

# MSP430<sup>™</sup> Bootloader With SimpleLink<sup>™</sup> MCUs

Nima Eskandari

MSP430 Applications

### ABSTRACT

The MSP430<sup>™</sup> bootloader (BSL) allows a host MCU to communicate with embedded memory in the MSP430 microcontroller (MCU). The 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<sup>™</sup> MSP432P401R, CC3220, and CC2640R2F devices as the hosts for the BSL communication with the target MSP430 devices. Both flash-based and FRAM-based MSP430 MCUs are used in this report to showcase the differences between their BSLs. Software examples are provided for each of the SimpleLink host MCUs. The software examples also make use of the SimpleLink Software Development Kit (SDK), making it easy to port the examples to other SimpleLink devices.

The source code and other files described in this application report can be downloaded from <a href="http://www.ti.com/lit/zip/slaa755">http://www.ti.com/lit/zip/slaa755</a>. The example source code demonstrates how a SimpleLink MCU can access the memory of a target MSP430 MCU (flash or FRAM based) through the UART BSL.

1	Introduction	on	2
	1.1 Su	upplementary Online Information	2
2	Software	Example	3
		oftware Example File Descriptions	
3		er (BSL) Connections	
	3.1 Bo	ootloader Connections for the Target MSP430 MCU	6
	3.2 Bo	ootloader Connections for the Host SimpleLink Device	8
4		se the Software Examples	
	4.1 Im	nport the Projects to CCS	11
		un the Software Examples	
5	Create Cu	ustom MSP430 Firmware Image	24
	5.1 M	ISP430 Flash Firmware Image	24
	5.2 M	ISP430 FRAM Firmware Image	25
	5.3 Ge	enerating Custom Firmware Image Header Files	27
6	Error Mes	ssages	30
		List of Figures	
1	MSP4300	G2553 BSL Entry Sequence Pins	6
2	MSP430F	FR2311 BSL Entry Sequence Pins	7
3	MSP4300	G2553 UART BSL Pins	7
4	MSP430F	FR2311 UART BSL Pins	8
5		P401R LaunchPad Development Kit UART BSL Host Pins	
6		SF LaunchPad Development Kit UART BSL Host Pins	
•		•	
7		R2 LaunchPad Development Kit UART BSL Host Pins	
8	Import Software Example Projects to CCS 11		
9	Import All	Projects to CCS	12

#### Contents



10	Building Software Example Projects	13
11	TI RTOS Build Project Dependency	14
12	MSP432P401R LaunchPad Development Kit Connected to MSP430G2553	15
13	MSP432P401R LaunchPad Development Kit UART BSL Host Connections	15
14	MSP430G2553 LaunchPad Development Kit UART BSL Connections	16
15	Application UART COM Port Number in Windows Device Manager	16
16	PuTTY Serial Communication Settings	17
17	MSP432Host_UART_BSL_MSP430 Project Debug Mode	18
18	MSP432Host_UART_BSL_MSP430 Project Serial Output	19
19	MSP430G2553 Programmed Successfully	20
20	MSP432P401R LaunchPad Development Kit Connected to MSP430FR2311	21
21	MSP430FR2311 LaunchPad Development Kit UART BSL Connections	21
22	MSP432Host_UART_BSL_MSP430FR Project Debug Mode	22
23	MSP430FR2311 Programmed Successfully	23
24	Enabling MSP430 Hex Utility	27
25	MSP430 TI-TXT Hex Format	27
26	MSP430FR2311 TI-TXT Hex Format Firmware Image	28
27	FRAM TI-TXT Hex to Firmware Image Converter	29
28	Firmware Image Python Script Console Output	29
29	UART BSL Error	30

### List of Tables

1	Top-Level Folder Description	3
2	CC2640R2 Host Example Project Content	4
3	CC3220 Host Example Project Content	4
4	MSP432 Host Example Project Content	5
5	BSL Folder Content	5

# Trademarks

2

Introduction

MSP430, SimpleLink are trademarks of Texas Instruments. All other trademarks are the property of their respective owners.

# 1 Introduction

The bootloader provides a method for the MSP430 MCUs to be programmed. In order for the MSP430 MCU to be programmed through BSL, commands must be sent using the UART protocol to the MSP430 MCU.

To invoke the bootloader, a BSL entry sequence must be applied to dedicated pins. After the BSL entry sequence is applied, UART commands can be sent to the target MSP430 MCU. The BSL UART 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 MSP432, CC3220 and CC2640 as the UART BSL host to the target MSP430 MCU. The UART BSL 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 and tools, visit Bootloader (BSL) for MSP low-power microcontrollers. This page contains links to additional BSL user's guides, source code, firmware images, and the BSL scripter with documentation and code examples.

For FRAM-based MSP430 MCUs:

MSP430FR5xx and MSP430FR6xx Bootloader (BSL) User's Guide

MSP430FR4xx and MSP430FR2xx Bootloader (BSL) User's Guide

For flash-based MSP430 MCUs:

MSP430<sup>™</sup> Flash Device Bootloader (BSL) User's Guide

Additional support is provided by the TI E2E Community.

# 2 Software Example

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

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

- SimpleLink<sup>™</sup> MSP432P401R LaunchPad<sup>™</sup> Development Kit
- SimpleLink™ Wi-Fi® CC3220SF Wireless Microcontroller LaunchPad™ Development Kit
- SimpleLink<sup>™</sup> Bluetooth<sup>®</sup> low energy CC2640R2F Wireless MCU LaunchPad<sup>™</sup> Development Kit

# 2.1 Software Example File Descriptions

The example software can be downloaded from http://www.ti.com/lit/zip/slaa755.

Table 1 describes the contents of the top-level folder.

Folder Name	Description		
CC2640R2Host_UART_BSL_MSP430	Example SimpleLink SDK project for CC2640R2. CC2640R2 acts as a UART BSL host for MSP430 flash-based MCUs. 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_UART_BSL_MSP430FR	Example SimpleLink SDK project for CC2640R2. CC2640R2 acts as a UART BSL host for MSP430 FRAM-based MCUs. 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_UART_BSL_MSP430	Example SimpleLink SDK project for CC3220. CC3220 acts as a UART BSL host for MSP430 flash-based MCUs. 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_UART_BSL_MSP430FR	Example SimpleLink SDK project for CC3220. CC3220 acts as a UART BSL host for MSP430 FRAM-based MCUs. 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.		

 Table 1. Top-Level Folder Description

TEXAS INSTRUMENTS

Software Example

www.ti.com

Folder Name	Description		
MSP432Host_UART_BSL_MSP430	Example SimpleLink SDK project for MSP432P401R. MSP432P401R acts as a UART BSL host for MSP430 flash-based MCUs. 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.		
MSP432Host_UART_BSL_MSP430FR	Example SimpleLink SDK project for MSP432P401R. MSP432P401R acts as a UART BSL host for MSP430 FRAM-based MCUs. 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.		
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_byte_image.py and Python_Scripts/FRAM_TI_txt_hex_to_byte_image.py	Converts TI HEX TXT file to the header file format used by the host. These files are python scripts and must be ran using python. Example: python flash_TI_txt_hex_to_byte_image.py		

The following tables describe the contents of each project.

Table 2.	CC2640R2	Host Exam	ple Projec	t Content
		I OUL ENGIN		

Name	Description		
BSL (Directory)	Contains the UART BSL 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_uart_bsl_msp430.c or cc2640r2host_uart_bsl_msp430fr.c	Driver Initialization. Starts the BSL programming with the example firmware image.		

# Table 3. CC3220 Host Example Project Content

Name	Description		
BSL (Directory)	Contains the UART BSL 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_uart_bsl_msp430.c or cc3220host_uart_bsl_msp430fr.c	Driver Initialization. Starts the BSL programming with the example firmware image.		

# Table 4. MSP432 Host Example Project Content

Name	Description		
BSL (Directory)	Contains the UART BSL 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_uart_bsl_msp430.c or msp432host_uart_bsl_msp430fr.c	Driver Initialization. Starts the BSL programming with the example firmware image.		

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

Table 5 describes the contents of the BSL folder.

### Table 5. BSL Folder Content

Name	Description			
image/msp430_image.h or image/msp430fr_image.h	Contains the example firmware image for flash or FRAM target MSP430. The msp430_image.h contains the blinky example for MSP430G2553 device, while the msp430fr_image.h contains the blinky example for MSP430FR2311 device.			
bsl.c	Implementation of the UART BSL commands. This file contains the commands used to program an MSP430 MCU through UART BSL. This file generates the BSL Command packages to send. It also receives and unpacks the BSL Responses from the target.			
bsl.h	Contains the function declarations for the BSL commands.			
bsl_main.c	Initializes the hosts peripherals and programs the target device with the specified example firmware image.			
bsl_main.h	Contains the function declarations for the bsl_main.c			
config.h	Contains configurable options such as UART read time-out 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				
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 BSL entry sequence pins.			
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 BSL commands.				
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 debugging.				
utils.h	Contains the function declarations for the utils.c			



### 3 Bootloader (BSL) Connections

The host SimpleLink MCU and target MSP430 MCU must be connected to through the BSL pins and share the GND signal. The BSL pins include the dedicated BSL entry pins and the UART BSL pins.

### 3.1 Bootloader Connections for the Target MSP430 MCU

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

To invoke the bootloader, the BSL entry sequence must be applied to the TST and RST pins on the target MSP430 MCU. For more information on the BSL entry sequence, see the MSP430<sup>™</sup> Flash Device Bootloader (BSL) User's Guide.

Figure 1 and Figure 2 show the dedicated pins for BSL entry sequence for the MSP430G2553 and the MSP430FR2311, respectively.

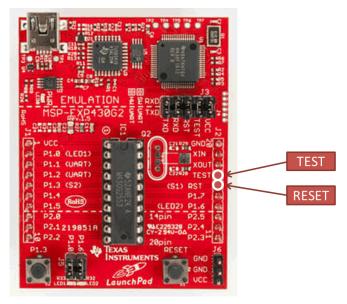


Figure 1. MSP430G2553 BSL Entry Sequence Pins



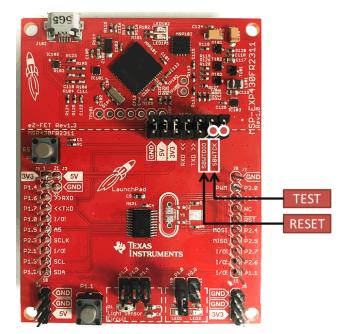


Figure 2. MSP430FR2311 BSL Entry Sequence Pins

These pins are connected to the GPIO of the SimpleLink host MCU. After the BSL entry sequence is completed, the host can use the dedicated UART BSL pins to send and receive BSL commands.

The pins for BSL entry sequence 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)

Figure 3 and Figure 4 show the dedicated UART pins for the MSP430G2553 and MSP430FR2311, respectively. To find the UART BSL pins for any MSP430 MCU, see the device-specific data sheet. The UART BSL pins are specific to each device and package. For example, the UART BSL pins for the MSP430G2553 are P1.1 and P1.5, while the UART BSL pins for the MSP430FR2311 are P1.6 and P1.7.

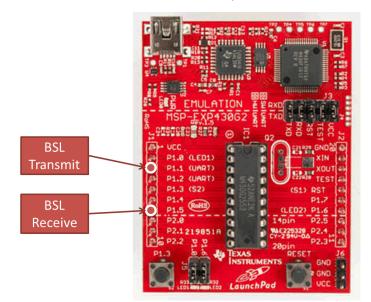


Figure 3. MSP430G2553 UART BSL Pins



Bootloader (BSL) Connections

www.ti.com

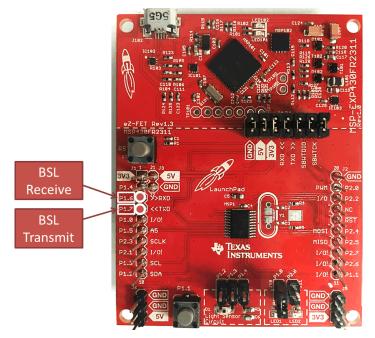


Figure 4. MSP430FR2311 UART BSL Pins

The final connections needed on the target MSP430 MCU is 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 VCC and GND pins.

# 3.2 Bootloader Connections for the Host SimpleLink Device

The host device uses two GPIO pins to execute the BSL entry sequence. The host also uses two UART modules. The first UART module communicates the status of code execution to the PC through the backchannel serial port. The second UART module communicates to the BSL target to transmit commands and receive the responses. These pins are defined in the board-specific file for each project (MSP\_EXP432P401R.c, CC3220SF\_LAUNCHXL.c and CC2640R2\_LAUNCHXL.c).

The software example provided for the supported LaunchPad development kits, utilizes 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
- BSL Communication UART:
  - TX: P3.3
  - RX: P3.2

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

- BSL Communication UART:
  - TX: PIN 07
  - RX: PIN 08

CC2640R2 Wireless Microcontroller LaunchPad development kit:

- Target Reset Pin: DIO23
- Target Test Pin: DIO25
- Backchannel UART to PC:
  - TX: DIO26 (Because CC2640R2 does not have two UART modules, software UART is used to communicate to the PC.)
  - RX: Unused
- BSL Communication UART:
  - TX: DIO03
  - RX: DIO02

Figure 5, Figure 6, and Figure 7 show each LaunchPad development kit and the pins used for the software example.

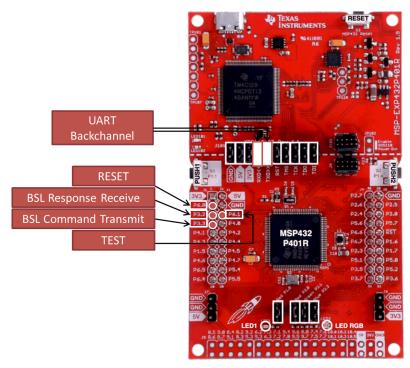


Figure 5. MSP432P401R LaunchPad Development Kit UART BSL Host Pins



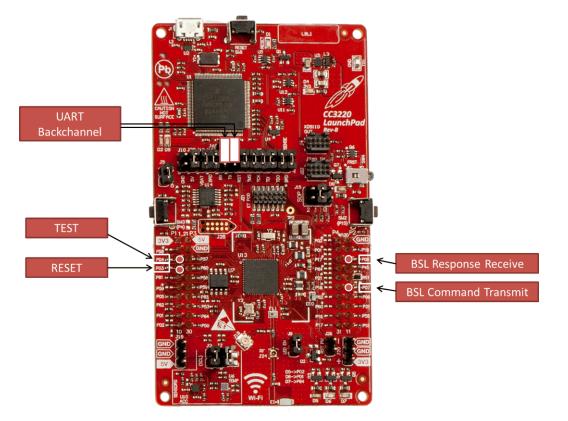


Figure 6. CC3220SF LaunchPad Development Kit UART BSL Host Pins

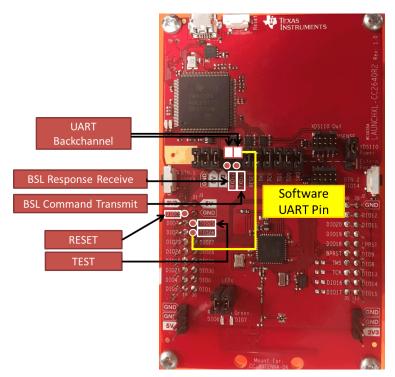


Figure 7. CC2640R2 LaunchPad Development Kit UART BSL Host Pins



The SimpleLink Host connects to the MSP430 MCU through the Reset (BSL entry), Test (BSL entry), BSL UART TX, and BSL UART RX 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 firmware update on the MSP430 MCU.

# 4 How to Use the Software Examples

The software examples in this application report use 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 IDE (CCS) 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:

- 1. Click Project > Import CCS Projects.
- 2. Select Browse and navigate to the extracted file from the software example zip file.

Figure 2015 Eclipse Projects			Ì	
Select CCS Projects to Imp Select a directory to search for			-	
Select search-directory:     Select archive file:		Browse		
Discovered projects:				
		Select All Deselect All	, ,	
Automatically import referen Copy projects into workspace Open <u>Resource Explorer</u> to brow	Korrection CC3220Host_UART_BSL_MSP430 Korrection CC3220Host_UART_BSL_MSP430	30 30FR FR 0		
0	Python_Scripts   Pitros_builds_CC2640R2_LAUNCH   Jitros_builds_CC3220SF_LAUNCH   Jitros_builds_MSP_EXP432P401R	HXL_release_ccs		
	Folder: SimpleLink Host for MSP430 UART BSL Make New Folder			OK Cancel

### Figure 8. Import Software Example Projects to CCS

3. 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.



How to Use the Software Examples

Import CCS Eclipse Projec	ts	
Select CCS Projects to I Select a directory to search	import 1 for existing CCS Eclipse projects.	
<ul> <li>Select search-directory:</li> <li>Select archive file:</li> </ul>	C:\Users\\workspace_linux\SimpleLink Hos	Browse Browse
CC2640R2Host U, CC3220Host UAR CC3220Host UAR CC3220Host UAR CMSP432Host UAR C	ART_BSL_MSP430 [C:\Users\workspace_in ART_BSL_MSP430FR [C:\Users\workspace_inux\ IT_BSL_MSP430FR [C:\Users\workspace_inux ART_BSL_MSP430FR [C:\Users\workspace_inux ART_BSL_MSP430FR [C:\Users\workspace_inux 40R2_LAUNCHXL_release_ccs [C:\Users\workspace_inux 220SF_LAUNCHXL_release_ccs [C:\Users\workspace_inux _EXP432P401R_release_ccs [C:\Users\workspace_inux]	Select All Deselect All Refresh
Copy projects into works	erenced projects found in same search-directory pace browse a wide selection of example projects	Cancel

Figure 9. Import All Projects to CCS

4. After importing, the projects must be built.

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





у.	▼ II (0:%)	▼ : ¢	2:☆▼:∥ ▼:⊑		<> ▼	
8	Project Ex	plore	er 🛛			□ 🕏 ▽ 🗖
R				ISP430 [Active - De	ebug	1
8			ost_UART_BSL_MSI			
			t_UART_BSL_MSP43	30		1
	CC322	_	New		•	
	<ul> <li>MISP43</li> <li>MSP43</li> </ul>	1	Show in Local Ter	minal	•	
	Initial Hard Stress		Add Files			
	▷ <sup>™</sup> tirtos_		Copy	Ctrl+	+C	
	▷ <sup>™</sup> tirtos_		Paste	Ctrl+		
		×	Delete	Dele	te	
			Refactor		•	
			Source		•	
			Move			
			Rename		F2	
		2	Import			
		è	Export			
			Show Build Settin	Rebuild the curren	tlv se	lected project(s)
			Build Project		ing be	
			Clean Project			
	•		Rebuild Project			
	Console		Refresh		F5	
	CDT Build Co		Close Project			
	IID.aem3	1	Make Targets		•	
	-l"C:/ti/		Index		•	ource/ti/drivers/rf/l
	-1"C:/ti/		Build Configuration	ons	•	urce/ti/drivers/lib/
	drivers_c	c	Debug As		•	
	-1"C:/ti/		Restore from Loca	al History		ource/ti/drivers/pdm/
	lib/pdm_c -l"C:/ti/		Team		•	rnel/tirtos/packages
	/ti/dp1/1		Compare With		•	ince, en cos, pacitoges
	-1"C:/ti/		Replace With			urce/ti/devices/cc26
	<pre>x0r2/driv <linking></linking></pre>	erlî	b/bin/ccs/driv	erlib.lib" -lli	DC.	а
	-	bui	lding target:	CC2640R2Host_UA	RT_ E	BSL_MSP430.out'
			0 0 0		_	-
	**** 0.13		nished ****			
	Bull	u F1	nisned			

# Figure 10. Building Software Example Projects

tirtos\_builds\_CC2640R2\_LAUNCHXL\_release\_ccs, tirtos\_builds\_CC3220SF\_LAUNCHXL\_release\_ccs, and tirtos\_builds\_MSP\_EXP432P401R\_release\_ccs are rebuilt automatically. These projects are referenced by the other projects with the same platform.

workspace_SimpleLink_AppReport - CCS Edi			
File Edit View Navigate Project Run S			
📑 🕶 🔚 🕼 🔦 🔻 🖉 🔅 🖉 🖉 📑 📴			
<pre> Project Explorer  Project  Projec</pre>	FR FR FR [Active - Debug] Properties for MSP432Host_UAI type filter text • Resource Linked Resources Resource Filters General • Build • MSP432 Compiler Processor Options Optimization Include Options ULP Advisor Predefined Symbols > Advanced Options > MSP432 Linker MSP432 L	Image: Steps in the second	Referenced Bu
<pre>-l"C:/ti/simplelink_msp432_sdk p4xx/driverlib/ccs/msp432p4xx_ <linking></linking></pre>	4	e 'General' for changing tool versions and de	vice settings
'Finished building target: MSP	Show advanced settings	ОК	Cancel
**** Build Finished ****			
BSP432Host UART BSL MSP430FR			

Figure 11. TI RTOS Build Project Dependency

# 4.2 Run the Software Examples

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

# 4.2.1 MSP430G2553 UART BSL Target Example

1. Connect the two devices.

The pin to pin connections are available in the README file.

Figure 12, Figure 13, and Figure 14 show the connections for the MSP432P401R and MSP430G2553 LaunchPad development kits.



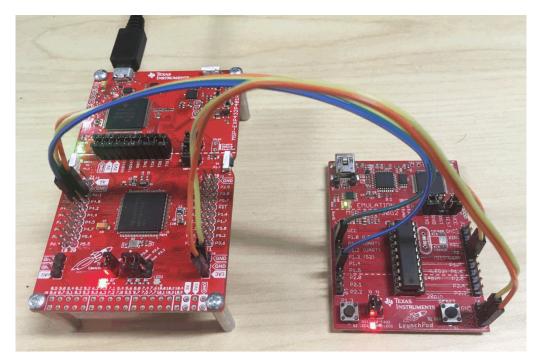


Figure 12. MSP432P401R LaunchPad Development Kit Connected to MSP430G2553

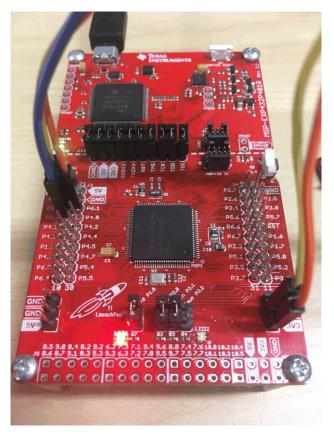
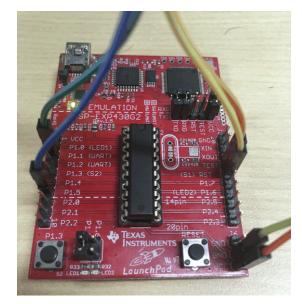


Figure 13. MSP432P401R LaunchPad Development Kit UART BSL Host Connections





### Figure 14. MSP430G2553 LaunchPad Development Kit UART BSL 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.

2. 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.

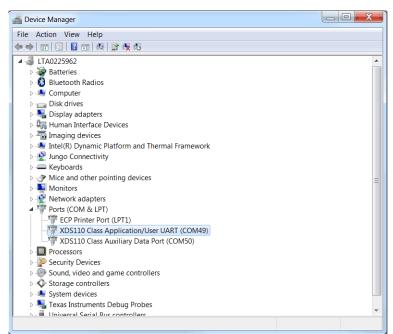


Figure 15. Application UART COM Port Number in Windows Device Manager



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.

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

1	PuTTY Configuration		? ×
	Category:		
	Session	Basic options for your PuTTY ses	sion
	Logging	Specify the destination you want to connect to	
	Keyboard	Serial line	Speed
🚔 Device Manager	-Bell	COM49	9600
File Action View	- Features - Window - Appearance	Connection type: Raw Telnet Rlogin SSH	<ul> <li>Serial</li> </ul>
▲ → LTA0225962	-Behaviour -Translation	Load, save or delete a stored session	
Batteries	-Selection	Saved Sessions	
Bluetooth	- Colours		
> - 🥾 Computer	- Data	Default Settings	Load
Disk drives	Proxy		Save
Display ad Display	– Telnet – Rlogin		
Finaging d			Delete
▷ - 🍂 Intel(R) Dy	- Serial		
⊳ 🔮 Jungo Con		Close window on exit	
Keyboards		Always Never Only on cle	an exit
🗅 💇 Mice and 🤇			
D - Monitors			
P - Ports (CON)	About Help	Open	Cancel
ECP Printe	ar Port (LDT1)		
	lass Application/User UART (	(COM49)	
	lass Auxiliary Data Port (COM		
Processors			
Device Security Device			
	and game controllers		
Storage control			
b - System device	es ients Debug Probes		
	al Rus controllers		-

### Figure 16. PuTTY Serial Communication Settings

- 3. Debug the project MSP432Host\_UART\_BSL\_MSP430 by Run > Debug.
- 4. After the binary firmware image is uploaded to MSP432P401R, run the example by clicking the green play button.



	View Project Tools Run Scripts Window Help			
<u>n</u> - 8 €	)   ≪ ▼   ∅   <   ↓ ▼   √ ▼   ⊆   ▶ = 3. ۞ () =   [] % @	▼ 18 18 🍓 ▼ 🕹 💣 ▼ 13.	<u>م</u>	Quick Access 📑 📑 🖏
🎋 Debug 🛛	3 ×	(x)= Variables 🛛 🕸 Expression	ons IIII Registers	🖆 🐗 🖻   📫 🛃 🔷 🔻 🗖
▲ 🕄 MSP43	32Host_UART_BSL_MSP430 [Code Composer Studio - Device Debugging	Name	Туре	Value
4 🧬 Tex	kas Instruments XDS110 USB Debug Probe/CORTEX_M4_0 (Suspended -	🕪 detachState	unknown	Memory map prevent
	main() at main_tirtos.c:57 0x00004D64	pAttrs struct pthread_attr_t		{priority=???
=.	_c_int00() at boot.asm:254 0x00005478 (_c_int00 does not contain frame	🖻 🥏 priParam	struct sched_param	{sched_priority=???
		(×)= retc	unknown	Memory map prevent
		⊗= thread	unknown	Memory map prevent
(	III	•	III	*
main tirte	os c X			
61 ir 62 ir 63 64 /*	,			=
66 67 /* 68 pt 69 pr 70	<pre>* Set priority and stack size attributes */ thread_attr_init(&amp;pAttrs); riParam.sched_priority = 1; etachState = PTHREAD_CREATE_DETACHED;</pre>			•
66 67 /* 68 pt 69 pr 70 71 d€ ∢	<pre>thread_attr_init(&amp;pAttrs); riParam.sched_priority = 1; etachState = PTHREAD_CREATE_DETACHED;</pre>			
66 67 /* 68 pt 69 pr 70 71 de < 2 Console	<pre>thread_attr_init(&amp;pAttrs); riParam.sched_priority = 1; etachState = PTHREAD_CREATE_DETACHED; 8</pre>			· · · · · · · · · · · · · · · · · · ·
66 67 /* 68 pt 69 pr 70 71 de € © Console ÷ MSP432Host CORTEX_M4 CORTEX_M4	<pre>thread_attr_init(&amp;pAttrs); riParam.sched_priority = 1; etachState = PTHREAD_CREATE_DETACHED;</pre>	o check if the device mory 0x00000000		e is in low power mode.
66 67 /* 68 pt 69 pr 70 71 de 8 8 9 Console 1 1 1 1 1 5 9 4 2 8 9 8 9 8 9 8 9 8 9 8 9 9 70 70 70 70 71 de 9 9 70 70 71 de 9 9 70 70 71 de 9 9 70 70 71 de 9 9 70 70 71 de 9 9 70 70 71 de 9 9 70 71 de 9 9 70 71 de 9 9 70 71 de 9 9 70 71 de 9 9 70 71 de 9 9 70 71 de 9 9 70 71 de 9 9 70 71 de 70 71 de 70 70 71 de 70 70 71 de 70 70 71 de 70 70 71 de 70 70 71 de 70 70 71 de 70 70 71 de 70 70 71 de 70 70 71 de 70 70 70 70 70 70 70 70 70 70 70 70 70	<pre>thread_attr_init(&amp;pAttrs); riParam.sched_priority = 1; etachState = PTHREAD_CREATE_DETACHED; &amp; t_UART_BSL_MSP430 L_0: WARNING : On MSP432P401R hitting a breakpoint of Click the pause button during debug t L_0: Flash Programmer: Erasing main memory _0: Flash Programmer: Writing 32752 bytes to flash me</pre>	o check if the device mory 0x00000000 ry 0x00007ff0		e is in low power mode.

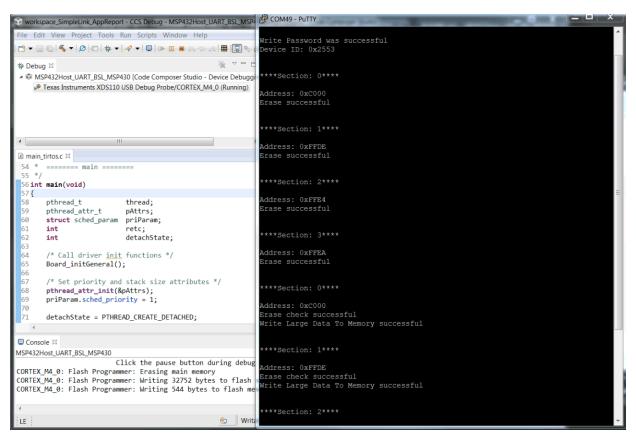
Figure 17. MSP432Host\_UART\_BSL\_MSP430 Project Debug Mode

5. View the serial output on PuTTY (see Figure 18).



**TEXAS** 

**NSTRUMENTS** 



# Figure 18. MSP432Host\_UART\_BSL\_MSP430 Project Serial Output

The MSP430G2533 device ID is 0x2553. All MSP430s have a unique device ID. For example, the Device ID for MSP430FR2311 is 0xFF80, which is shown in the next section.

6. The MSP430 MCU is programmed and the message, "MSP430 Programmed Successfully" is shown on the serial terminal.

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



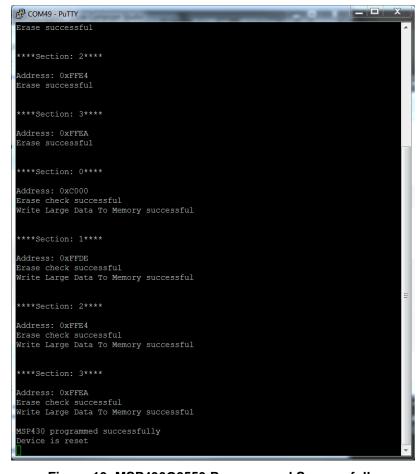


Figure 19. MSP430G2553 Programmed Successfully

7. Follow the same steps to program any flash-based MSP430 MCU using CC2640R2 and CC3220 device.

# 4.2.2 MSP430FR2311 UART BSL Target Example

To program an MSP430FR2311 using the MSP432P401R through UART BSL, follow the steps in Section 4.2.1.

Figure 20 shows the connections between the MSP430FR2311 and the MSP432P401R.





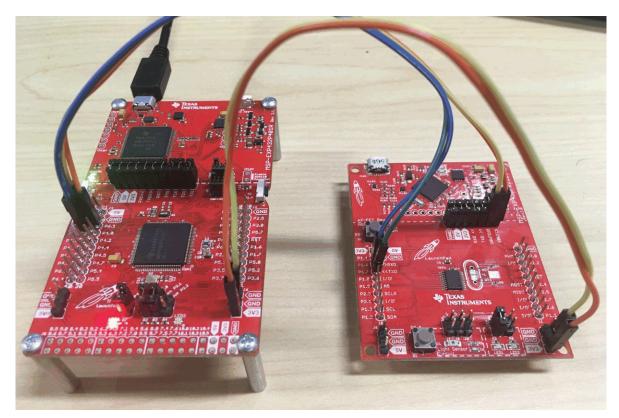


Figure 20. MSP432P401R LaunchPad Development Kit Connected to MSP430FR2311

The MSP432P401R connections are the same as Figure 13. Figure 21 shows the MSP430FR2311 connections.

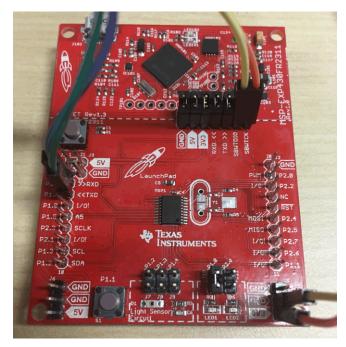


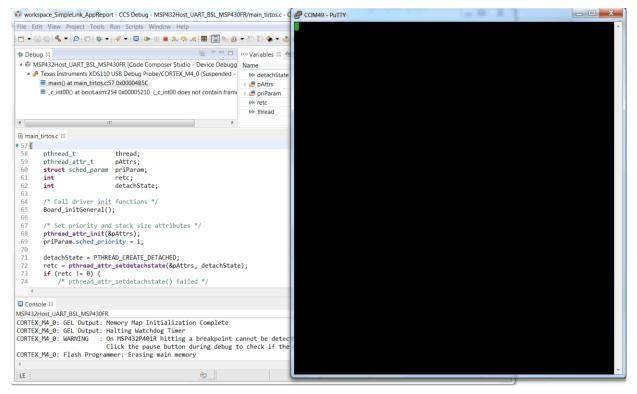
Figure 21. MSP430FR2311 LaunchPad Development Kit UART BSL Connections



#### How to Use the Software Examples

www.ti.com

After the connections are made, the example project MSP432Host\_UART\_BSL\_MSP430FR can be debugged and the result can be viewed using PuTTY (see Figure 22).



### Figure 22. MSP432Host\_UART\_BSL\_MSP430FR Project Debug Mode

After running the example, the result in Figure 23 is seen in the serial terminal.



Texas

*TRUMENTS* 

Dia D	dia Missi Dustant T I	Dura Cardinata MAG		
	dit View Project Tools			
3 🔻 [	∃ G 🔦 ▾ 🖉 🖾 🏘 ▾	🔗 ▾ 📮 🕩 🛛 🛢	*****New Attempt****	
\$ Deb	ug 🛙			
4 😳 N	ASP432Host UART BSL MSP	430FR [Code Composei	Write Password was successful <sup>Stu</sup> Device ID: 0xF082	
<b>N</b>	Texas Instruments XDS110	USB Debug Probe/COF		
			****Section: 0****	
			Address: 0xF100	
			Write Large Data To Memory successful	
	11	1		
mai	n_tirtos.c 🛛		****Section: 1****	
57 {				
58	pthread_t	thread;	Address: 0xFF80	
59 60	pthread_attr_t struct sched param	pAttrs; priParam;	Write Large Data To Memory successful	
61	int	retc;		
62	int	detachState;	****Section: 2****	
63 64	(* c-11 Jainer init	· (		
65	<pre>/* Call driver init Board initGeneral()</pre>		Address: 0xFFE2 Write Large Data To Memory successful	
66				
67	/* Set priority and	stack size attri	bu MSP430 programmed successfully	
68 69	<pre>pthread_attr_init(&amp;     priParam.sched_pric</pre>		Device is reset	
70	priparam.scheu_pric	)/1Ly = 1,		
71	detachState = PTHRE			
72		_setdetachstate(&	RpAttrs, detachState);	
73 74	<pre>if (retc != 0) {     /* nthread attr</pre>	setdetachstate()	) failed */	
1	/ peinedu_deen	()		•
1.0	sole 🛛		B	<b></b>
	2Host_UART_BSL_MSP430FR			La .
	X M4 0: GEL Output: M	emory Map Initial	ization Complete	
ORTE	X_M4_0: GEL Output: H	alting Watchdog T	imer	
ORTE			ting a breakpoint cannot be detected by the debugger when the device is in low power	mode.
ORTE	C X M4 0: Flash Program		tton during debug to check if the device is held at the breakpoint.	
(	A_re_o. riash Program	mer. crusting main		Þ

Figure 23. MSP430FR2311 Programmed Successfully

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

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

The default firmware image in all examples for FRAM-based MSP430 MCUs 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 IDE 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
*/
uint8_t flash[] =
{
//0xc000
0x21, 0x83, 0xB2, 0x40, 0x80, 0x5A, 0x20, 0x01,
0xD2, 0xD3, 0x22, 0x00, 0xD2, 0xE3, 0x21, 0x00,
0xB1, 0x40, 0x50, 0xC3, 0x00, 0x00, 0x91, 0x83,
0x00, 0x00, 0x81, 0x93, 0x00, 0x00, 0xF6, 0x27,
0xFA, 0x3F, 0x31, 0x40, 0x00, 0x04, 0xB0, 0x12,
0x42, 0xC0, 0x0C, 0x43, 0xB0, 0x12, 0x00, 0xC0,
0xB0, 0x12, 0x3C, 0xC0, 0x32, 0xD0, 0x10, 0x00,
0xFD, 0x3F, 0x03, 0x43, 0x03, 0x43, 0xFF, 0x3F,
0x03, 0x43, 0x1C, 0x43, 0x30, 0x41,
//0xffde
0xFF, 0xFF, 0x34, 0xC0,
//0xffe4
0x34, 0xC0, 0x34, 0xC0,
//0xffea
0x34, 0xC0, 0x34, 0xC0, 0x34, 0xC0, 0x34, 0xC0,
0x34, 0xC0, 0x34, 0xC0, 0x34, 0xC0, 0x34, 0xC0,
0x34, 0xC0, 0x34, 0xC0, 0x22, 0xC0,
};
const uint32_t flash_address[] =
0xc000, 0xffde, 0xffe4, 0xffea,
};
const uint32_t flash_length_of_sections[] =
{
70, 4, 4, 22,
};
```

# TEXAS INSTRUMENTS

www.ti.com

```
const uint32_t flash_sections = 4;
const uint32_t flash_termination = 0x00000000; /*Check device data sheet*/
const uint32_t flash_start = 0x00000000; /*Check device data sheet*/
const uint32_t flash_finish = 0x00000000; /*Check device data sheet*/
const uint32_t flash_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 70 bytes of data in flash must be written to the MSP430 MCU, starting at address 0xC000.

For address 0xFFDE

Starting from the 71<sup>st</sup> element in the flash variable, 4 bytes must be written to the MSP430 MCU, starting at address 0xFFDE.

For address 0xFFE4

Starting at the 75<sup>th</sup> element in the flash variable, 4 bytes must be written to the MSP430 MCU, starting at address 0xFFE4.

For address 0xFFEA

Starting at the 79<sup>th</sup> element in the flash variable, 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
* /
uint8_t fram[] =
{
//0xf100
0xB2, 0x40, 0x80, 0x5A, 0xCC, 0x01, 0xD2, 0xC3,
0x02, 0x02, 0xD2, 0xD3, 0x04, 0x02, 0x92, 0xC3,
0x30, 0x01, 0xD2, 0xE3, 0x02, 0x02, 0x1E, 0x14,
0x3D, 0x40, 0x3C, 0x82, 0x3E, 0x40, 0x0E, 0x00,
0x1D, 0x83, 0x0E, 0x73, 0xFD, 0x23, 0x0D, 0x93,
0xFB, 0x23, 0x1D, 0x16, 0xF2, 0x3F, 0x03, 0x43,
0x03, 0x43, 0xFF, 0x3F, 0x03, 0x43, 0x1C, 0x43,
0x10, 0x01, 0x31, 0x40, 0x00, 0x24, 0xB0, 0x13,
```



Create Custom MSP430 Firmware Image

```
0x36, 0xF1, 0x0C, 0x43, 0xB0, 0x13, 0x00, 0xF1,
0xB0, 0x13, 0x30, 0xF1, 0x32, 0xD0, 0x10, 0x00,
0xFD, 0x3F, 0x03, 0x43,
//0xff80
OxFF, OxFF, OxFF, OxFF, OxFF, OxFF, OxFF, OxFF, OxFF,
//0xffe2
0x4C, 0xF1, 0x4C, 0xF1, 0x4C, 0xF1, 0x4C, 0xF1,
0x4C, 0xF1, 0x4C, 0xF1, 0x4C, 0xF1, 0x4C, 0xF1,
0x4C, 0xF1, 0x4C, 0xF1, 0x4C, 0xF1, 0x4C, 0xF1,
0x4C, 0xF1, 0x4C, 0xF1, 0x3A, 0xF1,
};
const uint32_t fram_address[] =
0xf100, 0xff80, 0xffe2,
};
const uint32_t fram_length_of_sections[] =
{
84, 8, 30,
};
const uint32_t fram_sections
                                = 3;
const uint32_t fram_termination = 0x00000000; /*Check device data sheet*/
                            = 0x0000000; /*Check device data sheet*/
const uint32_t fram_start
const uint32_t fram_finish
                                = 0x0000000; /*Check device data sheet*/
                                = 0x0000000; /*Check device data sheet*/
const uint32_t fram_length
```

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

Texas Instruments

www.ti.com

# 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 24, Figure 25, and Figure 26 show how to generate a TI-TXT hex file in CCS.

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

Properties for FR2311_blinky	MSP430 Hex Utility	(2 ▼ □) ▼
<ul> <li>&gt; Resource</li> <li>General</li> <li>Build</li> <li>MSP430 Compiler</li> <li>Processor Options</li> </ul>	Configuration: Debug	[Active ]  Manage Configurations
Optimization Include Options ULP Advisor Advice Options Predefined Symbols	Command:	"\$(CG_TOOL_HEX)"
<ul> <li>&gt; Advanced Options</li> <li>&gt; MSP430 Linker</li> <li>MSP430 Hex Utility</li> <li>General Options</li> <li>Diagnostics Options</li> </ul>	Summary of flags set: memwidth=8romw	idth=8ti_txt
Output Format Options Load Image Options Debug		Ŧ
Show advanced settings		OK Cancel

Figure 24. Enabling MSP430 Hex Utility

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

type filter text	Output Format Options 🗢 🔻 🗘 🔻
<ul> <li>&gt; Resource</li> <li>General</li> <li>Build</li> <li>MSP430 Compiler</li> <li>Processor Options</li> <li>Optimization</li> </ul>	Configuration: Debug [Active]  Manage Configurations Output format Output TI-TXT hex format (ti txt)
Include Options ULP Advisor Advice Options Predefined Symbols Advanced Options MSP430 Linker MSP430 Hex Utility General Options Diagnostics Options Output Format Options Load Image Options Debug	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 TL Togged hex format (d_agged, -b) Output TL-TXT hex format (ti_txt)
Show advanced settings	OK Cancel

Figure 25. MSP430 TI-TXT Hex Format

4. After the project is built, the .txt containing the full firmware image is created. Figure 26 shows this file.

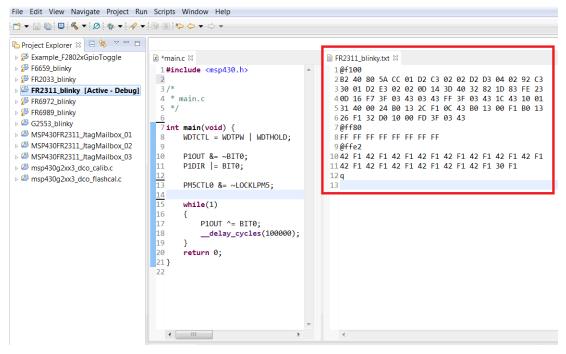


Figure 26. MSP430FR2311 TI-TXT Hex Format Firmware Image

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 (see Figure 27). The scripts are placed 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.

TEXAS

ISTRUMENTS

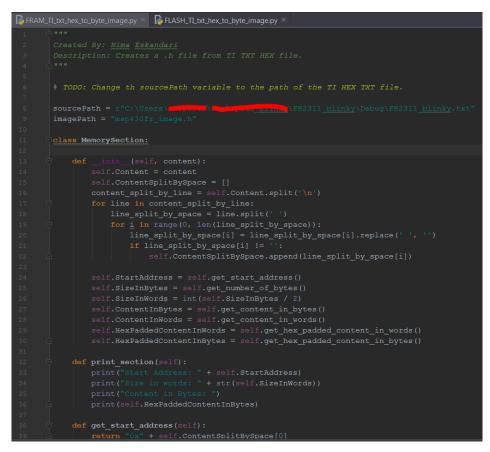


Figure 27. FRAM TI-TXT Hex to Firmware Image Converter

The variable *sourcePath* must be modified to point to the TI-TXT hex file to convert to the firmware image header file. The *imagePath* variable defines the location for the output firmware image header file. Use Python 3 to execute the script and generate the firmware image header file (see Figure 28).

C:\Users\	\Programs	\Python\Python35	\python.exe
C:/Users/			
/FRAM_TI_txt_hex_to_byte	e_image.py		
Start Address: 0xf100			
Size in words: 37			
Content in Bytes:			
['0xB2', '0x40', '0x80',	'0x5A', '0xCC',		
<pre>\'0x02', '0xD2', '0xD3',</pre>	'0x04', '0x02',		'0x30', '0x01', 2
<pre>\$'0xD2', '0xE3', '0x02',</pre>		'0x14', '0x3D',	'0x40', '0x32', 2
\$'0x82', '0x1D', '0x83',	'0xFE', '0x23',	'0x0D', '0x16',	'0xF7', '0x3F', 2
\$'0x03', '0x43', '0x03',	'0x43', '0xFF',	'0x3F', '0x03',	'0x43', '0x1C', 2
\$'0x43', '0x10', '0x01',	'0x31', '0x40',	'0x00', '0x24',	'0xB0', '0x13', 2
S'0x2C', '0xF1', '0x0C',	'0x43', '0xB0',	'0x13', '0x00',	'0xF1', '0xB0', 2
S'0x13', '0x26', '0xF1',	'0x32', '0xD0',	'0x10', '0x00',	'OxFD', 'Ox3F', 2
\$'0x03', '0x43']			
Start Address: 0xff80			
Size in words: 4			
Content in Bytes:			
['OxFF', 'OxFF', 'OxFF',	'OxFF', 'OxFF',	'OxFF', 'OxFF',	'OxFF']
Start Address: 0xffe2			
Size in words: 15			
Content in Bytes:			
['0x42', '0xF1', '0x42',	'0xF1', '0x42',	'0xF1', '0x42',	'0xF1', '0x42',
'0xF1', '0x42', '0xF1',	'0x42', '0xF1',	'0x42', '0xF1',	'0x42', '0xF1',
'0x42', '0xF1', '0x42',	'0xF1', '0x42',	'0xF1', '0x42',	'0xF1', '0x42',
'0xF1', '0x30', '0xF1']			
Process finished with ex:	it code 0		

Figure 28. Firmware Image Python Script Console Output



Error Messages

www.ti.com

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

### 6 Error Messages

If there are any errors in the setup or if the UART BSL state machine is out of sync with the host, error messages can be seen in the serial terminal (see Figure 29).

COM49 - PuTTY		
ISP430 programmed successfully		
Device is reset		
****New Attempt****		
Receiving Data:		
0x0		
x80 x2		
x2 )x0		
IX3b		
lx5		
lxc5		
x94		
one Receiving Data		
Irite password failed		
****New Attempt****		
A A NEW ALLEMPLANAAA		
Receiving Data:		
0		
x80 x2		

Figure 29. UART BSL Error

To debug this issue further, change the DEBUG macro in config.h to 1. More information is printed to the serial terminal.

The issue in is usually caused by the UART BSL state machine getting out of sync with the host. This is suggested by the fact that the 0x80 byte is received as the second byte in the packet.

If reviewing the error messages does not give any hints as to what the source of the failure is, try to debug and single step through the ProgramMSP430() in bsl\_main.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.

#### IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

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

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

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

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

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

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

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

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

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2017, Texas Instruments Incorporated