Application Report
SLAA659D–March 2015–Revised July 2017

Configuring Security and Bootloader (BSL) on MSP432P4xx Microcontrollers

Dung Dang *MSP Applications*

ABSTRACT

With the increasing threats of IP theft and cyberattack, security has changed from a feature to a basic requirement for any embedded application. The MSP432P4xx family of devices offers a number of security measures to meet the new security requirements including JTAG security, IP protection, and encryption-enabled bootloader (BSL) (formerly known as the bootstrap loader). In addition, the MSP432P4xx family does not compromise usability when adding the security options. In fact, it offers several methods to enable debugging when adding security or configuring BSL in the system. This application note covers the unique mailbox mechanism used to program and debug device security and BSL on MSP432™ MCUs. The application note also provides examples and guidance to add these features to an existing application.

Source code and related collateral discussed in this document can be downloaded from http://www.ti.com/lit/zip/slaa659.

Contents

List of Tables

Trademarks

MSP432 is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

1 MSP432™ Security and BSL Overview

MSP432P4xx family of devices supports the following security and BSL options:

- JTAG/SWD security: the device can be permanently locked from further JTAG/SWD access after programmed with the final firmware. The device can regain JTAG access only after going through the Factory Reset operation.
- IP Protection: the device has up to four regions that can be individually locked from debug and read accesses after the regions are programmed with the IP, which can be both functions (API) and data. An AES-CBC password is used and preprogrammed for each IP protection region. When a region is configured with IP protection, the rest of the device can still be programmed and debugged. All JTAG/SWD debug or memory accesses on the IP-protected zone will be stalled and returned with error code.
- Bootloader: the device firmware can be updated through the BSL without requiring any JTAG access. BSL can be configured to have multiple levels of security to support encryption, JTAG/SWD security, as well as for selective IP protection updates.
- BSL can be used for both unsecured and secured devices (JTAG/SWD security or IP Protection).
- BSL always requires an entry password to unlock further BSL commands.
- BSL payload encryption is possible; this is particularly useful for device secured with JTAG/SWD security. Using a preprogrammed AES-CBC password, the entire firmware payload can be securely transferred through BSL communication. The encrypted payload can then be decrypted by the device boot-code.
- BSL can be used to update individual IP protection regions. An encrypted IP using the prealigned AES-CBC password can be securely transferred through BSL into the device free memory. The protected IP can then be decrypted by the device boot-code to be programmed and secured into the IP protection region.
- Factory Reset: After the device is locked from JTAG/SWD access, the device can be accessed again only after going through the JTAG Factory Reset operation, which completely erases all of the device main flash and security content (information memory is not affected). Factory Reset can also be used to erase any existing IP Protection configuration on the device.

2 MSP432™ Security and BSL Configuration Mechanism

All of the security measures and field firmware update options described in Section 1 can be configured and executed through one consistent mechanism called boot-override using a special flash mailbox.

This application note describes how to use the flash mailbox with custom code. An alternative method to configure security through the flash mailbox is to use the graphical interface of the SimpleLink MSP432P4xx Security and Update Tool.

Boot overrides are special functions executed by the device boot-code. The device boot-code is provided by TI to execute code in a secure environment at device start-up or reset, before the device provides access to the application code or the debugger. To instruct the boot-code to execute these special boot override functions, a secure one-way communication is used. On the MSP432P4xx family specifically, a flash sector is used by the application or the debugger to pass in information and configuration data to be accessed by the boot-code after a Reboot Reset. This method is similar to a secure mailbox, hence the name for the flash sector is Boot-Override Flash Mailbox.

After a Reboot Reset, the boot code performs the commands and operations as programmed into the flash mailbox. The result of each boot-override operation is programmed back into the flash mailbox in the result/status memory address. The boot-override operation is complete at this point, allowing for the remaining boot-code operation to resume before turning the access over to the main application. Care must be taken however, to ensure that the flash mailbox content should be erased after a successful boot override operation. This procedure needs to be handled either through JTAG (if JTAG access was not locked) or through application code.

The flash mailbox is located at the flash memory region accessible to the application and debugger, usually residing together with the device's other information memory sectors. For example, on the MSP432P401R device, the flash mailbox is located at 0x0020.0000.

The Flash mailbox area consists of a fixed structure of commands and parameters to configure the security and BSL as described in Section 1. Each command or parameter has its fixed designated address, and they can be configured by writing directly to these addresses.

Because the flash mailbox is just a flash sector accessible by the application or by the debugger (JTAG/SWD), boot-override operations can be configured as part of the firmware (firmware binary containing data for the flash mailbox sector), as the final step after programming the application firmware (using JTAG), or by the application during runtime.

The structure and memory layout of these commands and parameters are detailed in the Device Security section of the *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual*. Table 1 is a high-level overview of the structure.

Table 1. MSP432P4xx Flash Mailbox Structure

Group	Offset	Description
Command and General Parameters	0x000	Command for the Boot-override operation
JTAG/SWD Lock Configuration	0x010	Enable or disable lock, AES-CBC password for encrypted update
IP Protection – Zone 0	0x060	Address and AES-CBC password for IP Protection - Zone 0
IP Protection – Zone 1	0x0C0	Address and AES-CBC password for IP Protection - Zone 1
IP Protection – Zone 2	0x120	Address and AES-CBC password for IP Protection - Zone 2
IP Protection – Zone 3	0x180	Address and AES-CBC password for IP Protection - Zone 3
BSL Configuration	0x1E0	BSL address, interface selection, and hardware invoke
JTAG/SWD Encryption Update	0x1F8	Address and length for encrypted update firmware
IPP– Update - Zone 0	0x20C	Address and length for encrypted update firmware for IPP-0
IPP– Update - Zone 1	0x21C	Address and length for encrypted update firmware for IPP-1
IPP– Update - Zone 2	0x22C	Address and length for encrypted update firmware for IPP-2
IPP– Update - Zone 3	0x23C	Address and length for encrypted update firmware for IPP-3

A complete boot-override configuration procedure involves writing data to two areas:

- First, into the command and general parameters group, specifying which security/BSL/update feature to configure.
- Second, into the data group associated with the selected security/BSL/update feature (see Table 1).

Table 2 provides the different types of boot-override commands available on MSP432P4xx family of devices.

Table 2. MSP432P4xx Boot-Override Commands

Boot-Override Command	Value	Description
JTAG and SWD lock	0x00080000	Secure the application code from debug through JTAG or Serial Wire Debug (SWD/2-wire JTAG debug) mechanisms. This is similar to JTAG and SBW lock available on the TI MSP430 product families.
Secure Zone X configure (X=0/1/2/3)	0x00100000, 0x00200000, 0x00400000, 0x00800000	Configure and secure the software IPs that the user wishes to create. Code executing in these secure zones has all the characteristics mentioned in Section 1.
Secure Zone X Update (X=0/1/2/3)	0x01000000, 0x02000000, 0x04000000, 0x08000000	Perform an in-field update to software IP already secured using the "Secure Zone X Enable" command.
JTAG and SWD lock encrypted update	0x10000000	Perform an encrypted update to any application code in the memory of the device which is not protected by "Secure Zone X Enable".
Bootloader (BSL) configuration	0x00020000	Configure TI BSL interface and/or hardware invocation method or enable a custom BSL.
Factory reset	0x00010000	Erase all application code from the flash main memory of the system and remove all device security protections after the flash main memory has been successfully erased.

This application note focuses on the configuration procedures required at development, debug, and production stages (that is, the options not grayed out in Table 2). For details on field firmware updates for both devices with JTAG/SWD lock and devices with IP protection, see the *MSP432™ SimpleLink™ Microcontrollers Bootloader (BSL) User's Guide* or *Software IP Protection on MSP432P4xx Microcontrollers*.

The Flash mailbox includes the Factory Reset option, where all Flash memory content as well as device security protections are erased, restoring the device completely back to factory condition. While this is possible with the flash mailbox (for details, see Section 3), it is also possible to invoke the Factory Reset through JTAG on locked and protected devices. This method is also available inside the IDEs that support MSP432. See the *Code Composer Studio™ IDE 7.1+ for SimpleLink™ MSP432™ MCUs User's Guide*, *IAR Embedded Workbench for ARM 7.x for SimpleLink™ MSP432™ MCUs User's Guide*, or *ARM® Keil® MDK Version 5 for SimpleLink™ MSP432™ MCUs User's Guide* for more information on this option.

Table 3. What to Use for Different Security Features

Goal	Method	Document
Adding one or more of the following options to an existing application: 1. Add BSL configuration 2. Add a new IP protection zone configuration 3. Lock device from JTAG/SWD access 4. Factory reset	Add flash mailbox array to application code	This document, *Configuring Security and Bootloader (BSL) on MSP432P4xx Microcontrollers* *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual*
Perform field firmware update (BSL) for nonsecure, encrypted, or IP-protected device	Add flash mailbox array as part of the BSL firmware	*MSP432™ SimpleLink™ Microcontrollers Bootloader (BSL) User's Guide* This document, *Configuring Security and Bootloader (BSL) on MSP432P4xx Microcontrollers*
Configure IP protection zone and IP protection for use or debug	Add flash mailbox array to application code	*Software IP Protection on MSP432P4xx Microcontrollers*
Perform a factory reset inside an IDE	Use IDE options	*Code Composer Studio™ IDE 7.1+ for SimpleLink™ MSP432™ MCUs User's Guide* *IAR Embedded Workbench for ARM 7.x for SimpleLink™ MSP432™ MCUs User's Guide* *ARM® Keil® MDK Version 5 for SimpleLink™ MSP432™ MCUs User's Guide*

3 Adding Flash Mailbox Configuration to an MSP432™ Project

The section focuses on the method to incorporate the Flash mailbox configuration into your application firmware. This can be achieved by adding an array of data to be programmed at the flash mailbox sector location. In other words, a complete flash mailbox configuration can be added as part of the data to be programmed together with the firmware. A flash mailbox c file template is provided to aid with the process.

After the application firmware and the flash mailbox content are programmed, the debugger probe or JTAG production programming tool can issue a Reboot Reset, which resets the device and triggers the boot-code to perform the commands as prescribed in the flash mailbox. The completion and success of this operation can be determined by the status flags set inside the flash mailbox.

The last step of the security/BSL configuration operation is to remove the flash mailbox content. Because in many use cases JTAG access is no longer available at this point due to JTAG/SWD lock, this procedure must be taken by the main application code. An API is provided to be added to the beginning of the application code to check for boot-override operation result and erase the flash mailbox content.

Source code discussed in the following sections can be downloaded from http://www.ti.com/lit/zip/slaa659.

3.1 Advantages

Security options can be configured at the same time as firmware programming during the production flow.

Multiple security options (JTAG lock, one or several IP Protection zones, and BSL configuration) can be programmed all at once.

Solution and data are contained within a c file to be compiled into one final firmware together with the application.

Solution can be debugged within the IDE right after the device is programmed with the firmware.

3.2 Flash Mailbox Configuration Flow

Step 1: Add *msp432_flashmailbox.c*, *flashmailbox.c*, and *flashmailbox.h* to your project.

Step 2: Add MSP432 DriverLib to your project. DriverLib is required because of the DriverLib Flash erase API.

Step 3: Edit the *FlashMailBox* array inside msp432_flashmailbox.c to configure your security and BSL options. Note that multiple options can be programmed at once. See Section 3.3 for details on configuring the FlashMailBox array.

Step 4: Add #include "flashmailbox.h" to the main application source code. See Section 3.4 for more information on how to use this function.

Step 5: Add the CheckAndEraseFlashMailBox API call to the beginning of the main application such the following. Pass in the same parameter that you used to configure the boot-override functions.

```
#include "flashmailbox.h"
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;
    if (!CheckAndEraseFlashMailBox(bootOverrideParameter))
    {
        // Add security failure handling code here
        while(1);
    }
}
```

3.3 Configuring FlashMailBox Array

Follow the comments and instructions for each line of the FlashMailBox[] array inside the msp432_flashmailbox.c to fill in the array data.

Only change the necessary fields, leave the rest of the fields unmodified as 0xFFFFFFFF.

For the command parameter, use one or a bit-field combination of COMMAND_xxxx definitions or COMMAND_NONE to select the desirable commands to be executed by the boot code.

3.4 Using CheckAndEraseFlashMailBox API

uint32_t CheckAndEraseFlashMailBox() takes in one parameter, uint32_t *flashMailBoxCommand*, which should be the same parameter used to configure the boot-override operations. This parameter can be constructed using the definitions found in the MSP432P4xx device header files:

```
#define COMMAND_FACTORY_RESET                  0x00010000
#define COMMAND_BSL_CONFIG                     0x00020000
#define COMMAND_JTAG_SWD_LOCK_SECEN            0x00080000
#define COMMAND_SEC_ZONE0_EN                   0x00100000
#define COMMAND_SEC_ZONE1_EN                   0x00200000
#define COMMAND_SEC_ZONE2_EN                   0x00400000
#define COMMAND_SEC_ZONE3_EN                   0x00800000
#define COMMAND_SEC_ZONE0_UPDATE               0x01000000
#define COMMAND_SEC_ZONE1_UPDATE               0x02000000
#define COMMAND_SEC_ZONE2_UPDATE               0x04000000
#define COMMAND_SEC_ZONE3_UPDATE               0x08000000
#define COMMAND_JTAG_SWD_LOCK_ENC_UPDATE       0x10000000
#define COMMAND_NONE                           0xFFFFFFFF
```

The main application should invoke this function call at the beginning. The application should proceed only if the function returns with success, indicating that all security features have been successfully configured, and the flash mailbox content has been successfully erased.

3.5 Example

Use the example project associated with this application note for an example on how to incorporate security and BSL configuration into an existing project.

Examples are provided to demonstrate common flash mailbox use cases. All examples leverage the FlashMailBox[] array to provide configuration and update for the flash mailbox. The main application code for each example also includes the necessary routines to verify that flash mailbox operation is completed and erase the flash mailbox sector after operation is complete. When using RevC silicon, it is no longer necessary to erase the flash mailbox.

Silicon RevB boot code does not clean up the mailbox. To clean the mailbox, either run the clean_mailbox example or include the clean up code in the other examples after checking the mailbox code.

NOTE: The flash mailbox structure between RevB and RevC material has slightly changed. If you still want to use these examples with RevB material, review the msp432_flashmailbox_RevB.c file delivered with the examples and change the flash mailbox structure accordingly.

- empty_project: Includes only the default FlashMailBox array[] and the flash mailbox functions required by the operation. This project can be used as a template to incorporate into your main application project.
- clean_mailbox: This example erases the mailbox in case of a misconfiguration that is not cleaned up by the boot loader. After doing this, the application exits.
- bsl_config: Add BSL configuration to a project. Can be part of the initial production programming procedure.
- disable_jtag_swd: Disable JTAG and SWD access to the device. This can be done as the final step of the production programming procedure.
- ipp_config: Configure one or multiple IP Protection zones. Two projects are provided for this example:
  - First project (ipp_config_1_secure_ip) to configure an IP Protection zone: can be used by one up to four IP vendors to program their secret IP into a memory region and then secure this zone.
  - Second project (ipp_config_2_run_ip) to demonstrate how to call into an API inside a IP protection zone from the final application.
- ipp_update_jtag: Demonstrate how to update a secured IP protection zone with a new version of the IP using JTAG or SWD. Two projects are provided for this example:
  - First project (ipp_jtag_1_secure_ip) to configure an IP Protection zone: This demonstrates how an software IP developer can develop IP and then secure it into devices at a fixed address. This allows IP developers to test and update their secured IP during development. The mailbox is also configured to accept an update to the secure IP using a password.
  - Second project (ipp_jtag_2_run_ip) to program an updated IP into address 0x0000.8000 along with a password into address 0x0000.9000 to update a secure IP.
- ipp_update_bsl: Demonstrate how to update a secured IP protection zone with BSL. This method is closer to a realistic secure IP and firmware update scenario than the previous example. Two examples are provided:
  - First project (ipp_bsl_1_secure_ip) to configure an IP Protection zone. In contrast to the JTAG secure update example, the JTAG and SWD are also disabled so only updates through BSL are allowed.
  - Second project (ipp_bsl_2_update_ip) to program an updated IP into address 0x0000.8000 along with a password into address 0x0000.9000 to update a secure IP. This code should be loaded to the device by using the BSL.

4 References

1. *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual*
2. *Code Composer Studio™ IDE 7.1+ for SimpleLink™ MSP432™ MCUs User's Guide*
3. *IAR Embedded Workbench for ARM 7.x for SimpleLink™ MSP432™ MCUs User's Guide*
4. *ARM® Keil® MDK Version 5 for SimpleLink™ MSP432™ MCUs User's Guide*
5. *Software IP Protection on MSP432P4xx Microcontrollers*
6. *MSP432™ SimpleLink™ Microcontrollers Bootloader (BSL) User's Guide*
7. *SimpleLink™ MSP432P4xx Security and Update Tool User's Guide*

Revision History

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

Changes from September 14, 2016 to July 14, 2017	Page
• Added that information memory is not affected by a factory reset in Section 1, *MSP432™ Security and BSL Overview*	2
• Throughout document, updated titles of referenced documents as necessary	2