User's Guide
SLAU622G–March 2015–Revised December 2017

MSP432P4xx SimpleLink™ Microcontrollers Bootloader (BSL)

The SimpleLink™ MSP432™ microcontroller (MCU) bootloader (BSL, formerly known as the bootstrap loader) lets users communicate with embedded memory in the MSP432 microcontroller during the prototyping phase, final production, and in service. Both the programmable memory (flash memory) and the data memory (RAM) can be modified as required. Do not confuse the bootloader with the bootstrap loader programs found in some digital signal processors (DSPs) that automatically load program code (and data) from external memory to the internal memory of the DSP.

Contents

Trademarks

1 Introduction

To use the bootloader, a user-selectable BSL entry sequence must be applied. An added sequence of commands initiates the desired function. A boot-loading session can be exited by continuing operation at a defined user program address or by the reset condition.

If the device is secured by disabling JTAG, it is still possible to use the BSL. Access to the MSP432 memory through the BSL is protected against misuse by a user-defined password.

To avoid accidental overwriting of the BSL code, the code is protected in the flash by default. To prevent unwanted source readout, any BSL command that directly or indirectly allows data reading is password protected. For more information about password-protected commands, see **Section 4.3**.

To invoke the bootloader, the BSL entry sequence must be applied to dedicated pins. After that, the BSL header character, followed by the data frame of a specific command, initiates the desired function.

1.1 Other Resources

MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual

Bootloader (BSL) for MSP Low-Power Microcontrollers

Additional support is provided in support forums at the TI E2E™ Community.

1.2 Overview of BSL Features

Table 1 summarizes the BSL features of the MSP devices, organized by device family.

Table 1. BSL Overview

			MSP430								MSP432
			G2xx0, G2xx1, G2xx2, I20xx	F1xx, F2xx, F4xx, G2xx3	F5xx, F6xx		FR5xx, FR6xx		FR2x33, FR231x	FR413x, FR211x	P401R
					Non-USB	USB	Factory	Crypto-Boot-loader[1]			
General	BSL memory type		No BSL	ROM	Flash[2]	Flash[2]	ROM	FRAM	ROM	ROM	Flash[2]
	BSL memory size		N/A	1 KB	2 KB	2 KB	2 KB	4 KB	3 KB	1 KB	8 KB
	Peripheral configured by TLV						✓	✓			✓
	User configuration										✓
	UART			✓	✓		✓	✓	✓	✓	✓
	I^2C				✓		✓	✓	✓		✓
	SPI										✓
	USB					✓					
Protocol	'1xx, 2xx, 4xx' protocol			✓							
	'5xx, 6xx' protocol				✓	✓	✓	✓	✓	✓	✓
Invoke mechanism	Entry sequence on I/Os	Sequence on TEST/RST		✓	✓		✓	✓	✓	✓	
		PUR pin tied to V_{USB}				✓					
		Sequence on defined I/O						✓			✓
	Empty reset vector invokes BSL					✓		✓	✓		✓
	Calling BSL from software application			✓	✓	✓	✓	✓	✓	✓	✓
	Invalid or incomplete application							✓			
Tools Support	Hardware	MSP-BSL 'Rocket'		✓	✓		✓	✓	✓	✓	✓
		MSP-FET		✓	✓		✓	✓	✓	✓	✓[3]
		USB cable				✓					
	Software[1]	BSL Scripter			✓	✓	✓	✓	✓	✓	✓
		BSLDEMO		✓							
		MSPBSL library		✓	UART only	✓	UART only			✓	
Security	Password protection			32 byte	32 byte[4]	32 byte	32 byte		32 byte	32 byte	256 byte
	Mass erase on incorrect password[5]			✓	✓	✓	✓		✓	✓	✓
	Completely disable the BSL using signature or erasing the BSL				✓	✓	✓	✓	✓	✓	✓
	BSL payload encryption							✓			✓[6]
	Update of IP protected regions through boot code										✓
	Authenticated encryption							✓			
	Additional security							✓[7]			

(1) All BSL software collateral (application, examples, source code, and firmware images) is available in the BSL tool folder. The MSP430 USB developers package includes additional USB BSL sample applications.
(2) BSL in flash memory allows to replace the BSL with a custom version.
(3) MSP-FET supports UART and I^2C BSL communication only.
(4) F543x (non A) has a 16-byte password.
(5) Some devices can disable mass erase on incorrect password. See the device family user's guide.
(6) The decryption of the payload is performed by the device bootcode.
(7) Firmware validation through CRC.

2 BSL Architecture

2.1 Physical Interfaces

The MSP432 BSL is implemented on UART, I^{2}C, and SPI serial interfaces. In MSP432 devices, the BSL can automatically select the interface used to communicate with the device. The specific instance of the peripheral interfaces that is used depends on the selected device and can be found in the device-specific data sheet. This information is also part of the Device Descriptor Table (TLV) table, which is used by the BSL to select the correct instance of the interface. The BSL parses the TLV information and dynamically assesses the ports and interfaces to use. All interface setup calls from the BSL use the Driver Library in the ROM of the device. For additional information about the TLV parameters, see Section A.1.

2.2 UART Protocol Definition

The UART protocol used by the BSL is defined as:

1. Automatic baud-rate detection is run by the BSL slave to determine the initial baud rate of the host that is trying to connect into the BSL. The host can connect to the BSL with any of the following initial baud rates:
   - 9600 baud
   - 19200 baud
   - 38400 baud
   - 57600 baud
   - 115200 baud

   The sync character for the baud rate to be transmitted by the host is 0xFF. The BSL responds with 0x00 in the BSL response cycle if the communication was successfully established with the BSL. If the sync character could not be detected successfully, it must be sent again. The baud-rate tolerance for the sync character is ±4% from the standard baud-rate mentioned above.
2. The protocol used for the communication is: Start bit, 8 data bits (LSB first), an even parity bit, 1 stop bit.
3. Handshake for commands is performed by an acknowledge character in the BSL core response format as specified in Table 12.
4. TI recommends waiting 1.2 ms after receiving a response from the BSL and sending the next command.

2.3 I^{2}C Protocol Definition

The I^{2}C protocol used by the BSL is defined as:

1. Master must request data from BSL slave.
2. 7-bit addressing mode is used, and by default, the slave listens to address 0x48. (Can be changed using a boot override).
3. In addition to the I^{2}C protocol based hardware ACK, handshake for commands is performed by an acknowledge character in the BSL core response format as specified in Table 14
4. Repeated starts are not required by the BSL but can be used.
5. TI recommends waiting 1.2 ms after sending a command to the BSL and starting to receive the response. TI also recommends waiting 1.2 ms before sending the next command after a response was received.
6. The I^{2}C BSL interface supports a maximum clock speed of 1 MHz.

2.4 SPI Protocol Definition

The SPI protocol used by the BSL is defined as:

1. 4-pin SPI slave mode is used. Data is changed on the first clock edge and captured on the following edge. (CKPH = 0). Clock is high when inactive (CKPL = 1). Slave transmit enable (STE) is active low.
2. Master must request data from the BSL slave by transmitting a dummy 0xFF byte. Dummy transmission is needed to drive the clock lines. Until the next BSL response is calculated, that last byte in the SPI buffer will be output.
3. 8-bit serial data character format (MSB first transmit and receive) is used.
4. Handshake for commands is performed by an acknowledge character in the BSL core response format as specified in Table 16.
5. TI recommends waiting 1.2 ms after sending a command to the BSL and starting to receive the response. TI also recommends waiting 1.2 ms before sending the next command after a response was received.
6. The SPI BSL interface supports a maximum clock speed of 1 MHz.

See Section 4.8.1 for more information on the BSL SPI communication.

3 General BSL Information

3.1 BSL Memory Layout

Table 2 shows the BSL code space.

Table 2. BSL Code Space

Space	Flash Region	Length (Bytes)	Start Address
Code	Flash information memory: Bank 2 sectors 1 and 2	0x2000	0x0020:2000
Data	SRAM CTL Bank 0	0x800	0x2000:0000

3.2 BSL Memory Considerations

After initialization, the BSL uses RAM between the addresses 0x2000:0000 and 0x2000:07FF for data buffer and local variables. When invoking the BSL from a main application, the contents of RAM may be lost.

3.3 BSL Invocation

The MSP432 BSL is invoked by any of the three following approaches. Any one of the conditions is a sufficient condition for BSL entry.

1. The BSL is called by the bootcode when the application memory is erased. Bootcode reads out addresses 0x0 and 0x4 from the application flash memory and compares it with 0xFFFFFFFF to determine whether or not the application memory is erased.
2. The BSL is called by application software (see Section 3.3.1).
3. The BSL is called by the device bootcode by applying a hardware entry sequence (see Section 3.3.2).

NOTE: When the bootcode calls the BSL, the watchdog timer is halted. This applies for the TI BSL as well as for custom BSLs. If a custom BSL requires a running watchdog timer, the custom BSL must configure and start it.

3.3.1 Software BSL Invocation

The BSL has an API table at address 0x0020:2000 that the application can call. The table contains the address of the function that starts the BSL execution. The application must pass a certain set of parameters that determines the following:

1. BSL interface (automatic, UART, I²C, SPI)
2. BSL I²C slave address (default = 0x48)

The API table contains the address of the BSL entry function. This function takes a 32-bit argument that is formatted the same as the BSL hardware invoke parameters in the Boot Override Flash Mailbox (see Table 3).

Table 3. BSL Entry Function Parameters

Bit	Value
31	N/A for BSL as entry function parameter.
30:26	Reserved. Default should be 0x1F, but N/A for BSL as entry function parameter.
25:16	I²C slave address. Default = 0x48.
15:13	Interface selection. 7h = Automatic 6h = UART 5h = SPI 4h = I²C 3h to 0h = Reserved for future expansion; defaults to automatic mode
12	N/A for BSL as entry function parameter.
11:7	Reserved. Default should be 0x1F, but N/A for BSL as entry function parameter.
6:4	N/A for BSL as entry function parameter.
3:0	N/A for BSL as entry function parameter.

The following C code example demonstrates how the address from the API table is used to start the BSL and pass the BSL entry function parameters by the application:

```
#define BSL_PARAM          0xFC48FFFF  // I2C slave address = 0x48, Interface selection = Auto
#define BSL_API_TABLE_ADDR  0x00202000  // Address of BSL API table
#define BSL_ENTRY_FUNCTION  (*((uint32_t *)BSL_API_TABLE_ADDR))

((void (*)())BSL_ENTRY_FUNCTION)((uint32_t)BSL_PARAM);  // Call the BSL with given BSL parameters
```

To call the BSL from interrupt routine service, the NVIC IABR register must be cleared before invoking the BSL. The following examples demonstrates the BSL invocation by P6.7 which is set with low active interrupt.

```
//variable detect if we interrupt occurs
volatile bool jumpToBsl = false;

int main()
{
   // Stop watchdog
   MAP_WDT_A_holdTimer();

   // Setup P1.0 with LED output to detect if the interrupt occurs
   MAP_GPIO_setAsOutputPin(GPIO_PORT_P1, GPIO_PIN0);

   // Configure P6.7 as an input and enabling the interrupt
   MAP_GPIO_setAsInputPinWithPullDownResistor(GPIO_PORT_P6, GPIO_PIN7);
   MAP_GPIO_interruptEdgeSelect(GPIO_PORT_P6, GPIO_PIN7, GPIO_HIGH_TO_LOW_TRANSITION);
   MAP_GPIO_clearInterruptFlag(GPIO_PORT_P6, GPIO_PIN7);
   MAP_GPIO_enableInterrupt(GPIO_PORT_P6, GPIO_PIN7);
   MAP_Interrupt_enableInterrupt(INT_PORT6);
   MAP_Interrupt_enableMaster();
```

```
   while(1)
   {
      MAP_GPIO_toggleOutputOnPin(GPIO_PORT_P1, GPIO_PIN0);
      __delay_cycles(2000000);

      if (jumpToBsl)
   {
         jumpToBsl = false;
         MAP_Interrupt_disableMaster();
         NVIC->ICER[0] = 0xFFFF;
         NVIC->ICPR[0] = 0xFFFF;
         NVIC->ICER[1] = 0xFFFF;
         NVIC->ICPR[1] = 0xFFFF;
         ((void (*)())BSL_ENTRY_FUNCTION)((uint32_t)BSL_PARAM); // Call the BSL with
given BSL parameters
      }
   }
}

…

void PORT6_IRQHandler(void)
{
    uint32_t status;

    status = MAP_GPIO_getEnabledInterruptStatus(GPIO_PORT_P6);
    MAP_GPIO_clearInterruptFlag(GPIO_PORT_P6, status);

    /* Toggling the output on the LED */
    if(status & GPIO_PIN7)
    {
        jumpToBsl = true;
    }
}
```

3.3.2 Hardware BSL Invocation

Hardware invocation on the MSP432 BSL differs from the hardware invocation on the MSP430™ devices. On MSP432 devices, the BSL invocation takes advantage of the boot-override mechanism (see the SYSCTL chapter in the *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual* for details on boot overrides). The boot-override command used for setting up BSL for hardware invocation is BSL_PARAMS.

The boot-override allows to:

- Enable or disable the BSL
- Configure the I^2C slave address
- BSL interface (UART, I^2C, or SPI)
- Hardware invoke through GPIO

Hardware invocation through GPIO is supported on Port 1 to Port 3, Bit 0 to Bit 7. It can be configured to invoke the BSL on a high or low level of the corresponding pin.

NOTE: The hardware-based BSL invocation enable is different from the "BSL enable" in the boot-override fields. The hardware-based BSL invocation enable is not affected by using the "BSL Enable" field of the boot override mailbox. The user must configure the hardware-based BSL invoke using the boot-override mailbox with the "BSL hardware invoke parameters".

For example, assume a user wants to use the I²C mode of BSL with a slave address of 0x42, and wants to use bit 0 of port 1 at level 1 as the BSL entry sequence. The user must populate the mailbox command of BSL_PARAMS with the following:

1. BSL enable = 0x0
2. BSL start address = TI BSL API table address
3. BSL hardware invocation parameters = 0x7C429F80.
   Bit 31 = 0 (Enable)
   Bits 25:16 = 0x42 (I²C address)
   Bits 15:13 = 0x4 (I²C)
   Bit 12 = 1 (Level 1)
   Bits 6:4 = 0x0 (BIT0)
   Bits 3:0 = 0x0 (PORT1)

After this, the user must issue a reboot reset into the device to enable the hardware-based BSL invocation in the system. From then on, the device enters BSL whenever it finds the sequence as provided in Figure 2 (this figure is applicable for the previous example).

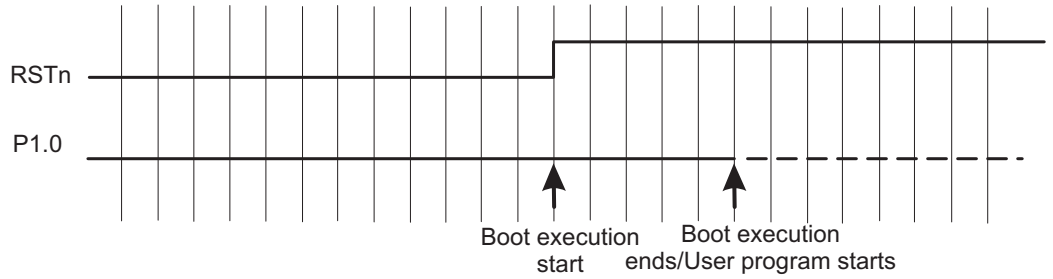

Figure 1. Standard Reset Sequence

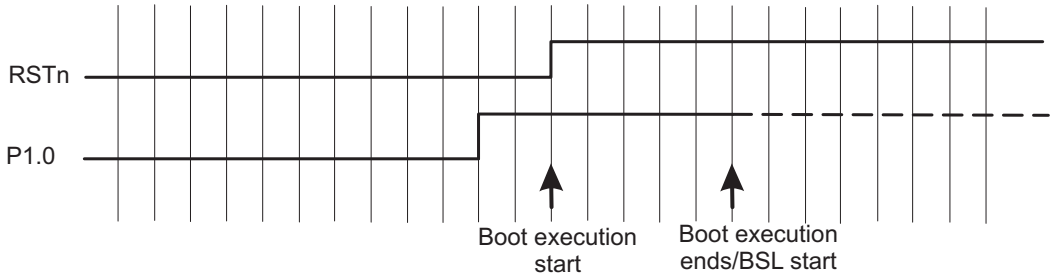

Figure 2. BSL Entry Sequence at Configured GPIO Pin

3.4 Low-Power Mode Support

The MSP432 BSL supports low-power mode of operation. If the device starts up into the BSL, and the BSL does not detect any user input within 10 seconds, the BSL puts the device into the lowest power mode of the device, LPM4.5. After the device BSL enters the LPM4.5 mode of operation, only a pulse on the device $\overline{\text{RST}}$ pin can bring the device back up again.

NOTE: Upon LPM4.5 entry, the BSL sets the I/Os to input direction. Make sure the device I/Os are terminated properly in the board design.

3.5 Debug Considerations

The BSL runs as a typical user application and can be debugged in the same way.

NOTE: If the debugger is used to wake up the BSL from LPM4.5, the BSL does not enter LPM4.5 again as long as the debugger keeps SYSPWRUPREQ alive.

Debug symbols for the MSP432 BSL are available for download in the "MSP432 MCUs Custom BSL Package" available on the MSP BSL tool folder. *Debugging DriverLib in ROM on MSP432P4xx Microcontrollers* describes how to load DriverLib debug symbols into an existing Code Composer Studio™ IDE project. The same method is applicable for the MSP432 BSL debug symbols to debug the BSL.

4 BSL Protocol

4.1 BSL Data Packet

The BSL data packet has a layered structure. The BSL core command contains the actual command data to be processed by the BSL. In addition the standard BSL commands, there can be wrapper data before and after each core command known as the peripheral interface code (PI Code). This wrapper data is information that is specific to the peripheral and protocol being used, and it contains information that allows for correct transmission of the BSL core command. Taken together, the wrapper and core command constitute a BSL data packet.

Table 4. BSL Data Packet

PI Code	BSL Core Command	PI code

4.2 BSL Security

To protect data within the device, most core commands are protected. A protected command is successfully complete only after the device has been unlocked by sending the RX Password command with the correct password. In addition, commands specific to the peripheral interface are not protected.

4.3 BSL Core Commands

Table 5 shows the format of the BSL core commands.

Table 5. BSL Core Commands

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
RX Data Block	Yes	0x10	(A0)	(A1)	(A2)	–	D1...Dn	Yes
RX Data Block 32	Yes	0x20	(A0)	(A1)	(A2)	(A3)	D1...Dn	Yes
RX Password	No	0x21	–	–	–	–	D1...D256	Yes
Erase Sector	Yes	0x12	(A0)	(A1)	(A2)	–	–	Yes
Erase Sector 32	Yes	0x22	(A0)	(A1)	(A2)	(A3)	–	Yes
Mass Erase	Yes	0x15	–	–	–	–	–	Yes
Reboot reset	No	0x25	–	–	–	–	–	No
CRC Check	Yes	0x16	(A0)	(A1)	(A2)	–	Length (low byte), Length (high byte)	Yes
CRC Check 32	Yes	0x26	(A0)	(A1)	(A2)	(A3)	Length (low byte), Length (high byte)	Yes
Load PC	Yes	0x17	(A0)	(A1)	(A2)	–	–	No
Load PC 32	Yes	0x27	(A0)	(A1)	(A2)	(A3)	–	No
TX Data Block	Yes	0x18	(A0)	(A1)	(A2)	–	Length (low byte), Length (high byte)	Yes
TX Data Block 32	Yes	0x28	(A0)	(A1)	(A2)	(A3)	Length (low byte), Length (high byte)	Yes
TX BSL Version	Yes	0x19	–	–	–	–	–	Yes
Factory Reset	No	0x30	–	–	–	–	D1...D16	No
Change Baud Rate	No	0x52	–	–	–	–	D1	No

A0, A1, A2, A3
Address bytes. These represent the 4 bytes from LSB to MSB, respectively, of a 32 bit address.

D1...Dn
Data bytes 1 through n.

Length
A byte containing a value from 1 to 255 describing the number of bytes to be transmitted or used in a CRC. In the case of multiple length bytes, they are combined together as described to form a larger value describing the number of required bytes.

–
No data required. No delay should be given, and any subsequently required data should be sent as the immediate next byte.

4.3.1 RX Data Block

Table 6 shows the RX data block format.

Table 6. RX Data Block

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
RX Data Block	Yes	0x10	(A0)	(A1)	(A2)	–	D1…Dn	Yes

Description
The BSL core writes bytes D1 through Dn starting from the location specified in the address fields. This command has been kept for backward compatibility with MSP430 BSL and allows addressing the lower 24 bits of the device.

Protection
This command is password protected and fails if the password has not been sent.

Command
0x10

Command Address
Address where the received data should be written.

Command Data
Command consists of the data D1 through Dn to be written. The command consists of n bytes, where n has a maximum of 256.

Command Returns
BSL acknowledgment and a BSL core response with the status of the operation.
See Table 9 for more information on BSL core responses.

Command Example
Write data 0x76543210 to address 0x0001:0000:

Header	Length	Length	CMD	A0	A1	A2	D1	D2	D3	D4	CKL	CKH
0x80	0x08	0x00	0x10	0x00	0x00	0x01	0x10	0x32	0x54	0x76	0x93	0xCA

BSL response for a successful data write:

ACK	Header	Length	Length	CMD	MSG	CKL	CKH
0x00	0x80	0x02	0x00	0x3B	0x00	0x60	0xC4

4.3.2 RX Data Block 32

Table 7 shows the RX data block 32 format.

Table 7. RX Data Block 32

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
RX Data Block 32	Yes	0x20	(A0)	(A1)	(A2)	(A3)	D1…Dn	Yes

Description

The BSL core writes bytes D1 through Dn starting from the location specified in the address fields. This command allows the BSL to address the device with the full 32 bit range.

Memory accesses to addresses below 0x20000000 are treated as flash accesses, accesses starting from 0x20000000 are treated as RAM accesses. To write to the SRAM region 0x1000000 to 0x1100000, the alias starting at 0x20000000 must be used.

Protection

This command is password protected and fails if the password has not been sent.

Command

0x20

Command Address

Address where the received data should be written.

Command Data

Command consists of the data D1 through Dn to be written. The command consists of n bytes, where n has a maximum of 256.

Command Returns

BSL acknowledgment and a BSL core response with the status of the operation.

See Table 9 for more information on BSL core responses.

Command Example

Write data 0x76543210 to address 0x0001:0000:

Header	Length	Length	CMD	A0	A1	A2	A3	D1	D2	D3	D4	CKL	CKH
0x80	0x09	0x00	0x20	0x00	0x00	0x01	0x00	0x10	0x32	0x54	0x76	0x66	0x96

BSL response for a successful data write:

ACK	Header	Length	Length	CMD	MSG	CKL	CKH
0x00	0x80	0x02	0x00	0x3B	0x00	0x60	0xC4

4.3.3 RX Password

Table 8 shows the RX password format.

Table 8. RX Password

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
RX Password	No	0x21	–	–	–	–	D1…D256	Yes

Description

The BSL core receives the password contained in the packet and unlocks the BSL protected commands if the password matches the top 256 bytes in the BSL interrupt vector table (located between addresses 0x0 to 0xFF). When an incorrect password is given, a Boot Override factory reset is initiated. This means all code in flash main memory is erased, but not Information Memory.

When a mass erase is performed, the password is always be 0xFF for all bytes. This is commonly used to gain access to an empty device or to load a new application to a locked device without password.

Protection

This command is not password protected.

Command

0x11

Command Address

N/A

Command Data

The command data is 256 bytes long and contains the device password.

Command Returns

BSL acknowledgment and a BSL core response with the status of the operation.

See Table 9 for more information on BSL core responses.

Command Example

Unlock a blank device:

Header	Length	Length	CMD	D1	D2	D3	D4	D5	D6
0x80	0x01	0x01	0x21	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

D7	D8	D9	D10	D11	D12	D13	D14	D15	D16
0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

D17	D18	D19	D20	D21	D22	D23	D24	D25	D26
0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

D27	D28	D29	D30	D31	D32	D33	D34	D35	D36
0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

D37	D38	D39	D40	D41	D42	D43	D44	D45	D46
0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

D47	...	...	...	...	...	...	..	...	D250
0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

D251	D252	D253	D254	D255	D256	CKL	CKH
0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xAD	0x08

BSL response for a successful password:

ACK	Header	Length	Length	CMD	MSG	CKL	CKH
0x00	0x80	0x02	0x00	0x3B	0x00	0x60	0xC4

4.3.4 Erase Sector

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
Erase Sector	Yes	0x12	(A0)	(A1)	(A2)	–	–	Yes

Description

All code flash (main or information flash) in the MSP432 MCU in the sector corresponding to the address is erased except for areas defined as IP Protected secure zones. This function does not erase RAM.

Protection

This command is password protected.

Command

0x12

Command Address

Any address (24 bits) in the sector for which erase is to be performed.

Command Data

N/A

Command Returns

BSL acknowledgment and a BSL core response with the status of the operation.

See Table 9 for more information on BSL core responses.

Command Example

Initiate a Erase sector at address 0x0020:0000:

Header	Length	Length	CMD	A0	A1	A2	CKL	CKH
0x80	0x04	0x00	0x12	0x00	0x00	0x20	0x6D	0x56

BSL response (successful operation):

ACK	Header	Length	Length	CMD	MSG	CKL	CKH
0x00	0x80	0x02	0x00	0x3B	0x00	0x60	0xC4

4.3.5 Erase Sector 32

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
Erase Sector 32	Yes	0x22	(A0)	(A1)	(A2)	(A3)	–	Yes

Description

All code flash (main or information flash) in the MSP432 MCU in the sector corresponding to the address is erased except for areas defined as IP Protected secure zones. This function does not erase RAM.

Protection

This command is password protected.

Command

0x22

Command Address

Any address (32 bits) in the sector for which erase is to be performed.

Command Data

N/A

Command Returns

BSL acknowledgment and a BSL core response with the status of the operation.

See Table 9 for more information on BSL core responses.

Command Example

Initiate a Erase sector at address 0x0020:0000:

Header	Length	Length	CMD	A0	A1	A2	A3	CKL	CKH
0x80	0x05	0x00	0x12	0x00	0x00	0x20	0x00	0x33	0x57

BSL response (successful operation):

ACK	Header	Length	Length	CMD	MSG	CKL	CKH
0x00	0x80	0x02	0x00	0x3B	0x00	0x60	0xC4

4.3.6 Mass Erase

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
Mass Erase	Yes	0x15	–	–	–	–	–	Yes

Description

All code flash in the MSP432 MCU is erased except for areas defined as IP protected secure zones. This function does not erase Information Memory and RAM.

Protection

This command is password protected and fails if the password has not been sent.

Command

0x15

Command Address

N/A

Command Data

N/A

Command Returns

BSL acknowledgment and a BSL core response with the status of the operation.

Command Example

Initiate a mass erase:

Header	Length	Length	CMD	CKL	CKH
0x80	0x01	0x00	0x15	0x64	0xA3

BSL response (successful operation):

ACK	Header	Length	Length	CMD	MSG	CKL	CKH
0x00	0x80	0x02	0x00	0x3B	0x00	0x60	0xC4

4.3.7 Reboot Reset

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
Reboot reset	No	0x25	–	–	–	–	–	No

Description

This command is used to initiate a reboot-reset into the MSP432 MCU. After the reboot reset it takes approximately 10 ms before the BSL is started again.

Protection

This command is not password protected.

Command

0x25

Command Address

N/A

Command Data

N/A

Command Returns

None. The devices resets after successfully receiving the command.

Command Example

Initiate a reboot reset into the MSP432 MCU:

Header	Length	Length	CMD	CKL	CKH
0x80	0x01	0x00	0x25	0x37	0x95

BSL Response:

None

4.3.8 CRC Check

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
CRC Check	Yes	0x16	(A0)	(A1)	(A2)	–	Length (low byte), Length (high byte)	Yes

Description

The MSP432 device performs a 16-bit CRC check using the CCITT standard. The address given is the first

byte of the CRC check. Two bytes are used for the length.

See the CRC chapter of the *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual* for more details on the CRC that is used.

This command is retained from MSP430 for backward compatibility and is capable of address device memory with up to 24 bits of address.

Protection

This command is password protected and fails if the password has not been sent.

Command

0x16

Command Address

Address to begin the CRC check.

Command Data

The 16-bit length of the CRC check. D1 is the low byte of the length, and D2 is the high byte of the length.

Command Returns

BSL acknowledgment and a BSL core response with the CRC value.

See Table 9 for more information on BSL core responses.

Command Example

Perform a CRC check from address 0x0000:4400 to 0x0000:47FF (size of 1024):

Header	Length	Length	CMD	A0	A1	A2	D1	D2	CKL	CKH
0x80	0x06	0x00	0x16	0x00	0x44	0x00	0x00	0x04	0x9C	0x7D

BSL response where 0x55 is the low byte of the calculated checksum and 0xAA is the high byte of the calculated checksum:

ACK	Header	Length	Length	CMD	D1	D2	CKL	CKH
0x00	0x80	0x03	0x00	0x3A	0x55	0xAA	0x12	0x2B

4.3.9 CRC Check 32

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
CRC Check 32	Yes	0x26	(A0)	(A1)	(A2)	(A3)	Length (low byte), Length (high byte)	Yes

Description

The MSP432 device performs a 16-bit CRC check using the CCITT standard. The address given is the first

byte of the CRC check. Two bytes are used for the length.

See the CRC chapter of the *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual* for more details on the CRC that is used.

Protection

This command is password protected and fails if the password has not been sent.

Command

0x26

Command Address

Address to begin the CRC check.

Command Data

The 16-bit length of the CRC check. D1 is the low byte of the length, and D2 is the high byte of the length.

Command Returns

BSL acknowledgment and a BSL core response with the CRC value.

See Table 9 for more information on BSL core responses.

Command Example

Perform a CRC check from address 0x0000:4400 to 0x0000:47FF (size of 1024):

Header	Length	Length	CMD	A0	A1	A2	A3	D1	D2	CKL	CKH
0x80	0x07	0x00	0x26	0x00	0x44	0x00	0x00	0x00	0x04	0xF7	0xE6

BSL response where 0x55 is the low byte of the calculated checksum and 0xAA is the high byte of the calculated checksum:

ACK	Header	Length	Length	CMD	D1	D2	CKL	CKH
0x00	0x80	0x03	0x00	0x3A	0x55	0xAA	0x12	0x2B

4.3.10 Load PC

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
Load PC	Yes	0x17	(A0)	(A1)	(A2)	–	–	No

Description

Causes the BSL to begin execution at the given address. As BSL code is immediately exited with this instruction, no core response can be expected.

This command is retained from MSP430 for backward compatibility and is capable of address device memory with up to 24 bits of address.

You must set the 'T bit' to indicate that Thumb code will be executed. For example, to set this bit when a Thumb code application at 0x8000 is to be executed, Load PC must be called with address 0x8001.

Protection

This command is password protected and fails if the password has not been sent.

Command

0x17

Command Address

Address to set the Program Counter.

Command Data

N/A

Command Returns

BSL acknowledgment

Load PC always return success or a BSL locked status. BSL core response with the status of the operation is sent only if the main application returns to the BSL.

See Table 9 for more information on BSL core responses.

Command Example

Set program counter to 0x0000:4451:

Header	Length	Length	CMD	A0	A1	A2	CKL	CKH
0x80	0x04	0x00	0x17	0x51	0x44	0x00	0xBC	0x66

The BSL does not respond after the application gains control.

4.3.11 Load PC 32

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
Load PC 32	Yes	0x27	(A0)	(A1)	(A2)	(A3)	–	No

Description

Causes the BSL to begin execution at the given address. As BSL code is immediately exited with this instruction, no core response can be expected.

Set the 'T bit' to indicate that Thumb code is to be executed. For example, to set this bit when a Thumb code application at 0x8000 is to be executed, Load PC 32 must be called with address 0x8001.

Protection

This command is password protected and fails if the password has not been sent.

Command

0x27

Command Address

Address to set the Program Counter.

Command Data

N/A

Command Returns

BSL acknowledgment

Load PC always return success or a BSL locked status. BSL core response with the status of the operation is sent only if the main application returns to the BSL.

See Table 9 for more information on BSL core responses.

Command Example

Set program counter to 0x0000:4451:

Header	Length	Length	CMD	A0	A1	A2	A3	CKL	CKH
0x80	0x05	0x00	0x27	0x51	0x44	0x00	0x00	0x8E	0xBC

The BSL does not respond after the application gains control.

4.3.12 TX Data Block

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
TX Data Block	Yes	0x18	(A0)	(A1)	(A2)	–	Length (low byte), Length (high byte)	Yes

Description

The BSL transmits data starting at the command address and with size command data. This command initiates multiple packets if the size is greater than or equal to the buffer size. This command is retained from MSP430 for backward compatibility and is supports addressing of device memory with up to 24 bits of address.

Protection

This command is password protected and fails if the password has not been sent.

Command

0x18

Command Address

Address to begin transmitting data from.

Command Data

The 16-bit length of the data to transmit. D1 is the low byte of the length, and D2 is the high byte of the length.

Command Returns

BSL acknowledgment and a BSL core response with *n* data packets where *n* is:

$$n = \text{ceiling}\left(\frac{length}{buffer\ size - 1}\right)$$

For example, if 512 bytes are requested, BSL sends two packets: the first packet with a length of 262 (1 CMD + 261 data bytes) and the second with a length of 252 (1 CMD + 251 data bytes).

See Table 9 for more information on BSL core responses.

Command Example

Transmit 4 bytes of data from address 0x0000:1C00:

Header	Length	Length	CMD	A0	A1	A2	D1	D2	CKL	CKH
0x80	0x06	0x00	0x18	0x00	0x1C	0x00	0x04	0x00	0x87	0x81

BSL response where D1..D4 are the data bytes requested:

ACK	Header	Length	Length	CMD	D1	D2	D3	D4	CKL	CKH
0x00	0x80	0x05	0x00	0x3A	0x11	0x33	0x55	0x77	0x90	0x55

4.3.13 TX Data Block 32

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
TX Data Block 32	Yes	0x28	(A0)	(A1)	(A2)	(A3)	Length (low byte), Length (high byte)	Yes

Description

The BSL transmits data starting at the command address and with size command data. This command initiates multiple packets if the size is greater than or equal to the buffer size.

Protection

This command is password protected and fails if the password has not been sent.

Command

0x28

Command Address

Address to begin transmitting data from.

Command Data

The 16-bit length of the data to transmit. D1 is the low byte of the length, and D2 is the high byte of the length.

Command Returns

BSL acknowledgment and a BSL core response with *n* data packets where *n* is:

$$n = \text{ceiling}\left(\frac{length}{buffer\ size - 1}\right)$$

For example, if 512 bytes are requested, BSL sends two packets: the first packet with a length of 262 (1 CMD + 261 data bytes) and the second with a length of 252 (1 CMD + 251 data bytes).

See Table 9 for more information on BSL core responses.

Command Example

Transmit 4 bytes of data from address 0x0000:1C00:

Header	Length	Length	CMD	A0	A1	A2	A3	D1	D2	CKL	CKH
0x80	0x07	0x00	0x28	0x00	0x1C	0x00	0x00	0x04	0x00	0x20	0x4F

BSL response where D1..D4 are the data bytes requested:

ACK	Header	Length	Length	CMD	D1	D2	D3	D4	CKL	CKH
0x00	0x80	0x05	0x00	0x3A	0x11	0x33	0x55	0x77	0x90	0x55

4.3.14 TX BSL Version

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
TX BSL Version	No	0x19	–	–	–	–	–	Yes

Description

BSL transmits its version information.

Protection

This command is password protected and fails if the password has not been sent.

Command

0x19

Command Address

N/A

Command Data

N/A

Command Returns

BSL acknowledgment and a BSL core response with its version number. The data is transmitted as it appears in memory with the following data bytes:

Version Byte	Data (High Byte, Low Byte)
BSL Vendor	D1, D2
Command Interpreter	D3, D4
API	D5, D6
Peripheral Interface	D7, D8
Build ID	D9, D10

See Table 9 for more information on BSL core responses.

Command Example

Request the BSL version:

Header	Length	Length	CMD	CKL	CKH
0x80	0x01	0x00	0x19	0xE8	0x62

BSL response (version 0011.2233.4455.6677.8899 of the BSL):

ACK	Header	Length	Length	CMD	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	CKL	CKH
0x00	0x80	0x0B	0x00	0x3A	0x00	0x11	0x22	0x33	0x44	0x55	0x66	0x77	0x88	0x99	0xCF	0x1D

4.3.15 Factory Reset

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
TX BSL Version	No	0x30	–	–	–	–	D1...D16	No

Description

BSL initiates a factory reset boot override request through the flash boot-override mailbox. This removes all security definitions and erases the flash main memory. After sending the factory reset command, it takes approximately 30 ms (depending on the time to erase the flash memory) before the BSL is started again.

Protection

This command is not password protected.

Command

0x30

Command Address

N/A

Command Data

D1...D16 is the 128-bit password for factory reset if any was saved into the device previously. If the password for factory reset was not enabled, the content of D1...D16 is ignored.

Command Returns

BSL acknowledgment

Command Example

Request factory reset with password 0x00, 0x01, ..., 0x0F:

Header	Length	Length	CMD	D1	D2	D3	D4	D5	D6	D7
0x80	0x11	0x00	0x30	0x00	0x01	0x02	0x03	0x04	0x05	0x06

D8	D9	D10	D11	D12	D13	D14	D15	D16	CKL	CKH
0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F	0xD2	0xB4

BSL Response:

None

4.3.16 Change Baud Rate

BSL Command	Protected	CMD	A0	A1	A2	A3	Data	BSL Core Response
Change Baud Rate	No	0x52	–	–	–	–	D1	No

Description

This command changes the baud rate for all subsequently received data packets. The command is acknowledged with either a single ACK or an error byte sent with the old baud rate before changing to the new one. No subsequent message packets can be expected.

Protection

This command is not password protected.

Command

0x52

Command Address

N/A

Command Data

Single byte, D1, that specifies the new baud rate to use.

D1	Baud Rate
0x01	9600
0x03	19200
0x04	38400
0x05	57600
0x06	115200

Command Returns

BSL acknowledgment (sent at the baud rate before the change)

Command Example

Change baud rate to 115200:

Header	Length	Length	CMD	D1	CKL	CKH
0x80	0x02	0x00	0x52	0x06	0x14	0x15

BSL Response:

ACK
0x00

4.4 BSL Core Responses

The BSL core responses are always wrapped in a peripheral interface wrapper with the identical format to that of received commands. The BSL core can respond in the format shown in Table 9. All numbers are in hexadecimal format.

Table 9. BSL Core Responses

BSL Response	CMD	Data
Data Block	0x3A	D1...Dn
BSL Version	0x3A	D1...D10
CRC Value	0x3A	DL, DH
Buffer Size	0x3A	NL, NH
Message	0x3B	MSG

CMD

A required field used to distinguish between a message from the BSL and a data transmission from the BSL.

MSG

A byte containing a response from the BSL core describing the result of the requested action. This can either be an error code or an acknowledgment of a successful operation. It should be noted, in cases where the BSL is required to respond with data (for example, memory, version, CRC, or buffer size), no successful operation reply occurs, and the BSL core immediately sends the data. See Table 10 for BSL core messages.

D1, Dx

Data bytes containing the requested data.

DL, DH

Data low and high bytes, respectively, of a requested 16-bit CRC value.

NL, NH

Data bytes describing the length of the buffer size in bytes. To manage sizes above 255, the size is broken up into a low byte and a high byte.

4.5 BSL Core Messages

Table 10 describes the BSL core messages

Table 10. BSL Core Messages

MSG	Meaning
0x00	Operation Successful
0x04	BSL Locked. The correct password has not yet been supplied to unlock the BSL.
0x05	BSL Password Error. An incorrect password was supplied to the BSL when attempting an unlock.
0x07	Unknown Command. The command given to the BSL was not recognized.

4.6 UART Peripheral Interface (PI)

4.6.1 UART Peripheral Interface Wrapper

The MSP432 UART BSL protocol interface is implemented in multiple packets. The first packet is transmitted to the BSL device and contains the BSL Core Command and its wrapper. This has the format shown in Table 11 . The second packet is received from the BSL device and contains the BSL acknowledgment and the BSL Core Response if one is required by the BSL Core Command that was sent (see Section 4.3 for more information about what commands return a BSL Core Response). The BSL response is shown in Table 12.

Table 11. UART BSL Command

Header	Length	Length	BSL Core Command	CKL	CKH
0x80	NL	NH	See Table 5	CKL	CKH

Table 12. UART BSL Response

ACK	Header	Length	Length	BSL Core Response	CKL	CKH
ACK from BSL	0x80	NL	NH	See Table 9	CKL	CKH

The BSL acknowledgment indicates any errors in the first packet. If a response other than ACK is received, the BSL Core Response is not sent. It is important that the host programmer check this first byte to determine if more data will be sent.

CKL, CKH

CRC checksum high and low bytes. The checksum is computed on bytes in BSL core command section only. The BSL uses CRC-CCITT for the checksum and computes it using the MSP432 CRC module (see the CRC chapter of the *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual* for more details about the CRC hardware).

NL, NH

Number of bytes in BSL core data packet, broken into high and low bytes.

ACK

Sent by the BSL after the packet is received to acknowledge receiving the data correctly. This does not imply the BSL core data is a correct command or that it was executed correctly. ACK signifies only that the packet was formatted as expected and had a correct checksum.

4.7 I²C Peripheral Interface (PI)

4.7.1 I²C Peripheral Interface Wrapper

The MSP432 I²C BSL protocol interface is implemented in multiple packets. The first packet is sent as a write request to the BSL slave address and contains the BSL Core Command and its wrapper. This has the format shown in Table 13. The second packet is sent as a read request to the BSL slave address and contains the BSL acknowledgment and the BSL Core Response if one is required by the BSL Core Command that was sent (see Section 4.3 for more information about what commands return a BSL Core Response). Table 14 shows the format of this BSL response.

Table 13. I²C BSL Command

I²C	Header	Length	Length	BSL Core Command	CKL	CKH
S/A/W	0x80	NL	NH	See Table 5	CKL	CKH

Table 14. I²C BSL Response

I²C	ACK	Header	Length	Length	BSL Core Response(1)	CKL	CKH	I²C
S/A/R	ACK from BSL	0x80	NL	NH	See Table 9	CKL	CKH	STOP

(1) BSL Core Response is not always included.

The BSL acknowledgment indicates any errors in the first packet. If a response other than ACK is received, the BSL Core Response is **not** sent. It is important that the host programmer check this first byte to determine if more data will be sent.

CKL, CKH

CRC checksum high and low bytes. The checksum is computed on bytes in BSL core command section only. The CRC is computed using the MSP432 CRC module specification (see the CRC chapter of the *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual* for implementation details).

NL, NH

Number of bytes in BSL core data packet, broken into high and low bytes.

ACK

Sent by the BSL after the packet is received to acknowledge receiving the data correctly. This does not imply the BSL core data is a correct command or that it was executed correctly. ACK signifies only that the packet was formatted as expected and had a correct checksum.

S/A/W

I²C start sequence sent by the host programmer to the MSP432 BSL slave device. This sequence specifies that the host would like to start a write to the device with the specified slave address. See the eUSCI I²C mode chapter of the *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual* for more details on I²C communication.

S/A/R

I²C start or re-start sequence sent by the host programmer to the MSP432 BSL slave device. This sequence specifies that the host would like to start a read from the device with the specified slave address. This does not need to be a re-start, the host programmer can send a stop followed by another start. See the eUSCI I²C mode chapter of the *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual* for more details on I²C communication.

STOP

I²C stop bit indicating the end of an I²C read or write.

4.8 SPI Peripheral Interface

4.8.1 SPI Peripheral Interface

The MSP432 SPI BSL protocol interface is implemented in multiple packets. The first packet contains the BSL Core Command and its wrapper. This has the format shown in Table 15. The second packet contains the BSL acknowledgment and the BSL Core Response if one is required by the BSL Core Command that was sent (see Section 4.3 for more information about what commands return a BSL Core Response). Table 16 shows the format of this BSL response.

Table 15. SPI BSL Command

Header	Length	Length	BSL Core Command	CKL	CKH
0x80	NL	NH	See Table 5	CKL	CKH

Table 16. SPI BSL Response

ACK	Header	Length	Length	BSL Core Response[1]	CKL	CKH
ACK from BSL	0x80	NL	NH	See Table 9	CKL	CKH

(1) BSL Core Response is not always included.

The BSL acknowledgment indicates any errors in the first packet. If a response other than ACK is received, the BSL Core Response is **not** sent. It is important that the host programmer check this first byte to determine if more data will be sent.

CKL, CKH

CRC checksum high and low bytes. The checksum is computed on bytes in BSL core command section only. The CRC is computed using the MSP432 CRC module specification (see the CRC chapter of the *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual* for implementation details).

NL, NH

Number of bytes in BSL core data packet, broken into high and low bytes.

ACK

Sent by the BSL after the packet is received to acknowledge receiving the data correctly. This does not imply the BSL core data is a correct command or that it was executed correctly. ACK signifies only that the packet was formatted as expected and had a correct checksum.

4.8.2 SPI Communication Example

The host and BSL communicate in master/slave model. The host is controlling the transmission and reception of data. A delay between each byte send might be necessary for the BSL to process the data. The time until the response is calculated is depending on the command executed. E.g. a CRC calculation over a large memory area takes longer than CRC over few bytes or reading the BSL version.

In the following example the TX BSL Version command is send to the BSL and the response is received:

1. The host sends the TX BSL Version command.
   Host: 0x80 0x01 0x00 0x19 0xE8 0x62
2. The host reads the acknowledgment response from the BSL (see Table 16). To receive the byte the hosts sends 0xFF to drive the clock line.
   Host: 0xFF (read)
   BSL: 0x00 (ACK)
3. The host continues to read.
   Host: 0xFF (read)
   BSL: 0x80 (header)
   Host: 0xFF (read)
   BSL: 0x0B (length low)
   Host: 0xFF (read)
   BSL: 0x00 (length high)
   The host can now read as many bytes as defined by the length field and receive the BSL core response and the remaining CRC.
   Host: 0xFF (read)
   BSL: 0x3A (data block, BSL version, CRC value, buffer size BSL core response)
   Host: 0xFF (read)
   BSL: 0x00 (BSL vendor)
   Host: 0xFF (read)
   BSL: 0x00 (BSL vendor)
   Host: 0xFF (read)
   BSL: 0x00 (BSL CI)
   Host: 0xFF (read)
   BSL: 0x02 (BSL CI)
   Host: 0xFF (read)
   BSL: 0x00 (BSL API)
   Host: 0xFF (read)
   BSL: 0x02 (BSL API)
   Host: 0xFF (read)
   BSL: 0x01 (BSL PI)
   Host: 0xFF (read)
   BSL: 0x01 (BSL PI)
   Host: 0xFF (read)
   BSL: 0x00 (BSL build)
   Host: 0xFF (read)
   BSL: 0x02 (BSL build)
   Host: 0xFF (read)
   BSL: 0xD3 (CRC low)
   Host: 0xFF (read)
   BSL: 0x6C (CRC high)

In the previous example, the BSL acknowledged the received command and responded with the BSL version BSL core response. The BSL transmitted BSL version 0000.0002.0002.0101.00.02 with CRC 0x6CD3.

4.9 BSL Messages

The peripheral interface section of the BSL software parses the wrapper section of the BSL data packet. If there are errors with the data transmission, an error message is sent immediately. An ACK is sent after all data has been successfully received and does not mean that the command has been correctly executed (or even that the command was valid) but, rather, that the data packet was formatted correctly and passed on to the BSL core software for interpretation.

The BSL protocol dictates that every BSL data packet sent is responded to with a single byte acknowledgment in addition to any BSL data packets that are sent. Table 17 lists the acknowledgment responses from the BSL. If an acknowledgment byte other than ACK is sent, the BSL does not send any BSL data packets. The host programmer must check the acknowledgment error and retry transmission.

Table 17. Error Messages

Data	Meaning
0x00	ACK
0x51	Header incorrect. The packet did not begin with the required value of 0x80.
0x52	Checksum incorrect. The packet did not have the correct checksum value.
0x53	Packet size zero. The size for the BSL core command was given as 0.
0x54	Packet size exceeds buffer. The packet size given is too big for the RX buffer.
0x55	Unknown error.
0x56	Unknown baud rate. The supplied data for baud rate change is not a known value (UART only).

5 Reprogram and Customize the BSL

In case the BSL code is accidentally erased or corrupted, the BSL firmware image is reprogrammable as it is programmed in Information Flash Memory . TI provides the MSP432P4xx BSL source and project files for CCS: http://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/MSPBSL_CustomBSL432/latest/index_FDS.html. In this package, the BSL firmware image programmed by the factory is also provided in the *Firmware* folder.

Table 18 lists the structure of the MSP432BSL source package 1.1.0 and later.

Table 18. MSP432BSL Source Package Structure

Folder			File	Additional information
Firmware	MSP432P401x	0000.0003.0007.0205.000A	BSL432_MSP432P401.map	Generated map file that shows memory used for BSL application code in BSL programmed by factory.
			BSL432_MSP432P401.out	BSL CCS out file, to be loaded by debugger and having the debug session. It is applicable for MSP432P401R and MSP432P401M.
			BSL432_MSP432P401.txt	BSL firmware image in TI TXT file format. It is programmed from factory for MSP432P401R and MSP432P401M.
	MSP432P4111	0000.0003.0007.0205.000D	BSL432_MSP432P4111.map	Generated map file that shows memory used for BSL application code in BSL programmed by factory.
			BSL432_MSP432P4111.out	BSL CCS out file, to be loaded by debugger and having the debug session. It is applicable for MSP432P4111, MSP432P411V, and MSP432P411Y.
			BSL432_MSP432P4111.txt	BSL firmware image in TI TXT file format. It is programmed from factory for MSP432P401R and MSP432P401M.

Table 18. MSP432BSL Source Package Structure (continued)

Folder			File	Additional information
Source	BSL432	MSP432P401x	.ccsproject	Code Composer Studio (CCS) project files. It is recommended to use minimum CCS v6.0.
			.cproject	
			.project	
			BSL432_device_file.h	Contains the version and build number, also the TLV information
			msp432p401m.cmd	Linker command file for MSP432P401M to be integrated in Release Build configuration
			msp432p401m_debug_flash.cmd	Linker command file for MSP432P401M to be integrated in Debug Build configuration
			msp432p401r.cmd	Linker command file for MSP432P401R to be integrated in Release Build configuration
			msp432p401r_debug_flash.cmd	Linker command file for MSP432P401R to be integrated in Debug Build configuration
		MSP432P4111	.ccsproject	Code Composer Studio (CCS) project files. It is recommended to use minimum CCS v6.0.
			.cproject	
			.project	
			BSL432_device_file.h	Contains the version and build number, also the TLV information
			msp432p4x1v.cmd	Linker command file for MSP432P411V to be integrated in Release Build configuration
			msp432p4x1v _debug_flash.cmd	Linker command file for MSP432P411V to be integrated in Debug Build configuration
			msp432p4x1y.cmd	Linker command file for MSP432P411Y to be integrated in Release Build configuration
			msp432p4x1y_debug_flash.cmd	Linker command file for MSP432P411Y to be integrated in Debug Build configuration
			msp432p4x11.cmd	Linker command file for MSP432P4111 to be integrated in Release Build configuration
			msp432p4x11_debug_flash.cmd	Linker command file for MSP432P4111 to be integrated in Debug Build configuration

Table 18. MSP432BSL Source Package Structure (continued)

Folder			File	Additional information
Source (continued)	DriverLib	3_21_00_05		DriverLib source code and library that is integrated for MSP432P401x BSL project
		3_40_00_11		DriverLib source code and library that is integrated for MSP432P4111 BSL project
			BSL432_API.h, BSL432_API.c	Header and source file for API (memory access for programming, reading, and erasing)
			BSL432_BSL432_Peripheral_Interface.h	Header and source file for peripheral functions (UART, I^2C, and SPI)
			BSL432_Peripheral_Interface_eUSCI_UART_SPI_I2C_IRQ.c	
			BSL432_Command_Interpreter.h, BSL432_Command_Interpreter.c	Header and source file for interpreter BSL command. The main function is written also in this file.
			msp432_startup_ccs.c	Standard interrupt interface for MSP432 MCUs
			manifest.htm	Software manifest for the software deliverable

Perform the following steps to import the project the CCS and configure the build.

1. Download the latest MSP432 BSL source and project from http://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/MSPBSL_CustomBSL432/latest/index_FDS.html.
2. Extract the files from the zip (see Figure 3).

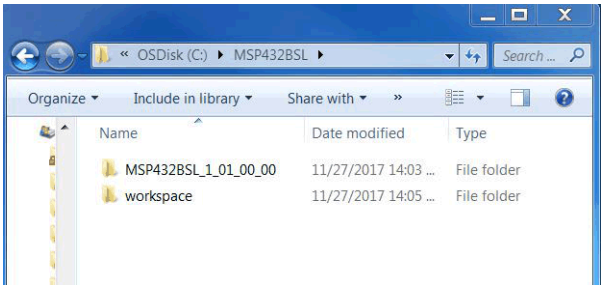

Figure 3. Extract Zip

3. Open the CCS workspace where we would like to have the BSL project (see Figure 4)

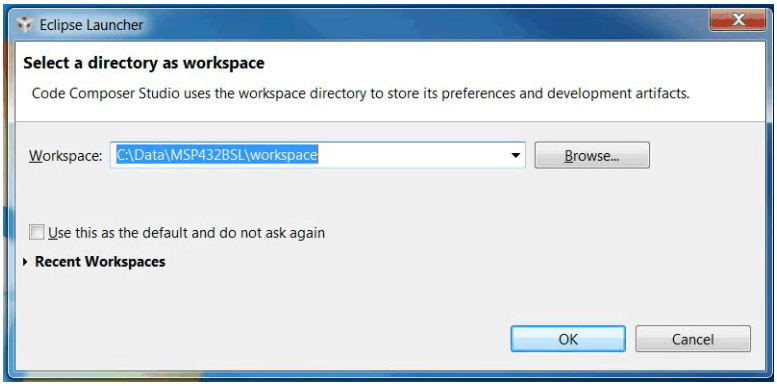

Figure 4. Select Workspace

4. To import the project, click Project → Import CCS Project (see Figure 5).

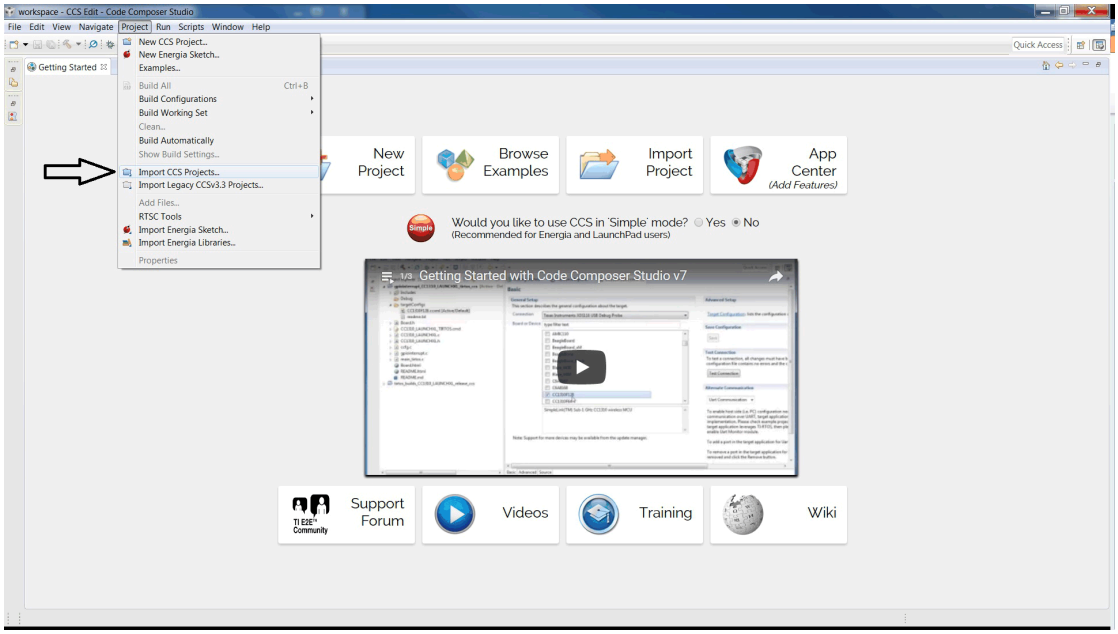

Figure 5. Import CCS Project

5. Select the project the corresponds to the MCU in use (see Figure 6).
   a. BSL432_MSP432P401x for MSP432P401R or MSP432P401M
   b. BSL432_MSP432P4111 for MSP432P4111, MSP432P411Y, or MSP432P411V

   To make a copy of the project in the current workspace, check *Copy projects into workspace*. If this option is not checked, the project and source code remain in the original location, and the workspace links to the files.

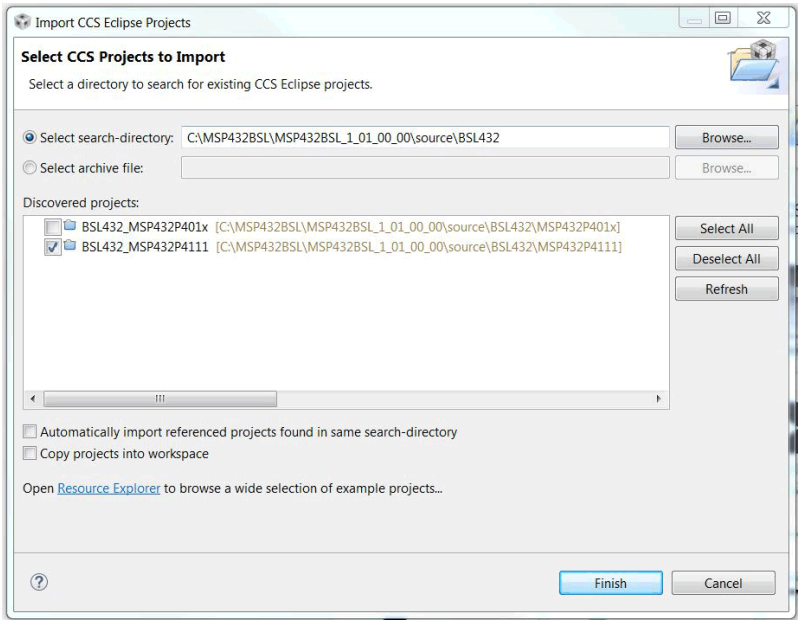

Figure 6. Select CCS Projects to Import

6. Click OK to start the import to the workspace (see Figure 7).

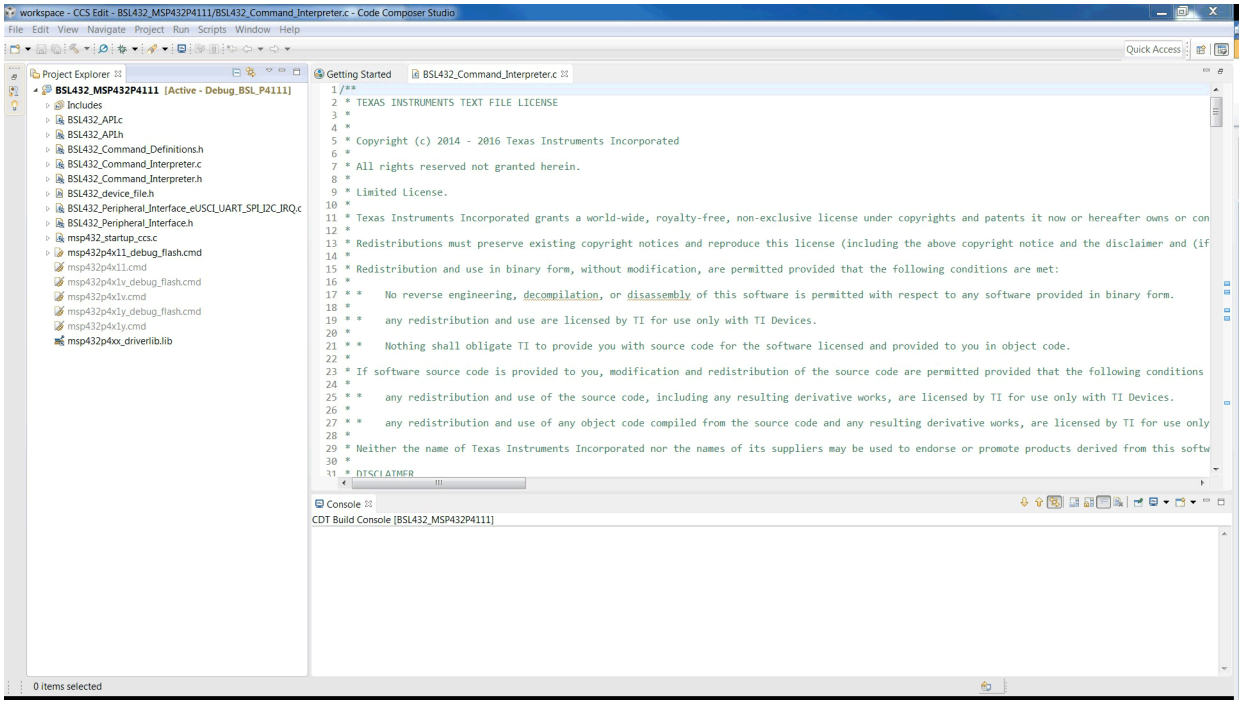

Figure 7. Project Imported

7. In the *Build Configuration* option of the Build icon, there are options available for the MSP432P4111 variants. Select the build option that is intended for the project (see Figure 8).

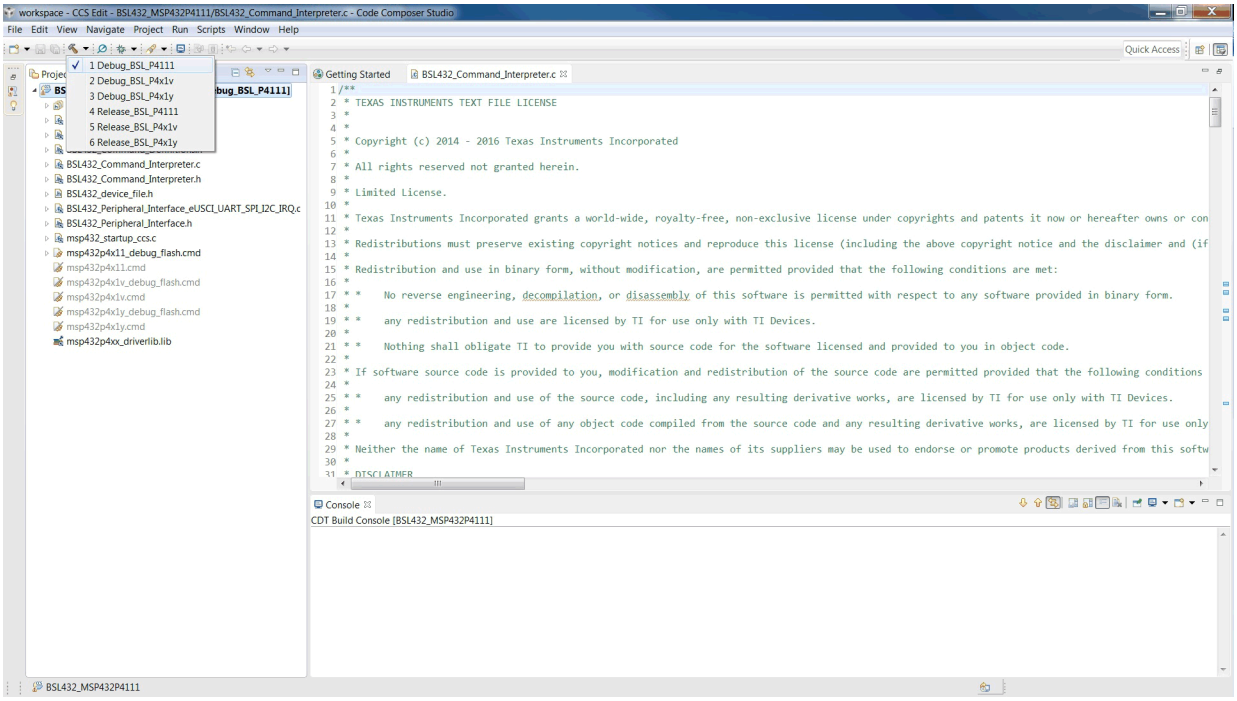

Figure 8. Select Build Option

8. Figure 9 shows an example in which build configuration of Debug_BSL_P4111 and Release_BSL_P4111 are selected. The firmware that is generated for Debug mode will be written to program flash memory, and the firmware that is generated for Release mode will be written to the BSL location in information memory at address 0x0020:0000 (see Figure 9).

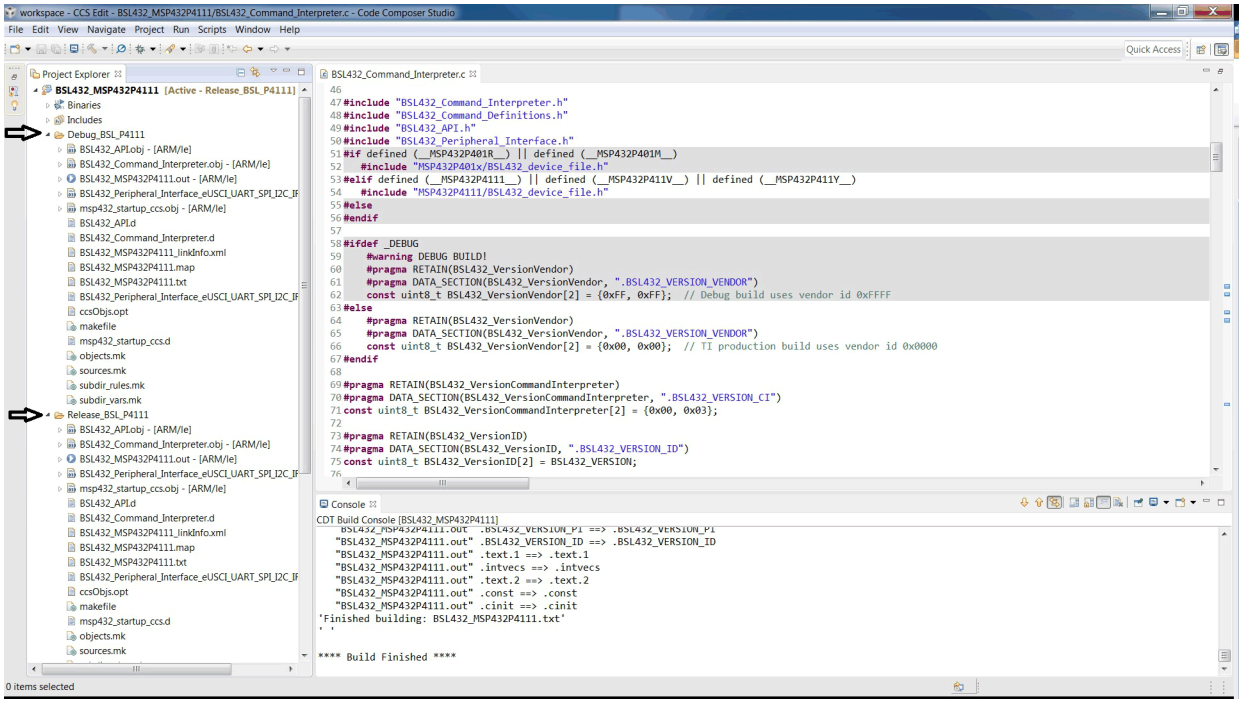

Figure 9. Debug and Release Builds

After the firmware image of the BSL is generated, perform the following steps:

1. Click *View* → *Target Configurations*. The *Target Configurations* window opens in the IDE (see Figure 10).

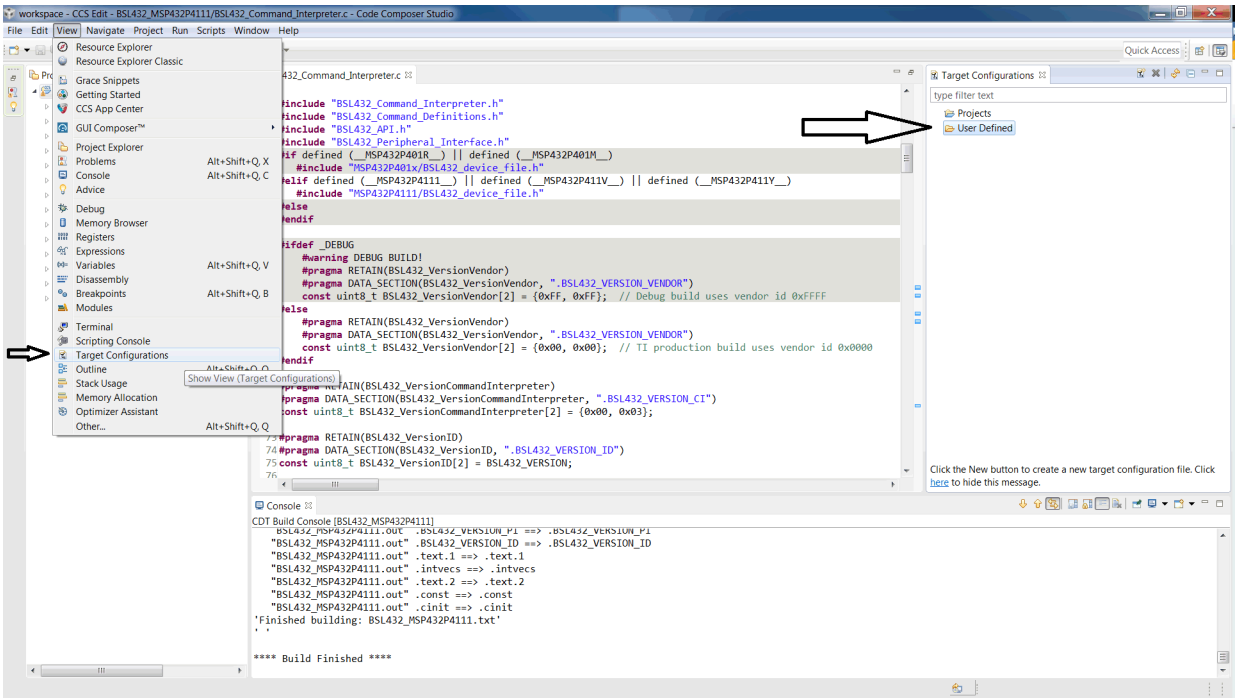

Figure 10. View Target Configurations

2. Right click on the *User Defined* folder, and choose *New Target Configuration* (see Figure 11).

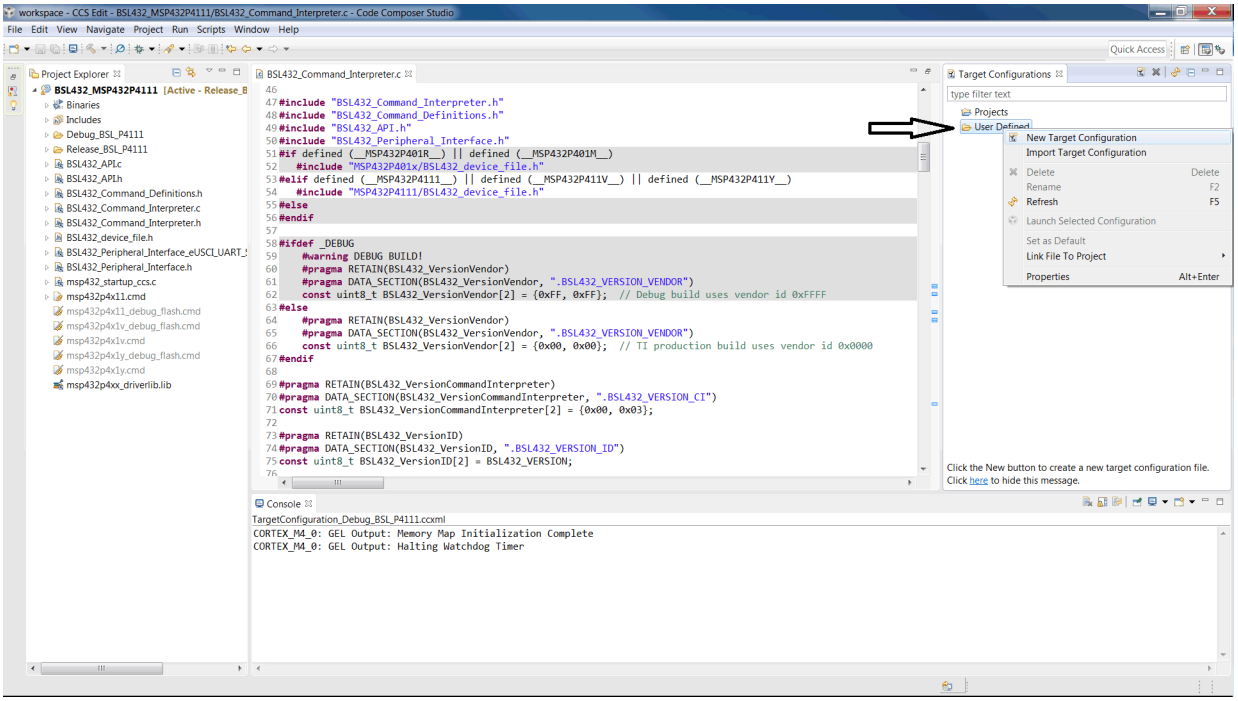

Figure 11. New Target Configuration

3. Enter the intended target configuration name, and also the location where to save this file. either in *File System*, or in this case the *Workspace* is chosen (see Figure 12).

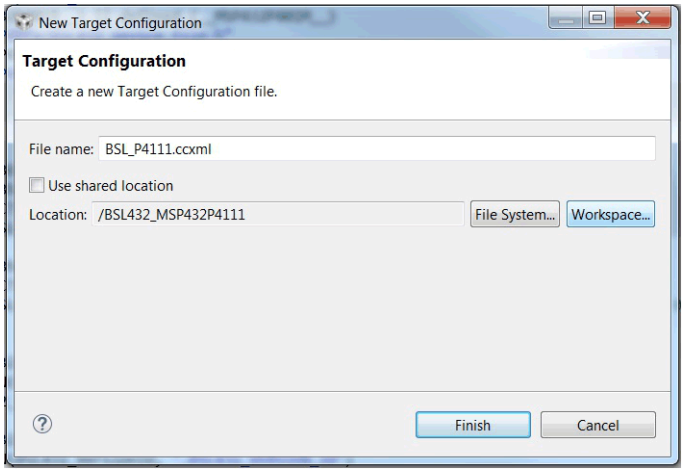

Figure 12. Target Configuration File

4. After the target configuration file is created, select the Debug Probe connection and the Device. Figure 13 shows an example in which the connection uses the XDS200 debug probe and the MSP432P4111 device. After making the selections, click *Save*.

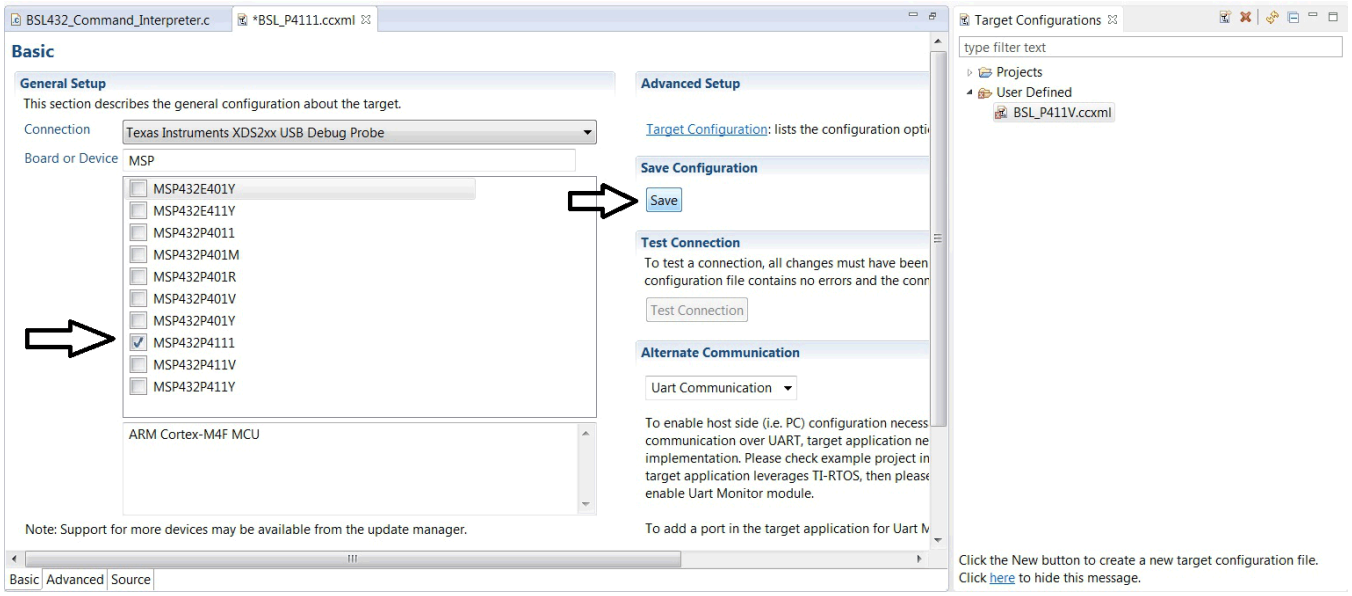

Figure 13. Setup Configuration and Save

5. Right-click on the target configuration BSL_P4111.ccxml, and select *Launch Selected Configuration* (see Figure 14).

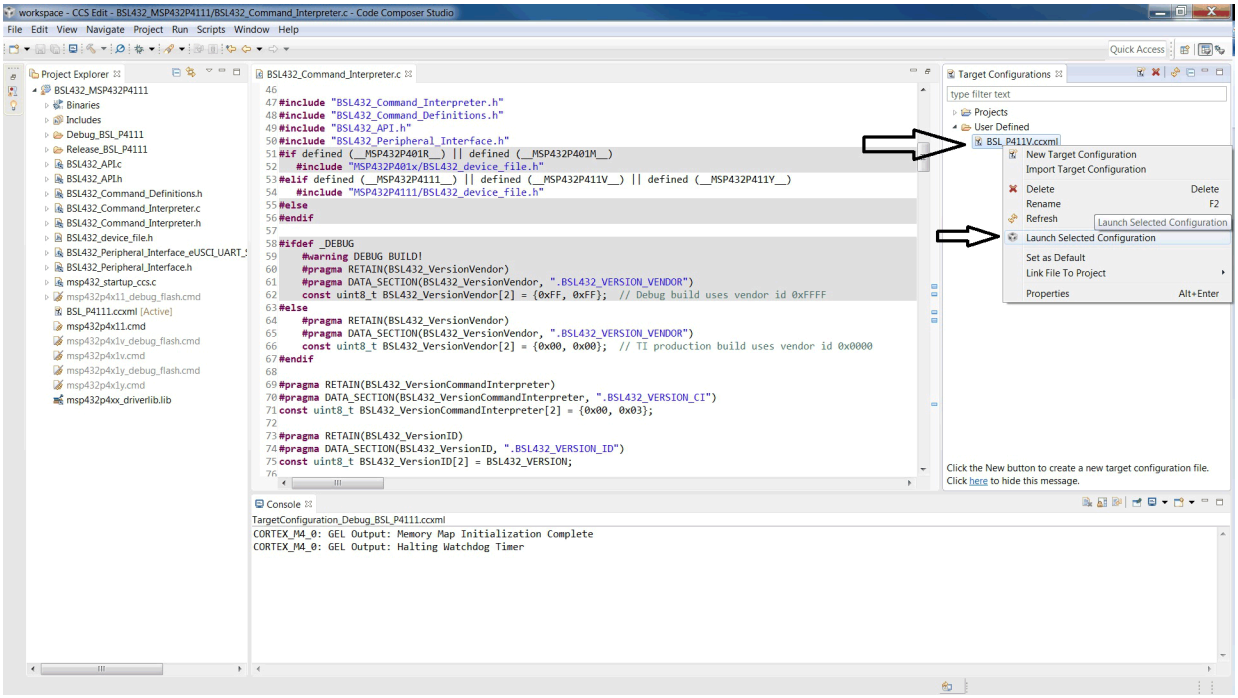

Figure 14. Launch Selected Configuration

6. The *Debug* window opens. Right-click on the XDS200 connection and select *Connect Target*. Next, click the *Pause* sign on the taskbar (see Figure 15).

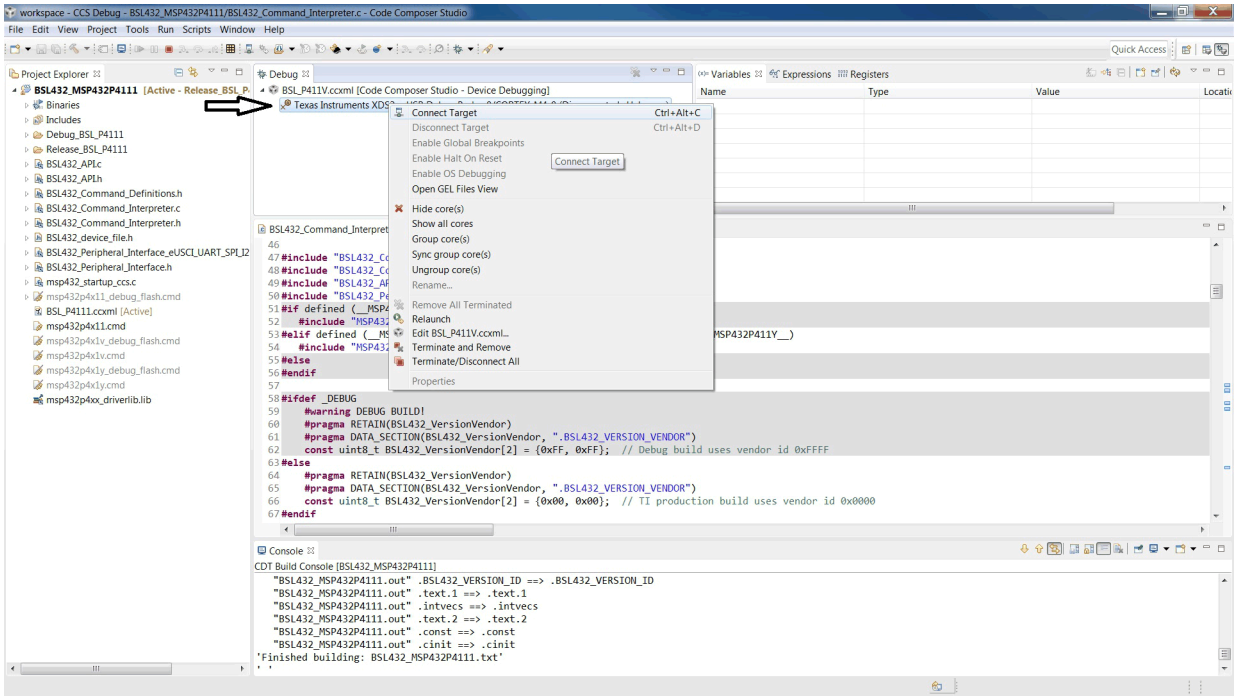

Figure 15. Connect Target

7. Select the BSL firmware image. Click *Run* → *Load* → *Load Program* (see Figure 16).

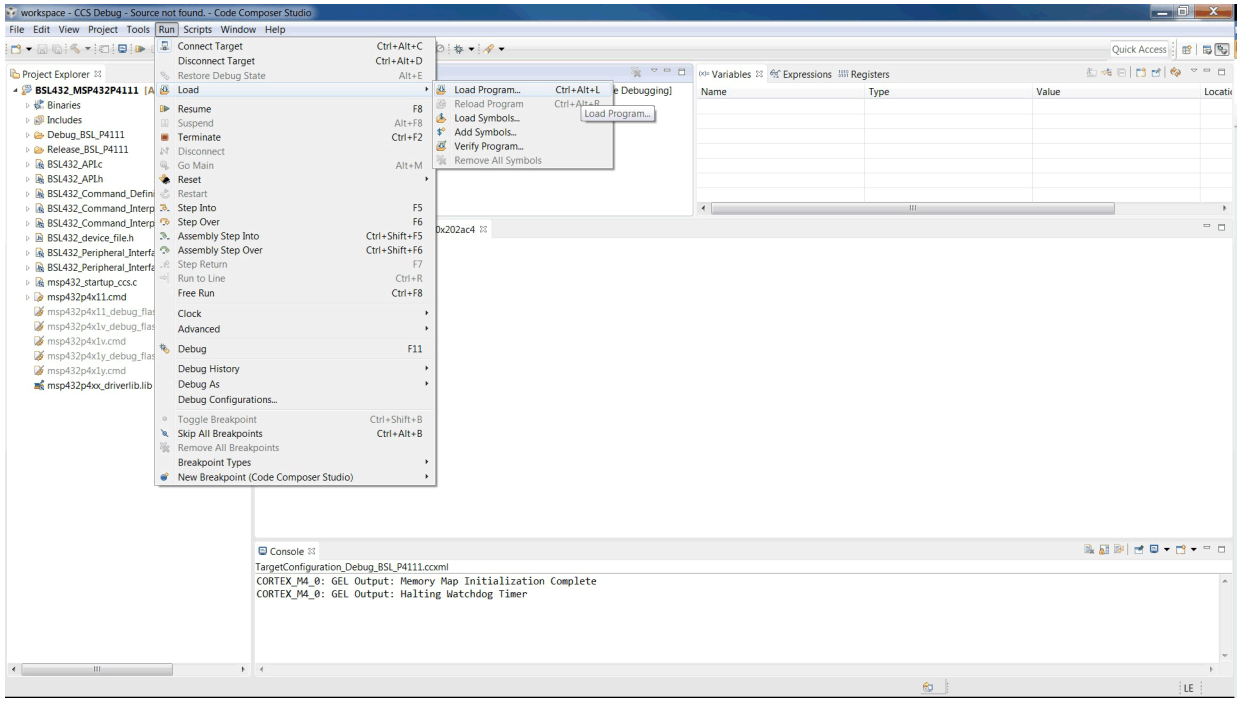

Figure 16. Load Program

8. In the *Load Program* window, click *Browse Project* and select the .out file (see Figure 17).

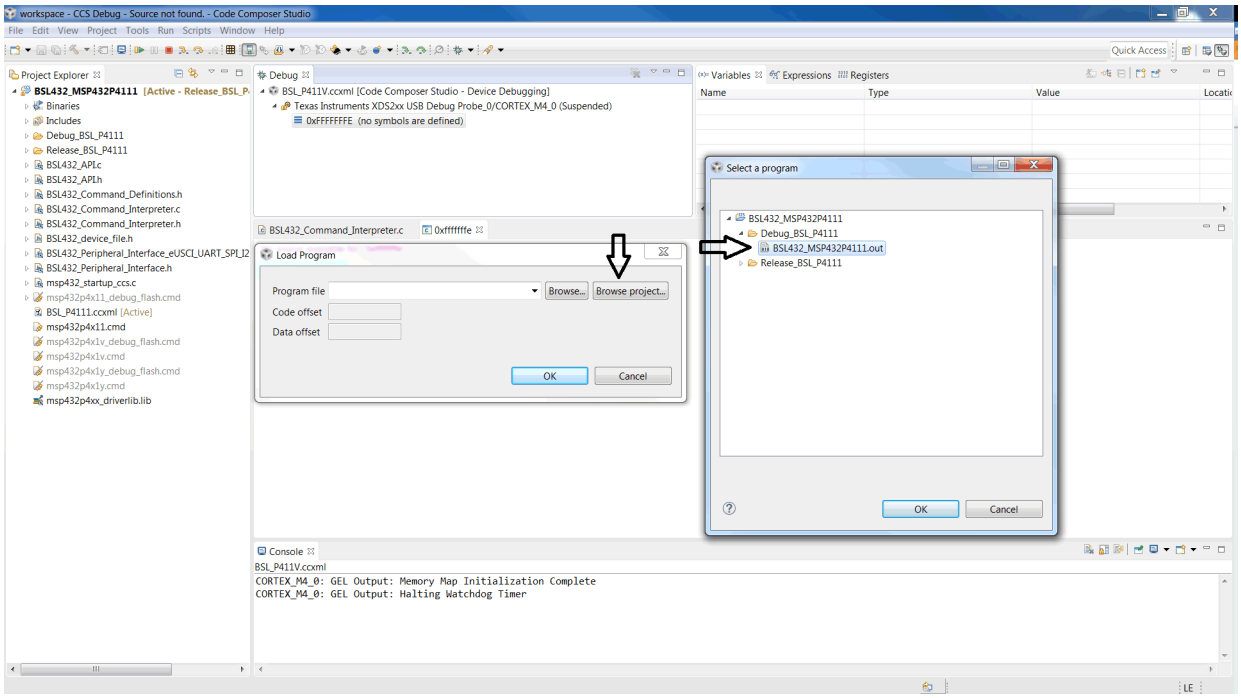

Figure 17. Select .out File

9. Optionally, to perform verification, click *Run → Load → Verify Program*. Select the same .out file as chosen in step 8. Figure 18 shows the console window in which the verification status is displayed.

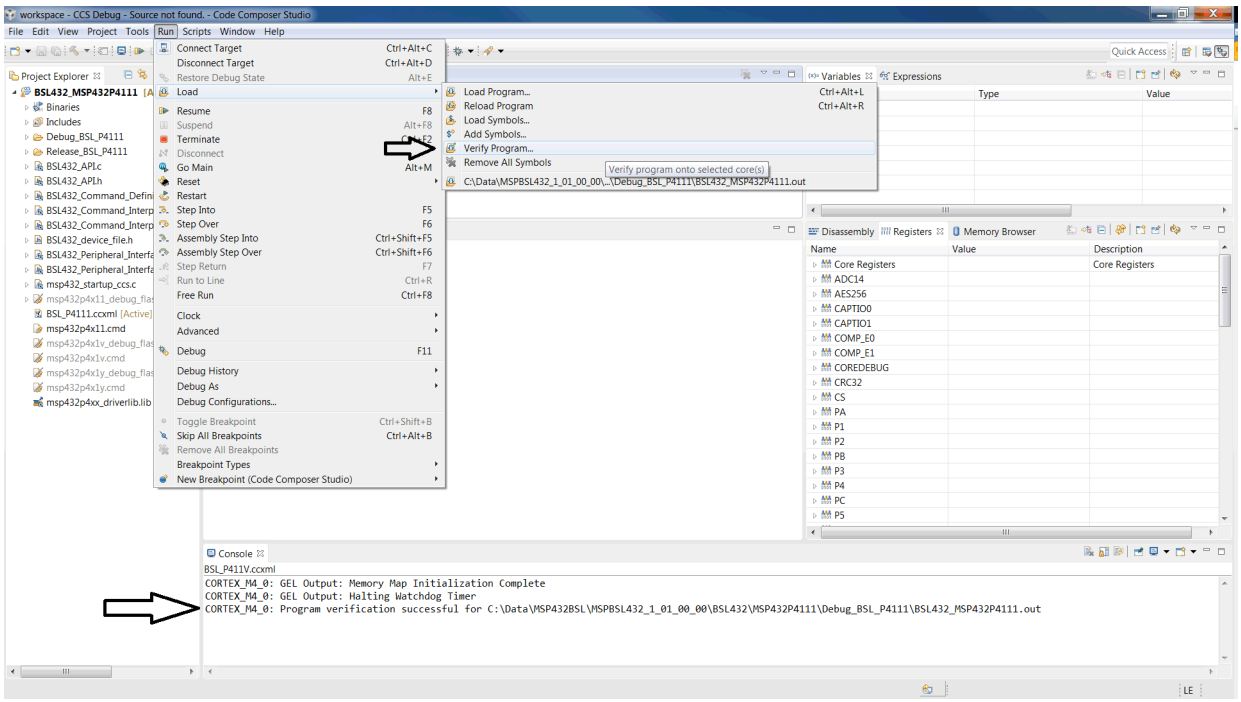

Figure 18. Verify Program

10. From the menu bar, open the *Registers* and *Memory Browser* views. In the memory browser, the BSL in flash is programmed at address 0x2000. The code itself start at address 0x2020 (see Figure 19).

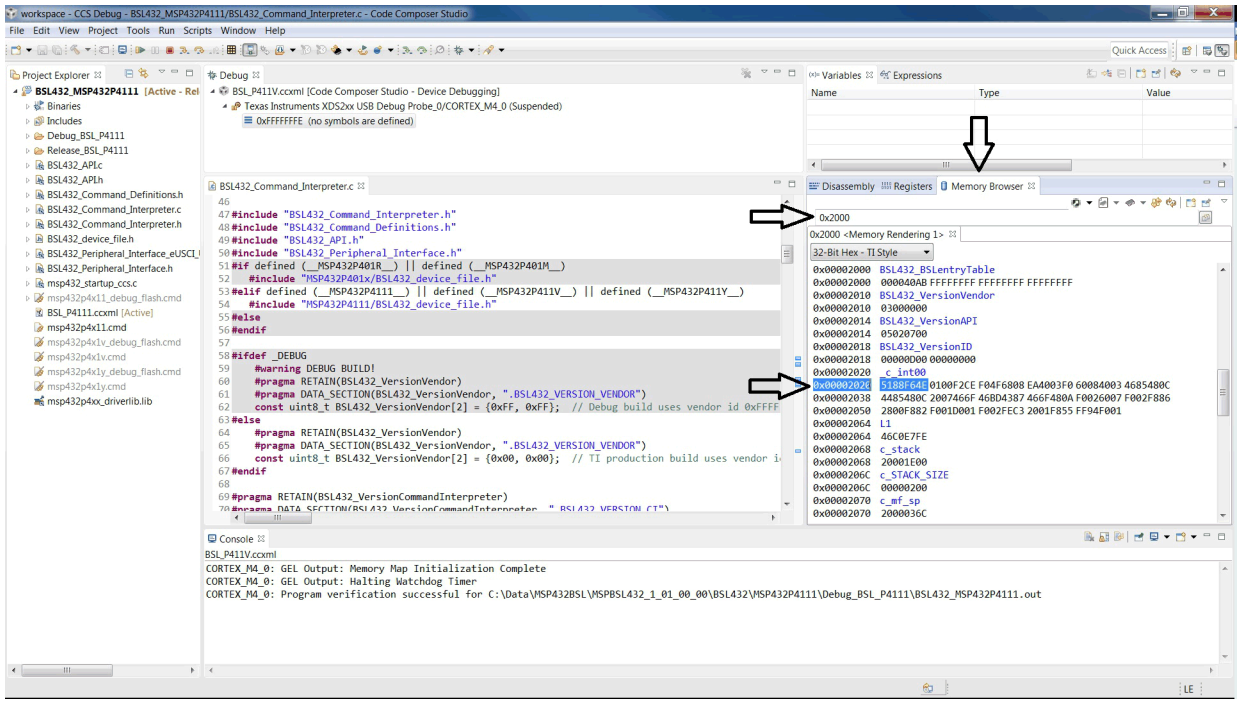

Figure 19. Memory Browser

11. Go to the Registers view → Core Registers → PC, enter the address 0x2020, and click *Enter* (see Figure 20).

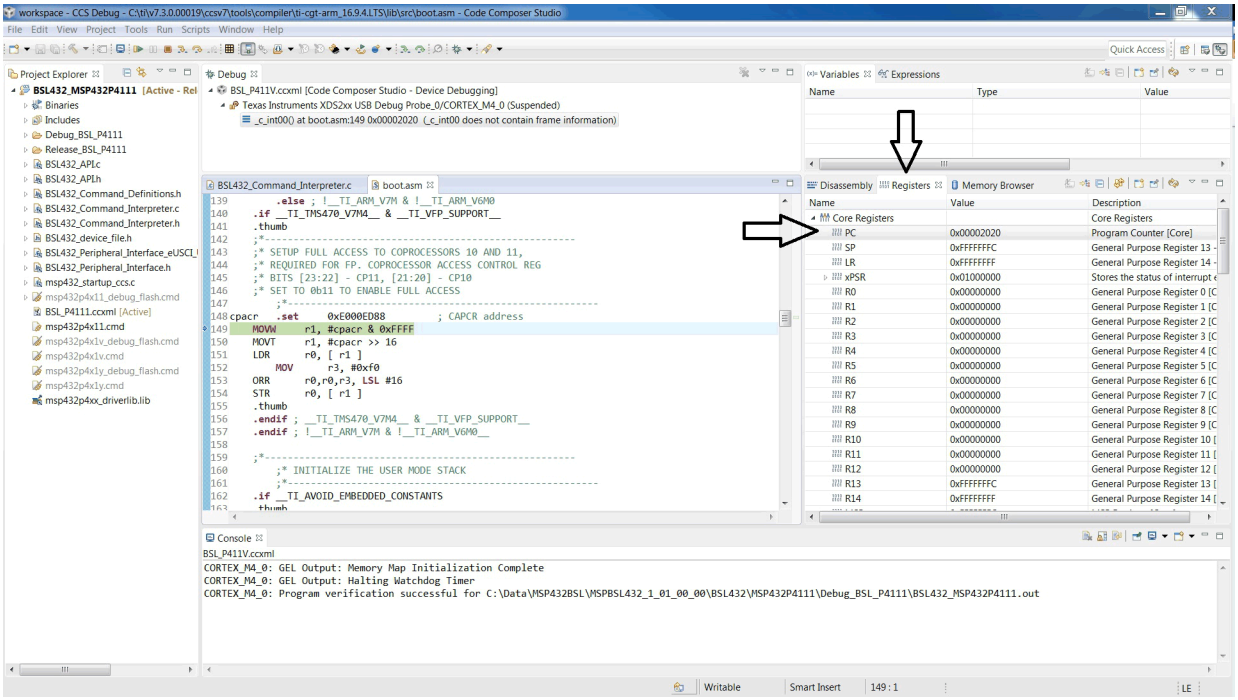

Figure 20. Set PC

To download or debug the Release build, perform the following steps:

1. Right click the project in *Target Configuration*, and select *Properties* (see Figure 21).

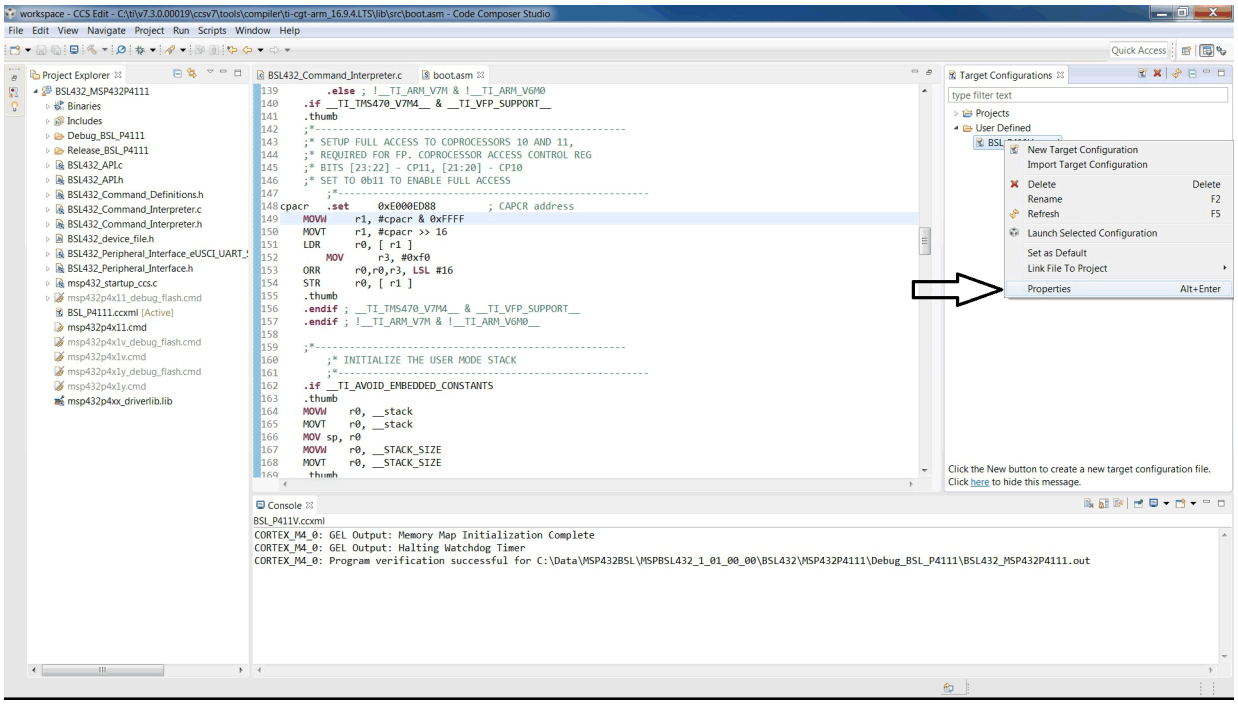

Figure 21. Target Properties

2. Select *MSP432 Setting* and then *Erase Configuration → Erase main and information memory*. Also check *Allow BSL information memory erase* (see Figure 22).

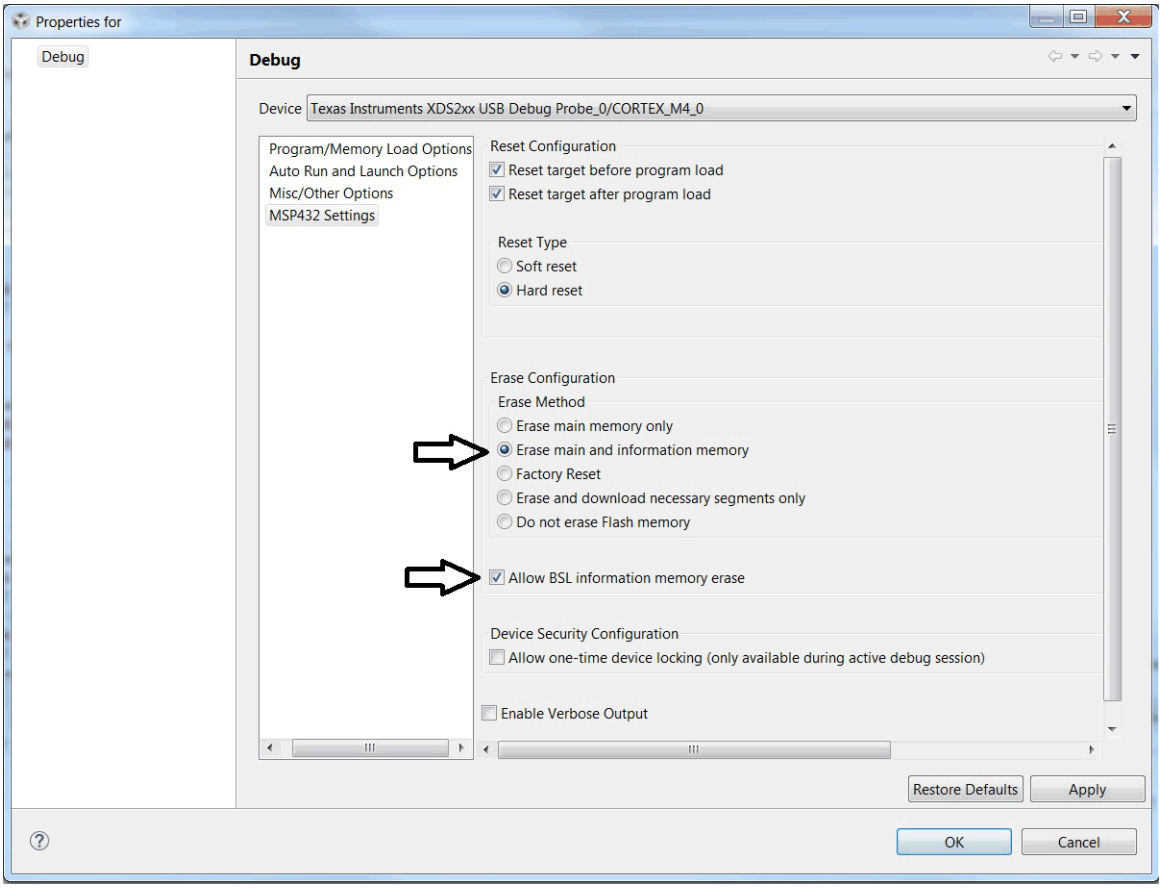

Figure 22. Properties for Debug

3. Follow the previous Step 5 to Step 11 to execute the BSL code. The BSL code for Release mode starts at address 0x202020.

6 MSP432P4xx BSL Version Information

Table 19 and Table 20 list the BSL information for the MSP432P401R and MSP432P401M devices, respectively.

Table 19. MSP432P401R BSL Information

BSL Information	Value
Device	MSP432P401R
BSL version	0000.0002.0003.0102.0003 (Rev B) 0000.0002.0004.0103.0004 (Rev C) 0000.0003.0006.0203.0008 (Rev C) 0000.0003.0007.0205.000A (Rev D)
Buffer size for core commands	262 bytes
Known limitations	Wake up from LPM (after BSL time-out) is not supported on GPIOs used by I^2C or SPI interface.

Table 20. MSP432P401M BSL Information

BSL Information	Value
Device	MSP432P401M
BSL version	0000.0003.0006.0203.0008 (Rev C) 0000.0003.0007.0205.000A (Rev D)
Buffer size for core commands	262 bytes
Known limitations	Wake up from LPM (after BSL time-out) is not supported on GPIOs used by I^2C or SPI interface.

Table 21, Table 22, and Table 23 list the BSL information for the MSP432P4x1x devices.

Table 21. MSP432P4x1V BSL Information

BSL Information	Value
Device	0000.0003.0007.0205.000D
BSL version	MSP432P401V (Rev A), MSP432P411V (Rev A)
Buffer size for core commands	262 bytes
Known limitations	Wakeup from LPM (after BSL time-out) is not supported on GPIOs used by I^2C or SPI interface.

Table 22. MSP432P4x1Y BSL Information

BSL Information	Value
Device	0000.0003.0007.0205.000D
BSL version	MSP432P401Y (Rev A), MSP432P411Y (Rev A)
Buffer size for core commands	262 bytes
Known limitations	Wakeup from LPM (after BSL time-out) is not supported on GPIOs used by I^2C or SPI interface.

Table 23. MSP432P4x11 BSL Information

BSL Information	Value
Device	0000.0003.0007.0205.000D
BSL version	MSP432P4111 (Rev A), MSP432P4011 (Rev A)
Buffer size for core commands	262 bytes
Known limitations	Wakeup from LPM (after BSL time-out) is not supported on GPIOs used by I^2C or SPI interface.

Technical BSL Information

A.1 BSL TLV Structure

During BSL initialization, the BSL parses the device TLV for configuration data. This TLV data is device specific and cannot be changed. The following subsections describe the configuration parameters in detail. Table 24 lists the BSL interface information in the Device Descriptor Table (TLV).

Table 24. BSL Interface Selection Location in TLV

Information	TLV Address
BSL Configuration Tag	Base address of BSL information in TLV (BA)
BSL Configuration Length	BA + 0x4
BSL Peripheral Interface Selection (BSL_PER_IF_SEL)	BA + 0x8
BSL Port Interface Configuration UART (BSL_PORTCNF_UART)	BA + 0xC
BSL Port Interface Configuration SPI (BSL_PORTCNF_SPI)	BA + 0x10
BSL Port Interface Configuration I^2C (BSL_PORTCNF_I2C)	BA + 0x14

A.1.1 BSL_PER_IF_SEL TLV Entry

BSL_PER_IF_SEL TLV entry helps the BSL to estimate the following:

1. Interface multiplexing status: If the corresponding interface is directly available on device pins or if it is multiplexed with GPIO.
2. Interface module: Module used to implement the interface (for example, eUSCIA or eUSCIB).
3. Instance of the module used by the BSL.

The BSL_PER_IF_SEL field is split as shown in Figure 23.

Figure 23. BSL_PER_IF_SEL

31	30	29	28	27	26	25	24
Reserved							
r	r	r	r	r	r	r	r
23	22	21	20	19	18	17	16
I2C_MUX	Reserved	I2C_MOD		I2C_INST			
r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8
SPI_MUX	Reserved	SPI_MOD		SPI_INST			
r	r	r	r	r	r	r	r
7	6	5	4	3	2	1	0
UART_MUX	Reserved	UART_MOD		UART_INST			
r	r	r	r	r	r	r	r

Table 25. BSL_PER_IF_SEL Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	FFh	Reserved for future use.
23	I2C_MUX		I^2C interface mux status
		1h	I^2C is muxed with GPIO
		0h	I^2C pins are dedicated on device pinout
22	Reserved	1h	Reserved for future use.
21-20	I2C_MOD		Module used to implement I^2C.
		0h	eUSCIB
		1h-3h	Reserved for future IPs implementing I^2C.
19-16	I2C_INST		Instance number of the IP specifically used to implement I^2C.
		0h	Use instance 0 of the IP used for implementing I^2C (for example, eUSCIB0 when I2C_MOD = 00)
		1h	Use instance 1 of the IP used for implementing I^2C (for example, eUSCIB1 when I2C_MOD = 00)
		⋮	⋮
		Fh	Use instance 15 of the IP used for implementing I^2C (for example, eUSCIB15, when I2C_MOD = 00)
15	SPI_MUX		SPI interface mux status
		1h	SPI is muxed with GPIO
		0h	SPI pins are dedicated on device pinout
14	Reserved	1h	Reserved for future use.
13-12	SPI_MOD		Module used to implement SPI.
		0h	eUSCIA
		1h	eUSCIB
		2h-3h	Reserved for future IPs implementing SPI.

Table 25. BSL_PER_IF_SEL Field Descriptions (continued)

Bit	Field	Value	Description
11-8	SPI_INST		Instance number of the IP specifically used to implement SPI.
		0h	Use instance 0 of the IP used for implementing SPI (for example, eUSCIA0 when SPI_MOD = 00 or eUSCIB0 when SPI_MOD = 01)
		1h	Use instance 1 of the IP used for implementing SPI (for example, eUSCIA1 when SPI_MOD = 00 or eUSCIB1 when SPI_MOD = 01)
		⋮	⋮
		Fh	Use instance 15 of the IP used for implementing SPI (for example, eUSCIA15 when SPI_MOD = 00 or eUSCIB15 when SPI_MOD = 01)
7	UART_MUX		UART interface mux status
		1h	UART is muxed with GPIO
		0h	UART pins are dedicated on device pinout
6	Reserved	1h	Reserved for future use.
5-4	UART_MOD		Module used to implement UART.
		0h	eUSCIA
		1h-3h	Reserved for future IPs implementing UART.
3-0	UART_INST		Instance number of the IP specifically used to implement UART.
		0h	Use instance 0 of the IP used for implementing UART (for example, eUSCIA0, when UART_MOD = 00)
		1h	Use instance 1 of the IP used for implementing UART (for example, eUSCIA1, when UART_MOD = 00)
		⋮	⋮
		Fh	Use instance 15 of the IP used for implementing UART (for example, eUSCIA15, when UART_MOD = 00)

A.1.2 BSL_PORTCNF_UART, BSL_PORTCNF_SPI, BSL_PORTCNF_I2C TLV Entries

The BSL needs the GPIO configuration details for the module type, module instance, and mux status of the interface being used (obtained from BSL_PER_IF_SEL). The BSL_PORTCNF_UART, BSL_PORTCNF_SPI, and BSL_PORTCNF_I2C TLV entries provide the BSL with these details.

If eUSCIA is the module used to implement UART, BSL_PORTCNF_UART field is split as shown in Figure 24.

Figure 24. BSL_PORTCNF_UART

31	30	29	28	27	26	25	24
Reserved				Reserved	Reserved	PSEL1_TXD	PSEL1_RXD
r	r	r	r	r	r	r	r
23	22	21	20	19	18	17	16
Reserved				Reserved	Reserved	PSEL0_TXD	PSEL0_RXD
r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8
Reserved				Reserved		BITPOS_TXD	
r	r	r	r	r	r	r	r
7	6	5	4	3	2	1	0
BITPOS_TXD	BITPOS_RXD			UART_PORT			
r	r	r	r	r	r	r	r

Table 26. BSL_PORTCNF_UART Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	3Fh	Reserved for future use.
25	PSEL1_TXD		PSEL1 value for GPIO mux of UCAxTXD line
		0h	PSEL1 bit for UCAxTXD should be programed with a value of 0.
		1h	PSEL1 bit for UCAxTXD should be programed with a value of 1.
24	PSEL1_RXD		PSEL1 value for GPIO mux of UCAxRXD line
		0h	PSEL1 bit for UCAxRXD should be programed with a value of 0.
		1h	PSEL1 bit for UCAxRXD should be programed with a value of 1.
23-18	Reserved	3Fh	Reserved for future use.
17	PSEL0_TXD		PSEL0 value for GPIO mux of UCAxTXD line
		0h	PSEL0 bit for UCAxTXD should be programed with a value of 0.
		1h	PSEL0 bit for UCAxTXD should be programed with a value of 1.
16	PSEL0_RXD		PSEL0 value for GPIO mux of UCAxRXD line
		0h	PSEL0 bit for UCAxRXD should be programed with a value of 0.
		1h	PSEL0 bit for UCAxRXD should be programed with a value of 1.
15-10	Reserved	3Fh	Reserved for future use.
9-7	BITPOS_TXD		Bit position of UCAxTXD in the 8-bit port. For example, in MSP432P401x devices, UCA0TXD is muxed on P1.3. This field therefore reads as 0x3 to represent BIT3.
		0h	Bit 0
		1h	Bit 1
		2h	Bit 2
		3h	Bit 3
		4h	Bit 4
		5h	Bit 5
		6h	Bit 6
		7h	Bit 7
6-4	BITPOS_RXD		Bit position of UCAxRXD in the 8-bit port. For example, in MSP432P401x devices, UCA0RXD is muxed on P1.2. This field therefore reads as 0x2 to represent BIT2.
		0h	Bit 0
		1h	Bit 1
		2h	Bit 2
		3h	Bit 3
		4h	Bit 4
		5h	Bit 5
		6h	Bit 6
		7h	Bit 7
3-0	UART_PORT		Port number on which the UART IP is muxed.
		0h	Port 1
		1h	Port 2
		⋮	⋮
		Fh	Port 16

If eUSCIA or eUSCIB is the module used to implement SPI, BSL_PORTCNF_SPI field is split as shown in Figure 25.

Figure 25. BSL_PORTCNF_SPI

31	30	29	28	27	26	25	24
Reserved				PSEL1_STE	PSEL1_CLK	PSEL1_SIMO	PSEL1_SOMI
r	r	r	r	r	r	r	r
23	22	21	20	19	18	17	16
Reserved				PSEL0_STE	PSEL0_CLK	PSEL0_SIMO	PSEL0_SOMI
r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8
BITPOS_STE			BITPOS_CLK			BITPOS_SIMO	
r	r	r	r	r	r	r	r
7	6	5	4	3	2	1	0
BITPOS_SIMO	BITPOS_SOMI			SPI_PORT			
r	r	r	r	r	r	r	r

Table 27. BSL_PORTCNF_SPI Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	Fh	Reserved for future use.
27	PSEL1_STE		PSEL1 value for GPIO mux of UCxxSTE line
		0h	PSEL1 bit for UCxxSTE should be programed with a value of 0.
		1h	PSEL1 bit for UCxxSTE should be programed with a value of 1.
26	PSEL1_CLK		PSEL1 value for GPIO mux of UCxxCLK line
		0h	PSEL1 bit for UCxxCLK should be programed with a value of 0.
		1h	PSEL1 bit for UCxxCLK should be programed with a value of 1.
25	PSEL1_SIMO		PSEL1 value for GPIO mux of UCxxSIMO line
		0h	PSEL1 bit for UCxxSIMO should be programed with a value of 0.
		1h	PSEL1 bit for UCxxSIMO should be programed with a value of 1.
24	PSEL1_SOMI		PSEL1 value for GPIO mux of UCxxSOMI line
		0h	PSEL1 bit for UCxxSOMI should be programed with a value of 0.
		1h	PSEL1 bit for UCxxSOMI should be programed with a value of 1.
23-20	Reserved	Fh	Reserved for future use.
19	PSEL0_STE		PSEL0 value for GPIO mux of UCxxSTE line
		0h	PSEL0 bit for UCxxSTE should be programed with a value of 0.
		1h	PSEL0 bit for UCxxSTE should be programed with a value of 1.
18	PSEL0_CLK		PSEL0 value for GPIO mux of UCxxCLK line
		0h	PSEL0 bit for UCxxCLK should be programed with a value of 0.
		1h	PSEL0 bit for UCxxCLK should be programed with a value of 1.
17	PSEL0_SIMO		PSEL0 value for GPIO mux of UCxxSIMO line
		0h	PSEL0 bit for UCxxSIMO should be programed with a value of 0.
		1h	PSEL0 bit for UCxxSIMO should be programed with a value of 1.
16	PSEL0_SOMI		PSEL0 value for GPIO mux of UCxxSOMI line
		0h	PSEL0 bit for UCxxSOMI should be programed with a value of 0.
		1h	PSEL0 bit for UCxxSOMI should be programed with a value of 1.

Table 27. BSL_PORTCNF_SPI Field Descriptions (continued)

Bit	Field	Value	Description
15-13	BITPOS_STE		Bit position of UCxxSTE in the 8-bit port. For example, in MSP432P401x devices, UCA0STE is muxed on P1.0. This field therefore reads as 0x0 to represent BIT0.
		0h	Bit 0
		1h	Bit 1
		2h	Bit 2
		3h	Bit 3
		4h	Bit 4
		5h	Bit 5
		6h	Bit 6
		7h	Bit 7
12-10	BITPOS_CLK		Bit position of UCxxCLK in the 8-bit port. For example, in MSP432P401x devices, UCA0CLK is muxed on P1.1. This field therefore reads as 0x1 to represent BIT1.
		0h	Bit 0
		1h	Bit 1
		2h	Bit 2
		3h	Bit 3
		4h	Bit 4
		5h	Bit 5
		6h	Bit 6
		7h	Bit 7
9-7	BITPOS_SIMO		Bit position of UCxxSIMO in the 8-bit port. For example, in MSP432P401x devices, UCA0SIMO is muxed on P1.3. This field therefore reads as 0x3 to represent BIT3.
		0h	Bit 0
		1h	Bit 1
		2h	Bit 2
		3h	Bit 3
		4h	Bit 4
		5h	Bit 5
		6h	Bit 6
		7h	Bit 7
6-4	BITPOS_SOMI		Bit position of UCxxSOMI in the 8-bit port. For example, in MSP432P401x devices, UCA0SOMI is muxed on P1.2. This field therefore reads as 0x2 to represent BIT2.
		0h	Bit 0
		1h	Bit 1
		2h	Bit 2
		3h	Bit 3
		4h	Bit 4
		5h	Bit 5
		6h	Bit 6
		7h	Bit 7
3-0	SPI_PORT		Port number on which the SPI IP is muxed.
		0h	Port 1
		1h	Port 2
		⋮	⋮
		Fh	Port 16

If eUSCIB is the module used to implement I²C, BSL_PORTCNF_I2C field is split as shown in Figure 26.

Figure 26. BSL_PORTCNF_I2C

31	30	29	28	27	26	25	24
Reserved				Reserved	Reserved	PSEL1_SDA	PSEL1_SCL
r	r	r	r	r	r	r	r
23	22	21	20	19	18	17	16
Reserved				Reserved	Reserved	PSEL0_SDA	PSEL0_SCL
r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8
Reserved			Reserved			BITPOS_SDA	
r	r	r	r	r	r	r	r
7	6	5	4	3	2	1	0
BITPOS_SDA	BITPOS_SCL			I2C_PORT			
r	r	r	r	r	r	r	r

Table 28. BSL_PORTCNF_I2C Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	3Fh	Reserved for future use.
25	PSEL1_SDA		PSEL1 value for GPIO mux of UCBxSDA line
		0h	PSEL1 bit for UCBxSDA should be programed with a value of 0.
		1h	PSEL1 bit for UCBxSDA should be programed with a value of 1.
24	PSEL1_SCL		PSEL1 value for GPIO mux of UCBxSCL line
		0h	PSEL1 bit for UCBxSCL should be programed with a value of 0.
		1h	PSEL1 bit for UCBxSCL should be programed with a value of 1.
23-18	Reserved	3Fh	Reserved for future use.
17	PSEL0_SDA		PSEL0 value for GPIO mux of UCBxSDA line
		0h	PSEL0 bit for UCBxSDA should be programed with a value of 0.
		1h	PSEL0 bit for UCBxSDA should be programed with a value of 1.
16	PSEL0_SCL		PSEL0 value for GPIO mux of UCBxSCL line
		0h	PSEL0 bit for UCBxSCL should be programed with a value of 0.
		1h	PSEL0 bit for UCBxSCL should be programed with a value of 1.
15-10	Reserved	3Fh	Reserved for future use.
9-7	BITPOS_SDA		Bit position of UCBxSDA in the 8-bit port. For example, in MSP432P401x devices, UCB3SDA is muxed on P10.2. This field therefore reads as 0x2 to represent BIT2.
		0h	Bit 0
		1h	Bit 1
		2h	Bit 2
		3h	Bit 3
		4h	Bit 4
		5h	Bit 5
		6h	Bit 6
		7h	Bit 7

Table 28. BSL_PORTCNF_I2C Field Descriptions (continued)

Bit	Field	Value	Description
6-4	BITPOS_SCL		Bit position of UCBxSCL in the 8-bit port. For example, in MSP432P401x devices, UCB3SCL is muxed on P10.3. This field therefore reads as 0x3 to represent BIT3.
		0h	Bit 0
		1h	Bit 1
		2h	Bit 2
		3h	Bit 3
		4h	Bit 4
		5h	Bit 5
		6h	Bit 6
		7h	Bit 7
3-0	I2C_PORT		Port number on which the I^2C IP is muxed.
		0h	Port 1
		1h	Port 2
		⋮	⋮
		Fh	Port 16

Revision History

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

Changes from November 14, 2017 to December 12, 2017 **Page**

- Added the paragraph that begins "To call the BSL from interrupt routine service..." and the following code example in Section 3.3.1, *Software BSL Invocation* 6
- Added Section 5, *Reprogram and Customize the BSL* 33
- Added Table 21, Table 22, and Table 23 in Section 6, *MSP432P4xx BSL Version Information* 46

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

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

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

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

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

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

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

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

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

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265