PSoC™ 61/62/63 RMA mode for field failure analysis

About this document

Scope and purpose

This document guides you on how to transition the PSoC™ 61/62/63 MCU into the return merchandise authorization (RMA) life cycle stage. Devices in RMA mode can be returned to Infineon for failure analysis. The companion code example in CE234992 - PSoC™ 6 MCU: Security application template demonstrates RMA mode transitioning of the PSoC™ 61/62/63 MCU.

Intended audience

This application note is intended for users who want to learn more about the RMA process on the PSoC™ 61/62/63 family of devices.

Table of contents

1 Introduction

Return merchandise authorization (RMA) is the life cycle stage of the PSoC™ 6 MCU series (PSoC™ 61/62/63) that allows the device to return to Infineon for the purpose of evaluation or failure analysis. To place a device in RMA mode, the application code must successfully execute the `TransitionToRMA` system call. This system call is only valid when the device is in the SECURE/SECURE_WITH_DEBUG stage. See the System call API for details about this system call.

To transition the device into RMA, it is required to generate an RMA certificate, copy it to the existing RMA code that is part of CE234992 - PSoC™ 6 MCU: Security application template code example (CE), and run the CE. For details on how to generate the RMA certificate and transition the device into the RMA stage; see the subsequent sections in this document.

For information on getting started with PSoC™ 6 MCU security, see AN221111 - PSoC™ 6 MCU designing a custom secured system.

2 Life cycle stages

PSoC™ 6 MCUs have configurable, non-volatile life cycle stages. Writing to the proper eFuse bits governs a strict, irreversible progression of life cycle stages. eFuse is a 1024-bit non-volatile memory area, with each bit being one-time programmable (OTP).

The life cycle stages can transition only from one stage to the next and cannot be reversed. For example, when the life cycle has transitioned from NORMAL to SECURE, it cannot be transitioned back to NORMAL. It may remain in the SECURE mode or transition to RMA.

Once the device has been transitioned to the RMA stage, the transition from RMA is irreversible. PSoC™ 6 MCU leaves the Infineon factory in the NORMAL stage.

For more information on life cycle stages and their transition, see AN221111 - PSoC™ 6 MCU designing a custom secured system.

3 RMA process

The RMA process involves the user transitioning the device into the RMA stage and shipping the device to Infineon, along with the RMA certificate. Before moving the device to RMA, you must erase any sensitive data on the device.

After the device is transitioned to the RMA stage, when it reboots, it does not attempt to execute any code in user flash. Instead, it waits for an `OpenRMA` system call. The `OpenRMA` system call requires the same RMA certificate as a parameter that was used to place the device in the RMA stage. When the part is sent to Infineon for failure analysis, the RMA certificate that contains the device unique ID should be supplied. The RMA certificate is unique for each device as it is generated based on the device's unique ID.

3 RMA process

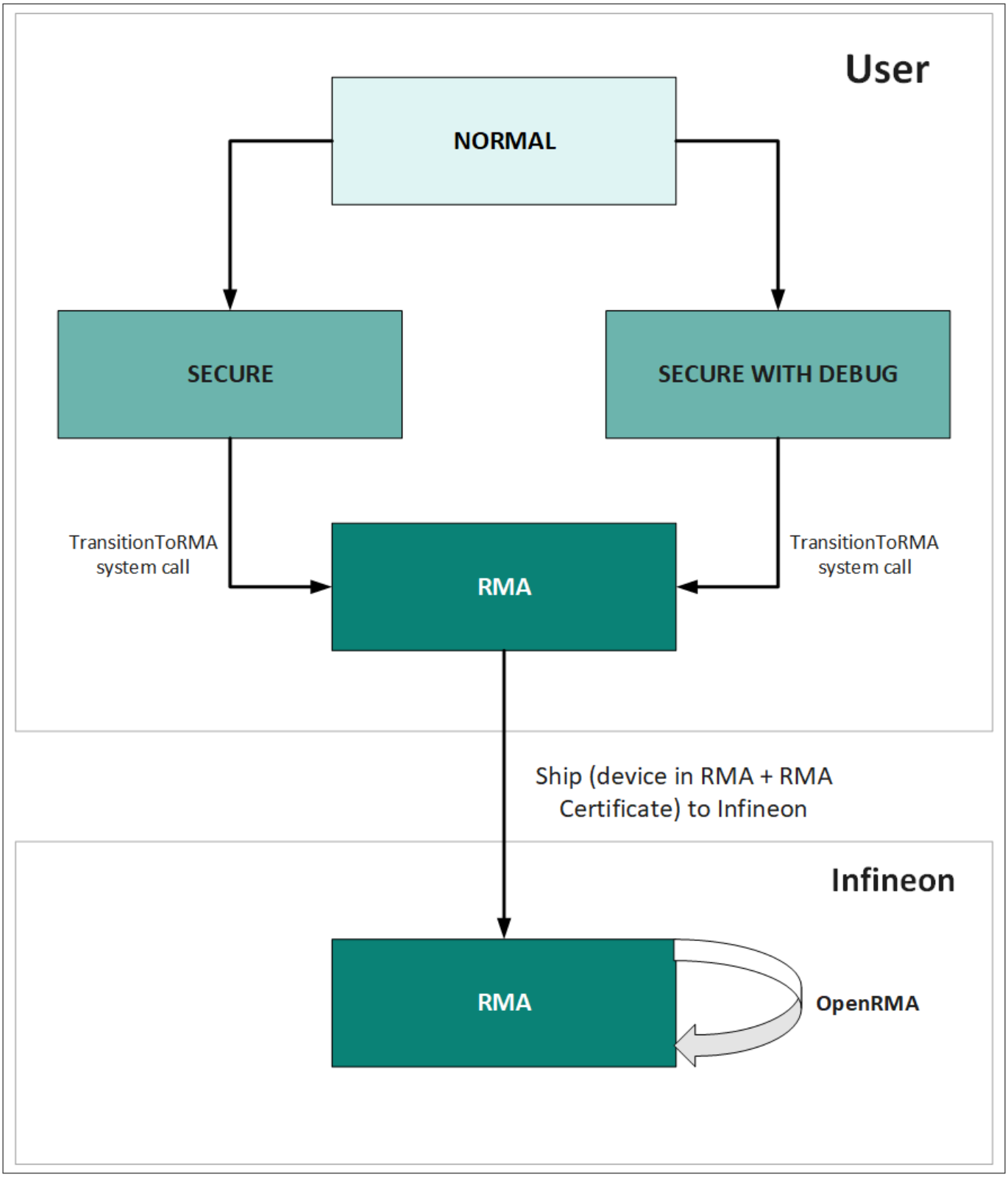

Figure 1 RMA process

To transition a device to the RMA stage, you must have access to the following:

- The device's unique ID.
- A private key that is paired with a public key is stored and authenticated in SFlash.

Send the following special commands as part of the application:

- Read the internal unique device ID.
- Invoke the transition to RMA.

You can implement the special commands in two ways:

1. Include these special commands as part of the customer application.
2. Create a special device code image that supports only the required commands and program it onto the device.
   - With this approach, when boot-loading this special code image, all proprietary and sensitive data can be erased at the same time. In addition, a special code image allows easy implementation of a standard interface such as a UART to implement the communication needed to invoke the commands.

In this application note, only sending the special commands as part of the existing application is described. See CE234992 – PSoC™ 6 MCU: Security application template that uses the Cortex®-M4 application to perform `TransitionToRMA`.

3.1 Transitioning the device to RMA mode

Follow these steps to transition the device to RMA mode:

1. Erase all sensitive or proprietary code stored on the device. This may be performed using a Flash API from the user's code. Erase the flash at least four times to ensure there is no possibility to detect any residual code. The public key stored in SFlash must remain because it is used to transition to the RMA life cycle stage and to allow Infineon to open the RMA later.
2. Read the device unique ID (12 bytes) stored in the device's SFlash.
3. Generate a certificate using the unique IDs. The certificate is hashed with SHA-256, and the generated hash is encrypted with the customer's private key to create a digital signature. The customer's private key is paired with the public key stored internally in SFlash. This is the same method that is used to sign a code, as described in "Code signing and verification" section in the AN221111 - PSoC™ 6 MCU designing a custom secured system document, using the same private/public key pairs. Figure 2 shows the format of the certificate.

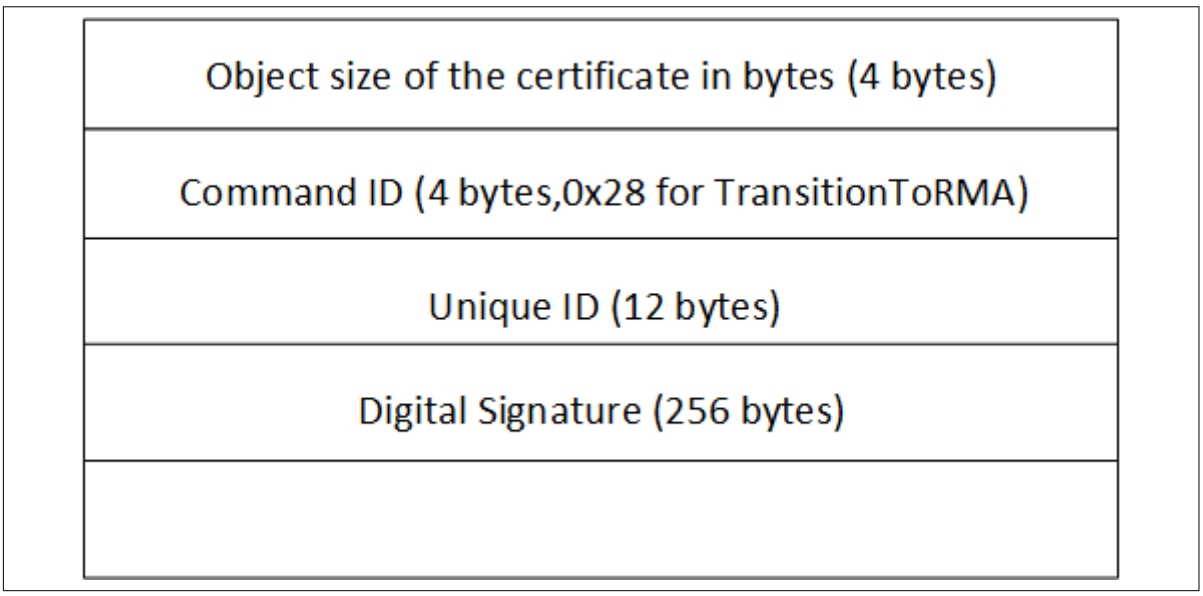

Figure 2 RMA certificate

4. The user's code must execute the `TransitiontoRMA` system call in order to transition the device to the RMA life cycle stage. The user must implement code to accept the RMA certificate and invoke the `TransitionToRMA` system call. This system call can only be executed in the SECURE/ SECURE_WITH_DEBUG life cycle stage. The system call may be performed from a user application if it is

supported or via the debug access port (DAP) port if it is enabled. The device VDDIO0 supply must be at 2.5 V before performing this step because the RMA eFuse is to be programmed (any programming of eFuse bits requires the VDDIO0 to be at 2.5 V). For VDDIO0 supply, see the "Pinouts" section in the PSoC™ 6 MCU datasheet.

5. Once the device is in the RMA stage, it may be shipped to Infineon along with the RMA certificate that was used to transition the device into RMA.
6. Infineon will perform an `OpenRMA` system call on the returned device to evaluate or perform a failure analysis.

4 System call API

The process of transitioning the device into the RMA stage is implemented as a system call. System calls can be made by Cortex®-M0+ and Cortex®-M4. Each has a reserved IPC structure through which they can request the execution of a system call. The opcodes (8-bit numbers) are used to identify different system calls. The caller sends an IPC structure pointer via IPC message to be written to the IPC data register. The IPC structure pointer points to the SRAM address that holds the LCS parameters. This results in an NMI interrupt in Cortex®-M0+, where the system call is executed. When the system call execution is finished, the IPC structure is released, and the caller can read a return code in the data register of the IPC structure. For more information about PSoC™ 6 system calls, see the device-specific documents in the PSoC™ 6 reference manuals.

The `TransitionToRMA` system call transitions the parts from SECURE/SECURE_WITH_DEBUG to the RMA life cycle stage. A certificate that contains the unique ID that is signed by a specified installed private key, which is passed as a parameter to the system call.

Table 1 and Table 2 show the arguments and return value of the system call to transition a device to the RMA life cycle stage.

The system call return code is a 32-bit value. The return code and output data are passed in a similar way as input parameters. The data register of the IPC structure can contain either the return code or a pointer to the SRAM, where the return code and output data are stored (if they are provided by the system call).

Table 1 Table 1: TransitionToRMA syscall API arguments

Address	Value to be written	Description
IPC_STRUCT.DATA register		
Bits[31:0]	SRAM_SCRATCH_ADDR	SRAM address where the API parameters are stored. This must be a 32-bit aligned address.
SRAM_SCRATCH		
Bits[31:24]	0x28	Opcode for TransitionToRMA
Bits[23:0]	0xXX	Not used
SRAM_SCRATCH + 0x04		
Bits[31:0]	0x14	Object size in bytes, including itself (it should always be 20 bytes for PSoC™ 6)
SRAM_SCRATCH + 0x08		
Bits[31:0]	0x120028F0	Command ID is fixed
SRAM_SCRATCH + 0x0C		
Bits[31:0]		Unique DIE ID word 0 (LSB)
SRAM_SCRATCH + 0x010		
Bits[31:0]		Unique DIE ID word 1 (Mid-MSB)
SRAM_SCRATCH + 0x14		
Bits[31:24]	Padding	1 byte padding (MSB)
Bits[23:0]		Unique DIE ID word 2
SRAM_SCRATCH + 0x18		
Bits[31:0]		Signature of the certificate (256 bytes)

Note: A 12-byte unique ID is formed by 11-byte of DIE ID from SFlash and 1 padding byte.

4 System call API

Table 2 Table 2: TransitionToRMA return value

Address	Return value	Description
Bits[31:28]	0xA = SUCCESS 0XF = ERROR	Status code
Bits[27:24]	0xXX	Not used
Bits[23:0]	Error code	See the SROM API status code in the PSoC™ 6 MCU architecture reference manual.

5 RMA certificate format

You must provide a certificate that authorizes Infineon to transition the part with a unique ID to the RMA life cycle stage. The certificate will be signed by the customer using the same private key that is used for signing the user application image. Flash boot uses the same algorithm for authenticating the user application when verifying the signature. The same public key (injected by the OEM) stored in SFlash is used for the verification. This is achieved using the `TransitionToRMA` ROM system call.

The following is a sample RMA certificate:

```
/* RMA Certificate to be sent to the device. It is unique per device */
transit_rma_param_t rmaParam =
{
.opcode = 0x28000000,
.obj_size = 0x00000014,
.cmd_id = 0x120028F0,
.unique_id_0 = 0x118c84f4,
.unique_id_1 = 0x01af2c10,
.unique_id_2 = 0x007a0415,
.signature = {0x84, 0x95, 0x4c, 0x6a, 0xf4, 0x80, 0x64, 0x73, 0xc2, 0x7f, 0x5d, 0x46, 0xaa,
0xc7, 0x77, 0x9b,
0xea, 0x20, 0xdf, 0x3e, 0x22, 0xd6, 0xd3, 0x07, 0x8b, 0x1d, 0x80, 0x50, 0xc3, 0xff, 0x82, 0x15,
0x60, 0xb3, 0x8b, 0x22, 0x8d, 0x6f, 0x79, 0x42, 0x25, 0xda, 0x05, 0x66, 0x75, 0x84, 0x3f, 0x56,
0x24, 0x37, 0x3f, 0xa1, 0x9b, 0x4b, 0x2f, 0x3e, 0x35, 0x6b, 0x07, 0x5e, 0x5d, 0xf5, 0xf6, 0x06,
0x44, 0xde, 0xa2, 0xe3, 0xb3, 0x4c, 0x8f, 0xfd, 0x79, 0x55, 0x63, 0x9a, 0xc2, 0x66, 0x0e, 0x87,
0xe1, 0x30, 0x02, 0x3a, 0xe5, 0x1f, 0x03, 0x3d, 0x1e, 0x31, 0x1e, 0xd3, 0xdc, 0x49, 0xe4, 0x8b,
0x28, 0xc6, 0xe6, 0x85, 0xac, 0xbd, 0xcf, 0x3f, 0xcd, 0xf7, 0x21, 0x76, 0xd6, 0x9c, 0x63, 0x32,
0x5d, 0xb5, 0xf7, 0x8b, 0x7a, 0x4c, 0x35, 0xcd, 0x0e, 0x64, 0x45, 0xed, 0x28, 0x40, 0xc7, 0x0d,
0xf4, 0xdc, 0x4b, 0x75, 0xe0, 0x6a, 0x1d, 0x84, 0xac, 0xef, 0x14, 0x4e, 0xdb, 0x6a, 0x6b, 0xad,
0xcf, 0xb2, 0xc9, 0x16, 0xba, 0x45, 0xea, 0xbb, 0x18, 0xb8, 0x21, 0x2a, 0x86, 0x5d, 0x46, 0x44,
0xa2, 0x7e, 0x68, 0x07, 0xb3, 0x9a, 0x41, 0xd6, 0x32, 0x6a, 0x01, 0x83, 0x31, 0xfb, 0x67, 0x0b,
0x28, 0x5a, 0x5a, 0x01, 0x0f, 0xa2, 0x93, 0x40, 0x55, 0xb6, 0x43, 0x64, 0x3b, 0xae, 0xbe, 0x7a,
0xa9, 0xfa, 0x31, 0xcc, 0x81, 0x36, 0x65, 0xd6, 0xf2, 0x58, 0xc0, 0x4c, 0x25, 0x58, 0xe7, 0x4e,
0x6d, 0xa4, 0x2c, 0x59, 0xa0, 0x6a, 0xae, 0x6c, 0xc4, 0xa8, 0x39, 0xa0, 0x6a, 0xc3, 0x58, 0xe8,
0x87, 0xb1, 0xaa, 0xfc, 0x86, 0x0f, 0x5c, 0x16, 0x11, 0x20, 0x8d, 0xa1, 0x73, 0xb4, 0x0d, 0x56,
0x2c, 0xa6, 0x62, 0x20, 0x6c, 0x28, 0x2d, 0xbf, 0x77, 0xaa, 0x16, 0x16, 0x49, 0xac, 0x4c, 0xc9}
};
```

You can either generate the certificate as mentioned in the RMA process or use a script provided along with the CE234992 – PSoC™ 6 MCU: Security application template code example. See "README" section in the CE234992 – PSoC™ 6 MCU: Security application template to generate the certificate using the script.

6 Code example

The CE234992 – PSoC™ 6 MCU: Security application template CE demonstrates the process of transitioning the PSoC™ 6x (61/62/63) MCU into RMA mode as one of its key features. In this example, the RMA system call is performed by the Cortex®-M4 application on the Cortex®-M0+ processor. By default, the RMA functionality is disabled in the application. Set the `TRANSITION_TO_RMA` flag to enable the RMA transition functionality.

For more information about this code example and the RMA transition, see the "README" section in the CE234992 – PSoC™ 6 MCU: Security application template.

Note: *Currently, this code example supports only PSoC™ 62/63 devices. To work on PSoC™ 61, see* Appendix A: Porting CE234992 - PSoC™ 6 MCU: Security application template on PSoC™ 61.

7 Appendix A: Porting CE234992 - PSoC™ 6 MCU: Security application template on PSoC™ 61

The existing security template code example cannot run directly on PSoC™ 61 as the CE is designed for dual-core applications (Cortex®-M0+ and Cortex®-M4) and PSoC™ 61 is a single-core (Cortex®-M4) CPU device. Therefore, make minor modifications to the existing code example to make it run on PSoC™ 61.

Follow these steps to port the security template code example on PSoC™ 61:

1. Customize the BSP for PSoC™ 61 in the context of this code example. See the AN236015 - ModusToolbox™ BSP Assistant user guide for detailed instructions. In addition, see "Custom BSP for PSoC™ 61" section in the CCE236897 - MCUboot based Basic Bootloader for PSoC™ 61.
2. Change the *CORE* and *CORE_NAME* in *./proj_btldr_cm0p/Makefile* file.

```
# CPU to target; CM4 is the default CPU when this variable is not present.
-CORE=CM0P
-CORE_NAME=CM0P_0
+CORE=CM4
+CORE_NAME=CM4_1
```

3. Create a new file named *'.cyignore'* in *./proj_cm0p/* and add the following IPC files to *.cyignore*. This will ignore the *'ipc_communication.c'* and *'ipc_communication.h'* files during the build process.

```
+$(SEARCH_mcuboot)
+./source/ipc_communication.c
+./source/ipc_communication.h
```

4. Change *CORE* name in *./proj_cm0p/Makefile* file.

```
# Set this application to run at the CM0+
-CORE=CM0P
+CORE=CM4
```

5. Remove *ipc_communication.h* header from *./proj_cm0p/source/main.c* file.

```
#include "cy_pdl.h"
#include "cyhal.h"
#include "cybsp.h"
-#include "ipc_communication.h"
```

6. Add the following macros in *./proj_cm0p/source/main.c* file.

```
+#define RESET_VECTOR_POS (0x04)
+#define UART_TX_COMPLETE_POLL_COUNT (10UL)
```

7. Delete *cm0p_msg_callback(void)* function prototype and definition from *./proj_cm0p/source/main.c* file.

8. Replace the `main()` function in *./proj_cm0p/source/main.c* file. with the following code:

```
int main(void)
{
    cy_rslt_t result;

    static uint32_t appStartAddr = 0u;
    static uint32_t appStackPtr = 0u;

    /* Enable global interrupts */
    __enable_irq();

    /* Initialize the device and board peripherals */
    result = cybsp_init() ;
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

    /* Initialize the LED pin to strong drive mode */
    cyhal_gpio_init(USER_LED, CYHAL_GPIO_DIR_OUTPUT, CYHAL_GPIO_DRIVE_STRONG, true);

    /* CM0P_APP_FLASH_START and MCUBOOT_HEADER_SIZE is defined in Makefiles */
    uint32_t *CM4_App_Stack_Ptr = (uint32_t *)(CM4_APP_FLASH_START);
    uint32_t *CM4_App_PC_Ptr    = (uint32_t *)(CM4_APP_FLASH_START + RESET_VECTOR_POS);

    /* Get the user application start address and stack pointer address */
    appStartAddr = *CM4_App_PC_Ptr;
    appStackPtr  = *CM4_App_Stack_Ptr;

    /* Set the main stack pointer to the user app stack pointer */
    __set_MSP(appStackPtr);

    ((void (*)(void))appStartAddr)();
} // End of main(void)
```

9. Build the application and follow the same steps as mentioned in the CE234992 – PSoC™ 6 MCU: Security application template to generate the certificate and transition the device into the RMA stage.

8 Abbreviations

Term	Description
API	Application programming interface
DAP	Debug access port
IPC	Inter-process communication
MCU	Microcontroller unit
NMI	Non-maskable interrupt
OTP	One-time programmable
RMA	Return merchandise authorization
SRAM	Static random-access memory
UART	Universal Asynchronous Receiver/Transmitter

Revision history

Document version	Date of release	Description of changes
**	2023-09-26	• Initial release

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2023-09-26

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-urx1690206198240

Important notice

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.