC2640R2 or CC3220 device.

The default firmware image in all MSP430 flash-based examples is the MSP430G2553 blinky firmware image. This program toggles P1.0 on and off.

The default firmware image in all MSP430 FRAM-based examples is the MSP430FR2311 blinky firmware image. This program toggles P1.0 on and off.

To create a new firmware image for any MSP430 MCU from an existing code composer studio project, see Section 5.

5 Create Custom MSP430 Firmware Image

The software example provides a default example firmware image, which is inside the msp430_image.h or msp430fr_image.h header file (based on whether the target MSP430 MCU is flash or FRAM based).

The following sections describe the format of the firmware image header file.

5.1 MSP430 Flash Firmware Image

The default example firmware image for the flash-based MSP430 MCUs is the blinky example for MSP430G2553.

```
/*
* This file was automatically generated.
* The memory sections should be quickly double checked.
* Some sections such ad Start, Finish, Termination and Length must be
* modified based on device datasheet. (These values aren't used by the
* default program.
* Created by: Nima Eskandari
*/

const uint16_t eprom[] =
{
//0xc000
0x8321, 0x40B2, 0x5A80, 0x0120, 0xD3D2, 0x0022, 0xE3D2, 0x0021,
0x40B1, 0xC350, 0x0000, 0x8391, 0x0000, 0x9381, 0x0000, 0x27F6,
0x3FFA, 0x4031, 0x0400, 0x12B0, 0xC042, 0x430C, 0x12B0, 0xC000,
0x12B0, 0xC03C, 0xD032, 0x0010, 0x3FFD, 0x4303, 0x4303, 0x3FFF,
0x4303, 0x431C, 0x4130,
//0xffde
0xFFFF, 0xC034,
//0xffe4
0xC034, 0xC034,
//0xffea
0xC034, 0xC034, 0xC034, 0xC034, 0xC034, 0xC034, 0xC034, 0xC034,
0xC034, 0xC034, 0xC022,

};


const uint32_t eprom_address[] =
{
0xc000, 0xffde, 0xffe4, 0xffea,
};
```

```
const uint32_t eprom_length_of_sections[] =
{
35, 2, 2, 11,
};


const uint32_t eprom_sections    = 4;

const uint32_t eprom_termination = 0x00000000; /*Check device data sheet*/
const uint32_t eprom_start       = 0x00000000; /*Check device data sheet*/
const uint32_t eprom_finish      = 0x00000000; /*Check device data sheet*/
const uint32_t eprom_length      = 0x00000000; /*Check device data sheet*/
```

The first variable is flash. The flash_sections variable, hold the number of start addresses that is required to be programmed. In this case the flash_sections variable is set to 4. This is consistent with the size of flash_address array. The flash_address holds 4 addresses. These addresses are 0xC000, 0xFFDE, 0xFFE4, and 0xFFEA. The flash_length_of_sections array, specifies the number of bytes in the flash variable for each of the addresses specified in flash_address array.

The example default image is interpreted as:

- For address 0xC000

  The first 35 16-bit words (70 bytes) of data in flash must be written to the MSP430 MCU, starting at address 0xC000.
- For address 0xFFDE

  Starting from the 36st element in the flash variable, 2 16-bit words (4 bytes) must be written to the MSP430 MCU, starting at address 0xFFDE.
- For address 0xFFE4

  Starting at the 38th element in the flash variable, 2 16-bit words (4 bytes) must be written to the MSP430 MCU, starting at address 0xFFE4.
- For address 0xFFEA

  Starting at the 40th element in the flash variable, 11 16-bit words (22 bytes) must be written to the MSP430 MCU, starting at address 0xFFEA.

5.2 MSP430 FRAM Firmware Image

The default example firmware image for the FRAM-based MSP430 MCUs is the blinky example for MSP430FR2311. The firmware image file is formatted the same as the flash-based MSP430 MCUs. The only difference is that variables are all renamed from *flash* to *fram*.

```
/*
* This file was automatically generated.
* The memory sections should be quickly double checked.
* Some sections such ad Start,Finish, Termination and Length must be
* modified based on device datasheet. (These values aren't used by the
* default program.
* Created by: Nima Eskandari
*/

uint16_t fram[] =
{
//0xf100
0x40B2, 0x5A80, 0x01CC, 0xC3D2, 0x0202, 0xD3D2, 0x0204, 0xC392,
0x0130, 0xE3D2, 0x0202, 0x140D, 0x403D, 0x8232, 0x831D, 0x23FE,
0x160D, 0x3FF7, 0x4303, 0x4303, 0x3FFF, 0x4303, 0x431C, 0x0110,
0x4031, 0x2400, 0x13B0, 0xF12C, 0x430C, 0x13B0, 0xF100, 0x13B0,
0xF126, 0xD032, 0x0010, 0x3FFD, 0x4303,
//0xff80
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
//0xffe2
0xF142, 0xF142, 0xF142, 0xF142, 0xF142, 0xF142, 0xF142, 0xF142,
0xF142, 0xF142, 0xF142, 0xF142, 0xF142, 0xF142, 0xF130,

};


const uint32_t fram_address[] =
{
0xf100, 0xff80, 0xffe2,
};


const uint32_t fram_length_of_sections[] =
{
37, 4, 15,
};


const uint32_t fram_sections    = 3;

const uint32_t fram_termination = 0x00000000; /*Check device data sheet*/
const uint32_t fram_start       = 0x00000000; /*Check device data sheet*/
const uint32_t fram_finish      = 0x00000000; /*Check device data sheet*/
const uint32_t fram_length      = 0x00000000; /*Check device data sheet*/
```

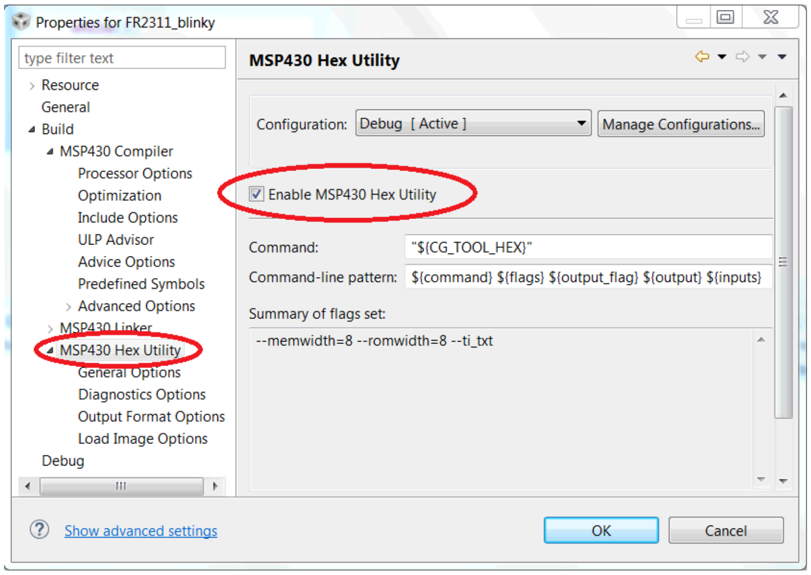
To generate a custom firmware image header file, follow the instructions in Section 5.3.

5.3 Generating Custom Firmware Image Header Files

To generate firmware image header files, the firmware must be compiled and the output must be in TI-TXT hex format.

Figure 23, Figure 24, and Figure 25 show how to generate a TI-TXT hex file in CCS.

1. Right click the project and click the properties option.
2. Check the *Enable MSP430 Hex Utility* checkbox.


Properties for FR2311_blinky

MSP430 Hex Utility

Configuration: Debug [Active]

[✓] Enable MSP430 Hex Utility

Command: "${CG_TOOL_HEX}"

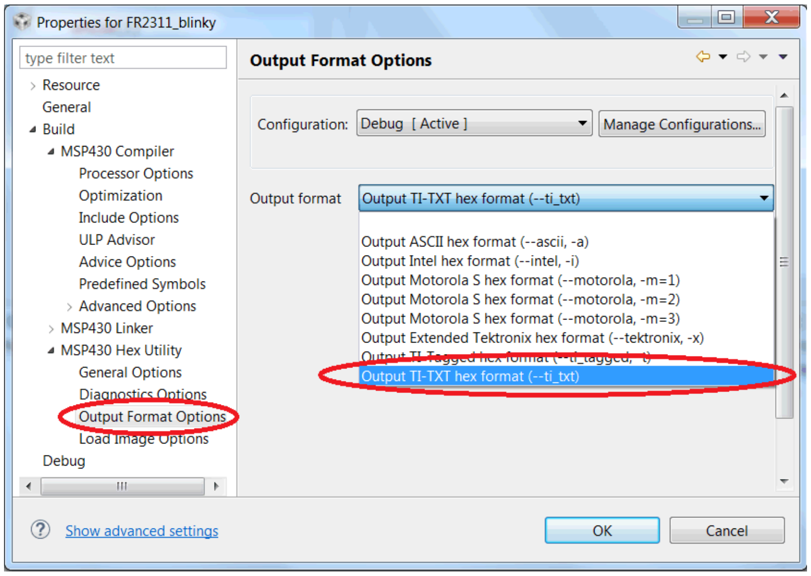
Command-line pattern: ${command} ${flags} ${output_flag} ${output} ${inputs}

Summary of flags set:

```
--memwidth=8 --romwidth=8 --ti_txt
```

Figure 23. Enabling MSP430 Hex Utility

3. In Output Format Options, select TI-TXT hex format.


Properties for FR2311_blinky

Output Format Options

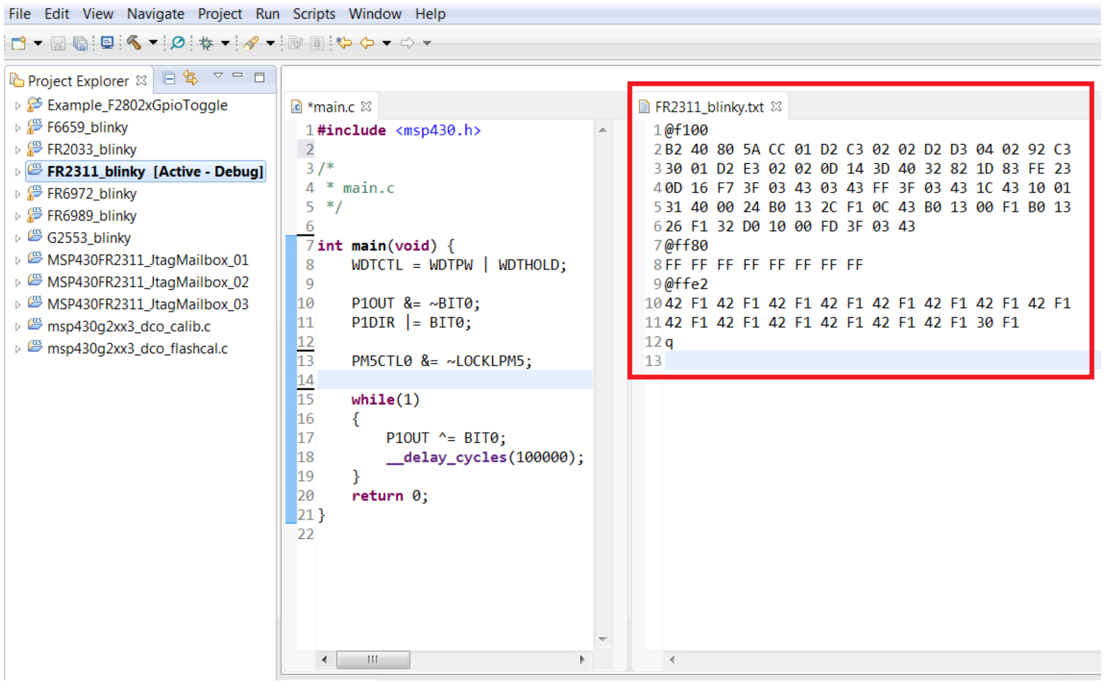
Configuration: Debug [Active]

Output format: Output TI-TXT hex format (--ti_txt)

Output ASCII hex format (--ascii, -a)
Output Intel hex format (--intel, -i)
Output Motorola S hex format (--motorola, -m=1)
Output Motorola S hex format (--motorola, -m=2)
Output Motorola S hex format (--motorola, -m=3)
Output Extended Tektronix hex format (--tektronix, -x)
Output TI-Tagged hex format (--ti_tagged, -t)
Output TI-TXT hex format (--ti_txt)

Figure 24. MSP430 TI-TXT Hex Format

4. After the project is built, the .txt containing the full firmware image is created. Figure 25 shows this file.


File Edit View Navigate Project Run Scripts Window Help
Project Explorer
Example_F2802xGpioToggle
F6659_blinky
FR2033_blinky
FR2311_blinky [Active - Debug]
FR6972_blinky
FR6989_blinky
G2553_blinky
MSP430FR2311_JtagMailbox_01
MSP430FR2311_JtagMailbox_02
MSP430FR2311_JtagMailbox_03
msp430g2xx3_dco_calib.c
msp430g2xx3_dco_flashcal.c
*main.c
FR2311_blinky.txt

```
#include <msp430.h>

/*
 * main.c
 */

int main(void) {
    WDTCTL = WDTPW | WDTHOLD;

    P1OUT &= ~BIT0;
    P1DIR |= BIT0;

    PM5CTL0 &= ~LOCKLPM5;

    while(1)
    {
        P1OUT ^= BIT0;
        __delay_cycles(100000);
    }
    return 0;
}
```

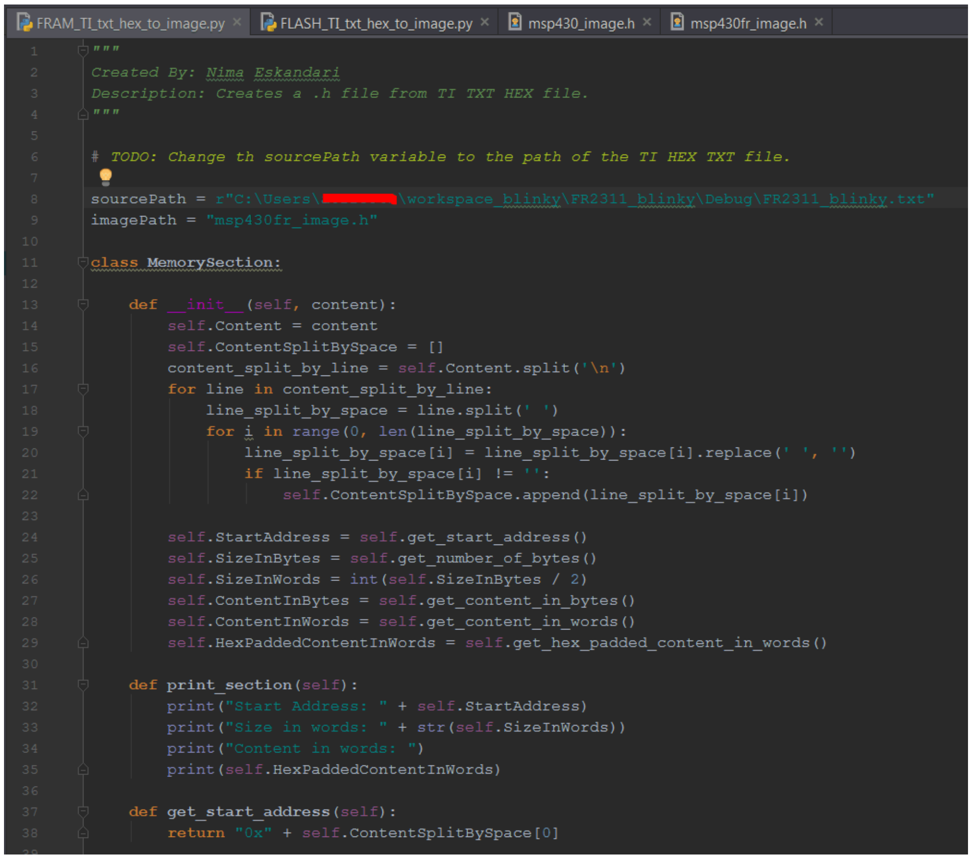
```
@f100
B2 40 80 5A CC 01 D2 C3 02 02 D2 D3 04 02 92 C3
30 01 D2 E3 02 02 0D 14 3D 40 32 82 1D 83 FE 23
0D 16 F7 3F 03 43 03 43 FF 3F 03 43 1C 43 10 01
31 40 00 24 B0 13 2C F1 0C 43 B0 13 00 F1 B0 13
26 F1 32 D0 10 00 FD 3F 03 43
@ff80
FF FF FF FF FF FF FF FF
@ffe2
42 F1 42 F1 42 F1 42 F1 42 F1 42 F1 42 F1 42 F1
42 F1 42 F1 42 F1 42 F1 42 F1 42 F1 30 F1
q
```

Figure 25. MSP430FR2311 TI-TXT Hex Format Firmware Image

The generated TI-TXT file can be converted to a firmware image header file.

Python scripts are provided to convert the TI-TXT file to a firmware image header file. The scripts are in the Python_Scripts folder of the example software zip file.

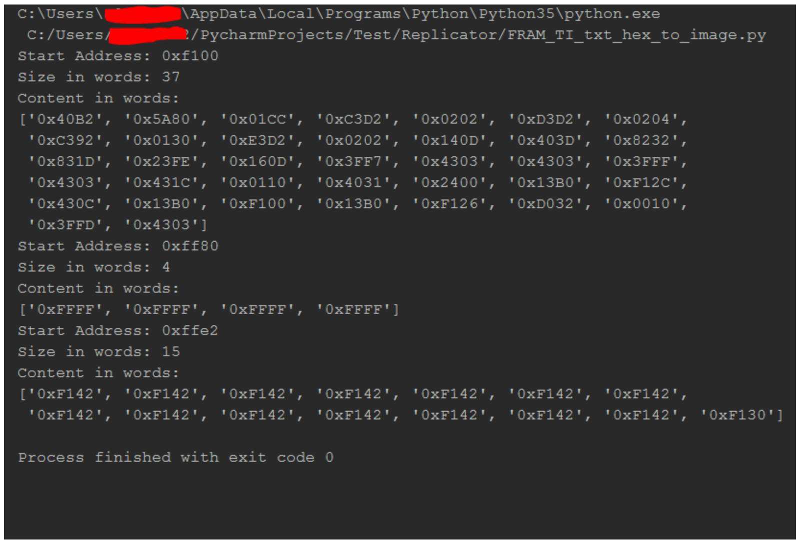
There are two Python scripts. The only difference between the two scripts is whether the created header file is for an FRAM based device or a flash-based device.



```python
"""
Created By: Nima Eskandari
Description: Creates a .h file from TI TXT HEX file.
"""

# TODO: Change th sourcePath variable to the path of the TI HEX TXT file.

sourcePath = r"C:\Users\[redacted]\workspace_blinky\FR2311_blinky\Debug\FR2311_blinky.txt"
imagePath = "msp430fr_image.h"

class MemorySection:

    def __init__(self, content):
        self.Content = content
        self.ContentSplitBySpace = []
        content_split_by_line = self.Content.split('\n')
        for line in content_split_by_line:
            line_split_by_space = line.split(' ')
            for i in range(0, len(line_split_by_space)):
                line_split_by_space[i] = line_split_by_space[i].replace(' ', '')
                if line_split_by_space[i] != '':
                    self.ContentSplitBySpace.append(line_split_by_space[i])

        self.StartAddress = self.get_start_address()
        self.SizeInBytes = self.get_number_of_bytes()
        self.SizeInWords = int(self.SizeInBytes / 2)
        self.ContentInBytes = self.get_content_in_bytes()
        self.ContentInWords = self.get_content_in_words()
        self.HexPaddedContentInWords = self.get_hex_padded_content_in_words()

    def print_section(self):
        print("Start Address: " + self.StartAddress)
        print("Size in words: " + str(self.SizeInWords))
        print("Content in words: ")
        print(self.HexPaddedContentInWords)

    def get_start_address(self):
        return "0x" + self.ContentSplitBySpace[0]
```

Figure 26. FRAM TI-TXT Hex to Firmware Image Converter

The variable sourcePath must be modified to point to the TI-TXT hex file to convert to a firmware image header file. The imagePath variable defines the location for the output firmware image header file. Using Python 3, the script can be executed and the firmware image header file is generated.



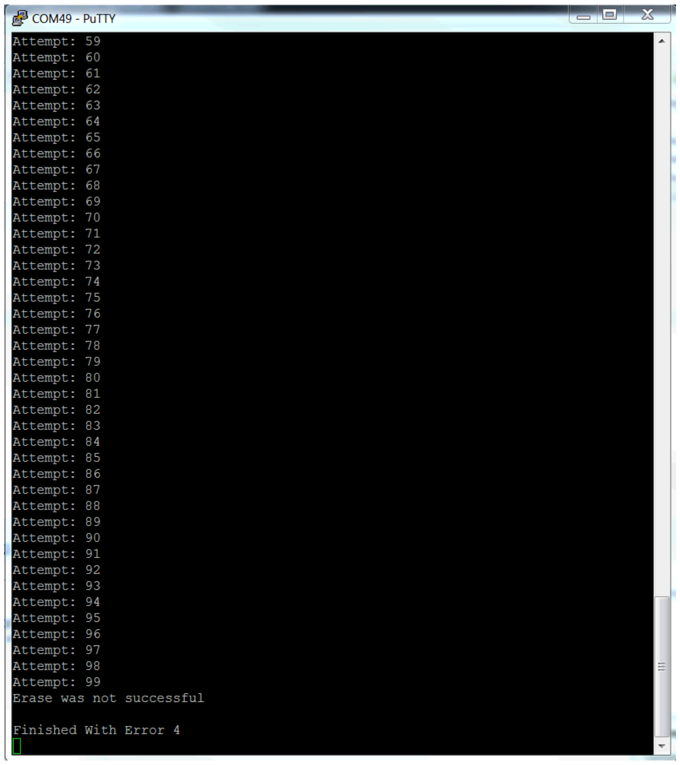
```
C:\Users\[redacted]\AppData\Local\Programs\Python\Python35\python.exe
 C:/Users/[redacted]2/PycharmProjects/Test/Replicator/FRAM_TI_txt_hex_to_image.py
Start Address: 0xf100
Size in words: 37
Content in words:
['0x40B2', '0x5A80', '0x01CC', '0xC3D2', '0x0202', '0xD3D2', '0x0204',
 '0xC392', '0x0130', '0xE3D2', '0x0202', '0x140D', '0x403D', '0x8232',
 '0x831D', '0x23FE', '0x160D', '0x3FF7', '0x4303', '0x4303', '0x3FFF',
 '0x4303', '0x431C', '0x0110', '0x4031', '0x2400', '0x13B0', '0xF12C',
 '0x430C', '0x13B0', '0xF100', '0x13B0', '0xF126', '0xD032', '0x0010',
 '0x3FFD', '0x4303']
Start Address: 0xff80
Size in words: 4
Content in words:
['0xFFFF', '0xFFFF', '0xFFFF', '0xFFFF']
Start Address: 0xffe2
Size in words: 15
Content in words:
['0xF142', '0xF142', '0xF142', '0xF142', '0xF142', '0xF142', '0xF142',
 '0xF142', '0xF142', '0xF142', '0xF142', '0xF142', '0xF142', '0xF142', '0xF130']

Process finished with exit code 0
```

Figure 27. Firmware Image Python Script Console Output

After the conversion is completed, the firmware image header file is generated and can be used as a replacement for the software example's default firmware image.

6 Error Messages

If the target to host connection setup is incorrect or the JTAG fuse is blown, the error in Figure 28 can be seen in the serial terminal.



```
Attempt: 59
Attempt: 60
Attempt: 61
Attempt: 62
Attempt: 63
Attempt: 64
Attempt: 65
Attempt: 66
Attempt: 67
Attempt: 68
Attempt: 69
Attempt: 70
Attempt: 71
Attempt: 72
Attempt: 73
Attempt: 74
Attempt: 75
Attempt: 76
Attempt: 77
Attempt: 78
Attempt: 79
Attempt: 80
Attempt: 81
Attempt: 82
Attempt: 83
Attempt: 84
Attempt: 85
Attempt: 86
Attempt: 87
Attempt: 88
Attempt: 89
Attempt: 90
Attempt: 91
Attempt: 92
Attempt: 93
Attempt: 94
Attempt: 95
Attempt: 96
Attempt: 97
Attempt: 98
Attempt: 99
Erase was not successful

Finished With Error 4
```

Figure 28. SBW Error

To debug this issue further, change the DEBUG macro in config.h to 1, which prints more information into the serial terminal. If that does not give any hints as to what the source of the failure is, debug and single step through the Program_MSPFR() or Program_MSP() in SBW430FR.c or SBW430.c. Also double check the connections between the host and the target. Finally, cycle power to the target MSP430 MCU and run the host again.