

## **MSP430™ FRAM Devices Bootloader (BSL)**

The bootloader (BSL) on MSP430™ microcontrollers (MCUs) lets users communicate with embedded memory in the MSP430 MCU 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.

### **Contents**

|   |   |    |
|---|---|----|
| 1 | Introduction .....                      | 2  |
|   | 1.1 BSL Limitations .....               | 2  |
|   | 1.2 Other Useful Documentation .....    | 2  |
| 2 | Overview of BSL Features.....           | 4  |
| 3 | BSL Architecture .....                  | 5  |
|   | 3.1 Communication Interface .....       | 5  |
|   | 3.2 BSL Memory.....                     | 5  |
|   | 3.3 BSL Invocation .....                | 6  |
|   | 3.4 BSL Time-out Feature.....           | 8  |
|   | 3.5 BSL Version Number.....             | 8  |
|   | 3.6 BSL (User) Configuration .....      | 9  |
| 4 | BSL Protocol .....                      | 12 |
|   | 4.1 BSL Data Packet .....               | 12 |
|   | 4.2 BSL Security .....                  | 26 |
| 5 | Common BSL Use Cases.....               | 27 |
|   | 5.1 Overview and Flow Chart .....       | 27 |
|   | 5.2 Establishing a Connection .....     | 28 |
|   | 5.3 Erasing the Device .....            | 28 |
|   | 5.4 Downloading the Application.....    | 28 |
|   | 5.5 Verifying the Application.....      | 28 |
|   | 5.6 Executing the Application .....     | 29 |
| 6 | Customizing the BSL .....               | 29 |
| 7 | Bootloader Versions .....               | 30 |
|   | 7.1 FR2xx BSL Versions .....            | 30 |
|   | 7.2 FR4xx BSL Versions .....            | 31 |
|   | 7.3 FR57xx BSL Versions .....           | 31 |
|   | 7.4 FR58xx and FR59xx BSL Versions..... | 32 |
|   | 7.5 FR6xx BSL Versions.....             | 33 |

### **List of Figures**

|   |   |    |
|---|---|----|
| 1 | Standard RESET Sequence .....               | 7  |
| 2 | BSL Entry Sequence at Shared JTAG Pins..... | 7  |
| 3 | Example BSL Version .....                   | 9  |
| 4 | Flowchart .....                             | 27 |

## Trademarks

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

## 1 Introduction

The MSP430 BSL lets users communicate with embedded memory in the MSP430 microcontroller (MCU) 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 must be applied to dedicated pins. BSL is also callable by 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, programming or reprogramming the memory, and verifying the written data). The bootloader 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 the BSL password. The BSL password is equal to the content of Interrupt Vector table on the device. For more information about password-protected commands, see [Section 4.2](#).

### 1.1 BSL Limitations

Compared to the BSL on MSP flash-based devices, the BSL on MSP FRAM-based devices bootloader is programmed in the read-only memory (ROM). See [MSP430FRBoot – Main Memory Bootloader and Over-the-Air Updates for MSP430 FRAM](#) for information on how to customize a bootloader in FR5xx and FR6xx devices. UART and I<sup>2</sup>C communication protocols are available. For FR4xx, FR21xx, and FR20xx devices, only UART is available at a rate of 9600 baud.

### 1.2 Other Useful Documentation

The device-specific data sheets include information on the pins used for BSL communication and the supported protocols.

#### Data Sheets

[MSP430FR573x Mixed-Signal Microcontrollers](#)

[MSP430FR572x Mixed-Signal Microcontrollers](#)

[MSP430FR59xx Mixed-Signal Microcontrollers](#)

[MSP430FR58xx Mixed-Signal Microcontrollers](#)

[MSP430FR698x\(1\), MSP430FR598x\(1\) Mixed-Signal Microcontrollers](#)

[MSP430FR21xx, MSP430FR2000 Mixed-Signal Microcontrollers](#)

#### Family User Guides

[MSP430FR58xx, MSP430FR59xx, and MSP430FR6xx Family User's Guide](#)

[MSP430FR57xx Family User's Guide](#)

[MSP430FR4xx and MSP430FR2xx Family User's Guide](#)

#### Tools and Application Notes

[MSP BSL Tool Folder](#)

**Training Videos**[MSP BSL Overview](#)[MSP BSL Options](#)[MSP Crypto bootloader Training Series](#)

## 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,<br>G2xx1,<br>G2xx2,<br>I20xx | F1xx,<br>F2xx,<br>F4xx,<br>G2xx3 | F5xx, F6xx             |                      | FR5xx, FR6xx |   | FR2x33,<br>FR231x | FR235x,<br>FR215x | FR413x,<br>FR211x | P401R                |
|                  |   |                                     |                                  | Non-<br>USB            | USB                  | Factory      | Crypto-<br>Boot-<br>loader <sup>(1)</sup> |                   |                   |                   |                      |
| General          | BSL memory type   | No BSL                              | ROM                              | Flash <sup>(2)</sup>   | Flash <sup>(2)</sup> | ROM          | FRAM                                      | ROM               | ROM               | ROM               | Flash <sup>(2)</sup> |
|                  | BSL memory size   | N/A                                 | 1 KB                             | 2 KB                   | 2 KB                 | 2 KB         | 4 KB                                      | 3 KB              | 3 KB              | 1 KB              | 8 KB                 |
|                  | Peripheral configured by TLV                                  |                                     |                                  |                        |                      | ✓            | ✓   |                   | ✓                 |                   | ✓                    |
|                  | User configuration  |                                     |                                  |                        |                      |              |   |                   | ✓                 |                   | ✓                    |
|                  | UART  |                                     | ✓                                | ✓                      |                      | ✓            | ✓   | ✓                 | ✓                 | ✓                 | ✓                    |
|                  | I <sup>2</sup> C  |                                     |                                  | ✓                      |                      | ✓            | ✓   | ✓                 | ✓                 |                   | ✓                    |
|                  | 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 <sub>USB</sub>    |                                  |                        | ✓                    |              |   |                   |                   |                   |                      |
|                  |   | 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                             | ✓                                | ✓                      |                      | ✓            | ✓   | ✓                 | ✓                 | ✓                 | ✓ <sup>(3)</sup>     |
|                  |   | USB cable                           |                                  |                        | ✓                    |              |   |                   |                   |                   |                      |
|                  | Software <sup>(1)</sup>                                       | BSL Scriptor                        |                                  | ✓                      | ✓                    | ✓            | ✓   | ✓                 | ✓                 | ✓                 | ✓                    |
|                  |   | BSLDEMO                             | ✓                                |                        |                      |              |   |                   |                   |                   |                      |
|                  |   | MSPBSL library                      | ✓                                | UART only              | ✓                    | UART only    |   |                   |                   | ✓                 |                      |
| Security         | Password protection   |                                     | 32 byte                          | 32 byte <sup>(4)</sup> | 32 byte              | 32 byte      |   | 32 byte           | 32 byte           | 32 byte           | 256 byte             |
|                  | Mass erase on incorrect password <sup>(5)</sup>               |                                     | ✓                                | ✓                      | ✓                    | ✓            |   | ✓                 | ✓                 | ✓                 | ✓                    |
|                  | Completely disable the BSL using signature or erasing the BSL |                                     |                                  | ✓                      | ✓                    | ✓            | ✓   | ✓                 | ✓                 | ✓                 | ✓                    |
|                  | BSL payload encryption  |                                     |                                  |                        |                      |              | ✓   |                   |                   |                   | ✓ <sup>(6)</sup>     |
|                  | Update of IP protected regions through boot code              |                                     |                                  |                        |                      |              |   |                   |                   |                   | ✓                    |
|                  | Authenticated encryption                                      |                                     |                                  |                        |                      |              | ✓   |                   |                   |                   |                      |
|                  | Additional security   |                                     |                                  |                        |                      |              | ✓ <sup>(7)</sup>                          |                   |                   |                   |                      |

<sup>(1)</sup> 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.

<sup>(2)</sup> BSL in flash memory allows to replace the BSL with a custom version.

<sup>(3)</sup> MSP-FET supports UART and I<sup>2</sup>C BSL communication only.

<sup>(4)</sup> F543x (non A) has a 16-byte password.

<sup>(5)</sup> Some devices can disable mass erase on incorrect password. See the device family user's guide.

<sup>(6)</sup> The decryption of the payload is performed by the device bootcode.

<sup>(7)</sup> Firmware validation through CRC.

## 3 BSL Architecture

### 3.1 Communication Interface

As [Table 1](#) shows, the communication protocols available for the BSL in FRAM devices are UART and I<sup>2</sup>C. Information regarding the available communication protocol and the hardware pins used for the BSL is available in the *Bootloader (BSL)* section of each device-specific data sheet.

#### 3.1.1 UART BSL

UART protocol used by BSL is defined as:

- Baud rate is configured to start at 9600 baud. The FR4xx, FR21xx, and FR20xx devices work at 9600 baud only, while other devices can change the baud rate to a higher speed after initialization.
- Works in half-duplex mode (one sender at a time)
- Handshake is performed by an acknowledge character
- The minimum time delay before sending new character after characters have been received from the MSP430 BSL is 1.2 ms.

---

**NOTE:** Applying a baud rate other than 9600 baud at initialization may result in communication problems.

---

#### 3.1.2 I<sup>2</sup>C BSL

I2C protocol used by BSL is defined as:

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

## 3.2 BSL Memory

### 3.2.1 BSL Memory Layout

BSL application is factory programmed in the read-only memory area of the MSP430 devices and cannot be customized. The size of the BSL application differs among the device families:

- FR4xx, FR21xx, and FR20xx BSL size is 1KB at address 0x1000 to 0x13FF.
- FR26xx, FR25xx, FR24xx, and FR23xx BSL size totals 3KB at address 0x1000 to 0x17FF for the first 2KB and 0xFFC00 to 0xFFFFF for the remaining 1KB.
- FR5xx and FR6xx BSL size is 2KB at address 0x1000 to 0x17FF.

#### 3.2.2 BSL Z-Area

When protected, the BSL memory cannot be read 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 startup or using certain public BSL functions.

The Z-Area is a special section of memory designed to allow for 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.

### 3.2.3 BSL Memory Consideration

The BSL partially clear 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 RAM for data buffer and local variables. When invoking the BSL from a main application, any RAM contents may be lost. The range of RAM that is cleared is listed in the tables in [Section 7](#).

## 3.3 BSL Invocation

The following methods can be used to invoke the BSL application on the MSP430 FRAM devices:

- The BSL can be called by application software (see [Section 3.3.1](#))
- The BSL can be called by the device boot code by applying a hardware entry sequence on TEST and RESET pins on the device (see [Section 3.3.2](#))
- The BSL can be invoked at start up if the device is blank (see [Section 3.3.3](#)). This is applicable only for FR26xx, FR25xx, FR24xx, and FR23xx devices

### 3.3.1 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.

#### 3.3.1.1 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 always reset, and RAM is cleared. Interrupts are not disabled by the BSL and should be disabled by the application before invoking the BSL.

TI recommends clearing the configuration of any module registers that are used in the BSL application, because the configuration for the external application can interrupt the BSL application and cause unexpected behavior. One example is that in the FR23xx and FR26xx MCUs, the Timer\_B module executes the time-out calculation of the BSL. If Timer\_B is also used in the external application and is not cleared before jumping to the BSL application, it could cause unexpected behavior.

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

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

---

**NOTE:** The BSL on FR5xx and FR6xx devices must be executed with a maximum frequency of 8 MHz. If the device operates at frequencies higher than 8 MHz, the MCLK frequency must be set to 8 MHz or lower before calling the BSL.

---

#### 3.3.1.2 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

The BSL on FR5xx and FR6xx devices always executes BSL Action function 2, regardless of the function number argument.

### 3.3.1.2.1 BSL Action Function 2

Function number: 2

Function Name: Return to BSL

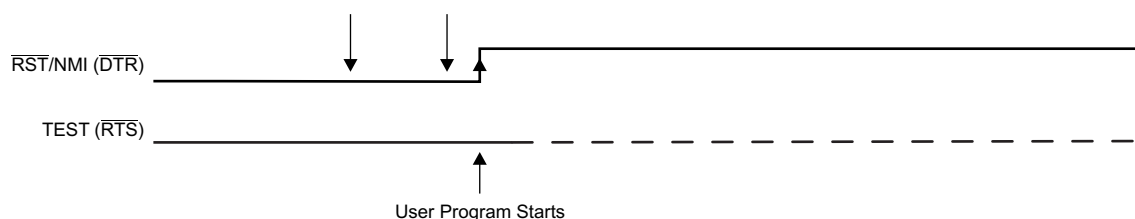
Description: Any supplied function number calls the return to BSL function. This function can be used if the BSL has written a program into FRAM or RAM, started that program by "Set PC", and then the program needs to return to the BSL. This function executes the following code:

```
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
```

### 3.3.2 Hardware BSL Invocation

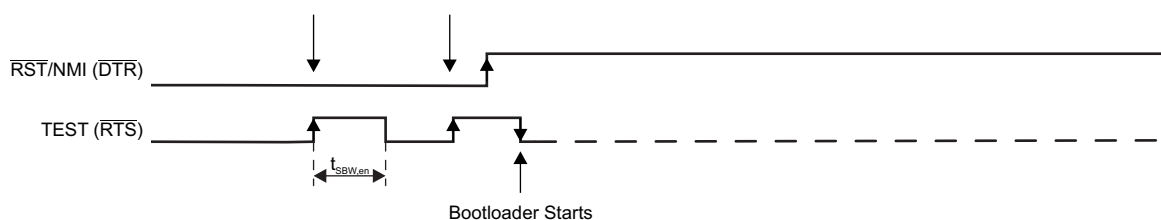
Applying an appropriate entry sequence on the  $\overline{\text{RST/NMI}}$  and TEST pins forces the MSP430 to start program execution at the BSL RESET vector instead of at the RESET vector located at address FFEh.

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  $\overline{\text{RST/NMI}}$  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  $\overline{\text{RST/NMI}}$  rises from low to high (BSL entry method, see [Figure 2](#)). This level transition triggering improves BSL startup reliability. The first high level of the TEST pin must be at least  $t_{\text{SBW, En}}$  (see device-specific data sheet for  $t_{\text{SBW, En}}$  parameter).



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  $\overline{\text{RST/NMI}}$  pins are also used to invoke the BSL. To invoke the BSL, the  $\overline{\text{RST/NMI}}$  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 after the  $\overline{\text{RST/NMI}}$  pin is released (see [Figure 2](#)).

### 3.3.2.1 Factors That Prevent Hardware BSL Invocation

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

- There are fewer than two negative edges at TCK pin while  $\overline{\text{RST}}/\text{NMI}$  is low.
- TCK is high if  $\overline{\text{RST}}/\text{NMI}$  rises from low to high.
- JTAG has control over the MSP430 MCU resources.
- The supply voltage,  $V_{\text{CC}}$ , drops below its threshold, and a power-on reset (POR) is executed.
- The  $\overline{\text{RST}}/\text{NMI}$  pin is configured for NMI functionality (the NMI bit is set).

### 3.3.3 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) for the BSL invocation sequence. 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 Time-out Feature

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.

## 3.5 BSL Version Number

The BSL version number can be requested by a host programmer using the TX BSL Version command (see [Section 4.1.5.7](#) for more information).

### Byte1: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 to 0x0F: Indicates that this BSL API interfaces with flash.

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

0x80 to 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 to 0x2F: Indicates a Timer\_A-based UART

0x30 to 0x4F: Indicates USB

0x50 to 0x6F: Indicates USCI-based UART

0x70 to 0x8F: Indicates eUSCI-based UART

0x90 to 0x9F: Indicates USCI-based I<sup>2</sup>C

0xA0 to 0xAF: Indicates eUSCI-based I<sup>2</sup>C

0xB0 to 0xBF: Indicates combined eUSCI I<sup>2</sup>C and UART



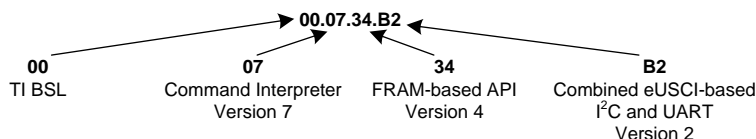


Figure 3. Example BSL Version

### 3.6 BSL (User) Configuration

The FR235x and FR215x BSL reads its configuration from the device descriptor (TLV) and additional settings from an user-configured area. The configuration data in the TLV is factory programmed and cannot be changed. Table 2 provides information about the peripheral modules and other data being used by the BSL. Section 3.6.1 describes the settings that can be modified by the user, including interface selection and RAM clearing. Initialization of the BSL that is based on the TLV data requires more time than other BSL versions. Initialization of the TLV-based BSL needs approximately 300 ms from the BSL entry invocation sequence until the BSL is ready to receive the first command.

Table 2. BSL Device Descriptor

| Description  | Address Offset     | Value    |
|--|--------------------|----------|
| BSL tag field  | 00h                | B5h      |
| BSL length field   | 01h                | 1Ch      |
| Timer base address (of the timer module used for time-out)                     | 02h, 03h           | Per unit |
| eUSCI_A base address (for UART)  | 04h, 05h           | Per unit |
| eUSCI_B base address (for I²C)   | 06h, 07h           | Per unit |
| Port base address (for UART pins)  | 08h, 09h           | Per unit |
| Port SELx and pin masks (for UART) (PxSEL0, PxSEL1, RX pin mask, TX pin mask)  | 0Ah, 0Bh, 0Ch, 0Dh | Per unit |
| Port base address (for I²C pins)   | 0Eh, 0Fh           | Per unit |
| Port SELx and pin masks (for I²C) (PxSEL0, PxSEL1, SDA pin mask, SCL pin mask) | 10h, 11h, 12h, 13h | Per unit |
| Reserved   | 14h, 15h           | –        |
| Reserved   | 16h, 17h           | –        |
| I²C slave address  | 18h                | Per unit |
| Tiny RAM length  | 19h                | Per unit |
| Tiny RAM start address   | 1Ah, 1Bh           | Per unit |
| Address of BSL user configuration data   | 1Ch, 1Dh           | Per unit |

#### 3.6.1 Configuring the BSL

The BSL user configuration is a data structure at a specific FRAM location. It is parsed by the BSL and used to configure features of the BSL.

The user configuration is optional. The BSL uses default values if no user configuration is found at the memory location.

To enable the user configuration, it must:

- Be placed at the correct location (see Section 7.1 for the device-specific address.)
- Start with the correct signature (see Table 3)

Table 3 lists the structure of the user configuration data. The address offset is with respect to the BSL user configuration location.

**Table 3. BSL User Configuration Structure**

| Description                    | Address Offset | Value    |   |
|--------------------------------|----------------|----------|---|
| BSL configuration signature    | 0h             | Bit 15-0 | Must be written as 695Ah or BSL user configuration will be ignored.   |
| BSL configuration              | 2h             | Bit 15-8 | Must be written as 5Ah or BSL user configuration will be ignored.   |
|                                |                | Bit 7-4  | Reserved  |
|                                |                | Bit 3    | Do not clear Tiny RAM on BSL invocation. A wrong BSL password always clears the memory. (Not every device has Tiny RAM.)<br>0b = Tiny RAM is cleared<br>1b = Tiny RAM is not cleared                    |
|                                |                | Bit 2    | Do not clear RAM on BSL invocation. A wrong BSL password always clears the memory. Refer to <a href="#">Section 7</a> for the RAM area being cleared.<br>0b = RAM is cleared<br>1b = RAM is not cleared |
|                                |                | Bit 1-0  | BSL interface selection:<br>00b = Automatic detection of UART or I2C BSL communication<br>01b = UART interface only<br>10b = I2C interface only<br>11b = Reserved                                       |
| Reserved                       | 4h-17h         | Bit 15-0 | Reserved  |
| I <sup>2</sup> C slave address | 18h            | Bit 15-7 | Reserved  |
|                                |                | Bit 6-0  | 7-bit address of the BSL I <sup>2</sup> C interface   |

### 3.6.1.1 Example of BSL User Configuration

[Table 4](#) is an example of user configuration for the MSP430FR2355 BSL. In this example, RAM clearing is disabled, and the I<sup>2</sup>C slave address is set to 0x60. The user configuration memory is at address FF88h.

**Table 4. Example of a BSL User Configuration**

| Address (Offset) | Value  | Description   |
|------------------|--------|---|
| 0xFF88 (0h)      | 0x695A | BSL configuration signature                         |
| 0xFF8A (2h)      | 0x5A04 | BSL configuration: RAM clearing disabled            |
| 0xFF8C to 0xFF9E | 0xFFFF | Reserved  |
| 0xFFA0           | 0xFF60 | 7-bit address of the BSL I <sup>2</sup> C interface |

The example in [Table 4](#) as TI-TXT:

```
@FF88
5A 69 04 5A FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF 60 FF
q
```

### 3.6.1.2 Implement BSL Configuration in Application Code

The linker command file for the FR235x and FR215x MCUs has the following sections:

- .bslconfigsignature
- .bslconfig
- .bsli2caddress

In the application code, define the following configuration values:

```
#define BSL_CONFIG_SIGNATURE                (0x695A) // BSL Configuration Signature
#define BSL_CONFIG_PASSWORD                (0x5A00) // BSL User's Configuration
Signature

#define BSL_CONFIG_TINY_RAM_ERASED_BY_INIT  (0xFFF7) // Tiny RAM is erased during
BSL invocation
#define BSL_CONFIG_TINY_RAM_NOT_ERASED_BY_INIT (0xFFFF) // (Default) Tiny RAM is not
erased during BSL invocation

#define BSL_CONFIG_RAM_ERASED_BY_INIT      (0xFFFB) // RAM is erased during BSL
invocation
#define BSL_CONFIG_RAM_NOT_ERASED_BY_INIT  (0xFFFF) // (Default) RAM is not erased
during BSL invocation

#define BSL_CONFIG_INTERFACE_UART_I2C      (0xFFFC) // Enable UART and I2C
#define BSL_CONFIG_INTERFACE_UART_ONLY    (0xFFFD) // Enable UART
#define BSL_CONFIG_INTERFACE_I2C_ONLY     (0xFFFE) // Enable I2C
#define BSL_CONFIG_INTERFACE_DEFAULT      (0xFFFF) // (Default / Reserved) Enable
UART and I2C

#define BSL_I2C_ADDRESS_7BIT               (0x60) // Customized I2C Address
```

Apply the intended configuration values to the BSL configuration section:

```
#pragma RETAIN(bslConfigurationSignature)
#pragma DATA_SECTION(bslConfigurationSignature, ".bslconfigsignature")
const uint16_t bslConfigurationSignature = BSL_CONFIG_SIGNATURE;

#pragma RETAIN(bslConfig)
#pragma DATA_SECTION(bslConfig, ".bslconfig")
const uint16_t bslConfig = (BSL_CONFIG_PASSWORD | BSL_CONFIG_INTERFACE_UART_ONLY) ;

#pragma RETAIN(bsI2CAddr)
#pragma DATA_SECTION(bsI2CAddr, ".bsli2caddr")
const uint16_t bsI2CAddr = (0xFF00 | BSL_I2C_ADDRESS_7BIT) ;
```

## 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 (see [Table 5](#)). See [Section 4.1.5](#) for details of the BSL Core Commands.

**Table 5. BSL Data Packet**

|         |                  |         |
|---------|------------------|---------|
| PI Code | BSL Core Command | PI Code |
|---------|------------------|---------|

The PI Code depends on the protocol in use, either UART or I<sup>2</sup>C. The following sections describe the UART and I<sup>2</sup>C peripheral interface wrappers.

#### 4.1.1 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. [Table 6](#) summarizes the packet format.

**Table 6. UART BSL Command**

| Header | Length | Length | BSL Core Command                  | CKL | CKH |
|--------|--------|--------|-----------------------------------|-----|-----|
| 0x80   | NL     | NH     | See <a href="#">Section 4.1.5</a> | CKL | CKH |

The second packet is received from the BSL device and contains the BSL acknowledgment and the BSL Core Response. [Table 7](#) lists the BSL response data packet structure.

**Table 7. UART BSL Response**

| Acknowledgment | Header | Length | Length | BSL Core Response <sup>(1)</sup>  | CKL | CKH |
|----------------|--------|--------|--------|-----------------------------------|-----|-----|
| ACK from BSL   | 0x80   | NL     | NH     | See <a href="#">Section 4.1.4</a> | CKL | CKH |

<sup>(1)</sup> BSL Core Response is not always included.

### 4.1.2 I<sup>2</sup>C Peripheral Interface Wrapper

The I<sup>2</sup>C 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. [Table 8](#) shows this format.

**Table 8. I<sup>2</sup>C BSL Command**

| I2C   | Header | Length | Length | BSL Core Command                  | CKL | CKH |
|-------|--------|--------|--------|-----------------------------------|-----|-----|
| S/A/W | 0x80   | NL     | NH     | See <a href="#">Section 4.1.5</a> | CKL | CKH |

The second packet is sent as a read request to the BSL slave address and contains the BSL acknowledgment and the BSL Core Response. [Table 9](#) shows the format of this BSL response.

**Table 9. I<sup>2</sup>C BSL Response**

| I2C   | ACK          | Header | Length | Length | BSL Core Response <sup>(1)</sup>  | CKL | CKH | I2C  |
|-------|--------------|--------|--------|--------|-----------------------------------|-----|-----|------|
| S/A/R | ACK from BSL | 0x80   | NL     | NH     | See <a href="#">Section 4.1.4</a> | CKL | CKH | STOP |

<sup>(1)</sup> BSL Core Response is not always included.

### 4.1.3 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 10](#) 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 needs to check the acknowledgment error and retry transmission.

**Table 10. 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 <sup>(1)</sup> | Packet size zero. The size for the BSL core command was given as 0.             |
| 0x54 <sup>(1)</sup> | Packet size exceeds buffer. The packet size given is too big for the RX buffer. |
| 0x55                | Unknown error   |
| 0x56 <sup>(1)</sup> | Unknown baud rate. The supplied data for baud rate change is not a known value. |
| 0x57                | Packet Size Error.  |

<sup>(1)</sup> Not supported for FR4xx, FR21xx, and FR20xx.

#### 4.1.4 BSL Core Response and BSL Core Message

For some commands, the BSL replies with certain BSL core response that contains single byte message on it. This 1-byte message is defined as BSL Core Message. For details about which command has the BSL core message on its response, see the [Section 4.1.5](#) examples.

**Table 11. BSL Core Response Structure (Has the BSL Core Message Byte)**

| CMD Byte | BSL Core Message                        |
|----------|---|
| 0x3B     | Message (see <a href="#">Table 12</a> ) |

**Table 12. BSL Core Messages**

| Message | Meaning  |
|---------|--|
| 0x00    | Operation successful   |
| 0x01    | Memory (for example, flash or FRAM) 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.  |

## 4.1.5 BSL Core Commands

Table 13 summarizes the BSL core commands.

**Table 13. BSL Core Commands**

| BSL Command                        | Protected | CMD  | AL   | AM   | AH   | Data                                     | BSL Core Response | Section                         |
|------------------------------------|-----------|------|------|------|------|--|-------------------|---------------------------------|
| RX Data Block                      | Yes       | 0x10 | (AL) | (AM) | (AH) | D1...Dn                                  | Yes               | <a href="#">Section 4.1.5.1</a> |
| RX Password                        | No        | 0x11 | –    | –    | –    | D1...D32                                 | Yes               | <a href="#">Section 4.1.5.2</a> |
| Mass Erase <sup>(1)</sup>          | No        | 0x15 | –    | –    | –    | –  | No                | <a href="#">Section 4.1.5.3</a> |
| CRC Check <sup>(1)</sup>           | Yes       | 0x16 | (AL) | (AM) | (AH) | Length (low byte),<br>Length (high byte) | Yes               | <a href="#">Section 4.1.5.4</a> |
| Load PC                            | Yes       | 0x17 | (AL) | (AM) | (AH) | –  | No                | <a href="#">Section 4.1.5.5</a> |
| TX Data Block                      | Yes       | 0x18 | (AL) | (AM) | (AH) | Length (low byte),<br>Length (high byte) | Yes               | <a href="#">Section 4.1.5.6</a> |
| TX BSL Version <sup>(1)</sup>      | Yes       | 0x19 | –    | –    | –    | –  | Yes               | <a href="#">Section 4.1.5.7</a> |
| RX Data Block Fast <sup>(1)</sup>  | Yes       | 0x1B | (AL) | (AM) | (AH) | D1...Dn                                  | No                | <a href="#">Section 4.1.5.8</a> |
| Change Baud Rate <sup>(1)(2)</sup> | No        | 0x52 | –    | –    | –    | D1                                       | No                | <a href="#">Section 4.1.5.9</a> |

<sup>(1)</sup> Not supported for FR4xx, FR21xx, and FR20xx

<sup>(2)</sup> Applicable for UART peripheral interface only

### 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.

#### 4.1.5.1 RX Data Block

##### Structure BSL Core Command

| 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.

**NOTE:** When a block is written partly outside the device's memory (for example, starting to write in FRAM but exceeding the end of the memory) the whole data block will not be written.

##### 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 4.1.4](#) for more information on BSL core responses.

##### Example for UART PI

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 |

##### Example for I<sup>2</sup>C PI

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 |



### 4.1.5.2 RX Password

#### Structure BSL Core Command

| 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, this is commonly used to gain access to an empty device or to load a new application to a locked device without password. The mass erase security feature can be disabled by setting the BSL signatures as described in the corresponding family user's guide (see [Section 1.2](#)).

#### 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 4.1.4](#) for more information on BSL core responses.

#### Example for UART PI

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 |

### Example for I<sup>2</sup>C PI

Unlock a blank 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 |

### 4.1.5.3 Mass Erase

#### Structure BSL Core Command

For FR23xx, FR25xx, and FR26xx:

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

For FR5xx and FR6xx:

| 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 4.1.4](#) for more information on BSL core responses.

#### Example for UART PI

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 |

#### Example for I<sup>2</sup>C PI

Initiate a mass erase:

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

BSL response (successful operation):

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

#### 4.1.5.4 CRC Check

##### Structure BSL Core Command

| 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 of each family user's guide (see [Section 1.2](#)) for more details on the CRC hardware calculation 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 4.1.4](#) for more information on BSL core responses.

##### Example for UART PI

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 |

##### Example for I<sup>2</sup>C PI

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 |

#### 4.1.5.5 Load PC

##### Structure BSL Core Command

| 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 3.3.1.2](#) for more information.

Be aware that password protection is not active at this time. Jumping to the user application does not reset the device and, therefore, the register configuration from the BSL application is kept. This might cause unexpected behavior in the user application. One example is the blink LED application, which does not have any clock module configuration (so it uses the default 1-MHz clock) and will blink faster, because the clock module is set by the BSL application to run at 8 MHz.

##### 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

The BSL does not return acknowledgment.

##### Example for UART PI

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.

##### Example for I<sup>2</sup>C PI

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.

#### 4.1.5.6 TX Data Block

##### Structure BSL Core Command

| 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:

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

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 4.1.4](#) for more information on BSL core responses.

##### Example for UART PI

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 |

##### Example for I<sup>2</sup>C PI

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 |

#### 4.1.5.7 TX BSL Version

##### Structure BSL Core Command

| 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 3.5](#) 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 4.1.4](#) for more information on BSL core responses.

##### Example for UART PI

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 |

##### Example for I<sup>2</sup>C PI

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 |

#### 4.1.5.8 RX Data Block Fast

##### Structure BSL Core Command

| 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.

**NOTE:** When a block is written partly outside the device's memory (for example, starting to write in FRAM but exceeding the end of the memory) the whole data block will not be written.

##### 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

##### Example for UART PI

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 |

##### Example for I<sup>2</sup>C PI

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 |



#### 4.1.5.9 Change Baud Rate

##### Structure BSL Core Command

| 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

##### Example for UART PI

Change baud rate to 115200:

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

BSL Response:

|      |
|------|
| ACK  |
| 0x00 |

## **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. In addition, commands specific to the peripheral interface are not protected.

### **4.2.2 RAM Erase**

At startup, the BSL performs a RAM erase, writing a constant word to certain RAM locations in a device. Usually these are the smallest shared RAM addresses within a family.

See [Section 7](#) for information about the RAM addresses that are erased.

### **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.

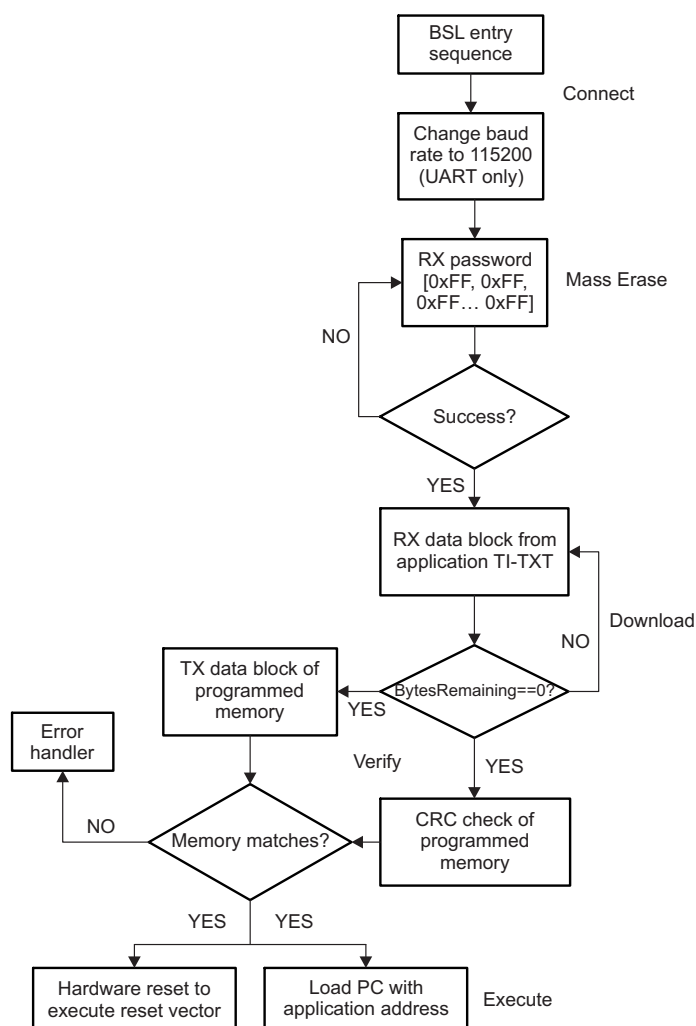
See [Section 3.2.2](#) for more information about the Z-Area.

## 5 Common BSL Use Cases

This section explains the flow of how a host programmer can establish a connection to the MSP430 BSL.

### 5.1 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**

## 5.2 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 3.3.2](#) to the 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 4.1.5.9](#).

## 5.3 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 4.1.5.2](#). For more information about the Mass Erase command, see [Section 4.1.5.3](#).

## 5.4 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 4.1.5.1](#).

## 5.5 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 two 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 4.1.5.4](#). For more information about the TX Data Block command, see [Section 4.1.5.6](#).

## 5.6 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 4.1.5.5](#). For more information about the TX Data Block command, see [Section 4.1.5.6](#).

## 6 Customizing the BSL

The BSL in MSP430 FRAM 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. [MSP430FRBoot – Main Memory Bootloader and Over-the-Air Updates for MSP430™ FRAM Large Memory Model Devices](#) describes the implementation of a main-memory resident bootloader for MSP430 FRAM microcontrollers using either universal asynchronous receiver/transmitter (UART) communication or a serial peripheral interface (SPI) bus and CC110x RF transceivers to accomplish over-the-air downloads (OAD). While still being highly flexible and modular, this bootloader maintains a small footprint, making it a very cost-effective solution, and supports the large memory model (devices with a memory footprint greater than 16KB).

## 7 Bootloader Versions

### 7.1 FR2xx BSL Versions

**Table 14.**

|                               |  |
|-------------------------------|--|
| BSL version                   | 00.08.35.B3  |
| Devices                       | MSP430FR2311, MSP430FR2310, MSP430FR2433, MSP430FR2633, MSP430FR2533, MSP430FR2632, MSP430FR2532, MSP430FR2522, MSP430FR2422 |
| RAM erased                    | 0x2000 to 0x23FF   |
| Buffer size for core commands | 260  |
| Notable information           | None   |
| Clock configuration           | Runs at 8 MHz  |
| Peripheral UART               | eUSCI_A  |
| Peripheral I <sup>2</sup> C   | eUSCI_B  |
| Timer module for time-out     | FR23xx: TIMER_B, FR26xx: TIMER_A   |
| Known bugs                    | None   |

**Table 15.**

|                               |  |
|-------------------------------|--|
| BSL version                   | 00.87.45.74  |
| Devices                       | MSP430FR2111, MSP430FR2110, MSP430FR2100, MSP430FR2000, MSP430FR2033, MSP430FR2032 |
| RAM erased                    | 0x2000 to 0x21FF   |
| Buffer size for core commands | 260  |
| Notable information           | None   |
| Clock configuration           | Runs at 8 MHz  |
| Peripheral UART               | eUSCI_A  |
| Peripheral I <sup>2</sup> C   | N/A  |
| Timer module for time-out     | N/A  |
| Known bugs                    | None   |

**Table 16.**

|                             |   |
|-----------------------------|---|
| BSL version                 | 00.09.36.B4   |
| Devices                     | MSP430FR2355, MSP430FR2353, MSP430FR2155, MSP430FR2153  |
| RAM erased                  | 0x2000 to 0x21FF  |
| Buffer size for core        | 260   |
| Notable information         | <ol style="list-style-type: none"> <li>1. The user overwrite configuration can be placed at 0xFF88. The device has no Tiny RAM and the clearing function has no effect.</li> <li>2. Initialization of the BSL that is based on the TLV data requires more time than other BSL versions. Initialization of the TLV-based BSL needs approximately 300 ms from the BSL entry invocation sequence until the BSL is ready to receive the first command.</li> </ol> |
| Clock configuration         | Runs at 8 MHz   |
| Peripheral UART             | eUSCI_A   |
| Peripheral I <sup>2</sup> C | eUSCI_B   |
| Timer module for time-out   | TIMER_B   |
| Known bugs                  | None  |

## 7.2 FR4xx BSL Versions

**Table 17.**

|                               |  |
|-------------------------------|--|
| BSL version                   | 00.87.45.74                              |
| Devices                       | MSP430FR4133, MSP430FR4132, MSP430FR4131 |
| RAM erased                    | 0x2000 to 0x21FF                         |
| Buffer size for core commands | 260                                      |
| Notable information           | None                                     |
| Clock configuration           | Runs at 8 MHz                            |
| Peripheral UART               | eUSCI_A                                  |
| Peripheral I <sup>2</sup> C   | N/A                                      |
| Timer module for time-out     | N/A                                      |
| Known bugs                    | None                                     |

## 7.3 FR57xx BSL Versions

**Table 18.**

|                               |  |
|-------------------------------|--|
| BSL version                   | 00.04.31.71  |
| Devices                       | MSP430FR5739, MSP430FR5738, MSP430FR5737, MSP430FR5736, MSP430FR5735, MSP430FR5734, MSP430FR5733, MSP430FR5732, MSP430FR5731, MSP430FR5730, MSP430FR5729, MSP430FR5728, MSP430FR5727, MSP430FR5726, MSP430FR5725, MSP430FR5724, MSP430FR5723, MSP430FR5722, MSP430FR5721, MSP430FR5720 |
| RAM erased                    | 0x1C00 to 0x1FFF   |
| Buffer size for core commands | 260  |
| Notable Information           | 1. TX and RX pins are noted in the device data sheet<br>2. A mass erase command or incorrect password triggers a BSL reset. This resets the BSL state to the default settings (9600 baud, password locked)   |
| Clock configuration           | Runs at 8 MHz  |
| Peripheral UART               | eUSCI_A  |
| Peripheral I <sup>2</sup> C   | N/A  |
| Timer module for time-out     | N/A  |
| Known bugs                    | 1. The baud rate of 115k cannot be ensured across all clock, voltage, and temperature variations.  |

## 7.4 FR58xx and FR59xx BSL Versions

**Table 19.**

|                               |   |
|-------------------------------|---|
| BSL version                   | 00.07.34.B2   |
| Devices                       | MSP430FR5969, MSP430FR59691, MSP430FR5968, MSP430FR5967, MSP430FR5949, MSP430FR5948, MSP430FR5947, MSP430FR59471, MSP430FR5959, MSP430FR5958, MSP430FR5957, MSP430FR5869, MSP430FR5868, MSP430FR5867, MSP430FR58671, MSP430FR5849, MSP430FR5848, MSP430FR5847, MSP430FR58471, MSP430FR5859, MSP430FR5858, MSP430FR5857, MSP430FR5972, MSP430FR59721, MSP430FR5872, MSP430FR58721, MSP430FR5989, MSP430FR59891, MSP430FR5988, MSP430FR59881, MSP430FR5987, MSP430FR59871, MSP430FR5986, MSP430FR5889, MSP430FR58891, MSP430FR5888, MSP430FR58881, MSP430FR5887, MSP430FR58871, MSP430FR5970, MSP430FR5870, MSP430FR5922, MSP430FR59221 |
| RAM erased                    | 0x1C00 to 0x1FFF  |
| Buffer size for core commands | 260   |
| Notable information           | None  |
| Clock configuration           | Runs at 8 MHz   |
| Peripheral UART               | (FR5xxx or FR6xxx) eUSCI_A0   |
| Peripheral I <sup>2</sup> C   | (FR5xxx1 or FR6xxx1) eUSCI_B0   |
| Timer module for time-out     | N/A   |
| Known bugs                    | None  |

**Table 20.**

|                               |  |
|-------------------------------|--|
| BSL version                   | 00.08.35.B3  |
| Devices                       | MSP430FR5994, MSP430FR59941, MSP430FR5992, MSP430FR5964, MSP430FR59641, MSP430FR5962, MSP430FR5894, MSP430FR58941, MSP430FR5892, MSP430FR5864, MSP430FR58641, MSP430FR5862 |
| RAM erased                    | 0x1C00 to 0x1FFF   |
| Buffer size for core commands | 260  |
| Notable information           | None   |
| Clock configuration           | Runs at 8 MHz  |
| Peripheral UART               | (FR5xxx or FR6xxx) eUSCI_A0  |
| Peripheral I <sup>2</sup> C   | (FR5xxx1 or FR6xxx1) eUSCI_B0  |
| Timer module for time-out     | N/A  |
| Known bugs                    | None   |



## 7.5 FR6xx BSL Versions

**Table 21.**

|                               |  |
|-------------------------------|--|
| BSL version                   | 00.07.34.B2  |
| Devices                       | MSP430FR6972, MSP430FR69721, MSP430FR6872, MSP430FR68721, MSP430FR6922, MSP430FR69221, MSP430FR6822, MSP430FR68221, MSP430FR6970, MSP430FR6870, MSP430FR6920, MSP430FR6820, MSP430FR6989, MSP430FR69891, MSP430FR6988, MSP430FR69881, MSP430FR6987, MSP430FR69871, MSP430FR6889, MSP430FR68891, MSP430FR6888, MSP430FR68881, MSP430FR6887, MSP430FR68871, MSP430FR6979, MSP430FR6879, MSP430FR6977, MSP430FR6877, MSP430FR6928, MSP430FR6927 |
| RAM erased                    | 0x1C00 to 0x1FFF   |
| Buffer size for core commands | 260  |
| Notable information           | None   |
| Clock configuration           | Runs at 8 MHz  |
| Peripheral UART               | (FR5xxx or FR6xxx) eUSCI_A0  |
| Peripheral I <sup>2</sup> C   | (FR5xxx1 or FR6xxx1) eUSCI_B0  |
| Timer module for time-out     | N/A  |
| Known bugs                    | None   |

**Table 22.**

|                               |  |
|-------------------------------|--|
| BSL version                   | 00.08.35.B3  |
| Devices                       | MSP430FR6047, MSP430FR60471, MSP430FR6045, MSP430FR6037, MSP430FR60371, MSP430FR6035, MSP430FR6043, MSP430FR60431, MSP430FR6041, MSP430FR5043, MSP430FR50431, MSP430FR5041 |
| RAM erased                    | 0x1C00 to 0x1FFF   |
| Buffer size for core commands | 260  |
| Notable information           | None   |
| Clock configuration           | Runs at 8 MHz  |
| Peripheral UART               | (FR5xxx or FR6xxx) eUSCI_A0  |
| Peripheral I <sup>2</sup> C   | (FR5xxx1 or FR6xxx1) eUSCI_B0  |
| Timer module for time-out     | N/A  |
| Known bugs                    | None   |

## Revision History

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

| Changes from April 12, 2018 to May 4, 2018  | Page |
|---|------|
| • Added column for FR235x and FR215x to <a href="#">Table 1</a> , <i>BSL Overview</i> .....             | 4    |
| • Added <a href="#">Section 3.6</a> , <i>BSL (User) Configuration</i> , for FR235x and FR215x BSL ..... | 9    |
| • Added <a href="#">Table 16</a> for FR235x and FR215x .....  | 30   |

## 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/sampterm.htm) (<http://www.ti.com/sc/docs/sampterm.htm>).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2018, Texas Instruments Incorporated