User's Guide
SLAU610C–October 2014–Revised March 2017

MSP430FR4xx and MSP430FR2xx Bootloader (BSL)

The bootloader (BSL, formerly known as the bootstrap loader) provides a method to program memory during MSP430™ MCU project development and updates. It can be activated by a utility that sends commands using a serial protocol. The BSL enables the user to control the activity of the MSP430 device and to exchange data using a personal computer or other device.

Contents

List of Figures

List of Tables

Trademarks

MSP430 is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Introduction

The MSP430 BSL enables users to communicate with embedded memory in the MSP430 microcontroller during the prototyping phase, final production, and in service. Both the programmable memory (FRAM memory) and the data memory (RAM) can be modified as required. Do not confuse the bootloader with 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. These programs are often referred to as bootloaders as well.

To start the bootloader, a specific BSL entry sequence or the application code sets the PC pointer to the BSL start address in Z-area. On FR26xx, FR24xx, and FR23xx MCUs, an empty reset vector (for example, as on a unprogrammed device) also invokes the BSL. After the BSL has started, a sequence of commands can be sent to the BSL to execute the desired functions (for example, unlocking the device, reprogramming the memory, or verifying the written data). A bootloading session can be exited by continuing operation at a defined user program address or by the reset condition.

Even if the device is secured by disabling JTAG, it is still possible to use the BSL.

To avoid accidental overwriting of the BSL code, the code is stored in a secure ROM memory location. To prevent unwanted memory readout, any BSL command that directly or indirectly allows data reading or writing is password protected. Using this method, access to the device memory through the BSL is protected against misuse by a user-defined password. For more information about password-protected commands, see Section 4.2.

To invoke the bootloader, a 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 BSL Limitations

The BSL on the FR4xx, FR21xx, and FR20xx MCUs supports UART at 9600 baud only. It has a limited Core Commands set (see Section 4.3.3) and responses.

1.2 Other Useful Documentation

MSP430FR4xx and MSP430FR2xx Family User's Guide

MSP430 Programming With the Bootloader

Creating a Custom Flash-Based Bootloader

MSP BSL Tool Folder

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.

3 BSL Architecture

3.1 Communication Interfaces

The BSL on the FR4xx, FR21xx, and FR20xx MCUs implements UART interface only. The BSL on the FR26xx, FR25xx, FR24xx, and FR23xx MCUs implements both UART and I²C serial interfaces. The BSL can be invoked by the device boot code or by the software application (see Section 3.5.1.1).

The BSL on the FR26xx, FR25xx, FR24xx, and FR23xx MCUs scans on both interfaces passively. This means the BSL is not driving any output pins. If data (for example, a BSL command) is sent to one of the BSL interfaces, this interface is used by the BSL, and the other interface is disabled. Only one interface can be active at a time. Data must be sent before the time-out occurs (see Section 3.2). If no communication is detected, the BSL times out, and the device enters LPM4.

3.1.1 UART BSL

The UART serial interface is used when the BSL Peripheral Interface is automatically detected. The BSL on the FR4xx, FR21xx, and FR20xx MCUs always uses the UART serial interface.

3.1.1.1 UART Hardware Pin Information

The pins that are used for communication by the UART BSL are described in the *Bootloader (BSL)* section of the device-specific data sheet.

3.1.1.2 UART Protocol Definition

The UART protocol used by the BSL is defined as:

- Baud rate is configured to start at 9600 baud in half-duplex mode (one sender at a time).
- Start bit, 8 data bits (LSB first), an even parity bit, 1 stop bit.
- Handshake is performed by an acknowledge character.
- The minimum time delay before sending new characters after characters have been received from the MSP430 BSL is 1.2 ms.

NOTE: Applying baud rates other than 9600 baud at initialization may result in communication problems.

3.1.2 I²C BSL

The I²C serial interface is used only when the BSL Peripheral Interface is automatically detected. The I²C slave address that is used by the BSL is 0x48.

3.1.2.1 I²C Hardware Pin Information

The pins that are used for communication by the I²C BSL are described in the *Bootloader (BSL)* section of the device-specific data sheet.

3.1.2.2 I²C Protocol Definition

The I²C protocol used by the BSL is defined as:

- Master must request data from BSL slave.
- 7-bit addressing mode is used, and the slave listens to address 0x48.
- Handshake is performed by an acknowledge character in addition to the hardware ACK.
- The minimum time delay before sending new characters after characters have been received from the MSP430 BSL is 1.2 ms.
- Repeated starts are not required by the BSL but can be used.

3.2 BSL Time-out

The BSL on the FR26xx, FR25xx, FR24xx, and FR23xx MCUs implements a low-power time-out feature for the automatic detection of the BSL interface. If no communication has been established within ten seconds, the device enters LPM4 mode. To invoke the BSL again, the device power must be cycled, or a reset or NMI must be received.

3.3 Blank Device Detection

The boot code on the FR26xx, FR25xx, FR24xx, and FR23xx MCUs supports blank device detection to avoid the BSL entry sequence. This saves time and also eliminates the need for two additional wires (TEST, RST). The BSL entry sequence can be bypassed when the blank detection is enabled. The device jumps directly to the BSL and bypasses the entry sequence only when the reset vector has a value of 0xFFFF.

3.4 BSL Memory

3.4.1 BSL Memory Layout

The BSL memory on the FR4xx, FR21xx, and FR20xx MCUs is a total 1KB section of read-only memory (ROM) and cannot be customized. The BSL is at addresses 0x1000 to 0x13FF (1KB).

The BSL memory on the FR26xx, FR25xx, FR24xx, and FR23xx MCUs is a total 3KB section of read-only memory (ROM) and cannot be customized. The BSL is at addresses 0x1000 to 0x17FF (2KB) and 0xFFC00 to 0xFFFFF (1KB).

3.4.1.1 BSL Z-Area

When protected, the BSL memory cannot be read from or jumped into from a location external to BSL memory. This makes the BSL more secure and prevents erroneous BSL execution. However, if the entire BSL memory space were protected in this way, it would also mean that user application code could not call the BSL in any way, such as an intentional BSL start-up or using certain public BSL functions.

The Z-Area is a special section of memory that allows a protected BSL to be publically accessible in a controlled way. The Z-Area is a section of BSL memory between addresses 0x1000 and 0x100F that can be jumped to and read by external application code. It functions as a gateway from which a jump can be performed to any location within the BSL memory. The default TI BSL uses this area for jumps to the start of the BSL and for jumps into BSL public functions.

3.4.2 BSL Memory Considerations

The BSL clears RAM contents at the beginning of BSL initialization to prevent reading out any application data or code that may reside there. After initialization, the BSL uses the RAM for data buffer and local variables. When invoking the BSL from a main application, RAM contents may be lost. The range of RAM that is cleared is listed in the tables in Section 6.

3.5 BSL Invocation

Three methods of BSL invocation are supported:

- The BSL can be called by application software (see Section 3.5.1).
- The BSL can be called by the device boot-code by applying a hardware entry sequence (see Section 3.5.2).
- The BSL can be invoked at start-up if the device is blank (on FR26xx, FR25xx, FR24xx, and FR23xx MCUs).

3.5.1 Software BSL Invocation

The BSL Z-Area is a small section of memory that can be read and invoked from application code. It is located at memory addresses 0x1000 to 0x100F.

3.5.1.1 Starting the BSL From an External Software Application

Memory location 0x1000 contains a jump instruction pointing to the BSL start and can be used to invoke the BSL from a running application by setting the program counter to 0x1000. The stack is reset, and RAM is cleared. Interrupts are not disabled by the BSL and should be disabled by the application before invoking the BSL.

The location 0x1000 may be called as a C function, as in the following example code:

```
__disable_interrupt(); // disable interrupts
((void (*)())0x1000)(); // jump to BSL
```

3.5.1.2 BSL Action

Memory location 0x1002 contains a jump to the "BSL Action" function. To invoke the action function, three parameters are needed. The first parameter is a number describing which function, and the other two parameters are known values that indicate that the function was called intentionally.

R12: The function number

R13: 0xDEAD

R14: 0xBEEF

3.5.1.2.1 BSL Action Function 2

Function number:	2
Function Name:	Return to BSL
Description:	Any supplied function number calls the return to BSL function. This function can be used if the BSL has written a program into FRAM or RAM, started that program by "Set PC", and then the program needs to return to the BSL. This function executes the following code:

```
RETURN_TO_BSL POP.W RET_low ; remove first word from return addr POP.W RET_high
                            ; remove second word from return addr RETA
                            ; should now return to the BSL location
```

3.5.2 Hardware BSL Invocation

Applying an appropriate entry sequence on the RST/NMI and TEST pins forces the device to start program execution at the BSL RESET vector instead of at the RESET vector located at address FFFEh.

If the application interfaces with a computer UART, these two pins may be driven by the DTR and RTS signals of the serial communication port (RS232) after passing level shifters. See the *Hardware Description* section of *MSP430 Programming With the Bootloader (BSL)* for hardware schematics. The standard user reset vector at 0xFFFE is used if TEST is kept low while RST/NMI rises from low to high (standard method, see Figure 1).

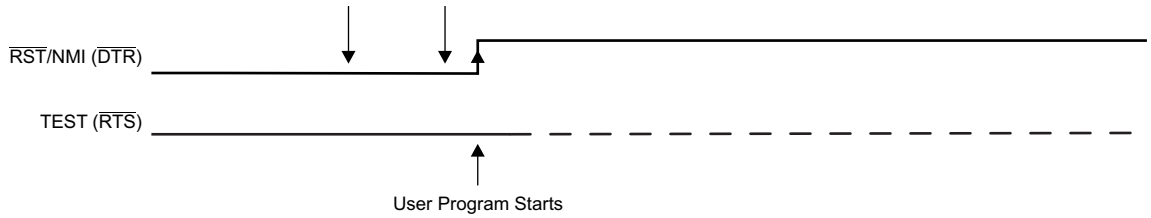
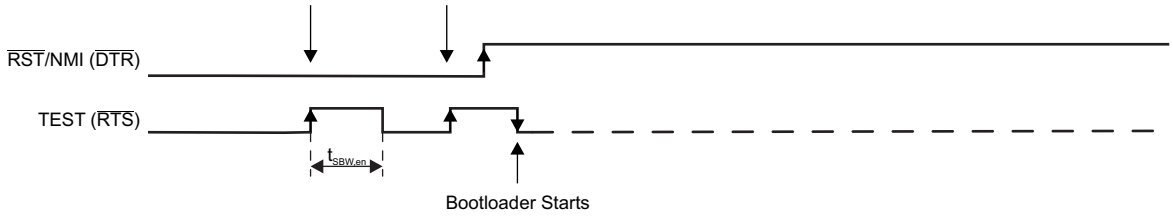


Figure 1. Standard Reset Sequence

The BSL program execution starts when the TEST pin has received a minimum of two positive transitions and if TEST is high while RST/NMI rises from low to high (BSL entry method, see Figure 2). This level transition triggering improves BSL start-up reliability.



NOTE: The recommended minimum time for pin states is 250 ns.

Figure 2. BSL Entry Sequence at Shared JTAG Pins

The TEST signal is typically used to switch the port pins between their application function and the JTAG function. In devices with BSL functionality, the TEST and RST/NMI pins are also used to invoke the BSL. To invoke the BSL, the RST/NMI pin must be configured as RST and must be kept low while pulling the TEST pin high and while applying the next two edges (falling, rising) on the TEST pin. The BSL is started after the TEST pin is held low for more than 100 µs and then the RST/NMI pin is released (see Figure 2).

The BSL is not started by the BSL RESET vector if:

- There are fewer than two positive edges at the TEST pin while RST/NMI is low.
- JTAG has control over the MSP430 resources.
- Supply voltage (V_{CC}) drops below its threshold, and a power-on reset (POR) is executed.
- RST/NMI pin is configured for NMI functionality (NMI bit is set).

3.6 BSL Version Number

The BSL version number can be requested by a host programmer using the TX BSL Version command (see Section 4.3.3.7 for more information).

Byte 1: BSL Vendor information

TI BSL is always 0x00. Non-TI BSLs may use this area in another manner.

Byte 2: Command Interpreter Version

The version number for the section of code that interprets BSL core commands.

Byte 3: API Version

The version number for the section of code that reads and writes to MSP430 memory.

Reserved bits:

0x00-0x0F: Indicates that this BSL API interfaces with flash.

0x30-0x3F: Indicates that this BSL API interfaces with FRAM.

0x40-0x4F: Indicates that this BSL API interfaces with FRAM using a reduced BSL Core Commands set.

0x80-0x8F: Indicates that this BSL can execute only the following commands:
RX Data Block Fast (and can only write to RAM)
RX Password
Set PC

Byte 4: Peripheral Interface Version

The version number for the section of code that manages communication.

Reserved numbers:

0x00-0x2F: Indicates a Timer_A-based UART

0x30-0x4F: Indicates USB

0x50-0x6F: Indicates USCI-based UART

0x70-0x8F: Indicates eUSCI-based UART

0x90-0x9F: Indicates USCI-based I^2C

0xA0-0xAF: Indicates eUSCI-based I^2C

0xB0-0xBF: Indicates combined eUSCI I^2C and UART

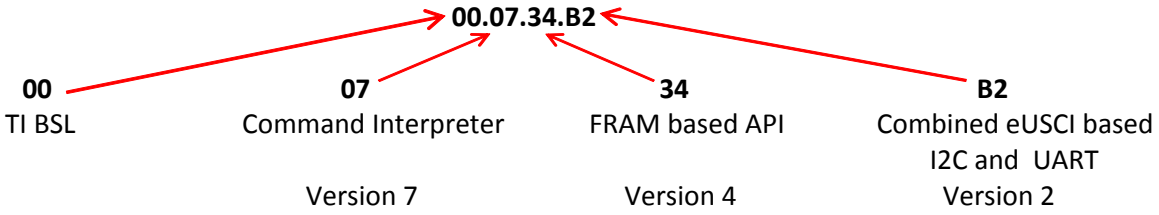


Figure 3. Example BSL Version

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 2. BSL Data Packet

PI Code	BSL Core Command	PI Code

4.2 BSL Security

4.2.1 Protected Commands

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. Commands specific to the peripheral interface are not protected.

4.2.2 RAM Erase

At start-up, the BSL performs a RAM erase, writing a constant word to certain RAM locations in a device, from the beginning of RAM to the current stack pointer.

4.2.3 BSL Entry

The BSL can only be accessed from the Z-Area at the beginning of the BSL. This restricts the BSL operation to a few controlled entry points. Read and execute access to the rest of the BSL code is not possible.

See Section 3.4.1.1 for more information about the Z-Area.

4.3 UART Peripheral Interface (PI)

4.3.1 UART Peripheral Interface Wrapper

The 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 3. 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 Table 6 for more information about what commands return a BSL Core Response). Table 4 lists the BSL response.

Table 3. UART BSL Command

Header	Length	Length	BSL Core Command	CKL	CKH
0x80	NL	NH	See Table 6	CKL	CKH

Table 4. UART BSL Response

Acknowledgment	Header	Length	Length	BSL Core Response[1]	CKL	CKH
ACK from BSL	0x80	NL	NH	See Table 14	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 BSL uses CRC-CCITT for the checksum and computes it using the MSP430 CRC module (see the CRC chapter 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.

4.3.2 UART BSL Acknowledgment

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 5 lists all possible 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 needs to check the acknowledgment error and retry transmission.

Table 5. UART 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(1)	Packet size zero. The size for the BSL core command was given as 0.
0x54(1)	Packet size exceeds buffer. The packet size given is too big for the RX buffer.
0x55	Unknown error
0x56(1)	Unknown baud rate. The supplied data for baud rate change is not a known value.
0x57	Packet Size Error.

(1) Not supported for MSP430FR4xx, MSP430FR21xx, and MSP430FR20xx.

4.3.3 UART BSL Core Commands

Table 6 summarizes the format of the BSL core commands.

Table 6. UART BSL Core Commands

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
RX Data Block	Yes	0x10	(AL)	(AM)	(AH)	D1…Dn	Yes
RX Password	No	0x11	–	–	–	D1…D32	Yes
Mass Erase[1]	No	0x15	–	–	–	–	No
CRC Check[1]	Yes	0x16	(AL)	(AM)	(AH)	Length (low byte), Length (high byte)	Yes
Load PC	Yes	0x17	(AL)	(AM)	(AH)	–	No
TX Data Block	Yes	0x18	(AL)	(AM)	(AH)	Length (low byte), Length (high byte)	Yes
TX BSL Version[1]	Yes	0x19	–	–	–	–	Yes
RX Data Block Fast[1]	Yes	0x1B	(AL)	(AM)	(AH)	D1...Dn	No
Change Baud Rate[1]	No	0x52	–	–	–	D1	No

(1) Not supported for MSP430FR4xx, MSP430FR21xx, and MSP430FR20xx.

AL, AM, AH

Address bytes. The low, middle, and upper bytes, respectively, of an address.

D1...Dn

Data bytes 1 through n (Note: n must be 4 less than the BSL buffer size.)

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 next byte.

4.3.3.1 RX Data Block

Table 7 shows the RX data block format.

Table 7. RX Data Block

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
RX Data Block	Yes	0x10	(AL)	(AM)	(AH)	D1…Dn	Yes

Description

The BSL core writes bytes D1 through Dn starting from the location specified in the address fields. This command differs from RX Data Block Fast in that it returns the status of the write operation.

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 maximum 256.

Command Returns

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

See Section 4.5 for more information on BSL core responses.

Command Example

Write data 0x76543210 to address 0x010000:

Header	Length	Length	CMD	AL	AM	AH	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

4.3.3.2 RX Password

Table 8 shows the RX password format.

Table 8. RX Password

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
RX Password	No	0x11	–	–	–	D1…D32	Yes

Description

The BSL core receives the password contained in the packet and unlocks the BSL protected commands if the password matches the top 16 words in the BSL interrupt vector table (located between addresses 0xFFE0 and 0xFFFF). When an incorrect password is given, a mass erase is initiated. This means all code FRAM is erased, but not Information Memory.

When a mass erase is performed, two conditions must be checked: the password is always 0xFF for all bytes, and the value of the BSL signature at 0xFF85 and 0xFF84 is not 0xAAAA and 0x5555. 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 32 bytes long and contains the device password.

Command Returns

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

See Section 4.5 for more information on BSL core responses.

Command Example

Unlock a blank device:

Header	Length	Length	CMD	D1	D2	D3	D4	D5	D6
0x80	0x21	0x00	0x11	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	CKL	CKH
0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0x9E	0xE6

BSL response for a successful password:

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

4.3.3.3 Mass Erase

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
Mass Erase	No	0x15	–	–	–	–	No

Description

All code FRAM in the device is erased. This function does not erase Information Memory and RAM.

As the BSL on the FR4xx, FR21xx, and FR20xx MCUs does not support the Mass Erase command, a RX Password command containing an incorrect password can be send instead to trigger a mass erase.

Protection

This command is not password protected.

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.

See Section 4.5 for more information on BSL core responses.

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

4.3.3.4 CRC Check

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
CRC Check	Yes	0x16	(AL)	(AM)	(AH)	Length (low byte), Length (high byte)	Yes

Description

The MSP430 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.

Refer to the CRC chapter 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

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 Section 4.5 for more information on BSL core responses.

Command Example

Perform a CRC check from address 0x4400 to 0x4800 (size of 1024):

Header	Length	Length	CMD	AL	AM	AH	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

4.3.3.5 Load PC

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
Load PC	Yes	0x17	(AL)	(AM)	(AH)	–	No

Description

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

The BSL can be returned to by the main application using the BSL Action function 2, Return to BSL. Refer to Section 3.5.1 for more information.

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

Load PC always return success or a BSL locked status. This status is sent only if the main application returns to the BSL.

See Section 4.5 for more information on BSL core responses.

Command Example

Set program counter to 0x4400:

Header	Length	Length	CMD	AL	AM	AH	CKL	CKH
0x80	0x04	0x00	0x17	0x00	0x44	0x00	0x42	0x0F

The BSL does not respond after the application gains control.

4.3.3.6 TX Data Block

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
TX Data Block	Yes	0x18	(AL)	(AM)	(AH)	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

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 with a buffer size of 260, the BSL sends two packets–the first packet with a length of 259 and the second with a length of 253.

See Section 4.5 for more information on BSL core responses.

Command Example

Transmit 4 bytes of data from RAM address 0x1C00:

Header	Length	Length	CMD	AL	AM	AH	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

4.3.3.7 TX BSL Version

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
TX BSL Version	Yes	0x19	–	–	–	–	Yes

Description

BSL transmits its version information (see Section 3.6 for more details).

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 Byte
BSL Vendor	D1
Command Interpreter	D2
API	D3
Peripheral Interface	D4

See Section 4.5 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 00.07.34.B2 of the BSL):

ACK	Header	Length	Length	CMD	D1	D2	D3	D4	CKL	CKH
0x00	0x80	0x05	0x00	0x3A	0x00	0x07	0x34	0xB2	0x14	0x90

4.3.3.8 RX Data Block Fast

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
RX Data Block Fast	Yes	0x1B	(AL)	(AM)	(AH)	D1…Dn	No

Description

This command is identical to RX Data Block, except there is no reply indicating the data was correctly programmed. It is used primarily to speed up USB programming on the MSP430F5xx and MSP430F6xx family of devices.

Protection

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

Command

0x1B

Command Address

Address to write the received data to.

Command Data

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

Command Returns

BSL acknowledgment

Command Example

Write data 0x76543210 to address 0x010000:

Header	Length	Length	CMD	AL	AM	AH	D1	D2	D3	D4	CKL	CKH
0x80	0x08	0x00	0x1B	0x00	0x00	0x01	0x10	0x32	0x54	0x76	0x3C	0x1C

BSL Response:

ACK
0x00

4.3.3.9 Change Baud Rate

BSL Command	Protected	CMD	AL	AM	AH	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
0x02	9600
0x03	19200
0x04	38400
0x05	57600
0x06	115200

Command Returns

BSL acknowledgment

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 I²C Peripheral Interface (PI)

4.4.1 I²C Peripheral Interface Wrapper

The 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 9. 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 Table 12 for more information about what commands return a BSL Core Response). Table 10 shows the format of this BSL response.

Table 9. I²C BSL Command

I2C	Header	Length	Length	BSL Core Command	CKL	CKH
S/A/W	0x80	NL	NH	See Table 12	CKL	CKH

Table 10. 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 14	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 MSP430 CRC module specification (see the CRC chapter 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 MSP430 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 for more details on I²C communication.

S/A/R

I²C start or re-start sequence sent by the host programmer to the MSP430 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, and the host programmer can send a stop followed by another start. See the eUSCI I²C mode chapter for more details on I²C communication.

STOP

I²C stop bit that indicates the end of an I²C read or write.

4.4.2 I²C BSL Acknowledgment

The peripheral interface section of the BSL software parses the wrapper section of the BSL data packet. The BSL responds after all data has been received and sends either an ACK indicating successful reception of the packet or one of several error codes. An ACK 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. If an error occurs with the packet structure an error response is sent instead of an ACK to the host.

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 11 lists all possible acknowledgment responses from the BSL. If an acknowledgment byte other than ACK is sent, the BSL does not send any BSL data packets in response. The host programmer needs to check the acknowledgment error and retry transmission.

Table 11. I²C Error Messages

Data	Meaning
0x00	ACK
0x51	Header incorrect. The packet did not begin with the required 0x80 value.
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

4.4.3 I²C BSL Core Commands

The BSL core command is transmitted in the format shown in Table 12. All numbers are in hexadecimal format.

Table 12. I²C BSL Core Commands

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
RX Data Block	Yes	0x10	(AL)	(AM)	(AH)	D1…Dn	Yes
RX Password	No	0x11	–	–	–	D1…D32	Yes
Mass Erase	No	0x15	–	–	–	–	No
CRC Check	Yes	0x16	(AL)	(AM)	(AH)	Length (low byte), Length (high byte)	Yes
Load PC	Yes	0x17	(AL)	(AM)	(AH)	–	No
TX Data Block	Yes	0x18	(AL)	(AM)	(AH)	Length (low byte), Length (high byte)	Yes
TX BSL Version	No	0x19	–	–	–	–	Yes
RX Data Block Fast	Yes	0x1B	(AL)	(AM)	(AH)	D1…Dn	No

AL, AM, AH

Address bytes. The low, middle, and upper bytes, respectively, of an address.

D1 ... Dn

Data bytes 1 through n (Note: n must be 4 less than the BSL buffer size.)

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 next byte.

4.4.3.1 RX Data Block

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
RX Data Block	Yes	0x10	(AL)	(AM)	(AH)	D1…Dn	Yes

Description

The BSL core writes bytes D1 through Dn starting from the location specified in the address fields. This command differs from RX Data Block Fast in that it returns the status of the write operation.

Protection

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

Command

0x10

Command Address

Address to write the received data to.

Command Data

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

Command Returns

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

See Section 4.5 for more information on BSL core responses.

Command Example

Write data 0x76543210 to address 0x010000:

I2C	Header	Length	Length	CMD	AL	AM	AH	D1	D2	D3	D4	CKL	CKH
S/A/W	0x80	0x08	0x00	0x10	0x00	0x00	0x01	0x10	0x32	0x54	0x76	0x93	0xCA

BSL response for a successful data write:

I2C	ACK	Header	Length	Length	CMD	MSG	CKL	CKH	I2C
S/A/R	0x00	0x80	0x02	0x00	0x3B	0x00	0x60	0xC4	STOP

4.4.3.2 RX Password

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
RX Password	No	0x11	–	–	–	D1…D32	Yes

Description

The BSL core receives the password contained in the packet and unlocks the BSL protected commands if the password matches the top 16 words in the BSL interrupt vector table (located between addresses 0xFFE0 and 0xFFFF). When an incorrect password is given, a mass erase is initiated. This means all code flash is erased, but not Information Memory.

When a mass erase is performed the password is always 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 32 bytes long and contains the device password.

Command Returns

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

See Section 4.5 for more information on BSL core responses.

In case a mass erase is initiated no command will be returned.

Command Example

Unlock an empty device:

I2C	Header	Length	Length	CMD	D1	D2	D3	D4	D5
S/A/W	0x80	0x33	0x00	0x11	0xFF	0xFF	0xFF	0xFF	0xFF

D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

D16	D17	D18	D19	D20	D21	D22	D23	D24	D25
0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

D26	D27	D28	D29	D30	D31	D32	CKL	CKH
0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0x9E	0xE6

BSL Response for a successful password:

I2C	ACK	Header	Length	Length	CMD	MSG	CKL	CKH	I2C
S/A/R	0x00	0x80	0x02	0x00	0x3B	0x00	0x60	0xC4	STOP

4.4.3.3 Mass Erase

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
Mass Erase	No	0x15	–	–	–	–	No

Description

All code FRAM in the MSP430 is erased. This function does not erase Information Memory and RAM.

Protection

This command is not password protected.

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.

See Section 4.5 for more information on BSL core responses.

Command Example

Initiate a mass erase:

I2C	Header	Length	Length	CMD	CKL	CKH
S/A/W	0x80	0x01	0x00	0x15	0x64	0xA3

BSL does not respond after a mass erase command.

4.4.3.4 CRC Check

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
CRC Check	Yes	0x16	(AL)	(AM)	(AH)	Length (low byte), Length (high byte)	Yes

Description

The MSP430 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 for more details on the CRC used.

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 Section 4.5 for more information on BSL core responses.

Command Example

Perform a CRC check from address 0x4400 to 0x4800 (size of 1024):

I2C	Header	Length	Length	CMD	AL	AM	AH	D1	D2	CKL	CKH
S/A/W	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:

I2C	ACK	Header	Length	Length	CMD	D1	D2	CKL	CKH	I2C
S/A/R	0x00	0x80	0x03	0x00	0x3A	0x55	0xAA	0x12	0x2B	STOP

4.4.3.5 Load PC

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
Load PC	Yes	0x17	(AL)	(AM)	(AH)	–	No

Description

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

The BSL can be returned to by the main application using the BSL Action function 2, Return to BSL. See Section 3.5.1.2 for more information.

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

There is no response after successfully executing from the given address.

Command Example

Set program counter to 0x4400:

I2C	Header	Length	Length	CMD	AL	AM	AH	CKL	CKH	I2C
S/A/R	0x80	0x04	0x00	0x17	0x00	0x44	0x00	0x42	0x0F	STOP

The BSL does not respond once the application gains control.

4.4.3.6 TX Data Block

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
TX Data Block	Yes	0x18	(AL)	(AM)	(AH)	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

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 with a buffer size of 260, the BSL sends two packets–the first packet with a length of 259 and the second with a length of 253.

See Section 4.5 for more information on BSL core responses.

Command Example

Transmit 4 bytes of data from RAM address 0x1C00:

I2C	Header	Length	Length	CMD	AL	AM	AH	D1	D2	CKL	CKH
S/A/W	0x80	0x06	0x00	0x18	0x00	0x1C	0x00	0x04	0x00	0x87	0x81

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

I2C	ACK	Header	Length	Length	CMD	D1	D2	D3	D4	CKL	CKH	I2C
S/A/R	0x00	0x80	0x05	0x00	0x3A	0x11	0x33	0x55	0x77	0x90	0x55	STOP

4.4.3.7 TX BSL Version

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
TX BSL Version	No	0x19	–	–	–	–	Yes

Description

BSL transmits its version information (see Section 3.6 for details).

Protection

This command is not password protected.

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 Byte
BSL Vendor	D1
Command Interpreter	D2
API	D3
Peripheral Interface	D4

See Section 4.5 for more information on BSL core responses.

Command Example

Request the BSL version:

I2C	Header	Length	Length	CMD	CKL	CKH
S/A/W	0x80	0x01	0x00	0x19	0xE8	0x62

BSL response (version 00.07.34.B2 of the BSL):

I2C	ACK	Header	Length	Length	CMD	D1	D2	D3	D4	CKL	CKH	I2C
S/A/R	0x00	0x80	0x05	0x00	0x3A	0x00	0x07	0x34	0xB2	0x14	0x90	STOP

4.4.3.8 RX Data Block Fast

BSL Command	Protected	CMD	AL	AM	AH	Data	BSL Core Response
RX Data Block Fast	Yes	0x1B	(AL)	(AM)	(AH)	D1…Dn	No

Description

This command is identical to RX Data Block, except there is no reply indicating the data was correctly programmed. It is used primarily to speed up USB programming on the MSP430F5xx and MSP430F6xx family of devices.

Protection

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

Command

0x1B

Command Address

Address to write the received data to.

Command Data

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

Command Returns

BSL acknowledgment.

Command Example

Write data 0x76543210 to address 0x010000:

I2C	Header	Length	Length	CMD	AL	AM	AH	D1	D2	D3	D4	CKL	CKH
S/A/W	0x80	0x08	0x00	0x10	0x00	0x00	0x01	0x10	0x32	0x54	0x76	0x93	0xCA

BSL Response:

I2C	ACK	I2C
S/A/R	0x00	STOP

4.5 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 13. All numbers are in hexadecimal format.

Table 13. BSL Response Interface

Header	Length	Length	BSL Core Response	CKL	CKH
0x80	NL	NH	See Table 14	CKL	CKH

Table 14 shows the Core Responses from the BSL

Table 14. BSL Core Responses

BSL Response	CMD	Data
Data Block	0x3A	D1 … Dn
BSL Version(1)	0x3A	D1 … D4
CRC Value(1)	0x3A	DL, DH
BSL Core Message	0x3B	MSG

(1) Not supported for MSP430FR4xx, MSP430FR21xx, and MSP430FR20xx.

CMD

Required field that distinguishes between a message from the BSL and a data transmission from the BSL.

D1 to Dn

Data bytes containing the requested data.

DL, DH

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

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.

4.5.1 Data Block

BSL Response	CMD	Data
Data Block	0x3A	D1 … Dn

Description

Data Block is the response for the TX Data Block command. The command data contains the requested data bytes. Some requests may be separated into multiple packets.

See Section 4.3.3.6 for more information about the TX Data Block command.

Response Command

0x3A

Response Data

The requested data bytes, D1 through Dn where n is the number of bytes requested.

Response Example

BSL response when the host requests 4 data bytes, ordered D1 to D4:

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

4.5.2 BSL Version

BSL Response	CMD	Data
BSL Version	0x3A	D1 … D4

Description

BSL Version is the response for a TX BSL Version command.

See Section 4.3.3.7 for more information about the TX BSL Version command.

Response

0x3A

Response Data

The BSL version stored as four bytes, D1 to D4.

Response Example

BSL response for version 00.07.34.B2 of the BSL:

Header	Length	Length	CMD	D1	D2	D3	D4	CKL	CKH
0x80	0x05	0x00	0x3A	0x00	0x07	0x34	0xB2	0x14	0x90

4.5.3 CRC Value

BSL Response	CMD	Data
CRC Value	0x3A	DL, DH

Description

CRC Value is the response for a CRC Check command.

See Section 4.3.3.4 for more information about the CRC Check command.

Response

0x3A

Response Data

The calculated CRC value, stored DL, DH where DL is the low byte of the calculated checksum, and DH is the high byte of the calculated checksum.

Response Example

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	DL	DH	CKL	CKH
0x00	0x80	0x03	0x00	0x3A	0x55	0xAA	0x12	0x2B

4.5.4 BSL Core Message

BSL Response	CMD	Data
BSL Core Message	0x3B	MSG

Description

BSL Core Message is a response for several of the BSL commands. The data contains a single byte message code which corresponds to a particular BSL message as shown in Table 15.

Response

0x3B

Response Data

The BSL Core Message data is a single byte code stored as MSG. This message can be decoded according to Table 15.

Table 15. BSL Core Messages

MSG	Meaning
0x00	Operation successful
0x01	Flash write check failed. After programming, a CRC is run on the programmed data. If the CRC does not match the expected result, this error is returned.
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.

Response Example

BSL response for a successful command:

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

4.5.4.1 BSL Host Example

This example shows how a host programmer can establish a connection to the MSP430 BSL, erase the contents of the device, download an application to the device, and begin executing the application. Refer to Section 4 for more information about formatting the commands for the BSL peripheral interface that is used.

4.6 Overview and Flow Chart

A common BSL use case is to erase a device and load new firmware onto it. This can be accomplished by issuing a mass erase either through a failed RX Password command or by the Mass Erase command. After the application has been erased, the password is always 0xFF for all bytes. A second RX Password command is sent with the correct password to gain access to the device. After the BSL is unlocked, the application can be loaded using the TX Data Block command. If additional verification is desired, the host programmer can request a CRC on the application or read the programmed application code. After the application has been verified, the device can be reset or the host programmer can provide an address for the BSL to set the program counter and begin execution.

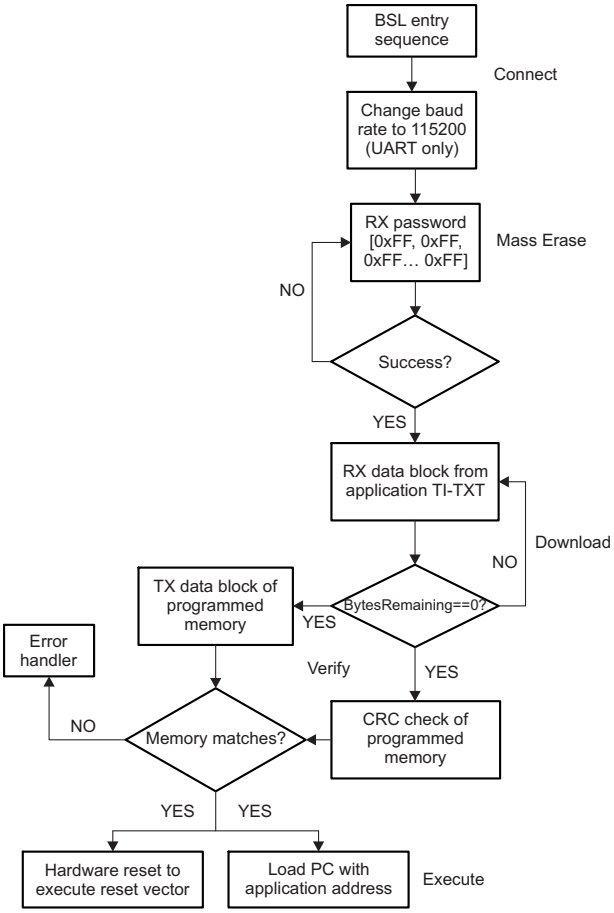


Figure 4. Flowchart

4.7 Establishing a Connection

Establishing a connection to the device BSL is the first step to loading an application through the BSL. The connection is initiated by providing the BSL entry sequence that is described in Section 3.5.2 to the $\overline{\text{RST}}$ and TEST pins.

For UART communication, the baud rate may be increased to speed up communication and downloads. The BSL begins operation at 9600 baud but allows the host to request a change to various preconfigured rates up to 115200 baud. TI recommends increasing the baud rate for large applications and production programming.

For more information about the Change Baud Rate command, see Section 4.3.3.9.

4.8 Erasing the Device

The next step in programming an application is to erase the device. This is possible either by using the Mass Erase command or by sending a wrong password with the RX Password command. The advantage of sending a wrong password instead of using a mass erase is the same programming flow can be used for an empty or programmed device. The host sends the password for an empty device, all 0xFF and checks the result. For an empty device, the password will be successful. For a programmed device, the password will fail, and the device will mass erase itself. The host can repeat this process and send the same password, which will now be successful for the erased device. This is especially useful for production programming from a host processor where the process flow is the same for the factory and the field.

For more information about the RX Password command, see Section 4.3.3.2. For more information about the Mass Erase command, see Section 4.3.3.3.

4.9 Downloading the Application

This step uses the RX Data Block command to download the application code to the device. A common file format that can easily be generated by MSP430 IDEs is TI-TXT. The download is split into smaller packets and sent with repeated RX Data Block commands. After all bytes have been sent and there is no remaining data, the host programmer can continue to the next step.

For more information about the RX Data Block command, see Section 4.3.3.1.

4.10 Verifying the Application

After downloading the application to the target, the next step is to verify the downloaded code. For this step, there are two methods that can be used, requesting a CRC check or reading out the application code.

The first method for verifying a download is to use the CRC Check command to request the device to perform a CRC check on a continuous section of the device memory. The calculated CRC value can be compared against the application CRC generated by the IDE or the host programmer.

The second method for verifying a download is to request the device to transmit the application code using the TX Data Block command. The data is sent from the device to the host, and the host can compare its original data to the application data that was downloaded from the device.

The advantage of using the CRC Check command over the TX Data Block for an application verify is that it is much quicker for a device to perform a CRC check on memory and then transmit a 2-byte CRC than it is to transmit every byte in the application back to the host (effectively doubling the time to program a device).

For more information about the CRC Check command, see Section 4.3.3.4. For more information about the TX Data Block command, see Section 4.3.3.6.

4.11 Executing the Application

The final step in to execute the application. There are two methods that can be used, setting the device program counter to an address to begin execution of the application, or performing a reset of the device. Often the simplest method when programming an entire application including the reset vector is to perform a reset, so that the bootcode invokes the application pointed to by the reset vector. The advantage of this is that no explicit address needs to be read or coded into the host programmer, and the same reset process can be used for any application downloaded as long as it includes the reset vector. Alternatively, the BSL can use the TX Data Block command to read the reset vector and then call the Load PC command to invoke the address that was read.

For more information about the Load PC command, see Section 4.3.3.5. For more information about the TX Data Block command, see Section 4.3.3.6.

5 Customizing the BSL

The BSL in FR4xx and FR2xx devices resides in ROM and does not allow for a custom BSL to be loaded into the standard BSL space. The standard BSL can be bypassed, and a custom start-up sequence that acts like a BSL can be used. This type of start-up sequence resides in main memory, giving full flexibility and control to the developer. MSPBoot is an example of such a bootloader.

MSPBoot – Main Memory Bootloader for MSP430 Microcontrollers describes the implementation of a bootloader that resides in the main memory in an MSP430 microcontroller and that uses I^2C communication. While highly flexible and modular, this bootloader has a small footprint, which makes it a very cost-effective solution.

6 Bootloader Versions

6.1 FR2xx BSL Versions

BSL Version	00.08.35.B3
Devices	MSP430FR2311, MSP430FR2310
RAM erased	0x2000-0x23FF
Buffer size for Core Commands	260
Notable Information	None
Known Bugs	None

BSL Version	00.08.35.B3
Devices	MSP430FR2433
RAM erased	0x2000-0x23FF
Buffer size for Core Commands	260
Notable Information	None
Known Bugs	None

BSL Version	00.08.35.B3
Devices	MSP430FR2633, MSP430FR2533, MSP430FR2632, MSP430FR2532
RAM erased	0x2000-0x23FF
Buffer size for Core Commands	260
Notable Information	None
Known Bugs	None

6.2 FR4xx, FR21xx, and FR20xx BSL Versions

BSL Version	00.87.45.74
Devices	MSP430FR4133, MSP430FR4132, MSP430FR4131, MSP430FR2111, MSP430FR2110, MSP430FR2033, MSP430FR2032
RAM erased	0x2000-0x21FF
Buffer size for Core Commands	260
Notable Information	None
Known Bugs	None

Revision History

Changes from April 21, 2016 to March 3, 2017 **Page**

- Corrected statement in Section 3.1.2 that the BSL's I^2C address is not parsed from TLV 4
- Added the last sentence in the first paragraph in Section 3.4.2, *BSL Memory Considerations* 5
- Renamed *FR4xx and FR20xx BSL Versions* table to *FR4xx, FR21xx, and FR20xx BSL Versions* 37
- Added MSP430FR2111 and MSP430FR2110 to BSL versions table 37

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