AN1303: Programming Series 2 Devices using the Debug Challenge Interface (DCI) and Serial Wire Debug (SWD)

This application note describes how to provision and configure Series 2 devices through the dedicated Debug Challenge Interface (DCI). The process to use the Serial Wire Debug (SWD) interface for programming the internal flash memory of Series 2 devices is also included.

For details on how to use the SWD interface to program devices, see AN0062: Programming Internal Flash over the Serial Wire Debug Interface.

KEY POINTS

- DCI overview
- SWD interface overview
- DCI programmer examples to provision and configure Series 2 devices
- SWD programmer examples to program Series 2 devices

1. Device Compatibility

This application note supports Series 2 device families, and some functionality is different depending on the device.

MCU Series 2 consists of:

- EFM32PG22

Wireless SoC Series 2 consists of:

- EFR32BG21A/EFR32BG21B/EFR32BG22
- EFR32FG22
- EFR32MG21A/EFR32MG21B/EFR32MG22

2. Introduction

On Series 2 devices, the security features are implemented by the Secure Engine. The Secure Engine may be hardware-based, or virtual (software). If hardware-based, the implementation may be either with or without Secure Vault. Throughout this document, the following conventions will be used.

- HSE - Hardware Secure Engine, either with or without Secure Vault if not specified
- VSE - Virtual Secure Engine
- SE - Secure Engine (either HSE or VSE, in general)

For more information about SE, see AN1190: Series 2 Secure Debug.

At the time of this writing, the latest SE firmware shipped with Series 2 devices and modules (if available) are listed in the following table.

Table 2.1. Latest SE Firmware Version Shipped on Series 2 Devices and Modules

Wireless SoC Series 2	SE	Shipped SE Firmware Version (Device and Module)
EFR32xG21A	HSE without Secure Vault	Version 1.2.6
EFR32xG21B	HSE with Secure Vault	Version 1.2.6
EFR32xG22	VSE	Version 1.2.6

The Debug Challenge Interface (DCI) is used to configure the security features of the Series 2 devices, whereas the Serial Wire Debug (SWD) interface is used to program the flash memory of Series 2 devices. A general overview of the DCI and SWD programming steps is described in the following sections.

3. Debug Challenge Interface (DCI)

Interaction with the SE is performed over a command interface that is available through a dedicated Debug Challenge Interface (DCI).

The DCI is intended to be used for various SE commands. The DCI is open while the SE is running.

3.1 DCI Connection

The DCI is made available through a connection on the Serial Wire Debug (SWD) port. The steps involved in connecting the DCI through the SWD port are as follows.

1. Send the JTAG-to-SWD switching sequence.
2. Read the IDCODE register (SWD DP register 0) to retrieve a identification value. On the Series 2 devices with a Cortex-M33 core, this value should be `0x6BA02477`.
3. Use the ABORT register (SWD DP register 0) to clear the error and sticky flag conditions.
4. Use the STAT register (SWD DP register 1) to generate a system and debug domain power-up request.
5. Use the SELECT register (SWD DP register 2) to set the SWD interface in the chip to communicate with the DCI.
6. Use the CSW register (SWD AP register 0) to set transfer size to 32-bit.

See AN0062: Programming Internal Flash over the Serial Wire Debug Interface for more information about the SWD port registers.

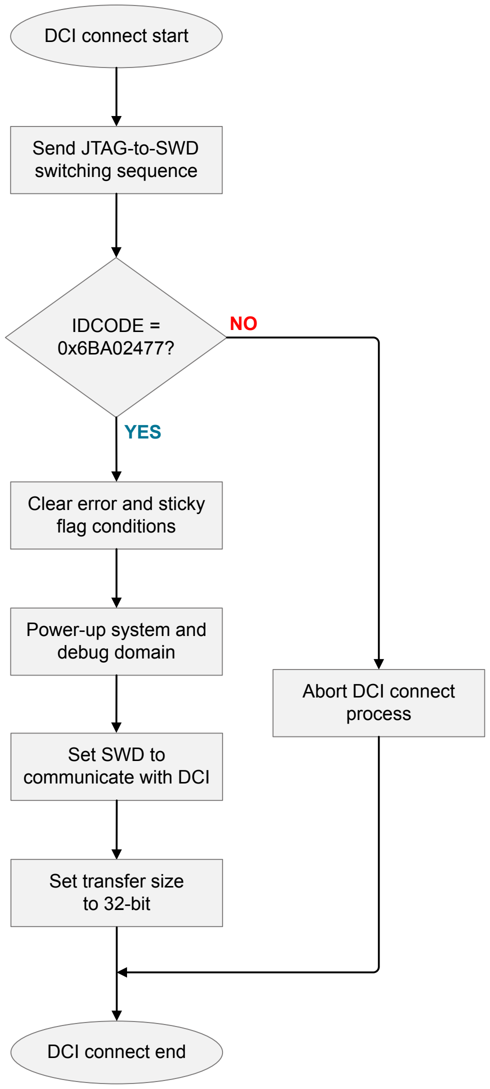

DCI connect start
Send JTAG-to-SWD switching sequence
IDCODE = 0x6BA02477?
NO
YES
Clear error and sticky flag conditions
Power-up system and debug domain
Set SWD to communicate with DCI
Set transfer size to 32-bit
Abort DCI connect process
DCI connect end

Figure 3.1. DCI Connection Flowchart

3.2 DCI Registers

For interacting with the DCI, there are three registers in Table 3.1 DCI Register on page 5 that can be accessed through SWD.

Table 3.1. DCI Register

Register	Description	Fields	Address	Remarks
DCI_WDATA	Write data to the DCI	WDATA [31:0]	0x1000	Command
DCI_RDATA	Read data from the DCI	RDATA [31:0]	0x1004	Response
DCI_STATUS	Status of DCI accesses	Bit 0 = WPENDING	0x1008	Write Request to the DCI is pending. Additional writes to DCI_WDATA are discarded when this bit is asserted.
"	"	Bit 8 = RDATAVALID	"	Response from the DCI is valid when this bit is asserted. Cleared on a read of DCI_RDATA.

3.3 DCI Calls

3.3.1 Command

All DCI calls start by writing a 32-bit word containing the length of the DCI data followed by the 32-bit Command ID into the DCI. The length includes the length word itself so the minimum value is 8 (4 bytes of length and 4 byes of Command ID). Then a variable length command payload (if applicable) is transferred into the DCI.

Table 3.2. DCI Command

Command Word	Description
Word 0	Length of packet in bytes (including word 0)
Word 1	Command ID
Words 2 – N	Command payload if applicable

3.3.2 Response

On completion, a 32-bit word consisting of the response length [15:0] and response code [31:16] is read from the DCI followed by the response payload, if present.

Table 3.3. DCI Response

Response Word	Description
Word 0	Total response length (including word 0) and response code: [15:0] – Total length; [31:16] – Response code
Words 1 – N	Response payload if applicable

All executed commands return a response code that classifies the result of the operation. The basic meaning of these response codes is given in the following table.

Table 3.4. DCI Response Codes

Response Code	Status	Description
0	SE_RESPONSE_OK	Command executed successfully or signature was successfully validated.
1	SE_RESPONSE_INVALID_COMMAND	Command was not recognized as a valid command, or is not allowed in the current context.
2	SE_RESPONSE_AUTHORIZATION_ERROR	User did not provide the required credentials to be allowed to execute the command.
3	SE_RESPONSE_INVALID_SIGNATURE	Signature validation command failed to verify the given signature as being correct.
4	SE_RESPONSE_BUS_ERROR	A command started in non-secure mode is trying to access secure memory.
5	SE_RESPONSE_INTERNAL_ERROR	Internal SE error.
6	SE_RESPONSE_CRYPTO_ERROR	Error in crypto operation.
7	SE_RESPONSE_INVALID_PARAMETER	One of the passed parameters is deemed invalid (for example, out of bounds), or the number of parameters is incorrect.
8	SE_RESPONSE_INTEGRITY_ERROR	Operation cannot be completed due to the SE having an invalid internal state.
9	SE_RESPONSE_SECUREBOOT_ERROR	The host application failed secure boot check.
10	SE_RESPONSE_SELFTEST_ERROR	Failure during self-test.
11	SE_RESPONSE_NOT_INITIALIZED	Feature or item is not present or not initialized.

3.4 DCI Operation

3.4.1 DCI Write

The user can write the DCI_WDATA register to pass command to the SE. The steps involved in writing a command to DCI are as follows.

1. Connect to DCI according to Figure 3.1 DCI Connection Flowchart on page 4
2. For each word in the command, follow these steps in sequence:
    a. Set the DCI to read from DCI_STATUS by setting SWD AP register 1 to 0x1008.
    b. Read DCI_STATUS by reading from SWD AP register 3.
    c. If WPENDING is high, go back to step (a). If WPENDING is low, continue. If RDATAVALID is high, then the SE has started issuing a reply and the current command needs to be aborted.
    d. Set the DCI to write to DCI_WDATA by setting SWD AP register 1 to 0x1000.
    e. Write the command word to SWD AP register 3.

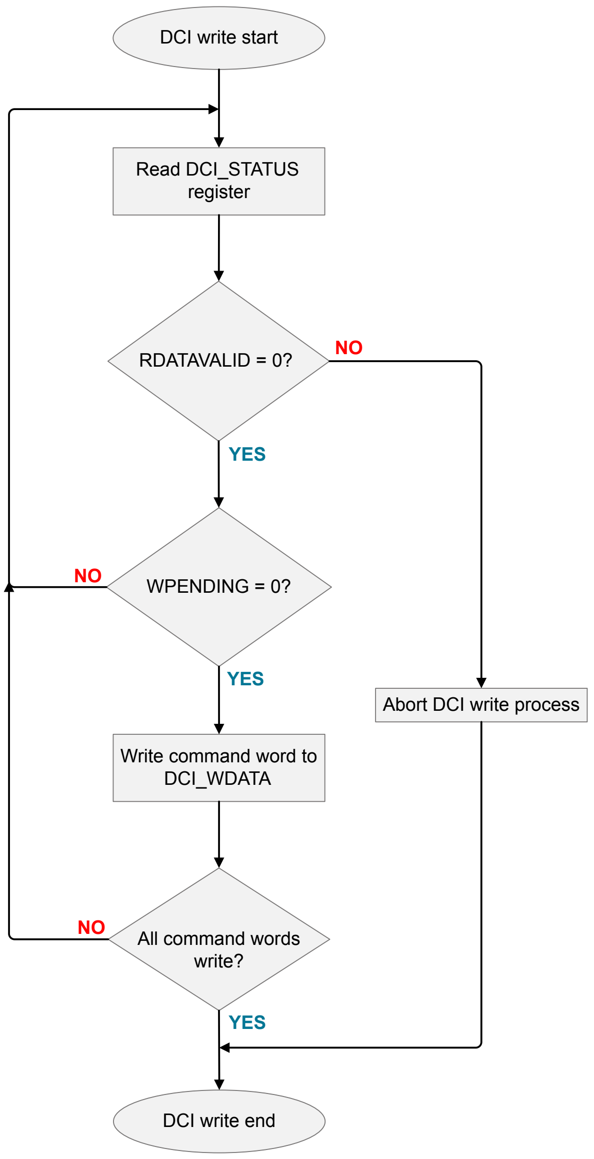


Figure 3.2. DCI Write Flowchart

3.4.2 DCI Read

The user can read the DCI_RDATA register to retrieve responses from the SE to a previously written command. If a user has sent a command according to Figure 3.2 DCI Write Flowchart on page 7, the steps to read the response from the DCI are as follows.

1. Set the DCI to read from DCI_STATUS by setting SWD AP register 1 to 0x1008.
2. Read DCI_STATUS by reading from SWD AP register 3.
3. If RDATAVALID is high, a response word is available to be read from DCI_RDATA. If RDATAVALID is low, go back to step (2) because the SE has not begun to reply.
4. Set the DCI to read from DCI_RDATA by setting SWD AP register 1 to 0x1004.
5. Read the first response word from the command to SWD AP register 3.
6. Total length of the response (including the first word) is in the lower 16 bits of that word. If this is larger than 4, for each response word:
    a. Set the DCI to read from DCI_STATUS by setting SWD AP register 1 to 0x1008.
    b. Read DCI_STATUS by reading from SWD AP register 3.
    c. If RDATAVALID is high, a response word is available to be read from DCI_RDATA. If RDATAVALID is low, go back to step (a) because the process is polling faster than the SE can send data.
    d. Set the DCI to read from DCI_RDATA by setting SWD AP register 1 to 0x1004.
    e. Read the sequential response word to SWD AP register 3.

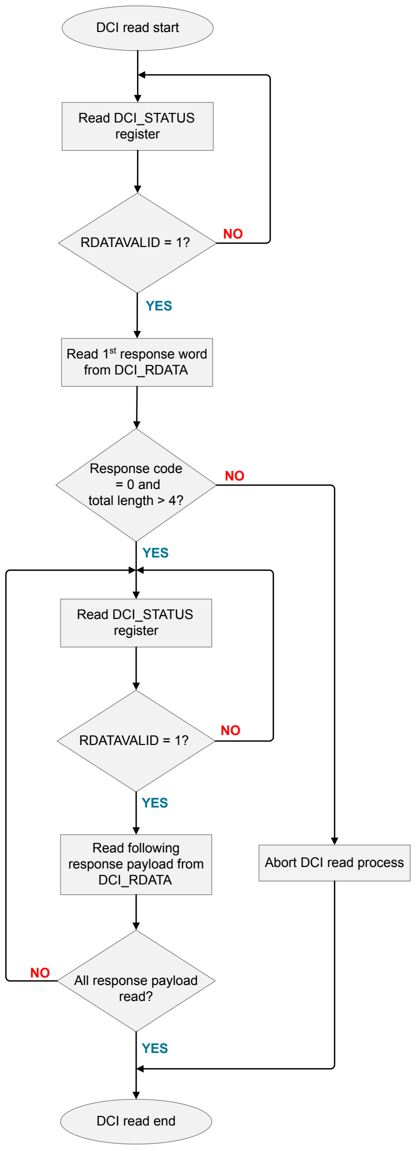

DCI read start
Read DCI_STATUS register
RDATAVALID = 1?
NO
YES
Read 1st response word from DCI_RDATA
Response code = 0 and total length > 4?
NO
YES
Read DCI_STATUS register
RDATAVALID = 1?
NO
YES
Read following response payload from DCI_RDATA
Abort DCI read process
NO
All response payload read?
YES
DCI read end

Figure 3.3. DCI Read Flowchart

4. Serial Wire Debug (SWD) Interface

The flash memory on Series 2 devices is divided into two blocks: the main block and the information block. Program code is normally written to the main block. The information block is available for special user data. Throughout this document, the flash or main flash is referred to as the main block and user data is referred to as the information block.

Program the Series 2 flash memory by writing directly to device's Memory System Controller (MSC) registers over the Serial Wire Debug (SWD) interface. This method is simple and easy to upgrade to support on new Series 2 devices.

For EFR32xG22 devices, the MSC bit in the CMU CLKEN1 register must be set to enable the clock source for the Memory System Controller (MSC).

Series 2 devices with a Cortex-M33 core should return the value `0x84770001` when reading the AP Identification Register (IDR) through SWD.

See AN0062: Programming Internal Flash over the Serial Wire Debug Interface for more information on how to access the SWD interface of the target device and how to use this interface to program devices.

4.1 Flash Erase

There are two ways to erase the flash of the target device through SWD interface:

Page Erase

- A page erase can be initiated from software using the ERASEPAGE bit in the MSC WRITECMD register. The page erase operations require that the address of main flash or user data is written into the MSC ADDRB register. To reduce the time needed for the flash erase process, the program should avoid erasing the target main flash page by page, especially for devices with larger flash memories.

Mass Erase

- A mass erase can be initiated from software using the ERASEMAIN0 bit in the MSC WRITECMD register. This erases the entire flash (excluding the user data).

Table 4.1. Flash Organization and Erase Timing

Device	Flash Size (kB)	Flash Page Size (kB)	User Data Size (kB)	Page Erase Time	Mass Erase Time
EFR32xG21A/B	1024 (maximum)	8	1	11.6 - 13.9 ms	11.7 - 14.1 ms
EFR32xG22	512 (maximum)	8	1	11.4 - 14.4 ms	11.7 - 14.3 ms

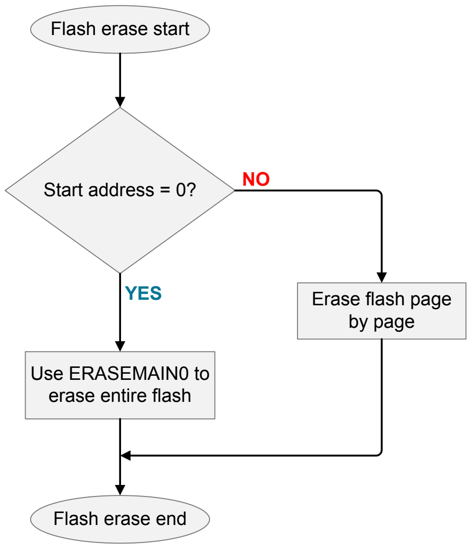


Figure 4.1. Flash Erase Flowchart

4.2 Flash Write

The write operation requires that the address be written into the MSC ADDRB register. After each 32-bit word is written, the internal address register is incremented automatically by 4. When a word is written to the MSC WDATA register, the WDATAREADY bit of the MSC STATUS register is cleared. When this status bit is set, software can write the next word.

- Flash program time (32-bit word) of EFR32xG21A and EFR32xG21B: 40.2 - 47.9 µs
- Flash program time (32-bit word) of EFR32xG22: 42.1 - 45.6 µs

To reduce the time for the flash write process, the program can omit polling the WDATAREADY bit in the MSC STATUS register after writing each 32-bit word, because the register read process is time consuming. The alternative is to add a fixed microseconds delay between each write to make sure the maximum write time can be met.

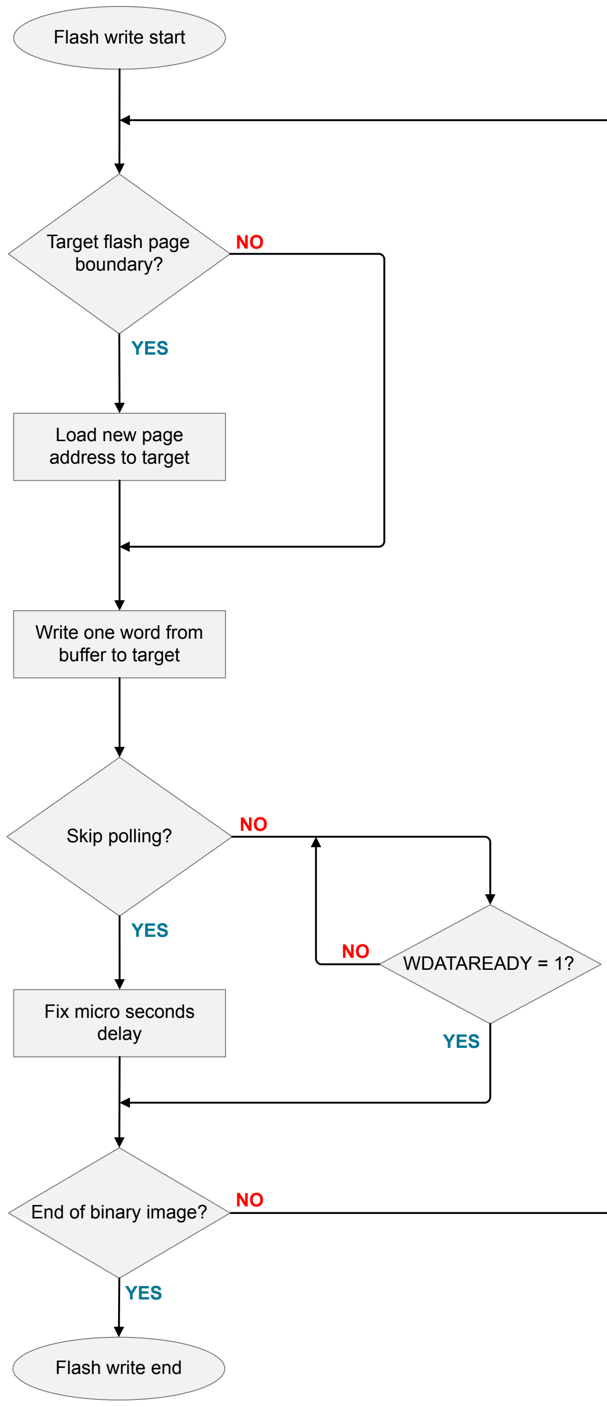


Figure 4.2. Flash Write Flowchart

4.3 Flash Verify

The program verifies the target flash contents with a buffer to make sure that no errors occurred during the programming process. The autoincrement of the Transfer Address Register (TAR) is for burst reads within the TAR wraparound boundary. The TAR must be initialized at every TAR wraparound boundary to set up the next flash read address.

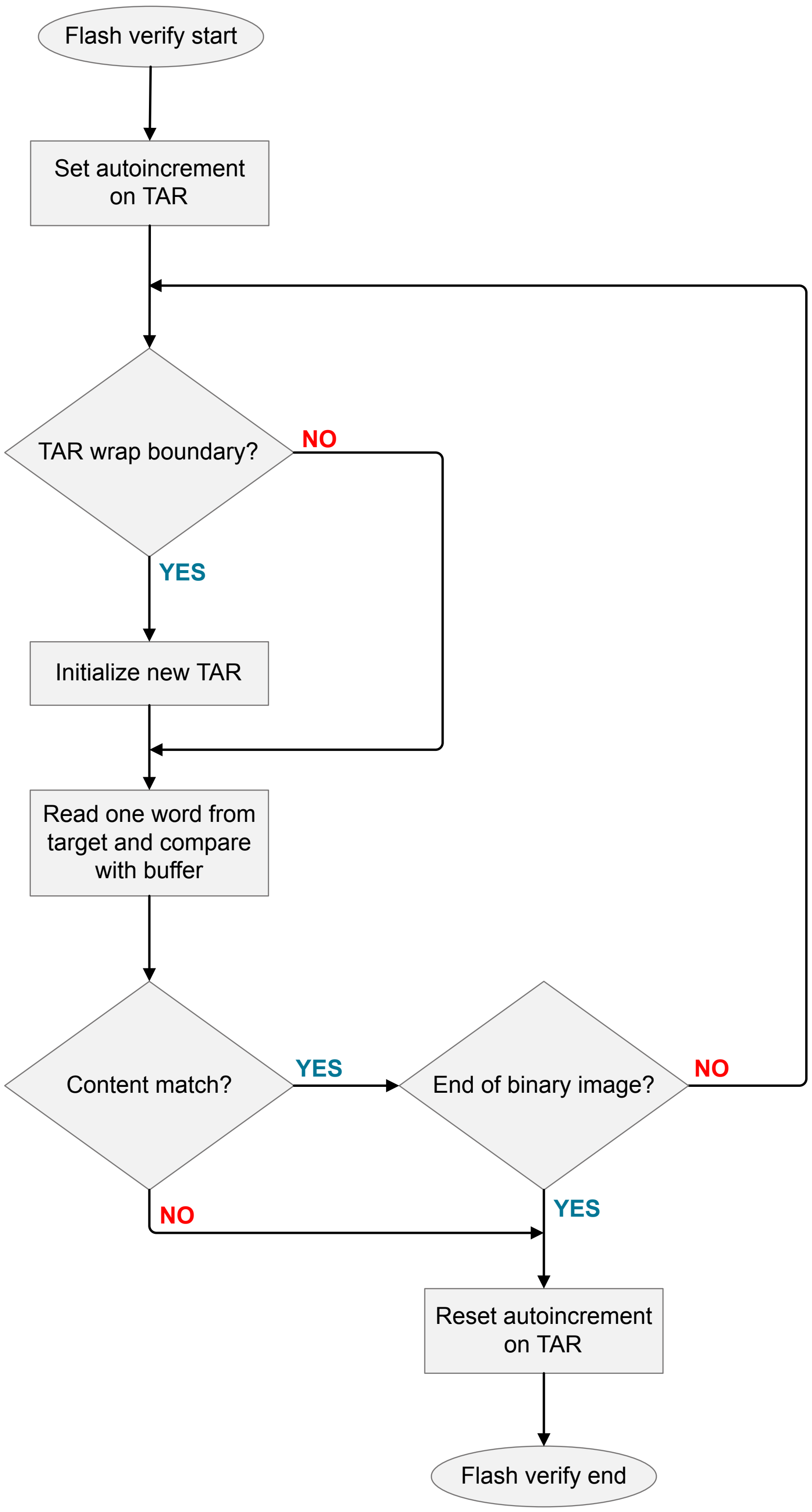


Figure 4.3. Flash Verify Flowchart

5. SE Command List

The SE commands can be issued over the DCI. The following sections contain information about each command's operation and arguments. The command payload and/or response payload on some commands are device-specific.

For more information about secure boot, see AN1218: Series 2 Secure Boot with RTSL. For more information about debug lock and secure debug, see AN1190: Series 2 Secure Debug.

5.1 SE Image Check

This command can be used to check the SE image before starting the upgrade process, in order to be able to abort early if the image is invalid or inapplicable.

Note: This command is only available on SE firmware version ≥ v1.2.2.

Table 5.1. SE Image Check Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0x4302	0x00	0x00	Address in internal flash where the SE upgrade image is stored - 4 bytes:	None

5.2 SE Image Apply

This command can be used to perform an upgrade of the SE firmware where the existing firmware will be overwritten with the one stored in the internal flash, if the upgrade image is valid and applicable. The system is restarted and no response code is returned if the SE image is successfully upgraded.

Note: This command is only available on SE firmware version ≥ v1.2.2.

Table 5.2. SE Image Apply Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0x4303	0x00	0x00	Address in internal flash where the SE upgrade image is stored - 4 bytes:	None

5.3 Apply Lock

This command enables the debug lock for the part.

Table 5.3. Apply Lock Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0x430C	0x00	0x00	None	None

5.4 Enable Secure Debug

This command enables the secure debug functionality. This command must be used before the debug port is locked and will fail if executed after locking debug access.

Table 5.4. Enable Secure Debug Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0x430D	0x00	0x00	None	None

5.5 Disable Secure Debug

This command disables the secure debug functionality, and is available even after the debug port has been locked.

Table 5.5. Disable Secure Debug Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0x430E	0x00	0x00	None	None

5.6 Erase Device

This command performs a device mass erase and resets the debug configuration to its initial unlocked state. It is only available if the Disable Device Erase command has not been executed.

This command clears and verifies the main flash and RAM of the system, excluding the user data and one-time programmable (OTP) commissioning information in the SE.

Table 5.6. Erase Device Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0x430F	0x00	0x00	None	None

5.7 Disable Device Erase

This command disables the Erase Device command. This command does not lock the debug interface to the part, but it is a permanent action for the part. This is a one-time command.

Table 5.7. Disable Device Erase Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0x4310	0x00	0x00	None	None

5.8 Read Serial Number

This command is used to read the Silicon Labs-provisioned serial number of the device.

Table 5.8. Read Serial Number Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0xFE00	0x00	0x00	None	16 bytes serial number

5.9 Get Status

This command is used to read out the status information from the SE.

Table 5.9. Get Status Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0xFE01	0x00	0x00	None	Varies by Secure Engine type, see following.

VSE - total 20 bytes:

- Boot status - 4 bytes
- VSE firmware version - 4 bytes
- MCU firmware version - 4 bytes
- Debug lock status - 4 bytes
- Secure boot configuration - 4 bytes

HSE without Secure Vault - total 36 bytes:

- Reserved - 16 bytes
- Boot status - 4 bytes
- HSE firmware version - 4 bytes
- MCU firmware version - 4 bytes
- Debug lock status - 4 bytes
- Secure boot configuration - 4 bytes

HSE with Secure Vault - total 36 bytes:

- Tamper status - 4 bytes
- Tamper time stamp - 4 bytes
- Tamper raw status - 4 bytes
- Time stamp - 4 bytes
- Boot status - 4 bytes
- HSE firmware version - 4 bytes
- MCU firmware version - 4 bytes
- Debug lock status - 4 bytes
- Secure boot configuration - 4 bytes

Note:

- Tamper status is a set of 32 flags that indicate which tamper events have occurred.
- Tamper time stamp is a HSE timer counter value for the last tamper event.
- Tamper raw status is encoded the same as tamper status but is an immediate value of the tamper event sources.
- The time stamp is a HSE timer counter value.
- Boot status:
  - Bit [7:0] - 0x20 if boot is successful.
  - Bit [15:8] - The response code if SE firmware version ≥ v1.2.0
- VSE or HSE firmware version: Bit [31:0]
- MCU firmware version:
  - Bit [31:0] - The MCU firmware version is not available if all set to 1 (0xFFFFFFFF)
- Debug lock status:
  - Bit [0] - Debug lock (configuration status) is enabled if set.
  - Bit [1] - Device erase is enabled if set.
  - Bit [2] - Secure debug is enabled if set.
  - Bit [5] - Debug lock (hardware status) is enabled if set.
- Secure boot configuration:
  - Bit [31:0] - SE OTP is not yet configured if all set to 1 (0xFFFFFFFF)
  - Bit [31:0] - SE OTP has been configured if Bit [31:1] are 0, secure boot is enabled if Bit [0] is set

5.10 Read User Configuration

This command is used to read non-reconfigurable user settings on the SE OTP for secure boot and tamper response.

Note: This command is only available on SE firmware versions ≥ v1.2.2.

Table 5.10. Read User Configuration Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0xFE04	0x00	0x00	None	Varies by Secure Engine type, see following.

VSE - total 4 bytes:

- MCU flags - 4 bytes

HSE without Secure Vault - total 24 bytes:

- MCU flags - 4 bytes
- Reserved - 20 bytes

HSE with Secure Vault - total 24 bytes:

- MCU flags - 4 bytes
- Tamper response levels (2 signals per byte) - 16 bytes
- Filter reset period - 1 byte
- Filter trigger threshold - 1 byte
- Tamper flags - 1 byte
- Tamper reset threshold - 1 byte

5.11 Initialize OTP

This command is used during factory initialization, to upload device-specific settings to the SE OTP. This is a one-time command.

Table 5.11. Initialize OTP Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0xFF00	0x00	0x01	Varies by Secure Engine type, see following.	None

VSE - total 12 bytes:

1. Parity (equal to item 3) - 4 bytes
2. Length of the following content - 4 bytes
3. MCU flags - 4 bytes

HSE without Secure Vault - total 32 bytes:

1. Parity (the XOR of 32-bit words from item 3 and 4) - 4 bytes
2. Length of the following content - 4 bytes
3. MCU flags - 4 bytes
4. Reserved - 20 bytes

HSE with Secure Vault - total 32 bytes:

1. Parity (the XOR of 32-bit words from item 3 to 8) - 4 bytes
2. Length of the following content - 4 bytes
3. MCU flags - 4 bytes
4. Tamper response levels (2 signals per byte) - 16 bytes
5. Filter reset period - 1 byte
6. Filter trigger threshold - 1 byte
7. Tamper flags - 1 byte
8. Tamper reset threshold - 1 byte

5.11.1 MCU Flags

The parameters of the MCU flags are described in the following table. For more information about these flags, see section "*Secure Boot Enabling*" in AN1222: Production Programming of Series 2 Devices.

Table 5.12. Parameters of MCU Flags

Fields	Description
Bit [15:0]	Reserved
Bit [16]	Secure boot enable
Bit [17]	Secure boot verify certificate
Bit [18]	Secure boot anti-rollback
Bit [19]	Secure boot page lock narrow
Bit [20]	Secure boot page lock full
Bit [31:21]	Reserved

5.11.2 Anti-Tamper Configuration

The 16 bytes of tamper response levels are described in the following table.

Table 5.13. Tamper Source Response Level

Tamper Response Level																																
Byte [Bit]	15 [7:4]	15 [3:0]	14 [7:4]	14 [3:0]	13 [7:4]	13 [3:0]	12 [7:4]	12 [3:0]	11 [7:4]	11 [3:0]	10 [7:4]	10 [3:0]	9 [7:4]	9 [3:0]	8 [7:4]	8 [3:0]	7 [7:4]	7 [3:0]	6 [7:4]	6 [3:0]	5 [7:4]	5 [3:0]	4 [7:4]	4 [3:0]	3 [7:4]	3 [3:0]	2 [7:4]	2 [3:0]	1 [7:4]	1 [3:0]	0 [7:4]	0 [3:0]
Tamper source	SE ICACHE	Digital glitch	SE debug	Secure lock	Voltage glitch rising	Voltage glitch falling	Temperature sensor	Decouple BOD	PRS7 only	PRS6	PRS5	PRS4	PRS3	PRS2	PRS1	PRS0	TRNG monitor	Self test	Reserved	Flash integrity	DCI authorization	Mailbox authorization	User secure boot	Reserved	SE software assertion	Reserved	SE hard fault	SE RAM CRC	Reserved	SE watchdog	Filter counter	Reserved
Level	7 only	1/2/4/7	1/2/4/7	7 only	1/2/4/7	1/2/4/7	1/2/4/7	7 only	1/2/4/7	1/2/4/7	1/2/4/7	1/2/4/7	1/2/4/7	1/2/4/7	1/2/4/7	1/2/4/7	1/2/4/7	7 only	—	7 only	1/2/4/7	1/2/4/7	1/2/4/7	—	7 only	—	7 only	7 only	—	7 only	1/2/4/7	—

Other anti-tamper settings are described in the following table.

Table 5.14. Anti-Tamper Configuration Settings

Setting	Value
Filter reset period	0 to 31
Filter trigger threshold	0 to 7
Tamper flags	Bit [1] - Digital glitch detector is always on if set
Tamper reset threshold	0 to 255

For more information about anti-tamper configuration, see section "*Anti-Tamper Configuration*" in AN1247: Anti-Tamper Protection Configuration and Use.

5.12 Initialize Public Key

This command is used to initialize the user public key(s) to the SE OTP. This is a one-time command.

Table 5.15. Initialize Public Key Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0xFF07	Key type: 0x01 - Public Sign Key; 0x02 - Public Command Key	0x01	Total 68 bytes: 4-byte Parity (the XOR of 32-bit words of option 2) plus 64-byte Public key in option 1	None

5.13 Read Public Key

This command can be used to read out one of the public keys that are permanently stored in SE OTP.

Table 5.16. Read Public Key Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0xFF08	Key type: 0x01 - Public Sign Key; 0x02 - Public Command Key	0x01	None	64 bytes: public key in option 1

5.14 Initialize AES Key

This command is used to initialize a 128-bit symmetric key to the SE OTP. This is a one-time command.

Note: This command is only available on HSE devices.

Table 5.17. Initialize AES Key Command

ID [31:16]	Option 1 [15:8]	Option 2 [7:0]	Command payload	Response payload
0xFF0B	Key type: 0x05 - AES-128 key	0x01	Total 20 bytes: 4-byte Parity (the XOR of 32-bit words of option 2) plus16-byte Symmetric key in option 1	None

6. DCI and SWD Programmer Examples

For more information about production programming steps, see section "Overview" in AN1222: Production Programming of Series 2 Devices.

6.1 Hardware Overview

A Wireless Starter Kit (WSTK) with the BRD4182A Radio Board (EFR32MG22C224F512IM40) is used as the hardware platform of the DCI and SWD programmer.

The programmer uses GPIO to emulate the Serial Wire Debug (SWD) interface to program the target device. The user interface is handled by push buttons and UART interface. The VCOM_TX is routed to the WSTK virtual COM port by setting the VCOM_ENABLE line high.

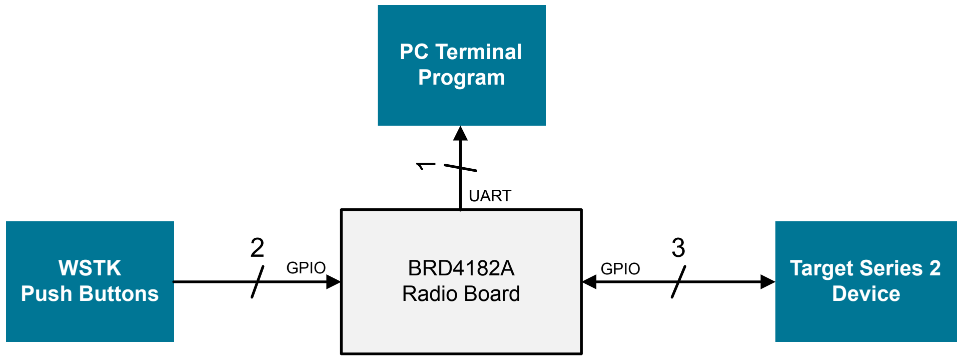


Figure 6.1. Block Diagram of DCI and SWD Programmer

Table 6.1. Resources of BRD4182A Used by Programmer

GPIO (SoC Peripheral)	WSTK Peripheral	Function	EXP Header Connection
PA01 (DBG_SWCLK)	DBG_TCK_SWCLK	Programmer debug SWCLK	—
PA02 (DBG_SWDIO)	DBG_TCK_SWDIO	Programmer debug SWDIO	—
PA05 (US1_TX)	VCOM_TX	WSTK Virtual COM port TX	Pin 12
PB00	UIF_BUTTON0	WSTK Push button 0 (PB0)	Pin 7
PB01	UIF_BUTTON1	WSTK Push button 1 (PB1)	Pin 9
PB04	VCOM_ENABLE	WSTK Virtual COM port enable	—
PC03	—	Target device RESET	Pin 10
PD02	UIF_LED0	Target device SWCLK (shared with WSTK LED0)	Pin 11
PD03	UIF_LED1	Target device SWDIO (shared with WSTK LED1)	Pin 13

The interconnection diagram used with the programmer is shown in Figure 6.2 Programmer Connection Diagram on page 19. The programmer is the WSTK with the BRD4182A Radio Board and the target device is the WSTK with the Series 2 radio board.

The target Series 2 radio board can be:

- EFR32xG21A — BRD4180A, BRD4180B, BRD4181A, and BRD4181B
- EFR32xG21B — BRD4181C
- EFR32xG22 — BRD4182A and BRD4183A

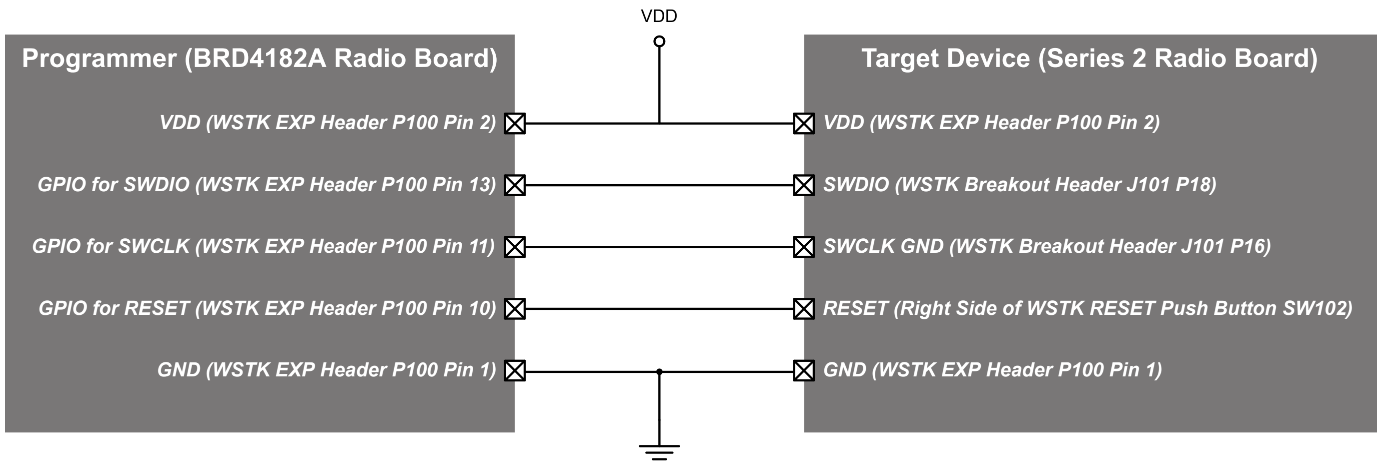

VDD
Programmer (BRD4182A Radio Board)
VDD (WSTK EXP Header P100 Pin 2)
GPIO for SWDIO (WSTK EXP Header P100 Pin 13)
GPIO for SWCLK (WSTK EXP Header P100 Pin 11)
GPIO for RESET (WSTK EXP Header P100 Pin 10)
GND (WSTK EXP Header P100 Pin 1)
Target Device (Series 2 Radio Board)
VDD (WSTK EXP Header P100 Pin 2)
SWDIO (WSTK Breakout Header J101 P18)
SWCLK GND (WSTK Breakout Header J101 P16)
RESET (Right Side of WSTK RESET Push Button SW102)
GND (WSTK EXP Header P100 Pin 1)

Figure 6.2. Programmer Connection Diagram

The target device on the Series 2 radio board is powered by the programmer, therefore the target device WSTK's power switch SW700 should be in the **BAT** position.

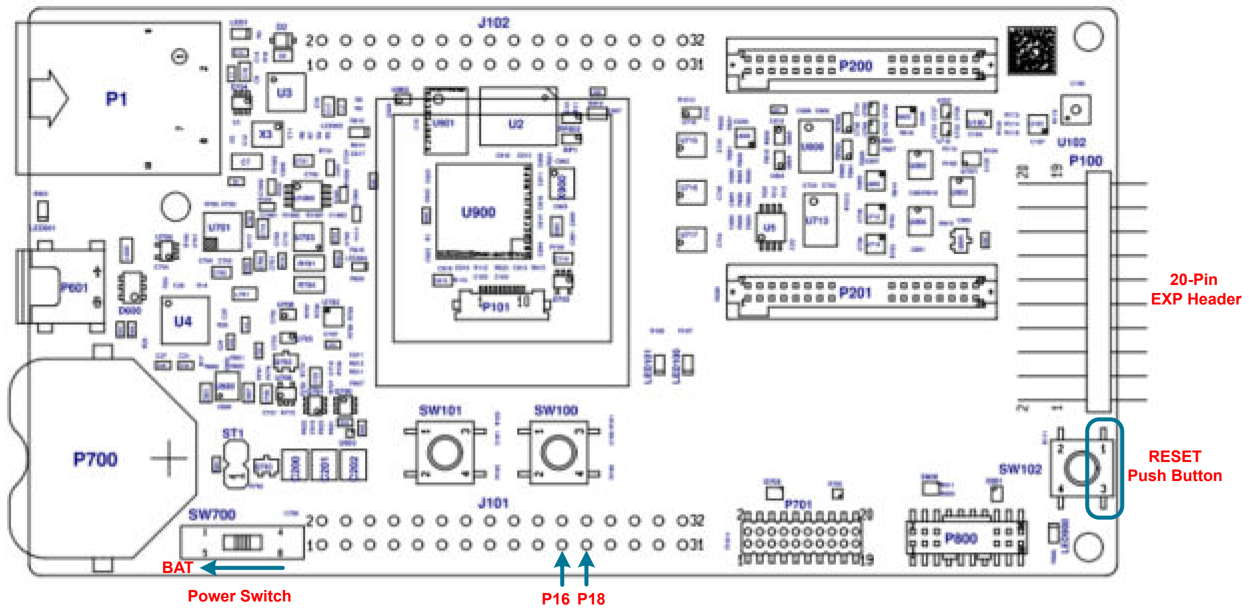
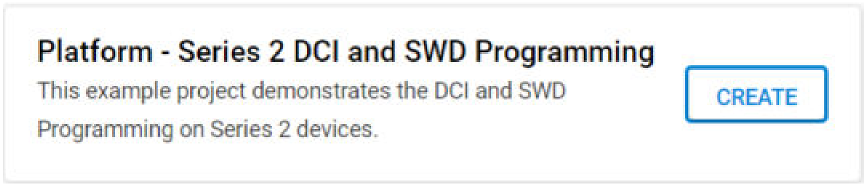


Figure 6.3. WSTK Assembly Drawing

6.2 Software Overview

The Gecko SDK Suite 3.1.2 or later in Simplicity Studio 5 includes the `Series 2 DCI and SWD Programming` platform example.


Platform - Series 2 DCI and SWD Programming

This example project demonstrates the DCI and SWD Programming on Series 2 devices.

Refer to the `readme.html` file in this example for procedures to create the project and run the example.

The location of the `readme.html` file in Windows is `C:\SiliconLabs\SimplicityStudio\v5\developer\sdks\gecko_sdk_suite\v3.1\app\common\example\dci_swd_programming`.

6.2.1 Bit-Bang

The major overhead for the method of writing directly to MSC registers is to emulate the SWDIO and SWCLK signals by bit-banging GPIO pins. In order to speed up this process, the GPIOs of SWCLK and SWDIO for the target device must be on the same GPIO port group (0-7 or 8-15). The target RESET line and other signals should not be connected to this port, because the software writes to the entire port at once when bit-banging the SWCLK and SWDIO signals.

6.2.2 Security Keys

The security keys for programmer examples are hard-coded in `app_dci_task.c` and described in the following table.

Table 6.2. Hard-coded Security Keys

Security Key Array	Usage	Source (1)
aes_key[]	Decrypt GBL payloads	A 16 bytes AES-128 key in encrypt-unsafe-key.prv (binary file).
public_sign_key[]	Secure boot	A 64 bytes Public Sign Key, the corresponding Private Sign Key in root-sign-unsafe-privkey.pem.
public_command_key[]	Secure debug unlock and Disable tamper	A 64 bytes Public Command Key, the corresponding Private Command Key in cmd-unsafe-privkey.pem.

Note:

(1) The location in Windows is `C:\SiliconLabs\SimplicityStudio\v5\developer\adapter_packs\secmgr\scripts\offline`

The public key can be derived from private key by using the OpenSSL.

```
openssl ec -in rootsign-unsafe-privkey.pem -pubout -text > public_sign_key.txt
```

```
openssl ec -in cmd-unsafe-privkey.pem -pubout -text > public_command_key.txt
```

6.2.3 OTP Settings

The OTP settings for programmer examples are hard-coded in `app_dci_task.c` and described in the following table.

Table 6.3. Hard-coded OTP Settings

OTP Setting Array	Usage	Value
xg22_conf[]	OTP settings for EFR32xG22	MCU flags: Secure boot enable and Secure boot anti-rollback
xg21a_conf[]	OTP settings for EFR32xG21A	MCU flags: Secure boot enable and Secure boot anti-rollback
xg21b_conf[]	OTP settings for EFR32xG21B	MCU flags: Secure boot enable and Secure boot anti-rollback
"	"	Tamper source levels: 0x40440410, 0x14040100, 0x77442211, 0x42042224
"	"	Filter reset period: 10
"	"	Filter trigger threshold: 6
"	"	Tamper flags: 0
"	"	Tamper reset threshold: 5

6.2.4 Firmware images

The firmware images for programmer examples are hard-coded in `app_firmware_image.c` and described in Table 6.4 Hard-coded Firmware Images on page 22.

The SE upgrade firmware image must be stored to the device internal flash in `.seu` format. The latest SE firmware image (`.sec` and `.hex`) can be found in the Windows folders below (`<version>` should be `v3.1.0` or above).

```
C:\SiliconLabs\SimplicityStudio\v5\developer\sdks\gecko_sdk_suite\<version>\util\se_release\public
```

The firmware image (`.bin` or `.sec`) can be converted to a C source file using the SEGGER free utility `Bin2C.exe` (https://www.segger.com/free-utilities/bin2c/).

Note: The last `NULL` (0x00) character in the converted firmware image array should be discarded.

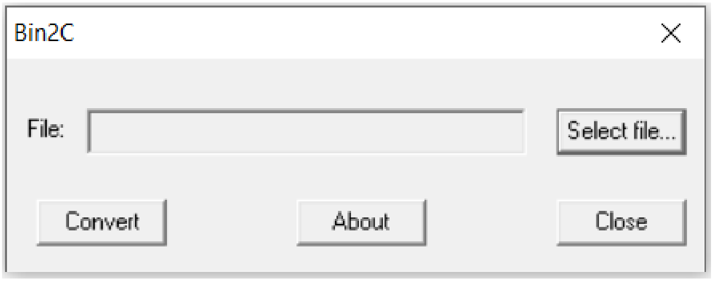


Figure 6.4. Bin2C Utility

Table 6.4. Hard-coded Firmware Images

Firmware Image Array	Size (Bytes)	Usage
xg21_hse_image[]	41125	EFR32xG21 HSE upgrade firmware image (v1.2.6)
xg22_vse_image[]	16549	EFR32xG22 VSE upgrade firmware image (v1.2.6)
xg21_app_image[]	8968	EFR32xG21 application firmware image to redirect `printf()` output on WSTK LCD
xg22_app_image[]	9744	EFR32xG22 application firmware image to redirect printf() output on WSTK LCD
xg21_signed_image[] (1)	11312	A signed EFR32xG21 UART XMODEM Bootloader image (v1.11.0)
xg22_signed_image[] (1)	14260	A signed EFR32xG22 UART XMODEM Bootloader image (v1.11.0)
erase_xg21_userdata[]	3068	Application firmware image to erase EFR32xG21 user data
write_xg21_userdata[]	4228	Application firmware image to program EFR32xG21 user data
prog_xg21_hse_upgrade[]	3236	Application firmware image to upgrade EFR32xG21 HSE firmware
prog_xg22_vse_upgrade[]	3844	Application firmware image to upgrade EFR32xG22 VSE firmware

Note:

(1) The image is signed by the Private Sign Key (rootsign-unsafe-privkey.pem) and this signed image is used to recover a secure boot failure device.

6.2.5 Compile Options

The programmer examples have corresponding header files to set up the software and hardware environment. The compile options for the three header files are shown in the following tables.

Table 6.5. app_dci_swd.h Compile Options

Parameter	Usage	Default Setting
RESET_PULSE	Pin reset pulse width in microseconds	1000 μs (1 ms)
RESET_DELAY	Delay in microseconds after issuing a soft or pin reset	50000 μs (50 ms)
DCI_RETRY_COUNT	Number of times to retry a DCI read or write operation. It must be high enough to receive a response from the SE command.	1001000
SWCLK_PORT	GPIO port for SWCLK	3 (Port D)
SWCLK_PIN	GPIO pin for SWCLK	2 (PD02)
SWDIO_PORT	GPIO port for SWDIO	3 (Port D)
SWDIO_PIN	GPIO pin for SWDIO	3 (PD03)
RESET_PORT	GPIO port for RESET	2 (Port C)
RESET_PIN	GPIO pin for RESET	3 (PC03)

Table 6.6. app_firmware_image.h Compile Options

Parameter	Usage	Default Setting
APP_START_ADDR	Application firmware image start address (aligned with flash page size)	0x00000000
SE_START_ADDR	SE upgrade firmware image start address (aligned with flash page size)	0x00040000
SE_UPGRADE_DELAY	Delay in microseconds after issuing a command to upgrade the SE firmware	2500000 μs (2.5 s)
USER_DATA_DELAY	Delay in microseconds after issuing a soft reset to run the application to erase or write the user data on HSE devices	1000000 μs (1 s)

Table 6.7. app_swd_task.h Compile Options

Parameter	Usage	Default Setting
ERASE_DELAY	Delay in micro seconds after a flash page erase or mass erase	12000 μs (12 ms)
SKIP_POLLING	Skip polling WDATAREADY bit in the MSC STATUS register after writing 32-bit word to target device flash	1 (Skip)
WRITE_DELAY	Fix delay in microseconds after writing a 32-bit word to target device flash (if SKIP_POLLING = 1)	11 μs

6.3 Menu Operation

The push buttons PB0 and PB1 on the Wireless Starter Kit (WSTK) are used to manipulate the menu of the programmer. The programmer redirects standard I/O to the virtual serial port (VCOM) of the WSTK. Open a terminal program (for example, Tera Term) and access the WSTK VCOM port (default setting is 115200 bps 8-N-1).

Interface Menu

```
Series 2 DCI and SWD Programmer Examples - Core running at 80000 kHz.
  . Current interface selection is DEBUG CHALLENGE INTERFACE (DCI).
  + Press PB0 to select an interface (DCI/SWD), press PB1 to select the task of the selected interface.
```

DCI Menu

```
  . Current DCI task is GET SE STATUS.
  + Press PB0 to select a DCI task, press PB1 to run the selected DCI task.
  + Current DCI task is READ SE OTP CONFIGURATION.
  + Current DCI task is READ SERIAL NUMBER.
  + Current DCI task is READ PUBLIC SIGN KEY.
  + Current DCI task is READ PUBLIC COMMAND KEY.
  + Current DCI task is ENABLE SECURE DEBUG.
  + Current DCI task is DISABLE SECURE DEBUG.
  + Current DCI task is LOCK DEVICE.
  + Current DCI task is ERASE DEVICE (UNLOCK).
  + Current DCI task is RECOVER SECURE BOOT FAILURE DEVICE.
  + Current DCI task is UPGRADE SE FIRMWARE THROUGH DCI.
  + Current DCI task is INITIALIZE AES-128 KEY (EFR32xG21).
  + Current DCI task is INITIALIZE PUBLIC SIGN KEY.
  + Current DCI task is INITIALIZE PUBLIC COMMAND KEY.
  + Current DCI task is INITIALIZE HSE OTP (EFR32xG21A).
  + Current DCI task is INITIALIZE HSE OTP (EFR32xG21B).
  + Current DCI task is INITIALIZE VSE OTP (EFR32xG22).
  + Current DCI task is DISABLE DEVICE ERASE.
```

From task `ENABLE SECURE DEBUG` to task `UPGRADE SE FIRMWARE THROUGH DCI`:

```
  + Current DCI task is ENABLE SECURE DEBUG.
  + Press PB0 to confirm or press PB1 to abort.
```

From task `INITIALIZE AES-128 KEY (EFR32xG21)` to task `DISABLE DEVICE ERASE`:

```
  + Current DCI task is DISABLE DEVICE ERASE.
  + Warning: This is a ONE-TIME command and the operation is IRREVERSIBLE!
  + Press PB0 to confirm or press PB1 to abort.
```

SWD Interface Menu

```
  . Current interface selection is DEBUG CHALLENGE INTERFACE (DCI).
  + Press PB0 to select an interface (DCI/SWD), press PB1 to select the task of the selected interface.
  + Current interface selection is SERIAL WIRE DEBUG (SWD) INTERFACE.

  . Current SWD task is ERASE MAIN FLASH.
  + Press PB0 to select a SWD task, press PB1 to run the selected SWD task.
  + Current SWD task is PROGRAM MAIN FLASH.
  + Current SWD task is ERASE USER DATA.
  + Current SWD task is PROGRAM USER DATA.
  + Current SWD task is UPGRADE SE FIRMWARE THROUGH APPLICATION FIRMWARE.
```

```
  + Current SWD task is ERASE MAIN FLASH.
  + Press PB0 to confirm or press PB1 to abort.
```

Error Handling

```
  + Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

6.4 DCI Programmer Examples

The following DCI programmer examples are based on BRD4181C as the target device (Figure 6.2 Programmer Connection Diagram on page 19). Software is compiled with -O2 optimization in Simplicity IDE of Simplicity Studio 5.

Get SE Status

- To get the current SE status.

```
. Current DCI task is GET SE STATUS.
+ Press PB0 to select a DCI task, press PB1 to run the selected DCI task.

. Read target device SE status... OK
+ SE firmware version  : 00010201
+ MCU firmware version : NA
+ Debug lock           : Disabled
+ Debug lock state     : False
+ Device Erase         : Enabled
+ Secure debug         : Disabled
+ Secure boot          : Disabled and SE OTP is not configured
+ Boot status          : Command successful.
```

Read SE OTP Configuration

- To read the current SE OTP configuration.

```
. Current DCI task is READ SE OTP CONFIGURATION.
+ Press PB0 to select a DCI task, press PB1 to run the selected DCI task.

. Read target device user (SE OTP) configuration... OK
+ Secure boot                    : Disabled
+ Secure boot verify certificate : Disabled
+ Secure boot anti-rollback      : Disabled
+ Secure boot page lock narrow   : Disabled
+ Secure boot page lock full     : Disabled
+ Tamper source level (valid on Secure Vault device)
  Filter counter          :  1
  SE watchdog             :  4
  SE RAM CRC              :  4
  SE hard fault           :  4
  SE software assertion   :  4
  User secure boot        :  0
  Mailbox authorization   :  1
  DCI authorization       :  0
  Flash integrity         :  4
  Self test               :  4
  TRNG monitor            :  1
  PRS0                    :  1
  PRS1                    :  1
  PRS2                    :  2
  PRS3                    :  2
  PRS4                    :  4
  PRS5                    :  4
  PRS6                    :  7
  PRS7                    :  7
  Decouple BOD            :  4
  Temperature sensor      :  2
  Voltage glitch falling :  2
  Voltage glitch rising   :  2
  Secure lock             :  4
  SE debug                :  0
  Digital glitch          :  2
  SE ICACHE               :  4
+ Reset period for the tamper filter counter: ~32 ms x 1024
+ Activation threshold for the tamper filter: 4
+ Digital glitch detector always on: Disabled
+ Tamper reset threshold: 5
```

- SE will return the SE_RESPONSE_INVALID_COMMAND code if the SE OTP data has not been initialized or SE firmware version is less than v1.2.2.

```
. Read target device user (SE OTP) configuration... Failed - Unsupported command.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

Read Serial Number

- To read the serial number of the Series 2 device.

```
. Current DCI task is READ SERIAL NUMBER.
+ Press PB0 to select a DCI task, press PB1 to run the selected DCI task.

. Read target device serial number... OK
+ The serial number (16 bytes): 0000000000000000014B457FFFE0F7762
```

Read Public Sign Key

- To read the Public Sign Key in SE OTP.

```
. Current DCI task is READ PUBLIC SIGN KEY.
+ Press PB0 to select a DCI task, press PB1 to run the selected DCI task.

. Read target device public sign key... OK
+ The public sign key (64 bytes): C4AF4AC69AAB9512DB50F7A26AE5B4801183D85417E729A56DA974F4E08A562C
                                  DE6019DEA9411332DC1A743372D170B436238A34597C410EA177024DE20FC819
```

- SE will return the SE_RESPONSE_INTERNAL_ERROR code if the Public Sign Key has not been provisioned.

```
. Read target device public sign key... Failed - Internal SE error.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

Read Public Command Key

- To read the Public Command Key in SE OTP.

```
. Current DCI task is READ PUBLIC COMMAND KEY.
+ Press PB0 to select a DCI task, press PB1 to run the selected DCI task.

. Read target device public command key... OK
+ The public command key (64 bytes): B1BC6F6FA56640ED522B2EE0F5B3CF7E5D48F60BE8148F0DC08440F0A4E1DCA4
                                     7C04119ED6A1BE31B7707E5F9D001A659A051003E95E1B936F05C37EA793AD63
```

- SE will return the SE_RESPONSE_INTERNAL_ERROR code if the Public Command Key has not been provisioned.

```
. Read target device public command key... Failed - Internal SE error.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

Enable Secure Debug

- To enable the secure debug functionality.

```
+ Current DCI task is ENABLE SECURE DEBUG.
+ Press PB0 to confirm or press PB1 to abort.

. Enable secure debug of target device... OK

. Read target device SE status... OK
+ SE firmware version  : 00010204
+ MCU firmware version : 010A0003
+ Debug lock           : Disabled
+ Debug lock state     : False
+ Device Erase         : Disabled
+ Secure debug         : Enabled
+ Secure boot          : Enabled
+ Boot status          : Command successful.
```

- SE will return the SE_RESPONSE_INVALID_PARAMETER code if the Public Command Key has not been provisioned.

```
. Enable secure debug of target device... Failed - Parameters are invalid or buffer is too small.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

- SE will return the SE_RESPONSE_INVALID_COMMAND code if the secure debug has already enabled.

```
. Enable secure debug of target device... Failed - Unsupported command.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

Disable Secure Debug

- To disable the secure debug functionality.

```
+ Current DCI task is DISABLE SECURE DEBUG.
+ Press PB0 to confirm or press PB1 to abort.

. Disable secure debug of target device... OK

. Read target device SE status... OK
+ SE firmware version  : 00010201
+ MCU firmware version : NA
+ Debug lock           : Disabled
+ Debug lock state     : False
+ Device Erase         : Enabled
+ Secure debug         : Disabled
+ Secure boot          : Disabled and SE OTP is not configured
+ Boot status          : Command successful.
```

Lock Device

- To enable the debug lock of the Series 2 device.

```
+ Current DCI task is LOCK DEVICE.
+ Press PB0 to confirm or press PB1 to abort.

. Lock target device... OK

. Read target device SE status... OK
+ SE firmware version  : 00010201
+ MCU firmware version : NA
+ Debug lock           : Enabled
+ Debug lock state     : True
+ Device Erase         : Enabled
+ Secure debug         : Disabled
+ Secure boot          : Disabled and SE OTP is not configured
+ Boot status          : Command successful.
```

Erase Device (Unlock)

- To perform a device mass erase and unlock the standard debug lock of the Series 2 device.

```
+ Current DCI task is ERASE DEVICE (UNLOCK).
+ Press PB0 to confirm or press PB1 to abort.

. Erase (unlock) target device... OK

. Read target device SE status... OK
+ SE firmware version  : 00010201
+ MCU firmware version : NA
+ Debug lock           : Disabled
+ Debug lock state     : False
+ Device Erase         : Enabled
+ Secure debug         : Disabled
+ Secure boot          : Disabled and SE OTP is not configured
+ Boot status          : Command successful.
```

- SE will return the SE_RESPONSE_INVALID_COMMAND code if the Device Erase is disabled.

```
. Erase (unlock) target device... Failed - Unsupported command.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

Recover Secure Boot Failure Device

- Issue a device erase to unlock the target device through DCI.

```
+ Current DCI task is RECOVER SECURE BOOT FAILURE DEVICE.
+ Press PB0 to confirm or press PB1 to abort.

. Issue a device erase through DCI.
+ Erase target device... OK
```

- A correctly-signed firmware image xg21_signed_image[] is programmed to the target device main flash (APP_START_ADDR) through the SWD interface to recover a secure boot failure device.

```
+ Initialize DP... OK - IDCODE = 0x6BA02477
+ Read AP... OK - IDR = 0x84770001
+ Set up AHB-AP and halt target... OK
+ Get device information... OK - Target device is EFR32MG21B010F1024 and unique ID is 0x14B457FFFE0F7777

. Program a correctly-signed image to recover device.
+ EFR32xG21 signed firmware image size is 11312 bytes and start address is 0x00000000.
+ Erase-Program-Verify the EFR32xG21 main flash for signed firmware image... OK (cycles: 13710930 time: 171 ms)

. Read target device SE status... OK
+ SE firmware version  : 00010204
+ MCU firmware version : 010B0000
+ Debug lock           : Disabled
+ Debug lock state     : False
+ Device Erase         : Enabled
+ Secure debug         : Disabled
+ Secure boot          : Enabled
+ Boot status          : Command successful.
```

- The device cannot be recovered if the image is not correctly-signed.

```
. Read target device SE status... OK
+ SE firmware version  : 00010204
+ MCU firmware version : NA
+ Debug lock           : Disabled
+ Debug lock state     : False
+ Device Erase         : Enabled
+ Secure debug         : Disabled
+ Secure boot          : Enabled
+ Boot status          : Failure while checking the host for secure boot.
```

Upgrade SE Firmware Through DCI

- A SE upgrade firmware image xg21_hse_image[] is programmed to the target device main flash (SE_START_ADDR) through the SWD interface.

```
+ Current DCI task is UPGRADE SE FIRMWARE THROUGH DCI.
+ Press PB0 to confirm or press PB1 to abort.

. Program a SE firmware image to target device.
+ Initialize DP... OK - IDCODE = 0x6BA02477
+ Read AP... OK - IDR = 0x84770001
+ Set up AHB-AP and halt target... OK
+ Get device information... OK - Target device is EFR32MG21B010F1024 and unique ID is 0x14B457FFFE0F7777
+ EFR32xG21 HSE firmware image version: 00010206
+ EFR32xG21 HSE firmware image size is 41125 bytes and start address is 0x00040000.
+ Erase-Program-Verify the EFR32xG21 main flash for HSE firmware image... OK (cycles: 52306285 time: 653 ms)
```

- Validate the SE upgrade firmware image and start the upgrade process if the image is valid.

```
. Validate SE firmware image in target device... OK

. Upgrade SE firmware image, delay few seconds to check SE status... Done

. Read target device SE status... OK
+ SE firmware version  : 00010206
+ MCU firmware version : NA
+ Debug lock           : Disabled
+ Debug lock state     : False
+ Device Erase         : Enabled
+ Secure debug         : Disabled
+ Secure boot          : Disabled and SE OTP is not configured
+ Boot status          : Command successful.
```

- SE will return the SE_RESPONSE_INVALID_PARAMETER code if the SE upgrade firmware image is invalid.

```
. Validate SE firmware image in target device... Failed - Parameters are invalid or buffer is too small.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

- SE will return the SE_RESPONSE_INVALID_COMMAND code if the SE firmware version is less than v1.2.2.

```
. Validate SE firmware image in target device... Failed - Unsupported command.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

Initialize AES-128 Key (EFR32xG21)

- To initialize the aes_key[] to HSE OTP.

```
+ Current DCI task is INITIALIZE AES-128 KEY (EFR32xG21).
+ Warning: This is a ONE-TIME command and the operation is IRREVERSIBLE!
+ Press PB0 to confirm or press PB1 to abort.

. Initialize AES-128 key of target device... OK
```

- SE will return the SE_RESPONSE_INVALID_PARAMETER code if the AES-128 key has already been initialized.

```
. Initialize AES-128 key of target device... Failed - Parameters are invalid or buffer is too small.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

Initialize Public Sign Key

- To initialize the public_sign_key[] to SE OTP.

```
+ Current DCI task is INITIALIZE PUBLIC SIGN KEY.
+ Warning: This is a ONE-TIME command and the operation is IRREVERSIBLE!
+ Press PB0 to confirm or press PB1 to abort.

. Initialize public sign key of target device... OK

. Read target device public sign key... OK
+ The public sign key (64 bytes): C4AF4AC69AAB9512DB50F7A26AE5B4801183D85417E729A56DA974F4E08A562C
                                  DE6019DEA9411332DC1A743372D170B436238A34597C410EA177024DE20FC819
```

- SE will return the SE_RESPONSE_INVALID_PARAMETER code if the Public Sign Key has already been initialized.

```
. Initialize public sign key of target device... Failed - Parameters are invalid or buffer is too small.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

Initialize Public Command Key

- To initialize the public_command_key[] to SE OTP.

```
+ Current DCI task is INITIALIZE PUBLIC COMMAND KEY.
+ Warning: This is a ONE-TIME command and the operation is IRREVERSIBLE!
+ Press PB0 to confirm or press PB1 to abort.

. Initialize public command key of target device... OK

. Read target device public command key... OK
+ The public command key (64 bytes): B1BC6F6FA56640ED522B2EE0F5B3CF7E5D48F60BE8148F0DC08440F0A4E1DCA4
                                     7C04119ED6A1BE31B7707E5F9D001A659A051003E95E1B936F05C37EA793AD63
```

- SE will return the SE_RESPONSE_INVALID_PARAMETER code if the Public Command Key has already been initialized.

```
. Initialize public command key of target device... Failed - Parameters are invalid or buffer is too small.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

Initialize HSE OTP (EFR32xG21B)

- To initialize the xg21b_conf[] to the HSE OTP.

```
+ Current DCI task is INITIALIZE SE OTP (EFR32xG21B).
+ Warning: This is a ONE-TIME command and the operation is IRREVERSIBLE!
+ Press PB0 to confirm or press PB1 to abort.

. Initialize SE OTP of EFR32xG21B device... OK

. Read target device user (SE OTP) configuration... OK
+ Secure boot                    : Enabled
+ Secure boot verify certificate : Disabled
+ Secure boot anti-rollback      : Enabled
+ Secure boot page lock narrow   : Disabled
+ Secure boot page lock full     : Disabled
+ Tamper source level (valid on Secure Vault device)
  Filter counter         :  1
  SE watchdog            :  4
  SE RAM CRC             :  4
  SE hard fault          :  4
  SE software assertion  :  4
  User secure boot       :  0
  Mailbox authorization  :  1
  DCI authorization      :  0
  Flash integrity        :  4
  Self test              :  4
  TRNG monitor           :  1
  PRS0                   :  1
  PRS1                   :  1
  PRS2                   :  2
  PRS3                   :  2
  PRS4                   :  4
  PRS5                   :  4
  PRS6                   :  7
  PRS7                   :  7
  Decouple BOD           :  4
  Temperature sensor     :  2
  Voltage glitch falling :  2
  Voltage glitch rising  :  2
  Secure lock            :  4
  SE debug               :  0
  Digital glitch         :  2
  SE ICACHE              :  4
+ Reset period for the tamper filter counter: ~32 ms x 1024
+ Activation threshold for the tamper filter: 4
+ Digital glitch detector always on: Disabled
+ Tamper reset threshold: 5
```

- SE will return the SE_RESPONSE_INVALID_COMMAND code if the HSE OTP has already programmed.

```
. Initialize SE OTP of EFR32xG21B device... Failed - Unsupported command.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

- SE will return the SE_RESPONSE_INVALID_PARAMETER code if secure boot option is enabled and the Public Sign Key has not been provisioned.

```
. Initialize SE OTP of EFR32xG21B device... Failed - Parameters are invalid or buffer is too small.
+ Press PB0 to issue a pin reset to the target, press PB1 to skip.
```

Disable Device Erase

- To disable the Erase Device command.

```
+ Current DCI task is DISABLE DEVICE ERASE.
+ Warning: This is a ONE-TIME command and the operation is IRREVERSIBLE!
+ Press PB0 to confirm or press PB1 to abort.

. Disable device erase of target device... OK

. Read target device SE status... OK
+ SE firmware version  : 00010204
+ MCU firmware version : 010A0003
+ Debug lock           : Disabled
+ Debug lock state     : False
+ Device Erase         : Disabled
+ Secure debug         : Disabled
+ Secure boot          : Enabled
+ Boot status          : Command successful.
```

6.5 SWD Programmer Examples

The following SWD programmer examples are based on BRD4181C and BRD4182A as target devices (Figure 6.2 Programmer Connection Diagram on page 19). Software is compiled with -O2 optimization in the Simplicity IDE of Simplicity Studio 5.

If the secure boot option is enabled in the SE OTP or bootloader, the application firmware must be signed.

Some operations are implemented by the SE Manager APIs in an application firmware that is programmed to the target device.

- For more information about SE Manager, see AN1190: Series 2 Secure Debug.
- The SE Manager is available in Gecko SDK Suite 3.0.0 or later.
- The SE Manager APIs are fully described in the Silicon Labs online documentation located at https://docs.silabs.com/gecko-platform/latest/service/api/group-sl-se-manager.

Erase Main Flash

- This example demonstrates the flash page erase or mass erase operation through the SWD interface.

```
+ Current SWD task is ERASE MAIN FLASH.
+ Press PB0 to confirm or press PB1 to abort.

. Connect to target device through SWD interface.
+ Initialize DP... OK - IDCODE = 0x6BA02477
+ Read AP... OK - IDR = 0x84770001
+ Set up AHB-AP and halt target... OK
+ Get device information... OK - Target device is EFR32MG21B010F1024 and unique ID is 0x14B457FFFE0F7777

. Erase main flash of target device... OK (cycles: 1538485 time: 19 ms)
```

Program Main Flash

- This example demonstrates the flash erase, flash write, and flash verify operations through the SWD interface. In this example, an application firmware image xg21_app_image[] is programmed to the target device main flash (APP_START_ADDR).

```
+ Current SWD task is PROGRAM MAIN FLASH.
+ Press PB0 to confirm or press PB1 to abort.

. Connect to target device through SWD interface.
+ Initialize DP... OK - IDCODE = 0x6BA02477
+ Read AP... OK - IDR = 0x84770001
+ Set up AHB-AP and halt target... OK
+ Get device information... OK - Target device is EFR32MG21B010F1024 and unique ID is 0x14B457FFFE0F7777

. Program an application firmware image to target device.
+ EFR32xG21 application firmware image size is 8968 bytes and start address is 0x00000000.
+ Erase-Program-Verify the EFR32xG21 main flash for application firmware image... OK (cycles: 11043525 time: 138 ms)
+ Issue a soft reset to run the application firmware... OK
```

Erase User Data (HSE Devices)

- For HSE devices (BRD4181C), the user data can only be erased by issuing a command to the HSE through the SE Manager API.
- In this example, an application firmware image erase_xg21_userdata[] (see source code below) is programmed to the target device main flash (APP_START_ADDR).

```
#include "em_chip.h"
#include "em_cmu.h"
#include "sl_se_manager.h"
#include "sl_se_manager_util.h"

/***************************************************************************//**
 * Main function
 ******************************************************************************/
int main(void)
{
  // Command context
  sl_se_command_context_t cmd_ctx;

  // Chip errata
  CHIP_Init();

  // Switch SYSCLK to 38 MHz HFRCO
  CMU_HFRCODPLLBandSet(cmuHFRCODPLLFreq_38M0Hz);

  // Initialize SE Manager
  sl_se_init();

  // Erase user page
  sl_se_erase_user_data(&cmd_ctx);

  while (1) ;
}
```

- Issue a soft reset to run the application above to erase the user data.

```
  + Current SWD task is ERASE USER DATA.
  + Press PB0 to confirm or press PB1 to abort.

  . Connect to target device through SWD interface.
  + Initialize DP... OK - IDCODE = 0x6BA02477
  + Read AP... OK - IDR = 0x84770001
  + Set up AHB-AP and halt target... OK
  + Get device information... OK - Target device is EFR32MG21B010F1024 and unique ID is 0x14B457FFFE0F7777

  . Erase EFR32xG21 user data through application firmware.
  + EFR32xG21 application firmware image size is 3068 bytes and start address is 0x00000000.
  + Erase-Program-Verify the EFR32xG21 main flash for application firmware image... OK (cycles: 4466426 time: 55 ms)
  + Issue a soft reset to run the application firmware to erase the EFR32xG21 user data... OK
```

Erase User Data (VSE Devices)

- For VSE devices (BRD4182A), the user data can be erased in the same way as any page in the main flash.

```
  + Current SWD task is ERASE USER DATA.
  + Press PB0 to confirm or press PB1 to abort.

  . Connect to target device through SWD interface.
  + Initialize DP... OK - IDCODE = 0x6BA02477
  + Read AP... OK - IDR = 0x84770001
  + Set up AHB-AP and halt target... OK
  + Get device information... OK - Target device is EFR32MG22C224F512 and unique ID is 0x680AE2FFFE287808

  . Erase EFR32xG22 user data... OK (cycles: 2008948 time: 25 ms)
```

Program User Data (HSE Devices)

- For HSE devices (BRD4181C), the user data can only be written by issuing a command to the HSE through the SE Manager API.
- In this example, an application firmware image write_xg21_userdata[] (see source code below) is programmed to the target device main flash (APP_START_ADDR).

```
#include "em_chip.h"
#include "em_cmu.h"
#include "sl_se_manager.h"
#include "sl_se_manager_util.h"

// User data
SL_ALIGN(4) static const uint32_t user_data[256] SL_ATTRIBUTE_ALIGN(4) = {
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
  0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55, 0x55AA55AA, 0xAA55AA55,
};

/***************************************************************************//**
 * Main function
 ******************************************************************************/
int main(void)
{
  // Command context
  sl_se_command_context_t cmd_ctx;

  // Chip errata
  CHIP_Init();

  // Switch SYSCLK to 38 MHz HFRCO
  CMU_HFRCODPLLBandSet(cmuHFRCODPLLFreq_38M0Hz);

  // Initialize SE Manager
  sl_se_init();

  // Erase user data
  sl_se_erase_user_data(&cmd_ctx);

  // Write user data
  sl_se_write_user_data(&cmd_ctx, 0, (uint32_t *)user_data, sizeof(user_data));

  while (1) ;
}
```

- Issue a soft reset to run the application above to write the user data.

```
+ Current SWD task is PROGRAM USER DATA.
+ Press PB0 to confirm or press PB1 to abort.

. Connect to target device through SWD interface.
+ Initialize DP... OK - IDCODE = 0x6BA02477
+ Read AP... OK - IDR = 0x84770001
+ Set up AHB-AP and halt target... OK
+ Get device information... OK - Target device is EFR32MG21B010F1024 and unique ID is 0x14B457FFFE0F7777

. Program EFR32xG21 user data through application firmware.
+ EFR32xG21 application firmware image size is 4228 bytes and start address is 0x00000000.
+ Erase-Program-Verify the EFR32xG21 main flash for application firmware image... OK (cycles: 5759117 time: 71 ms)
+ Issue a soft reset to run the application firmware to program the EFR32xG21 user data... OK
```

Program User Data (VSE Devices)

- For VSE devices (BRD4182A), the user data can be written in the same way as any page in the main flash.

```
+ Current SWD task is PROGRAM USER DATA.
+ Press PB0 to confirm or press PB1 to abort.

. Connect to target device through SWD interface.
+ Initialize DP... OK - IDCODE = 0x6BA02477
+ Read AP... OK - IDR = 0x84770001
+ Set up AHB-AP and halt target... OK
+ Get device information... OK - Target device is EFR32MG22C224F512 and unique ID is 0x680AE2FFFE287808

. Program EFR32xG22 user data.
+ User data size is 1024 bytes and start address is 0x0FE00000.
+ Erase-Program-Verify the EFR32xG22 user data... OK (cycles: 2157417 time: 26 ms)
```

Upgrade SE Firmware Through Application Firmware

- If the target device SE firmware version is less than v1.2.2, the SE firmware can only be upgraded by issuing a command to the SE through the SE Manager API.
- Connect to the target device through the SWD interface.
- An application firmware image prog_xg21_hse_upgrade[] (see source code below) is programmed to the target device main flash (APP_START_ADDR).

```
#include "em_chip.h"
#include "em_cmu.h"
#include "sl_se_manager.h"
#include "sl_se_manager_util.h"

// SE firmware image start address
#define SE_START_ADDR           (0x00040000UL)

// Current SE firmware version
static uint32_t current_version;

// Upgrade SE firmware version
static volatile uint32_t upgrade_version;

/***************************************************************************//**
 * Main function
 ******************************************************************************/
int main(void)
{
  // Command context
  sl_se_command_context_t cmd_ctx;

  // Chip errata
  CHIP_Init();

  // Switch SYSCLK to 38 MHz HFRCO
  CMU_HFRCODPLLBandSet(cmuHFRCODPLLFreq_38M0Hz);

  // Initialize SE Manager
  sl_se_init();

  // Get current SE firmware version
  if (sl_se_get_se_version(&cmd_ctx, &current_version) != 0) {
    goto exit;
  }

  // Get upgrade SE firmware version
  upgrade_version = *((uint32_t *)SE_START_ADDR + 3);

  // Check if upgrade version > current version
  if (upgrade_version <= current_version) {
    goto exit;
  }

#if !defined(CRYPTOACC_PRESENT)
  // Validate the SE firmware image
  if (sl_se_check_se_image(&cmd_ctx, (uint32_t *)SE_START_ADDR) != 0) {
    goto exit;
  }
#endif

  // Upgrade the SE firmware image
  sl_se_apply_se_image(&cmd_ctx, (uint32_t *)SE_START_ADDR);

exit:
  while (1) ;
}
```

- The original application firmware image on `APP_START_ADDR` will be erased.

```
+ Current SWD task is UPGRDE SE FIRMWARE THROUGH APPLICAION FIRMWARE.
+ Press PB0 to confirm or press PB1 to abort.

. Connect to target device through SWD interface.
+ Initialize DP... OK - IDCODE = 0x6BA02477
+ Read AP... OK - IDR = 0x84770001
+ Set up AHB-AP and halt target... OK
+ Get device information... OK - Target device is EFR32MG21B010F1024 and unique ID is 0x14B457FFFE0F7777

. Program an application firmware image to target device to upgrade the SE firmware.
+ EFR32xG21 HSE upgrade application firmware image size is 3236 bytes and start address is 0x00000000.
+ Erase-Program-Verify the EFR32xG21 main flash for application to upgrade
  HSE firmware... OK (cycles: 4652183 time: 58 ms)
```

- A SE upgrade firmware image xg21_hse_image[] is programmed to the target device main flash (SE_START_ADDR). The `APP_START_ADDR` cannot overlap with `SE_START_ADDR`. This means the `SE_START_ADDR` must be greater than the `APP_START_ADDR` plus the size (3236 bytes) of the application firmware image (`prog_xg21_hse_upgrade[]`).

```
. Program a SE firmware image to target device.
+ EFR32xG21 HSE firmware image version: 00010206
+ EFR32xG21 HSE firmware image size is 41125 bytes and start address is 0x00040000.
+ Erase-Program-Verify the EFR32xG21 main flash for HSE firmware image... OK (cycles: 52209679 time: 652 ms)
```

- Issue a pin reset to run the application firmware in `prog_xg21_hse_upgrade[]`.
- Check the SE firmware version after a few seconds to verify the SE firmware has been upgraded.

```
+ Issue a pin reset to run the application to upgrade the SE firmware.
+ Delay few seconds to check SE status... Done

. Read target device SE status... OK
+ SE firmware version  : 00010206
+ MCU firmware version : NA
+ Debug lock           : Disabled
+ Debug lock state     : False
+ Device Erase         : Enabled
+ Secure debug         : Disabled
+ Secure boot          : Disabled and SE OTP is not configured
+ Boot status          : Command successful.
```

- The SE firmware cannot be downgraded so the upgrade will be ignored if SE firmware with the same or a lower version is applied to the device.

6.6 Benchmark

The xg21_app_image[] or xg22_app_image[] is replaced by a 256 kB application firmware image and the test results in Table 6.8 Erase-Program-Verify Time for Different Target Devices on page 38 are based on following conditions.

- Gecko SDK Suite v3.1.2
- Erase, program, and verify the main flash
- Compile options are set to default values
- Software is compiled with -O2 in Simplicity IDE of Simplicity Studio 5
- The programmer (EFR32MG22C224F512IM40) is running at 80 MHz

Table 6.8. Erase-Program-Verify Time for Different Target Devices

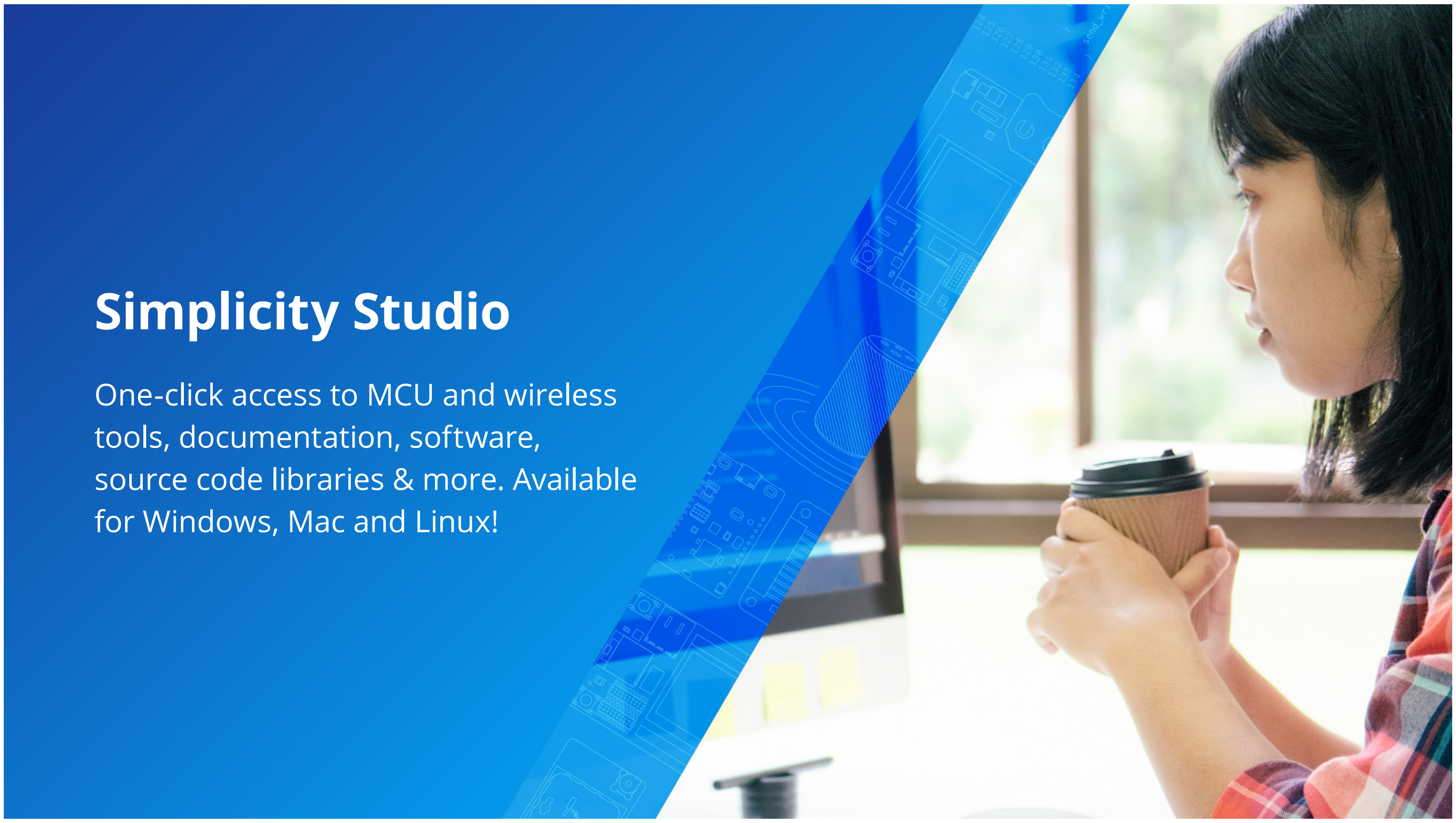
Target Device	Application Firmware Image Size	Erase-Program-Verify Time
EFR32xG21A (BRD4180A)	256 kB (xg21_app_image[])	3.67 s
EFR32xG21B (BRD4181C)	256 kB (xg21_app_image[])	3.67 s
EFR32xG22 (BRD4182A)	256 kB (xg22_app_image[])	3.67 s

7. Revision History

Revision 0.1

April 2021

- Initial Revision.


Simplicity Studio
One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com