Getting started with Trusted Firmware-M (TF-M) on PSOC™ Edge

About this document

Scope and purpose

This application note explains how to get started with Trusted Firmware-M (TF-M) on the PSOC™ Edge MCU. It describes how to build a TF-M-based application, add TF-M to an existing application, and use the services offered by TF-M with ModusToolbox™.

Intended audience

This is an introductory document for getting started with TF-M that is intended for users who are familiar with application development with PSOC™ Edge in ModusToolbox™ and PSOC™ Edge Protect Bootloader. For more information, see AN235935 Getting started with PSOC™ Edge E8 on ModusToolbox™ software.

Table of contents

Table of contents

1 Introduction

Trusted Firmware-M (TF-M) is an open-source initiative that implements the Arm® Platform Security Architecture (PSA) specifications. This implementation enables chips, real-time operating systems, and devices to become PSA Certified. TF-M leverages the security features of the MCU hardware to establish an isolated secure processing environment (SPE), where sensitive information can be securely processed and stored. In Armv8-M and Armv8.1-M architectures, TF-M relies on the isolation capability of Arm® TrustZone®, while in architectures earlier than Armv8-M, physical core isolation is necessary.

TF-M consists of the following components:

- **TF-M core:** TF-M core manages isolation among partitions of SPE and between the SPE and non-secure processing environment (NSPE). It manages communication among partitions of SPE and between SPE and NSPE. The TF-M core also manages execution of SPE partitions NSPE application
- **Secure services:** TF-M includes secure services such as Crypto, Internal Trusted Storage (ITS), Protected Storage (PS), firmware update, attestation, and platform. The secure services are utilized by NSPE application via PSA APIs. The secure services are organized into a cohesive software unit referred to as "partition"

Figure 1 illustrates the high level design of TF-M. For more information about TF-M, see Trusted Firmware-M documentation.

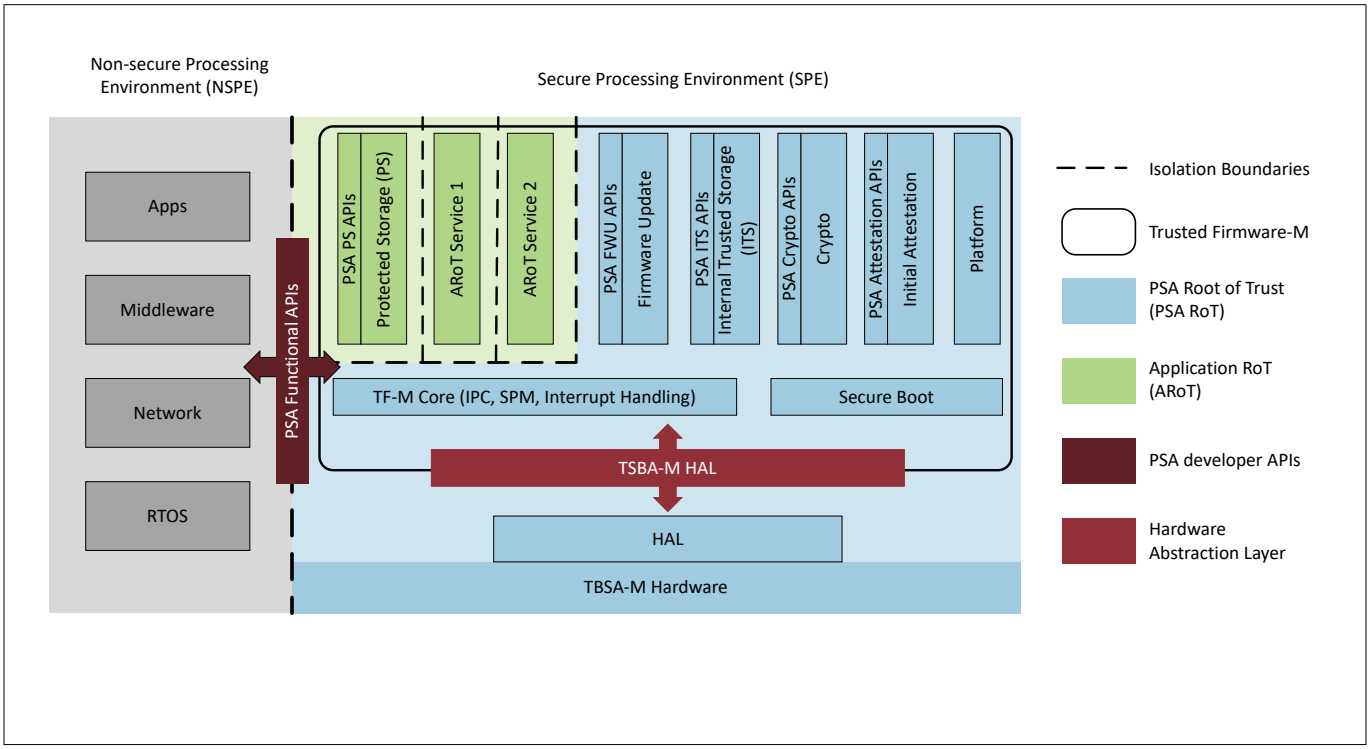

Figure 1 TF-M overview

2 Getting started with TF-M

This section describes how to build an application based on TF-M in ModusToolbox™. A starter template application "PSOC™ Edge basic TF-M application" is available in ModusToolbox™.

TF-M is available for both Edge Protect Category 2 (EPC2) and Edge Protect Category 4 (EPC4) parts. For more information about Edge Protect Category see here.

2.1 Prerequisites

It is assumed that the reader is familiar with PSOC™ Edge application development with ModusToolbox™ software. If you are new to PSOC™ Edge application development software, see the AN235935 Getting started with PSOC™ Edge E8 on ModusToolbox™ software application note for getting started with application development.

TF-M is designed to work together with the Edge Protect Bootloader, so the reader is expected to have a basic understanding of the Edge Protect Bootloader. If you are new to Edge Protect Bootloader see AN237857 Edge Protect Bootloader for PSOC™ Edge.

2.2 Boot flow

When the PSOC™ Edge device comes out of reset, the immutable boot firmware on the Secure Enclave (SE) is executed first. In EPC 2 device, the immutable boot firmware then initializes basic run time services (Basic RT services), enables the CM33 CPU, which executes extended boot from RRAM. In case of EPC4 device, the immutable boot firmware initializes secure enclave run time services (SE RT services), enables the CM33 CPU, which executes the extended boot from RRAM. Extended boot can launch the next application from two addresses. When the BOOT SW (DIP switch), connected to GPIO 17.6, is set to OFF (LOW), the extended boot launches the image from primary boot address and if BOOT SW is set to ON (HIGH), the extended boot launches the image from the alternate boot address. By default, the primary boot address is configured as 0x32011000 (RRAM) while the alternate boot address is 0x70100000 (external flash). These boot addresses can be changed by provisioning the device with the appropriate OEM policy. See "Alternate boot location" section from AN237849 Getting started with PSOC™ Edge security.

The extended boot launches the first OEM application, that is the Edge Protect Bootloader from RRAM (default placement). The Edge Protect Bootloader is placed at address 0x32011000. The Edge Protect Bootloader authenticates the TF-M (M33 SPE), M33 NSPE, and M55 NSPE applications and transfers control to TF-M. In EPC2 devices, TF-M is launched from external flash. In case of EPC4 devices, TF-M is loaded into SRAM from external flash by Edge Protect Bootloader and executed from SRAM. TF-M performs necessary device initialization, sets up isolation boundaries using various protection units, and starts the M33 NSPE.

Figure 2 summarizes the above-described boot flow:

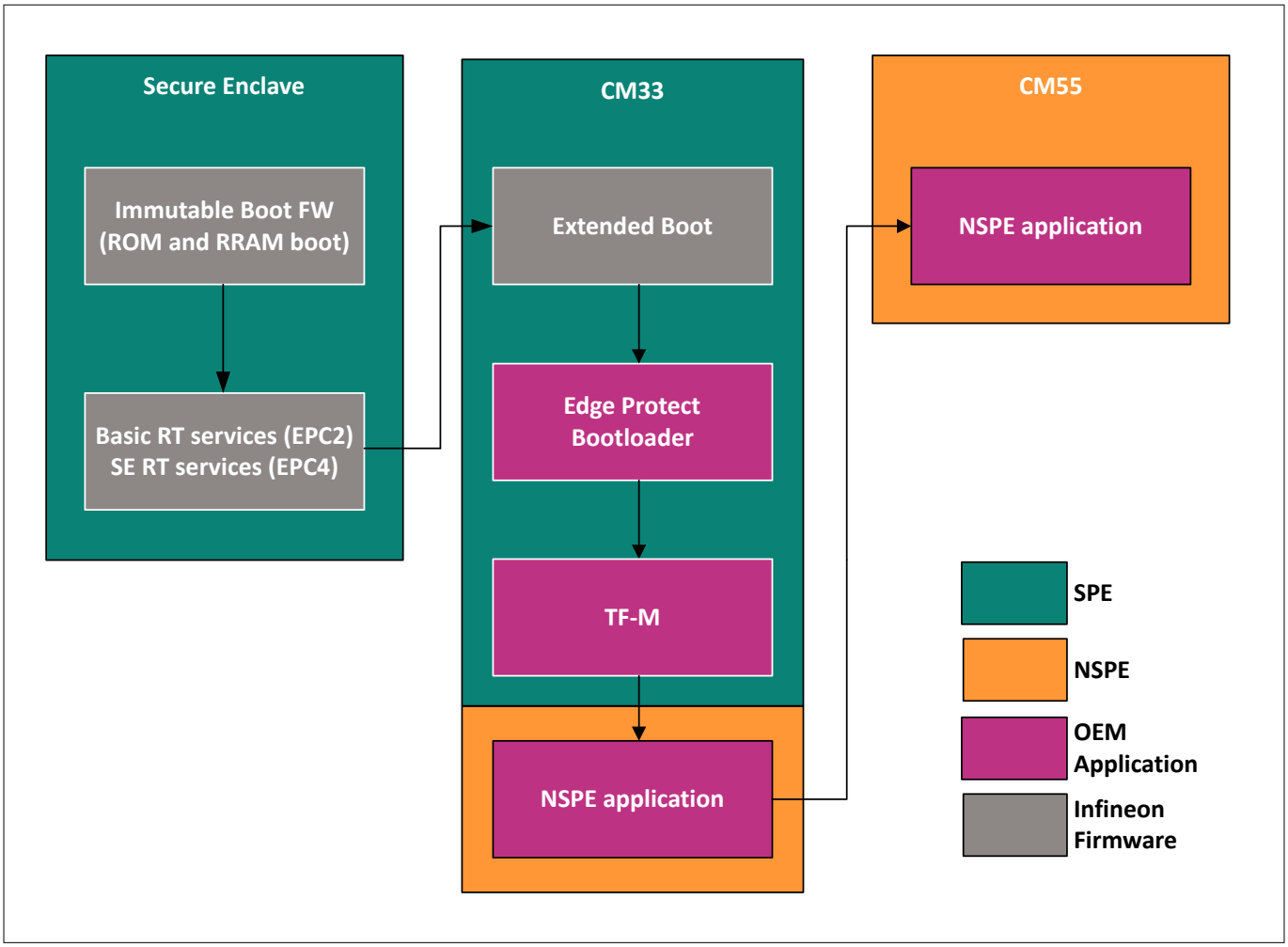

Figure 2 **Boot flow in PSOC™ Edge MCU with Edge protect Bootloader**

2.3 Application creation and programming procedure

This section provides step-by-step instructions to get started with PSOC™ Edge basic TF-M application. It provides guidance on how the create the PSOC™ Edge basic TF-M application and how to integrate Edge Protect Bootloader into it. If you need to add TF-M to an existing application, please refer to Add TF-M to an existing application for instructions.

2.3.1 Create basic TF-M based application

Create basic Trusted Firmware-M (TF-M) based application for `KIT_PSE84_EVAL_EPC2` or `KIT_PSE84_EVAL_EPC4` as shown below. The example is available under the **Security** section of the code example category.

***Note**: Windows has 260 character path length limit. A longer path to workspace may exceed this limit and cause build failure with TF-M application. Ensure that the path `<Workspace_Path>\<TF-M_app_name>\` (including back slashes) must be less than or equal to 32-characters. For example `C:\TF-M_workspace` can be used as workspace path and TF-M application name can be set as `TF-M`.*

2 Getting started with TF-M

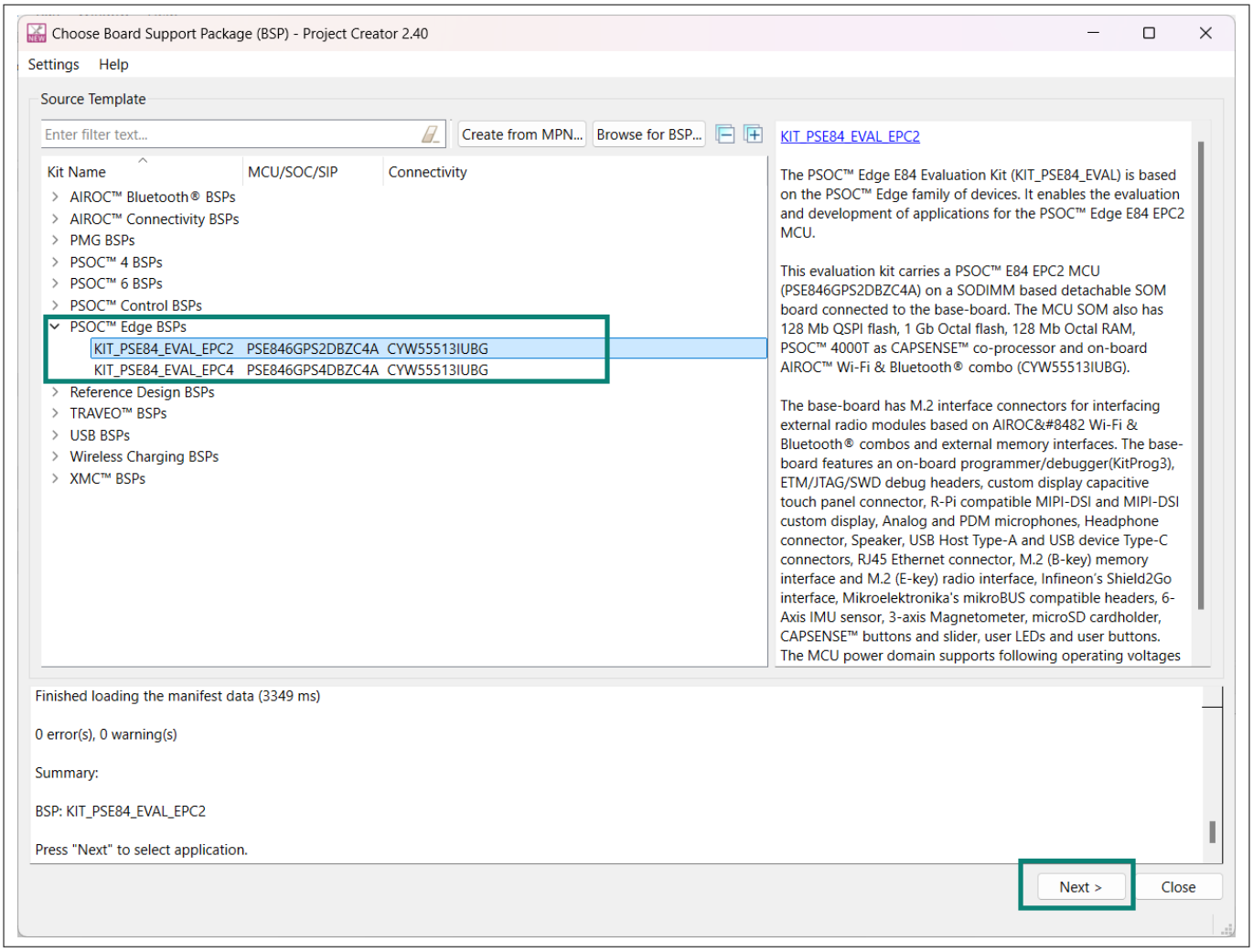

Figure 3 Select BSP for PSOC™ Edge MCU

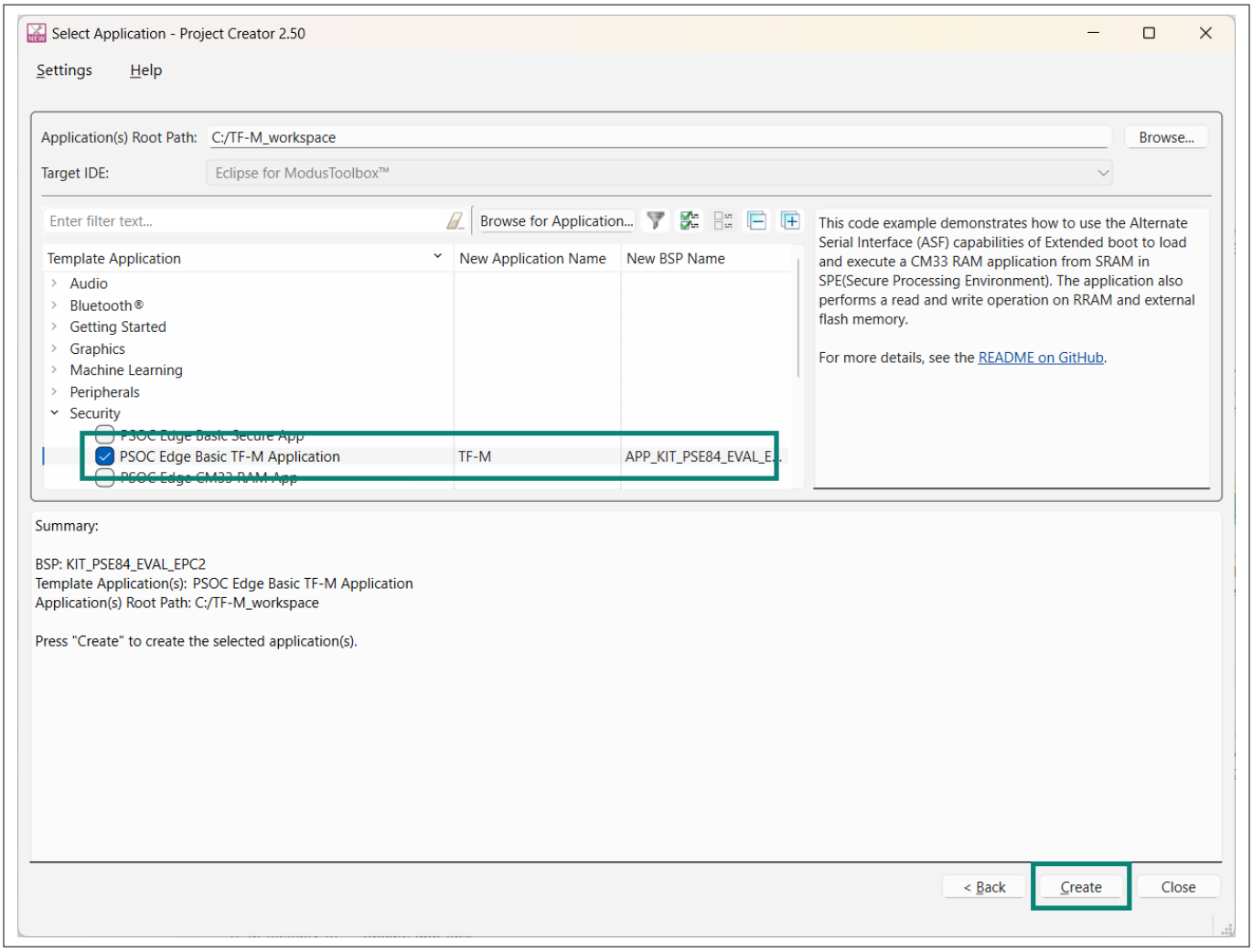

Figure 4 Creation of Basic TF-M application for PSOC™ Edge MCU

2.3.2 Add Edge Protect Bootloader to TF-M application

The TF-M based application is designed to work together with Edge Protect Bootloader. To ensure seamless integration between the TF-M application and the Edge Protect Bootloader application, it is imperative for the bootloader application to have detailed knowledge of the TF-M application's configuration parameters such as memory location, size, upgrade method, etc. To achieve this, the Edge Protect Bootloader application must be added as a project within the TF-M application.

The Edge Protect Bootloader is available as a separate code example with the title PSOC™ Edge Protect Bootloader in ModusToolbox™. Follow the steps provided below to integrate Edge Protect Bootloader in TF-M application

1. Create the PSOC™ Edge Protect Bootloader application for `KIT_PSE84_EVAL_EPC2` or `KIT_PSE84_EVAL_EPC4` BSP which is used for creating TF-M application as shown in Figure 5. The example is available under the **Security** section of the code example category

2 Getting started with TF-M

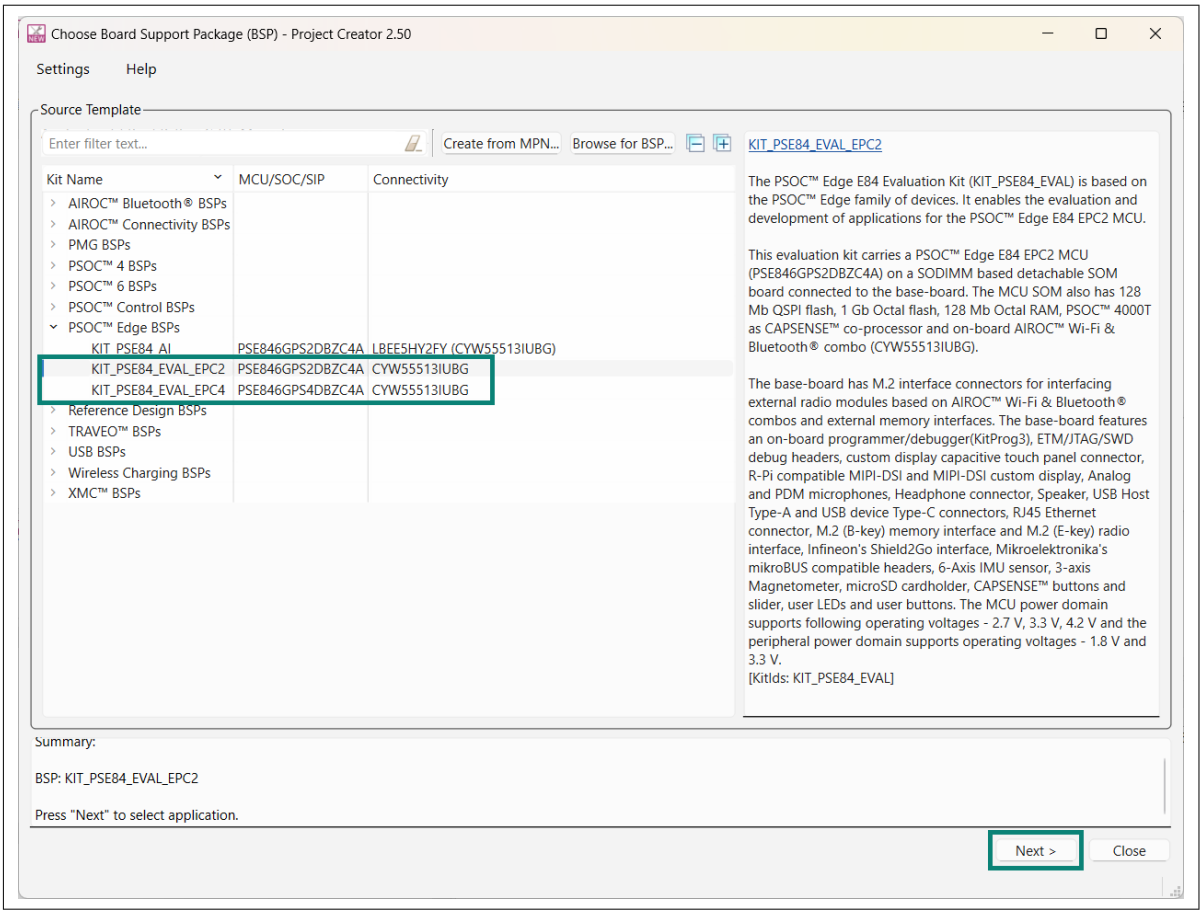

Figure 5 Select BSP for PSOC™ Edge MCU

2 Getting started with TF-M

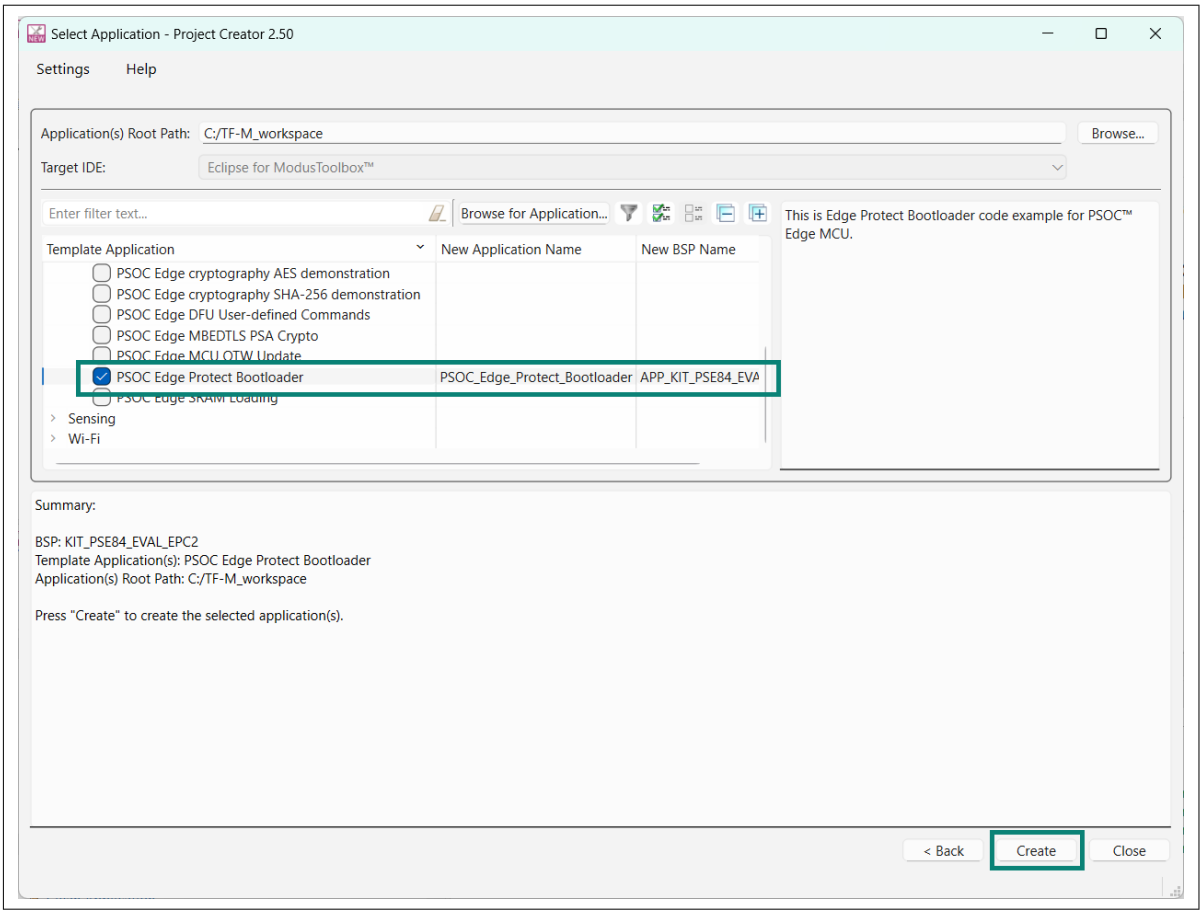

Figure 6 Creation of Edge Protect Bootloader application for PSOC™ Edge MCU

Verify that both Edge Protect Bootloader and TF-M application are created successfully and visible in project explorer of IDE

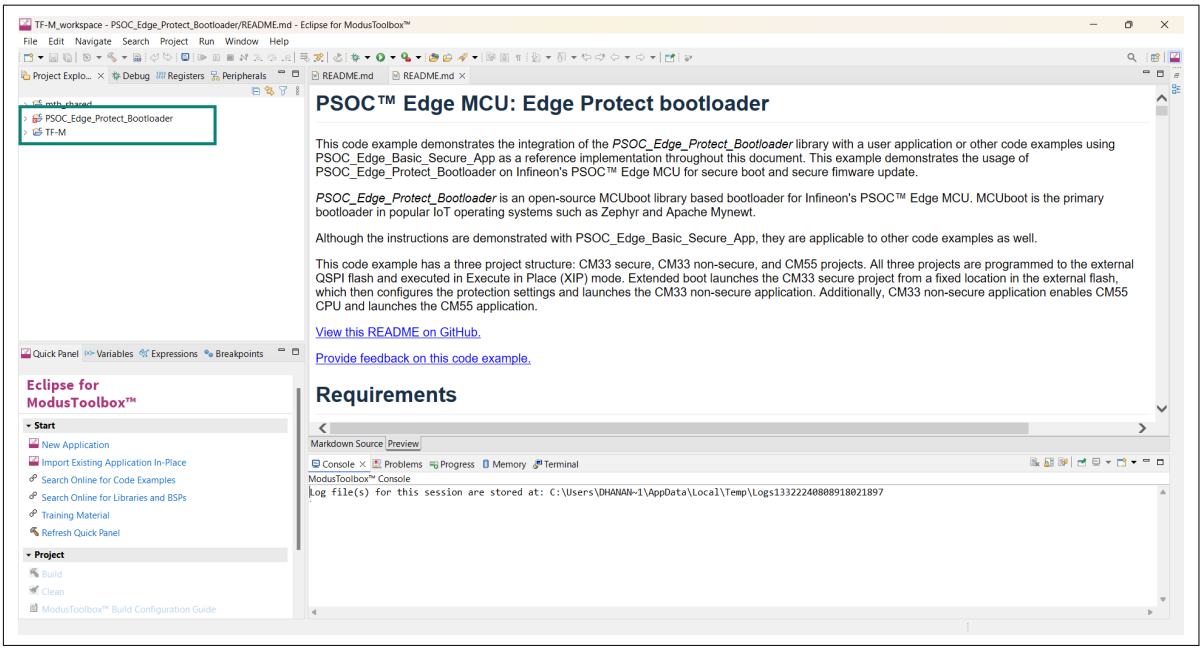

Figure 7 Eclipse IDE Project Explorer

2. Navigate to project explorer and select the `proj_bootloader` project from Edge Protect Bootloader application and copy it

2 Getting started with TF-M

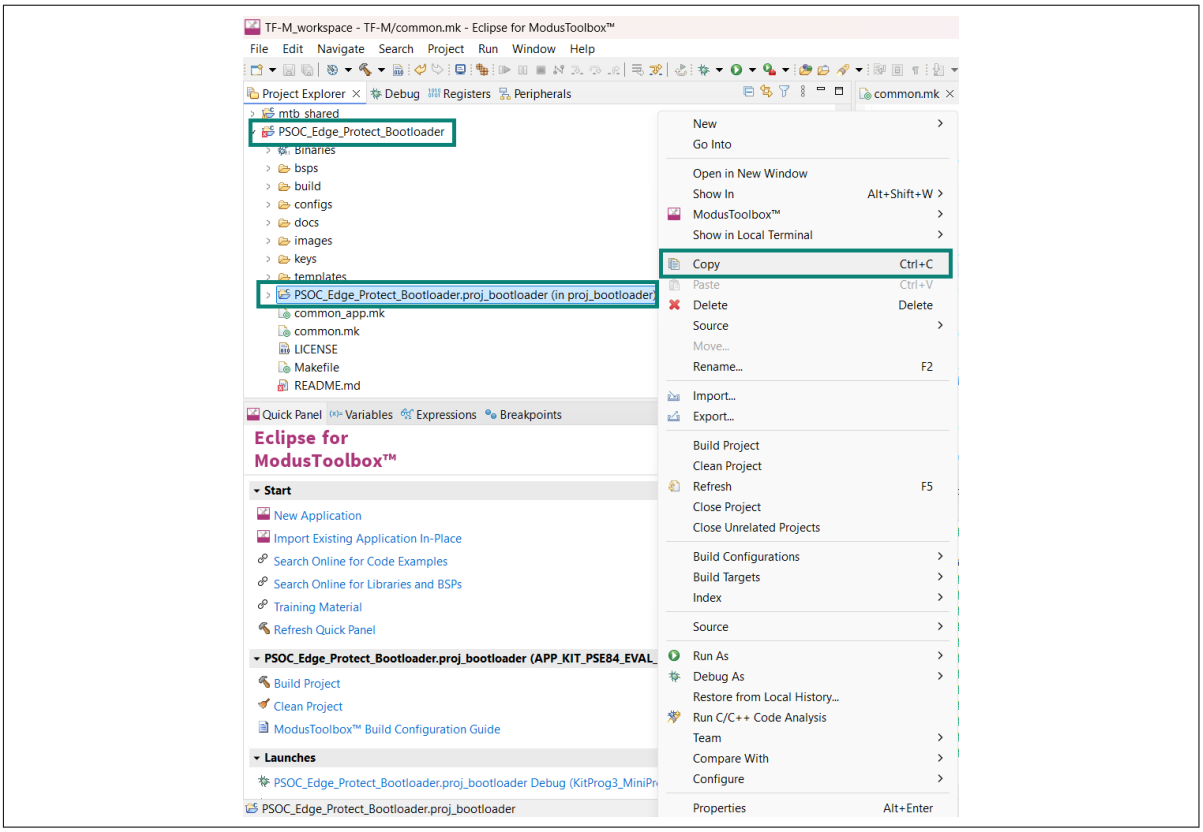

Figure 8 Copy proj_bootloader project

3. Select the TF-M application, press right click and select paste option

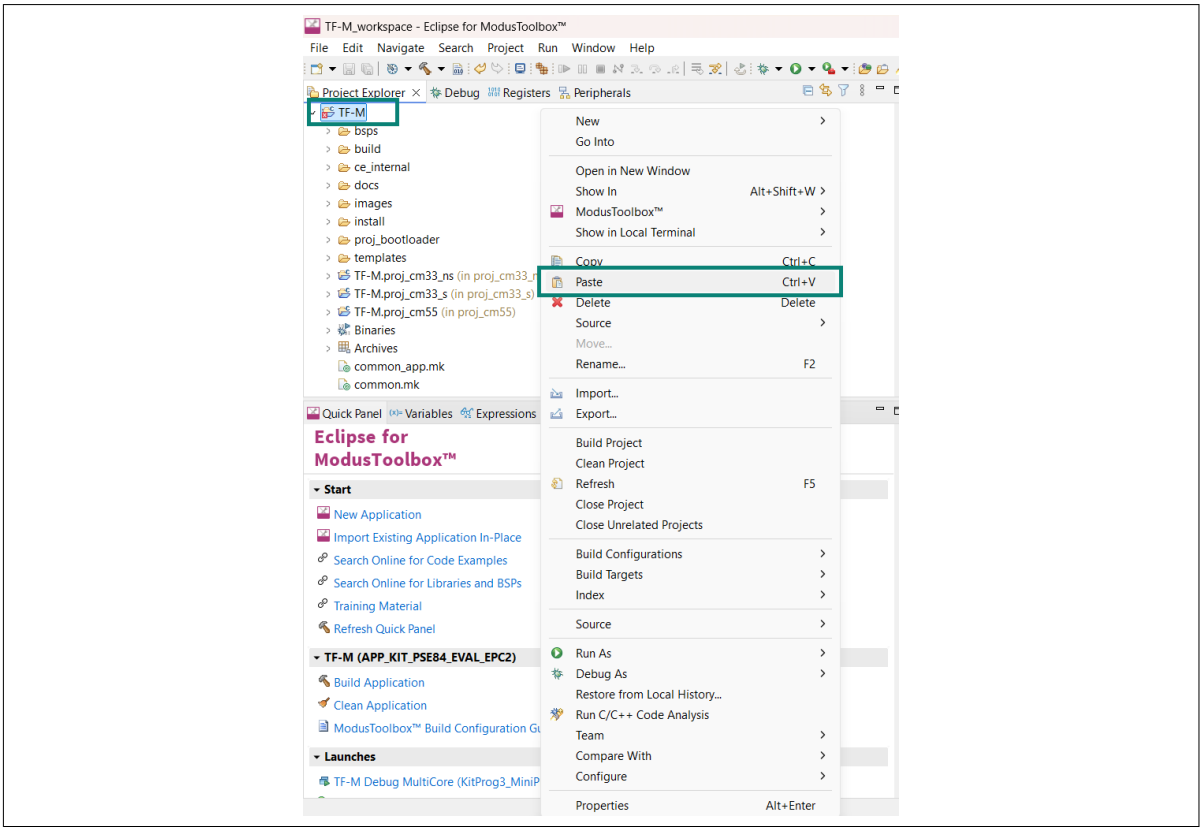

Figure 9 Paste proj_bootloader project

4. During the copy operation a copy project window pops up. Change the project name to `TF-M.proj_bootloader`. Deselect the `Use deafult location` option and choose browse option to set the path to TF-M application. Append \proj_bootloader to the TF-M application path as shown below

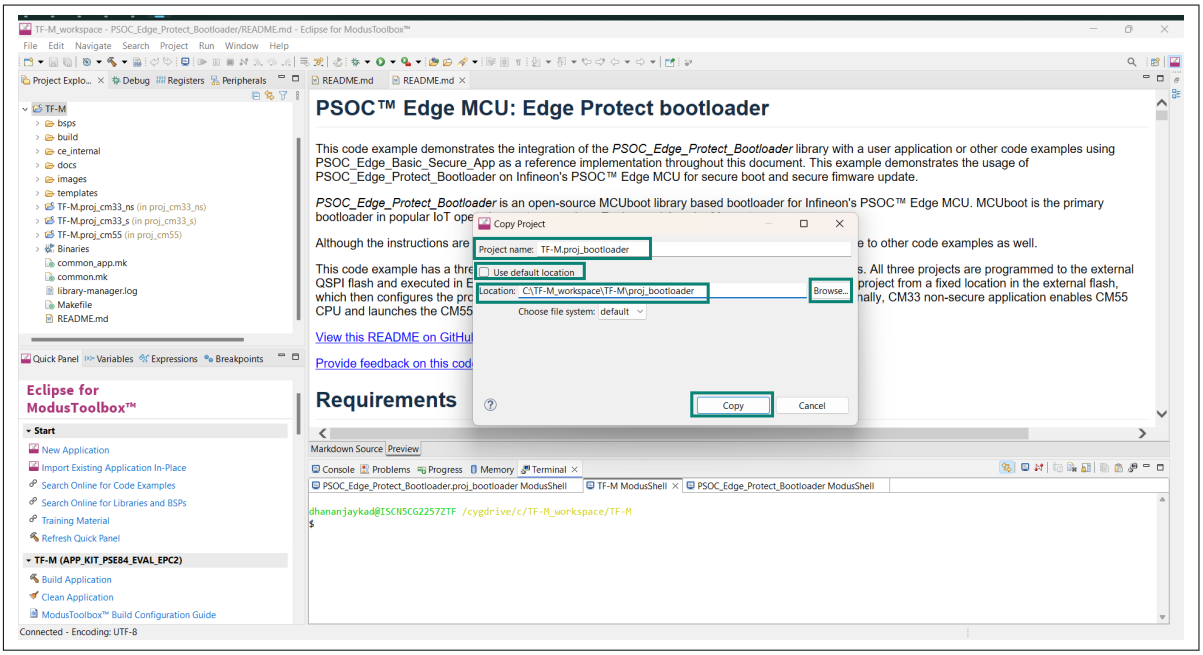

Figure 10 Rename project and change location

5. After successfully copy, the Edge Protect Bootloader project is added to TF-M application. Select the TF-M project, navigate to Quick panel and select `Generate Launches for TF-M`

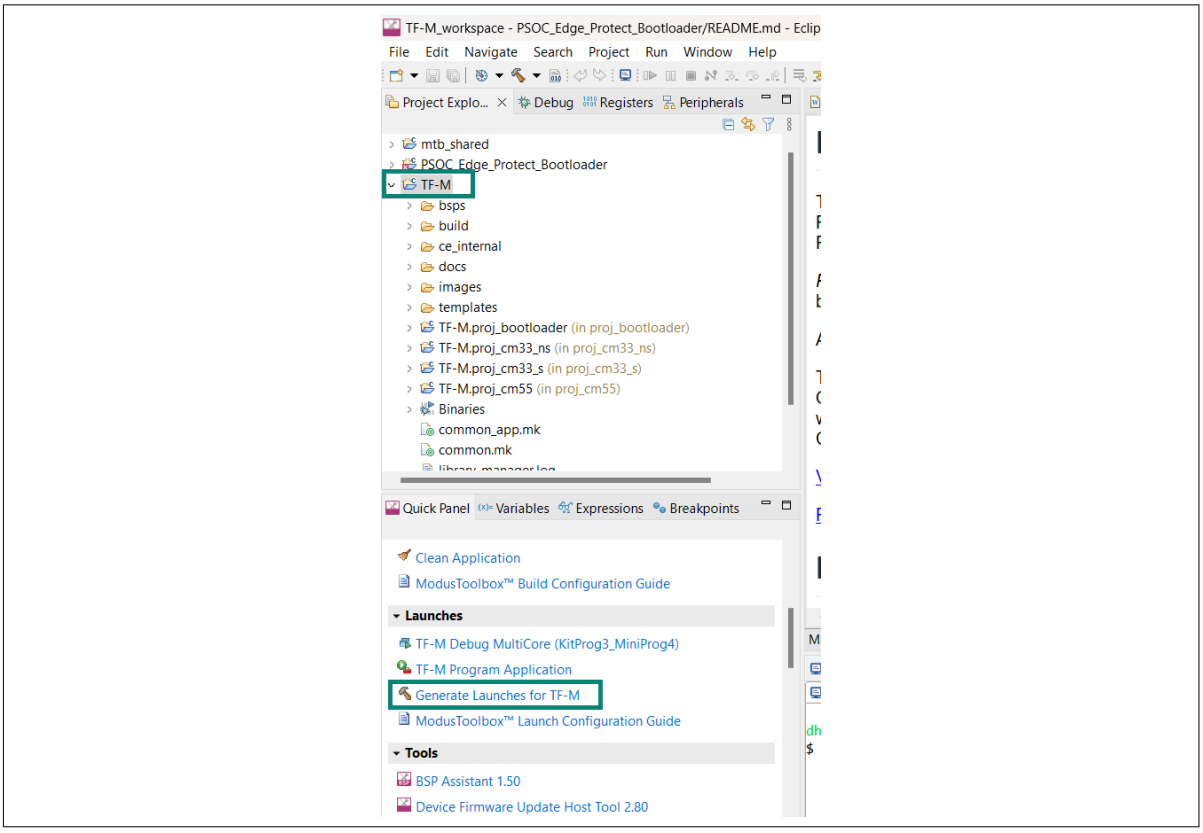

Figure 11 Generate launches

6. Open the Makefile for TF-M application which is located in `<Workspace_Path>\TF-M` directory and add `proj_bootloader` to `MTB_PROJECTS` as shown below

```
MTB_PROJECTS=proj_cm33_s proj_cm33_ns proj_cm55 proj_bootloader
```

7. Open the common.mk file from TF-M application which is located in `<Workspace_Path>\TF-M` directory and update the `COMBINE_SIGN_JSON` to point to signer combiner JSON file which is generated by Device Configurator

```
COMBINE_SIGN_JSON?=./bsps/TARGET_$(TARGET)/config/GeneratedSource/boot_with_bldr.json
```

2.3.3 Build and Program TF-M application

Once the integration of Edge Protect Bootloader and TF-M application is completed, it is ready for build and program operation. Select the TF-M application and click on Program application to build and program TF-M application as shown in below figure.

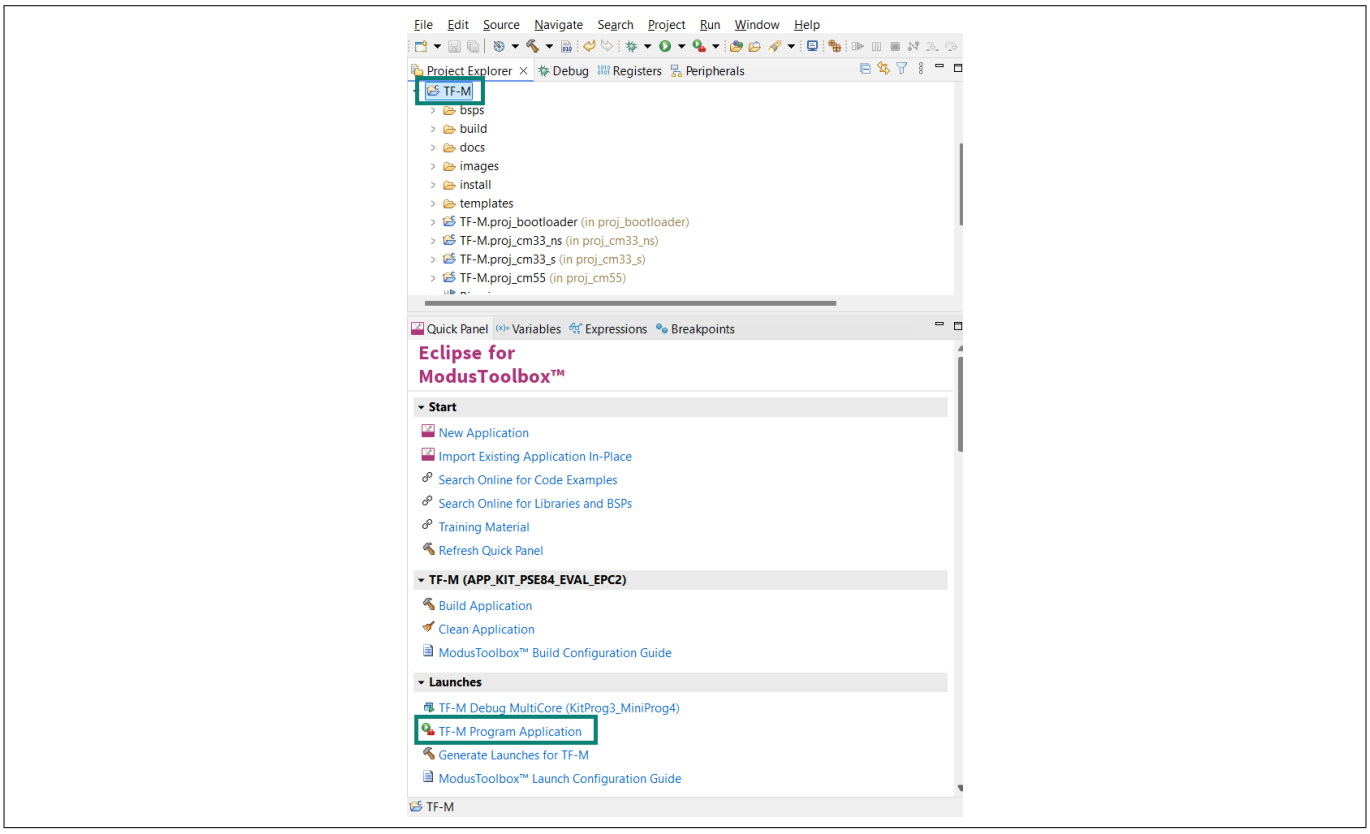

Figure 12 Build and Program TF-M application

After successfully programming the application, the extended boot launches the Edge Protect Bootloader. The Edge Protect Bootloader verifies image magic number of next all three images from the external flash (with the quad SPI interface), and launches the TF-M image. TF-M initializes the SPE and transfers control to M33 NSPE. The M33 NSPE enables M55 NSPE. Upon successful launch, you will see the following logs on the serial terminal as shown in Figure 13:

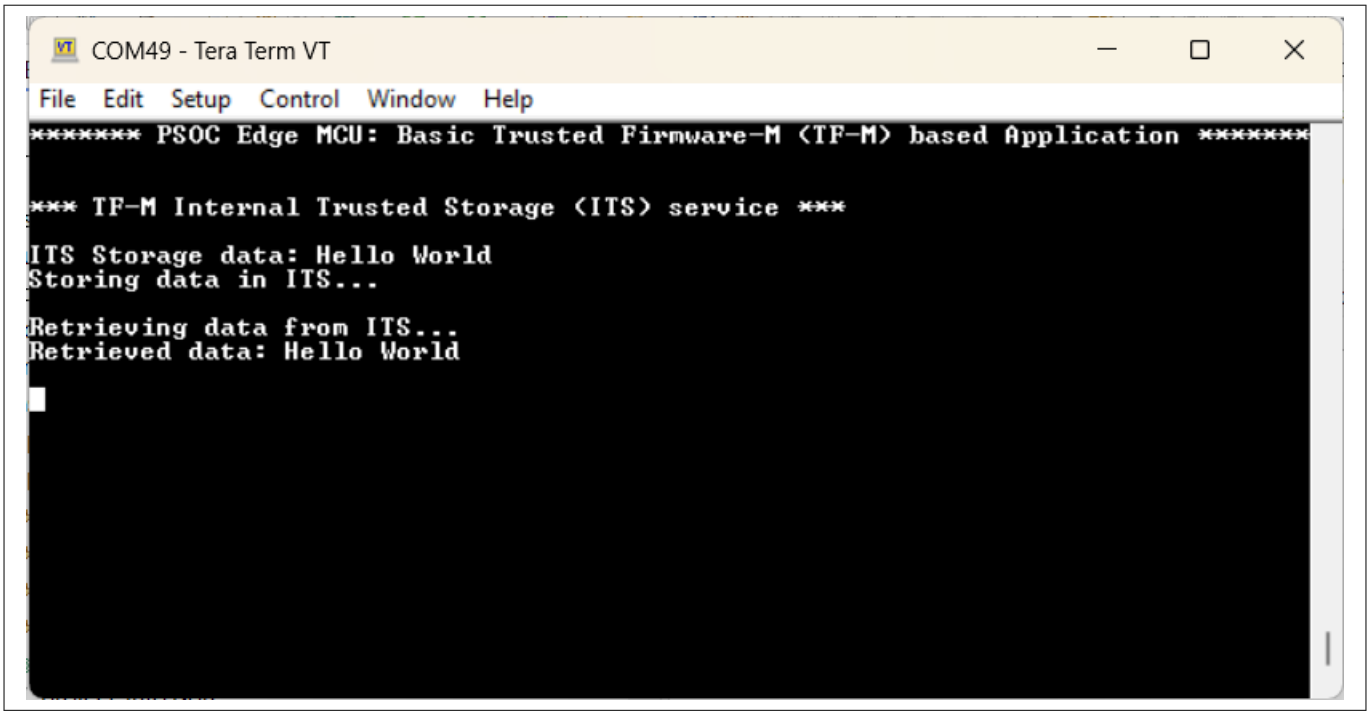

Figure 13 PSOC™ Edge basic TF-M application logs

***Note**: On Windows systems, if the TF-M application path exceeds 32 characters, an error stating `File not found` may occur during TF-M build process. It may be caused by a path length exceeding 260 characters on Windows. To resolve this, shorten the path of workspace and application name.*

2.4 Application structure and build artifacts

2.4.1 Application structure

The PSOC™ Edge basic TF-M application consists of three projects:

- proj_cm33_s
- proj_cm33_ns
- proj_cm55
- proj_bootloader (added separately)

proj_cm33_s: This is a CM33 secure project that contains the TF-M. TF-M is available in source code format and is placed in the `ifx-tf-m-pse84epc2`(EPC2 device) and `ifx-tf-m-pse84epc4`(EPC4 device) library that can be managed using the Library Manager offered by ModusToolbox™. The library is placed in the `<Workspace_Path>/mtb_shared` directory and functions as an independent, self-contained application. It contains all the necessary components and dependencies required to operate independently. The `proj_cm33_s` project directory contains only the essential Makefile and dependencies; it does not contain any source code. Any source code placed in this directory will be excluded from the build process. Instead, ModusToolbox™ compiles the TF-M application provided within the `ifx-tf-m-pse84epc2/ifx-tf-m-pse84epc4` library.

proj_cm33_ns: This is the CM33 non-secure project. This project has the TF-M NSPE interface and can call PSA APIs to request secure services from TF-M. The TF-M interface is available in a separate `ifx-tf-m-ns` library that is placed in `<Workspace_Path>/mtb_shared` directory. The interface must be initialized by CM33 non-secure project before requesting service from TF-M.

proj_cm55: This is the CM55 non-secure project. This project also has TF-M NSPE interface, that is, this project is an off-core NSPE, which can request services from TF-M. The TF-M interface is available in a separate `ifx-tf-m-ns` library and is placed in `<Workspace_Path>/mtb_shared` directory. The interface must be initialized by the CM55 project before requesting service from TF-M.

proj_bootloader: This is the EPB project integrated in TF-M application. This project is launched by extended boot and executes in M33 SPE. The EPB is based on MCUBoot bootloader which is available in `ifx-mcuboot-pse84` library. The EPB authenticates the other three projects and transfers control to TF-M.

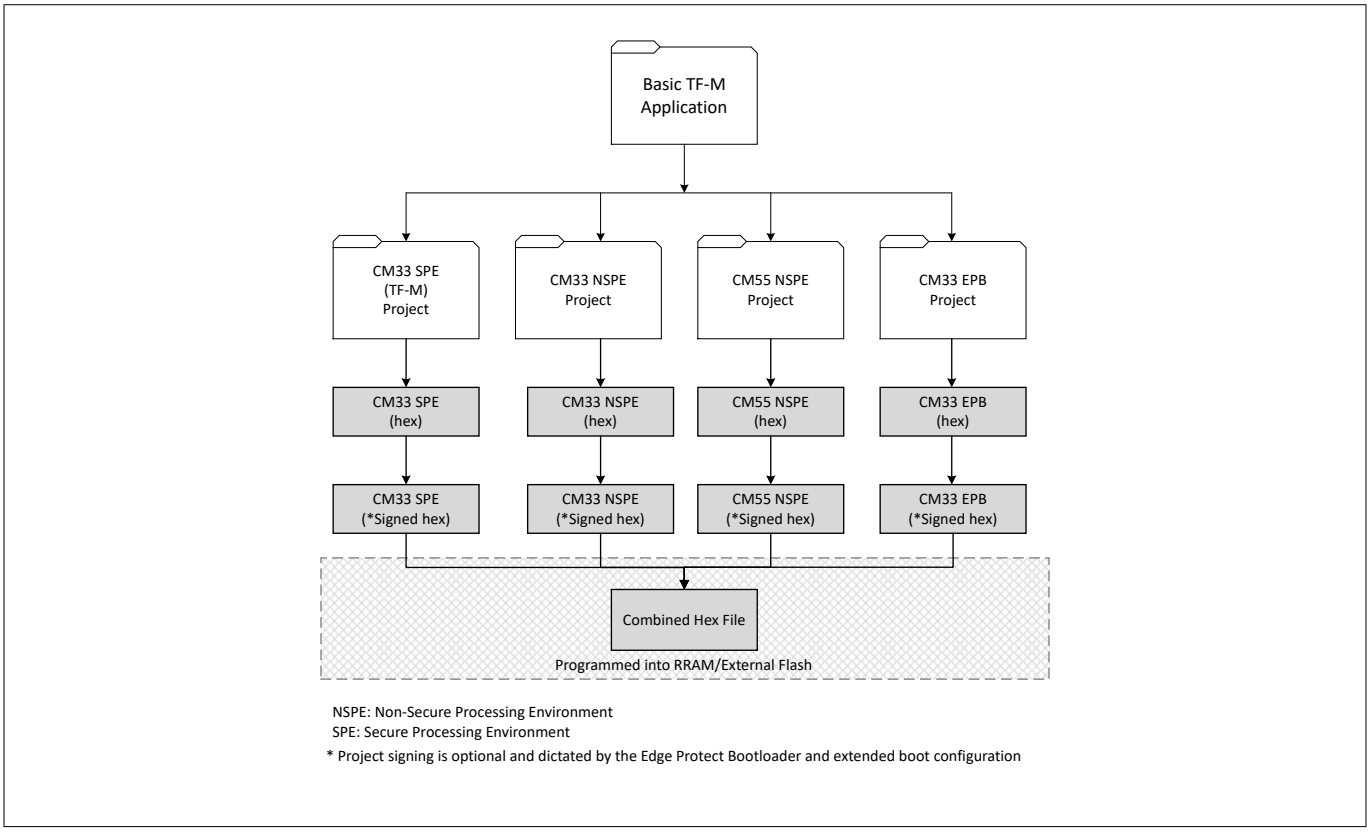

Figure 14 TF-M application structure

2.4.2 Build artifacts

The build artifacts (hex, bin, elf, etc.) of each project of PSOC™ Edge basic TF-M application are placed into their individual project directories.

- **proj_cm33_s:** When building TF-M, the normal ModusToolbox™ application build flow is bypassed. Instead, TF-M uses `CMake` as the build system generator and GNU Make as the build system. The output of `CMake` is placed in the `proj_cm33_s/build/APP_<BSP_NAME>/<CONFIG>` directory. The build system compiles the TF-M source code from the `mtb-shared/ifx-trusted-firmware-m` directory and generates the `tfm_s.hex`, `tfm_s.bin`, `tfm_s.map`, `tfm_s.elf`, and `tfm_s.axf` artifact files and places them in the `proj_cm33_s/build/APP_<BSP_NAME>/<CONFIG>/bin` directory. The object files (.o) and library files (.a) are placed in `proj_cm33_s/build/APP_<BSP_NAME>/<CONFIG>` directory. The `tfm_s.hex` and `tfm_s.elf` artifacts are copied to the `proj_cm33_s/build/APP_<BSP_NAME>/<CONFIG>` directory and renamed as `<APPNAME>.hex` and `<APPNAME>.elf` by TF-M build system.

The TF-M build system also installs the essential non-secure interface files in `<TF-M_app_name>/install`. These files enable a non-secure interface to request secure services from TF-M. As part of the build process, ModusToolbox™ copies the `<APPNAME>.hex` to `<TF-M_app_name>/build/project_hex` directory

- **proj_cm33_ns, proj_cm55 and proj_bootloader:** Like any other ModusToolbox™ application, the artifacts of the build process are placed in their respective `build/APP_<BSP_NAME>/<CONFIG>` directories and the hex files are copied to `<TF-M_app_name>/build/project_hex` directory

After building all four projects, ModusToolbox™ looks for all four `<APPNAME>.hex`, signs the individual images and merges them to create a `app_combined.hex` file, which is the combination of all four projects. The signing and merging of all four projects is governed by `boot_with_bldr.json` that is located in `bsps/TARGET_<BSP_NAME>/config/GeneratedSource` directory. This json file is auto generated based on the memory map configured in Edge Protect Bootloader Solution configuration which is available under Solutions tab in Device Configurator.

2.5 TF-M Logging

TF-M provides an option to log status, exceptions, and faults that are useful to debug the application. By default, TF-M logs are enabled in the code example.

The logs can be disabled by commenting the following line from `<Workspace_Path>/<TF-M_app_name>/proj_cm33_s/Makefile`:

```
#TFM_CONFIGURE_EXT_OPTIONS+= -DTFM_EXCEPTION_INFO_DUMP=ON -DPLATFORM_EXCEPTION_INFO=ON
-DIFX_FAULTS_INFO_DUMP=ON -DTFM_SPM_LOG_LEVEL=TFM_SPM_LOG_LEVEL_DEBUG
-DTFM_PARTITION_LOG_LEVEL=TFM_PARTITION_LOG_LEVEL_DEBUG
```

When the logs are enabled, TF-M logs all the Secure Partition Manager (SPM) and all other partition's status, faults, and exceptions. TF-M dumps the exception stack frame, type of fault, the bus master that caused the fault, etc through UART. For example, if the M33 NSPE tries to change the protection context (PC) of the M33 CPU, it leads to PPC fault as shown in Figure 15.

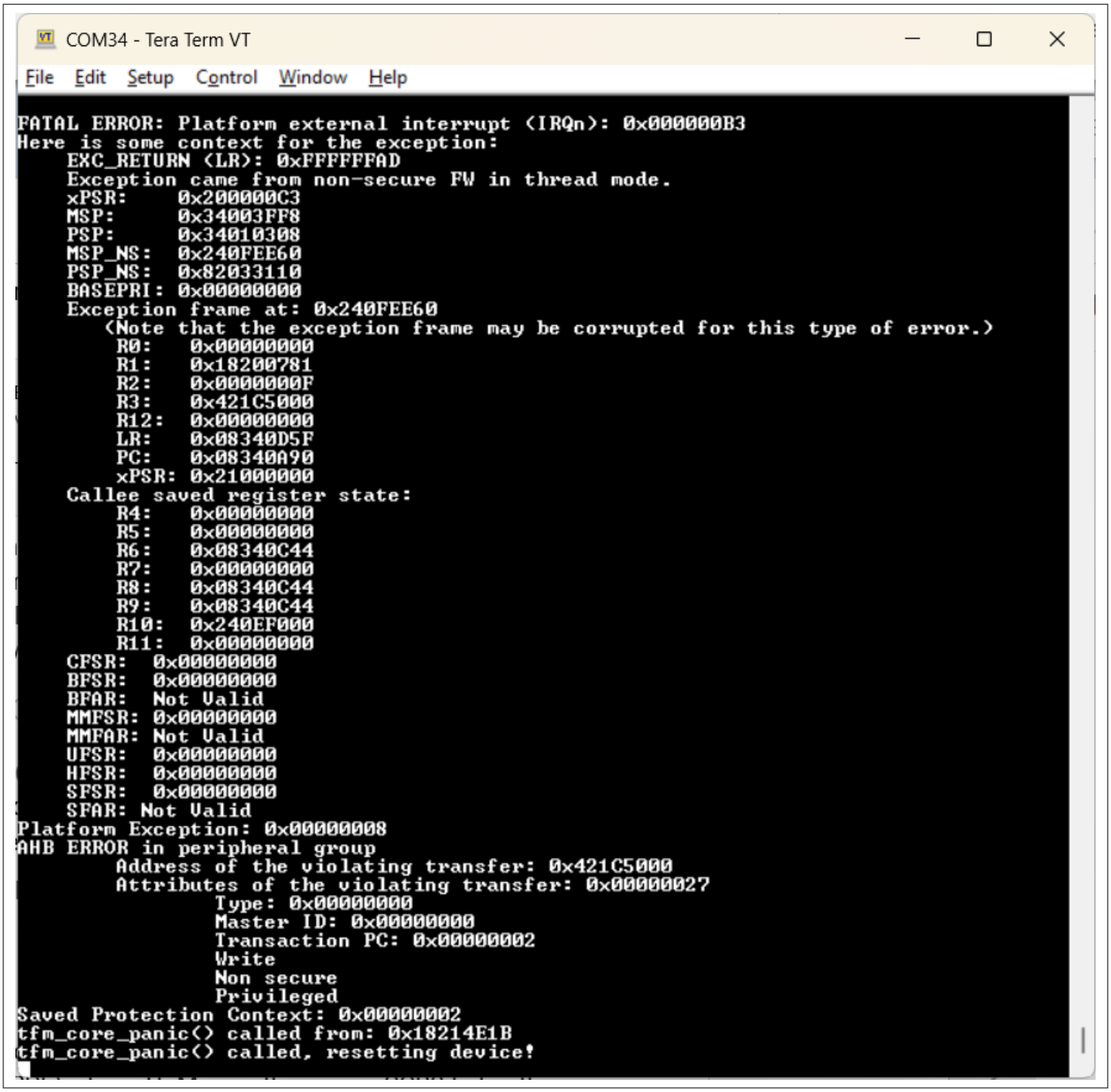

Figure 15 TF-M fault and exception logging

Serial Communication Block 2 (SCB2) UART lines are connected to KitProg3 on the PSOC™ Edge Evaluation Kit; therefore, it is used for logging by TF-M. When logging is enabled in TF-M, SCB2 must be assigned to M33 secure (M33S) domain, that is, it is marked as secure and cannot be accessed directly by NSPEs. This configuration is set from Peripheral Protection Controller (PPC) configuration, available under System tab in Device Configurator. It is possible to use some other instance of SCB for logging. To enable logging in such cases:

1. Enable the desired SCB instance with the `IFX_TFM_SPM_UART` alias and set the personality to UART. Configure the clock and Rx/Tx lines accordingly
2. Update the peripheral protection controller (PPC) protection settings of the new SCB instance by assigning it to M33S domain through PPC configuration in device configurator. Ensure that Rx/Tx pins have secure attribute set to Secure access

It is important to note that the secure SCB cannot be accessed by NSPE for logging or communication. If accessed, it will result in a fault. To prevent such faults, users can choose any of the following options:

- **Use platform service (for logging):** TF-M offers a platform service for message logging. The logging service transfers the message from NSPE to TF-M (SPE) where TF-M uses the secure SCB2 to log the messages. This is the default logging method used in the TF-M code example. See the Platform service section for instruction on how to use the platform partition's logging service
- **Use other SCB (for logging and communication):** Use any other SCB than secure SCB for NSPE logging or communication. Configure the new SCB with desired personality (UART/I2C/SPI) along with its associated Rx/Tx pins. The SCB must be assigned to non-secure domain (M33_M55 domain) with PPC configuration (available under System tab in Device Configurator) and the pin's secure attribute must be set as non-secure access to make it accessible to NSPEs

See ModusToolbox™ Device Configurator user guide for more information about PPC configurations in device configurator.

3 Add TF-M to an existing application

TF-M can be added to an existing application using Library Manager provided by ModusToolbox™. Adding TF-M to an existing project involves the following steps:

- Add TF-M libraries to the application
- Update application's memory and protection configuration with Edge Protect Configurator
- Integrate Edge Protect Bootloader in the application
- Add and Initialize the TF-M interface in NSPE

***Note**: After adding TF-M library, all source code placed in `proj_cm33_s` directory will be ignored by the build process. Instead the TF-M source code available in `ifx-tf-m-pse84epc2` or `ifx-tf-m-pse84epc4` library is built.*

3.1 Add TF-M libraries to application

TF-M source code is available in the in `ifx-tf-m-pse84epc2` (for EPC2 device) and `ifx-tf-m-pse84epc4` (for EPC4 device) library and the NSPE interface is available in `ifx-tf-m-ns` library. The following steps show how to add TF-M to PSOC™ Edge Hello World code example, but the same steps are applicable to other applications.

1. Select your application and open Library Manager.

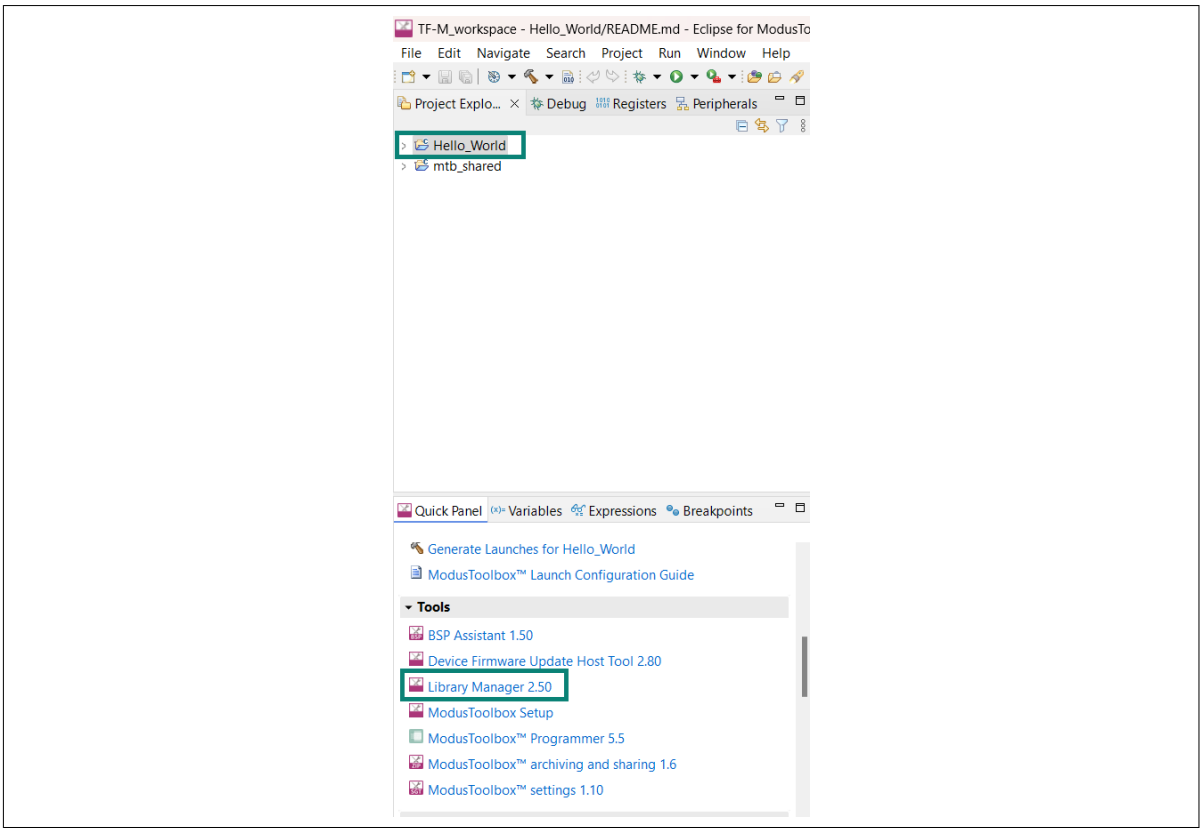

Figure 16 Open Library Manager

2. Add `ifx-tf-m-pse84epc2` or `ifx-tf-m-pse84epc4` library to `proj_cm33_s` project as per target device

3 Add TF-M to an existing application

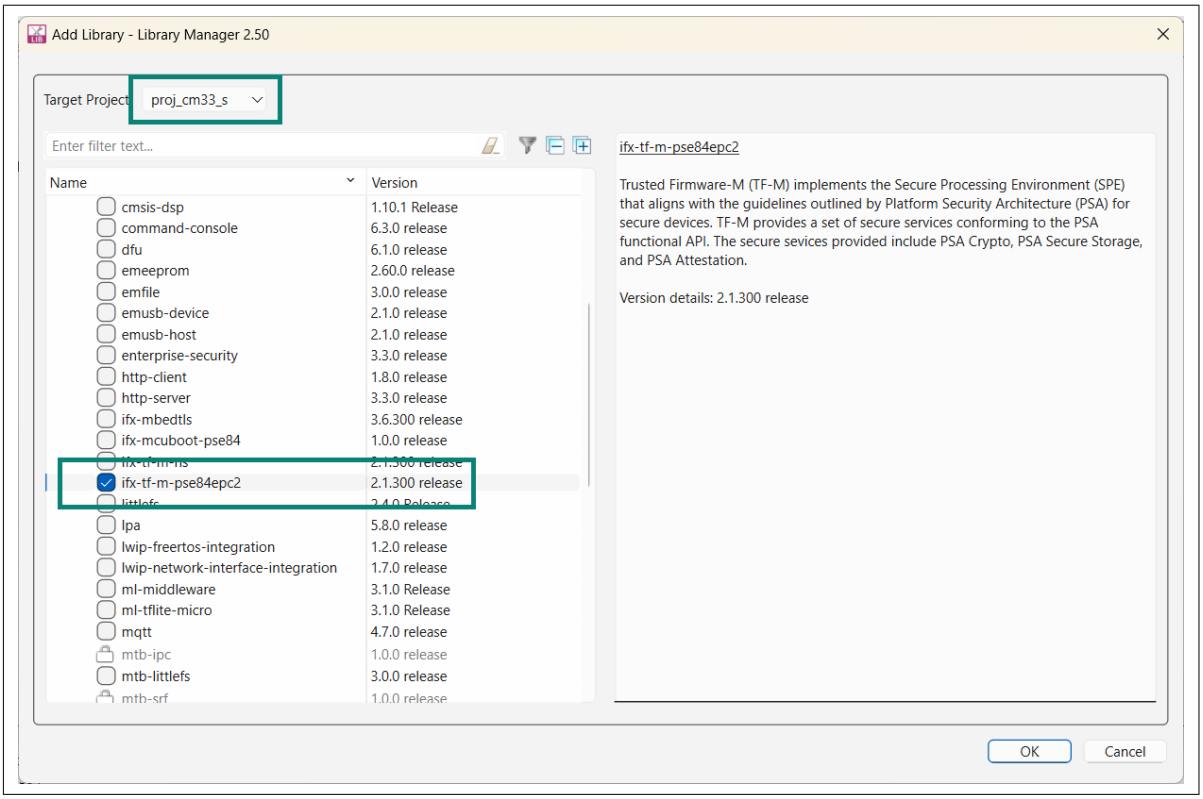

Figure 17 Add TF-M library

3. Add TF-M interface library to `proj_cm33_ns` project (M33 NSPE)

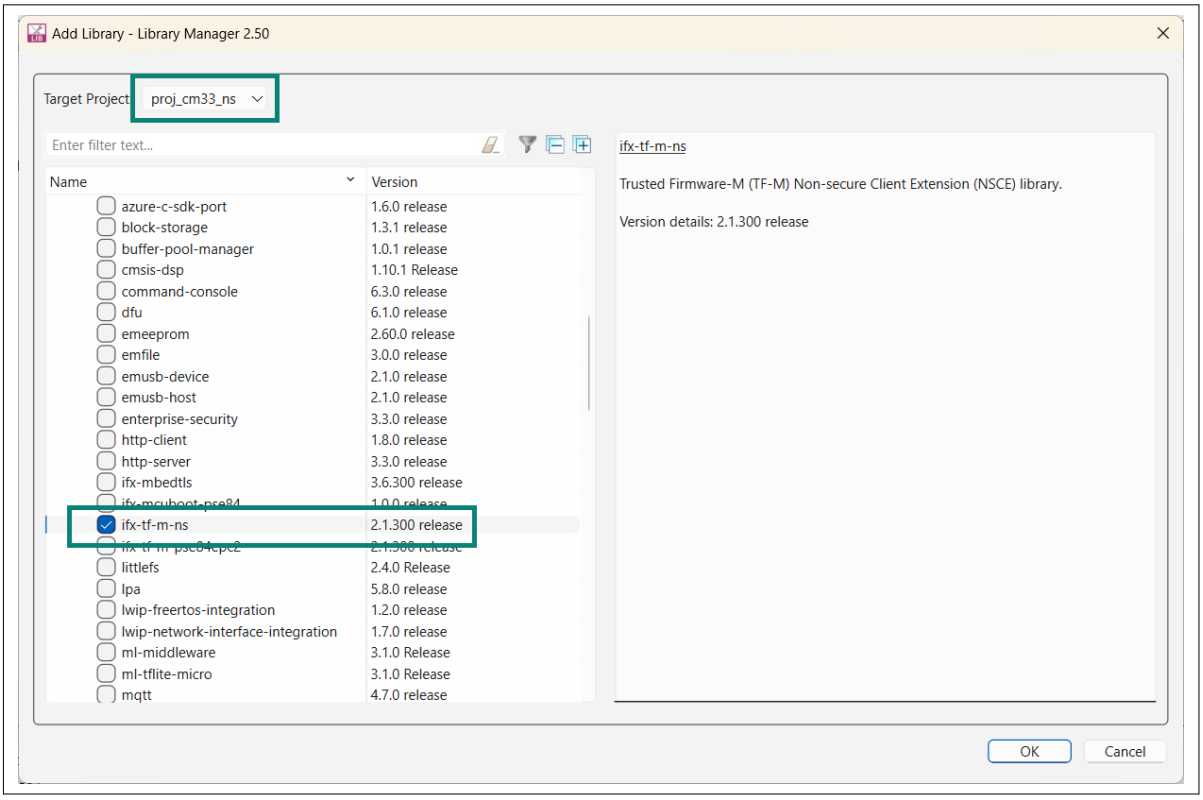

Figure 18 Add TF-M interface to M33 NSPE

4. Add TF-M interface library to `proj_cm55` project (M55 NSPE). This is an optional step required only if M55 NSPE uses TF-M services or security aware PDL

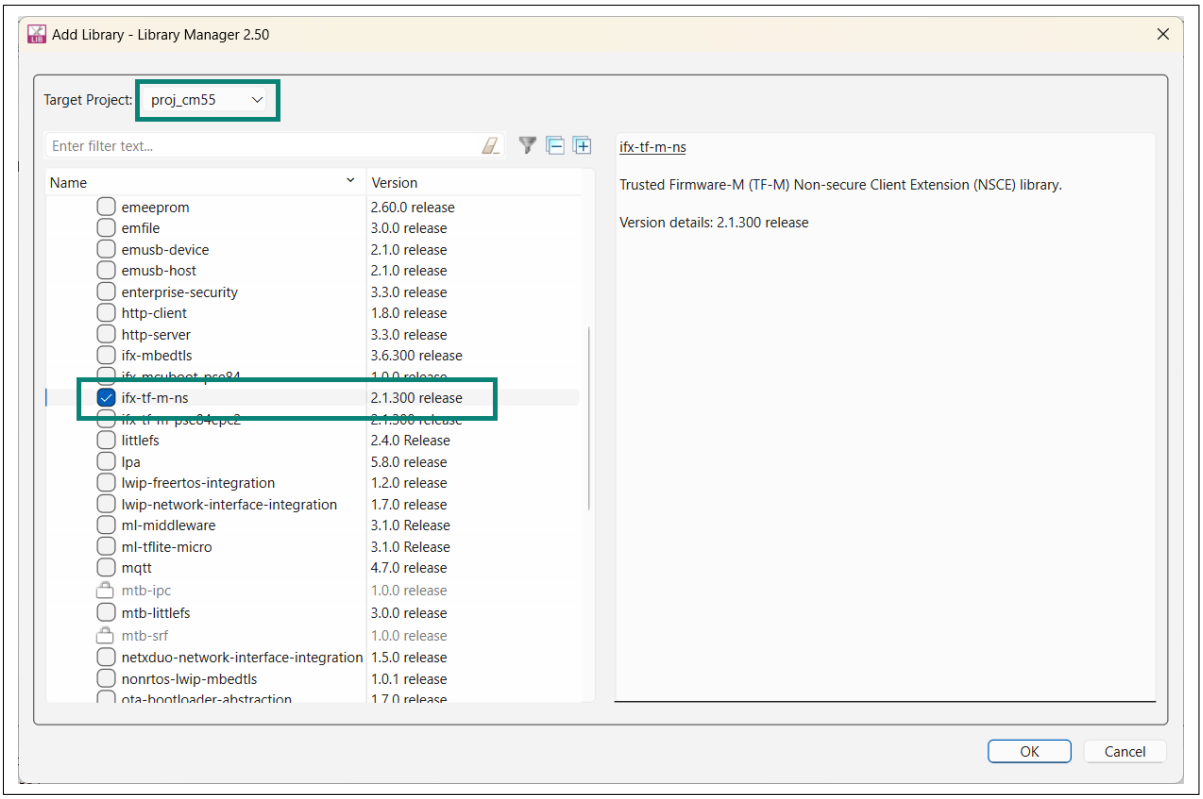

Figure 19 Add TF-M interface to M55 NSPE

5. Update the application after adding the necessary libraries

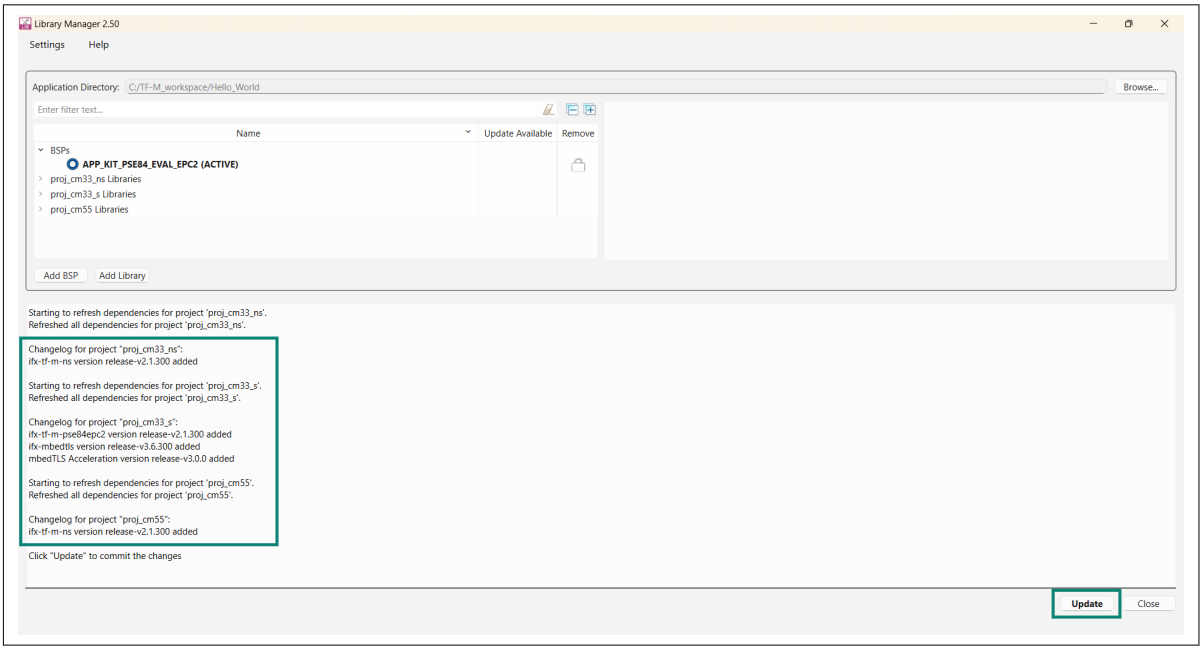

Figure 20 Update application libraries

3.2 Update memory and protection configuration

After adding TF-M libraries, the next step is to add the necessary memory regions and update the protection configuration. This can be achieved with Edge Protect Configurator.

1. Select your application and open Edge Protect Configurator

3 Add TF-M to an existing application

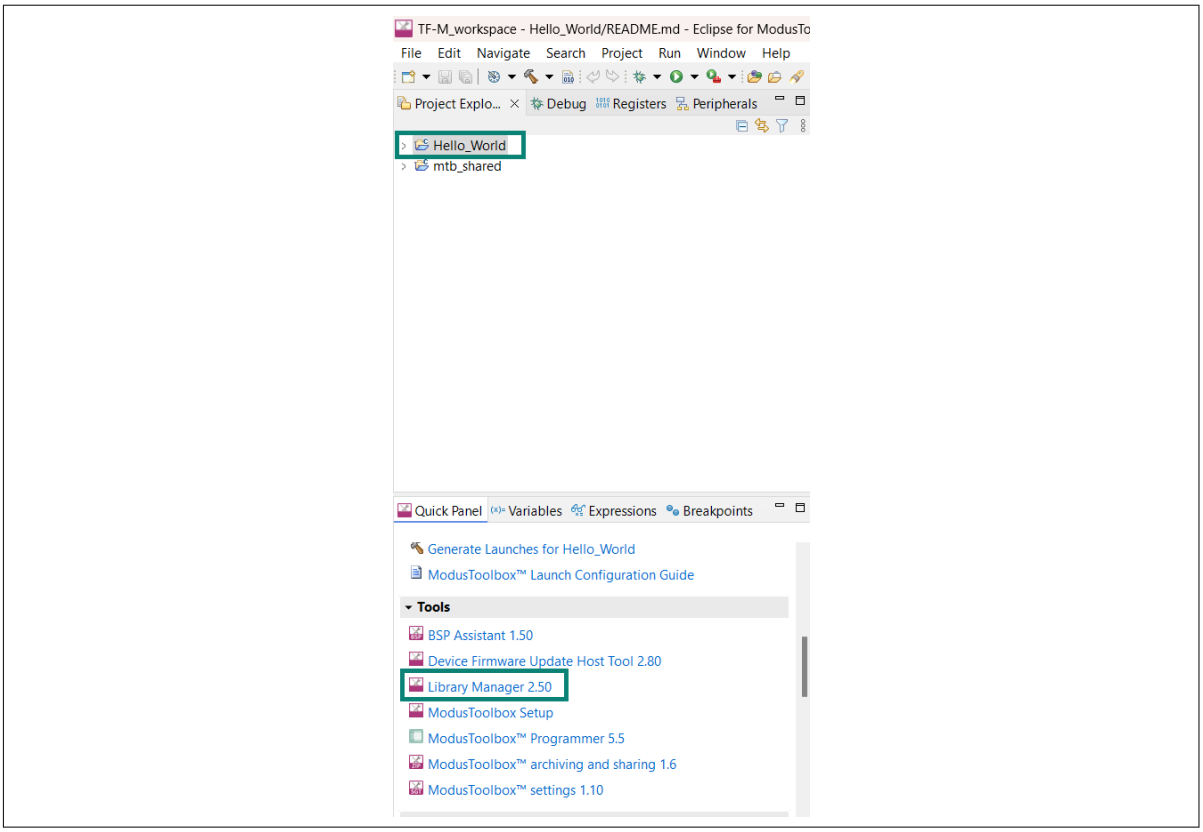

Figure 21 Open Edge Protect Configurator

2. After opening the Edge Protect Configurator, the Edge Protect Configurator provides an option to add the default memory regions and update security configurations for TF-M. Select yes to update the application with these changes

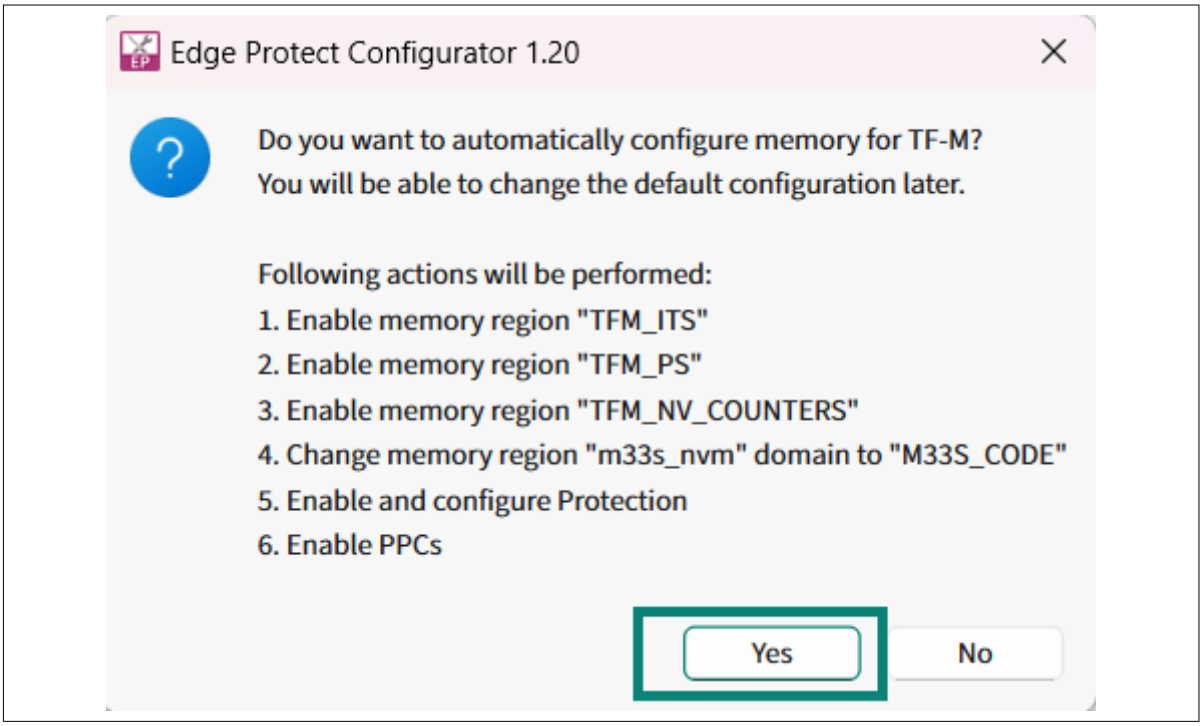

Figure 22 Update application with Edge Protect Configurator

3. Optionally you can select which TF-M profile, isolation level and partitions will be enabled. Additionally, user defined Application RoT partitions can be created from Edge Protect Configurator

3 Add TF-M to an existing application

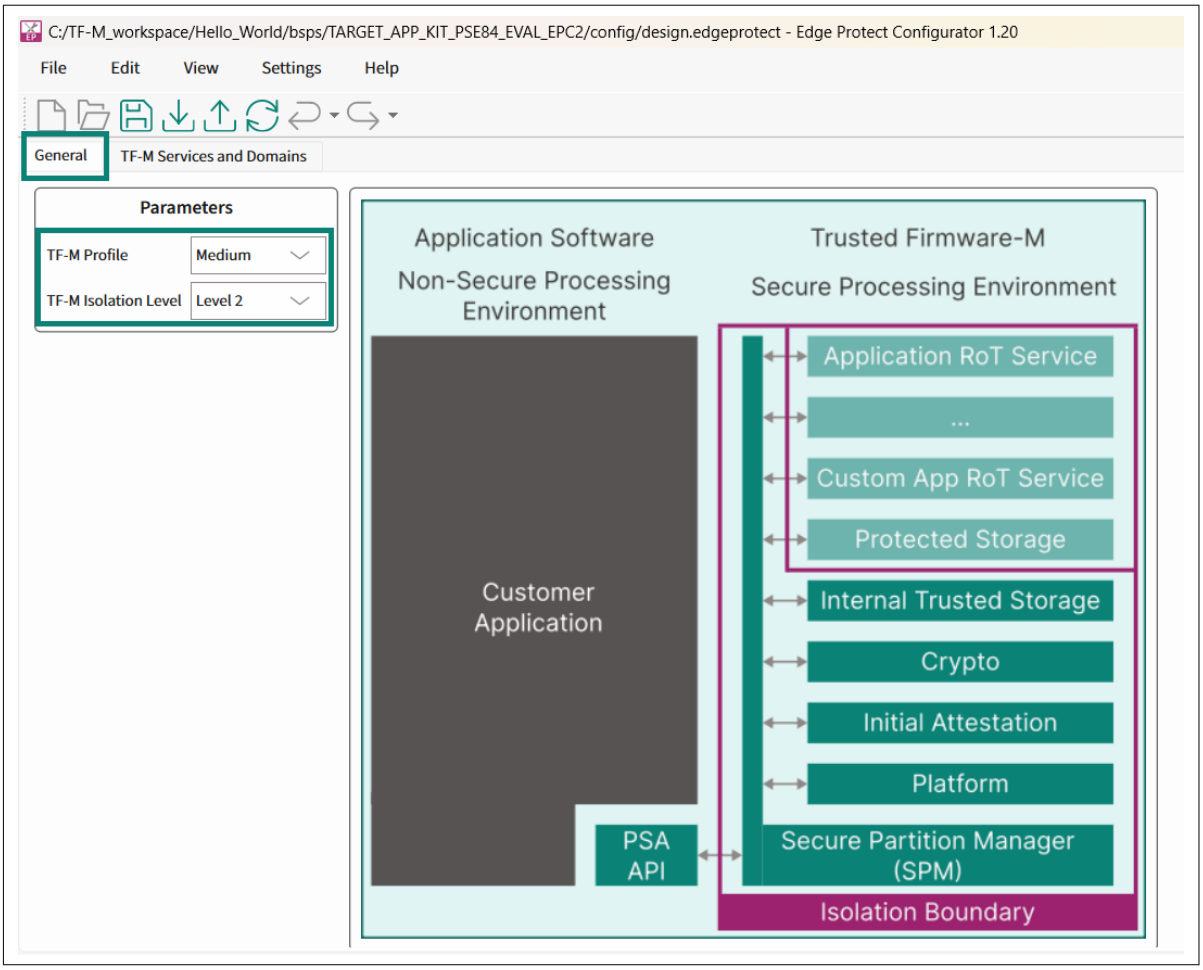

Figure 23 Change TF-M Profile and Isolation Level

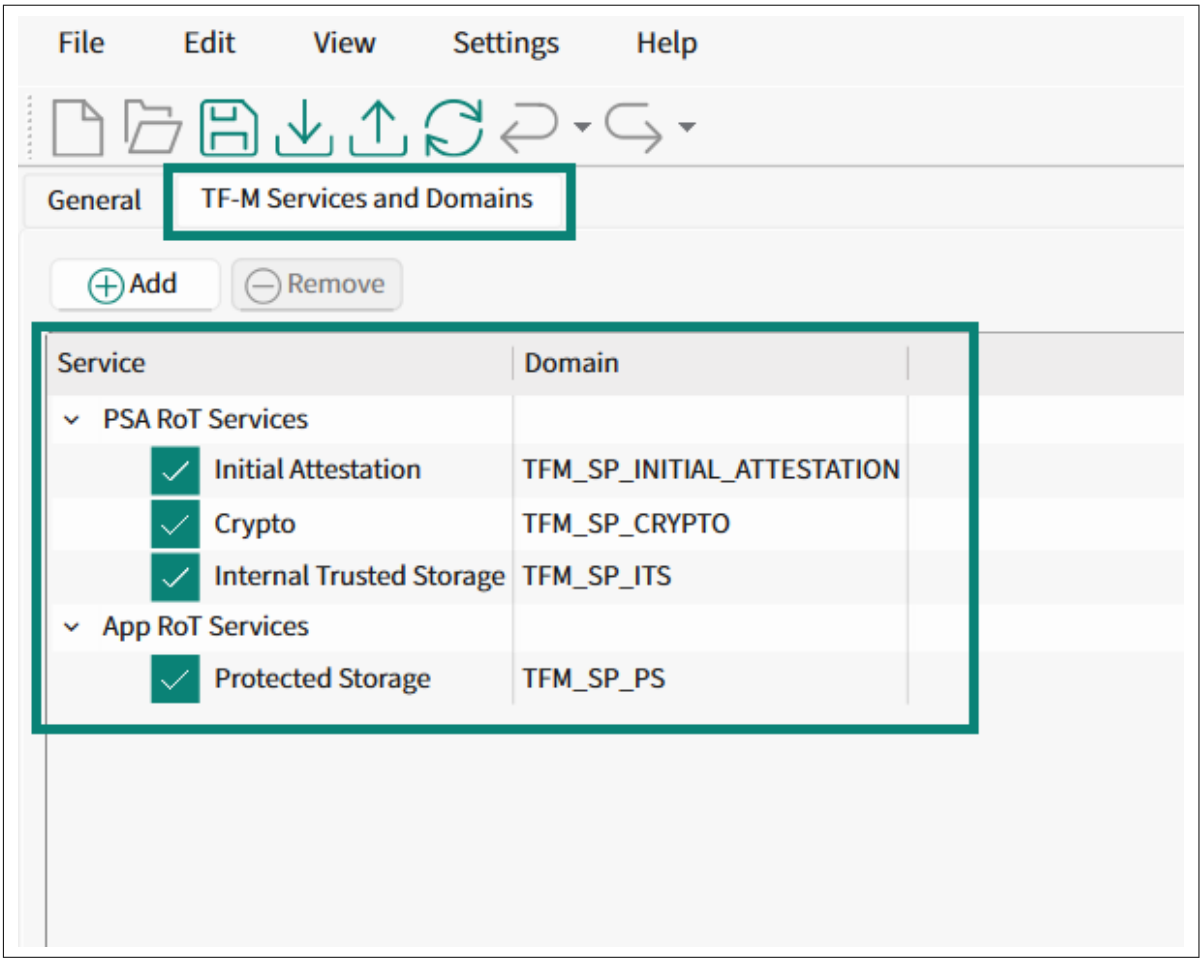

Figure 24 Enable/Disable TF-M Partitions

3.3 Integrate Edge Protect Bootloader

This step is optional and only required if the Edge Protect Bootloader is not already integrated into the application. To add Edge Protect Bootloader:

1. Create Edge Protect Bootloader application and import it as a project in your application as described in Add Edge Protect Bootloader to TF-M application
2. Add `ifx-mcuboot-pse84` to `proj_cm33_s` project
3. Open Device Configurator, navigate to Solution tab and enable Edge Protect Bootloader Solution. Select the `Add memory region for "bootloader_nvm"` to add the region. This is a 160KB region and is associated with M33S domain. Ensure that it is placed at 0x00011000 offset in RRAM. This is the location from which extended boot launches the next application when boot switch is low. The

3 Add TF-M to an existing application

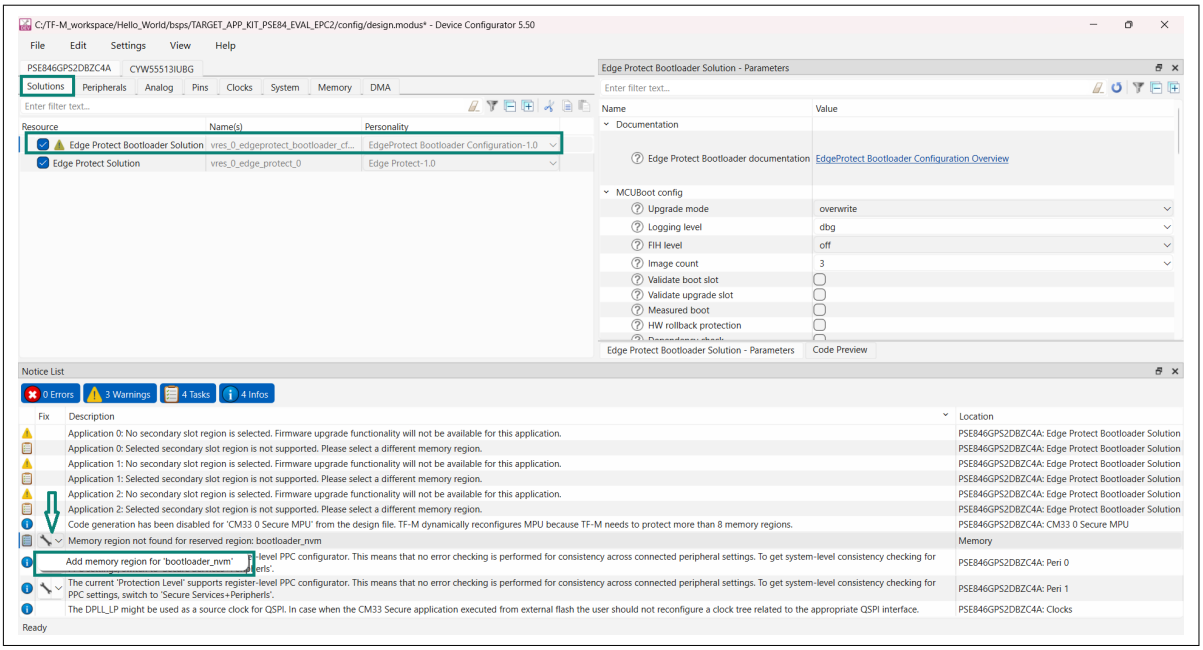

Figure 25 Add Edge Protect Bootloader NVM memory region

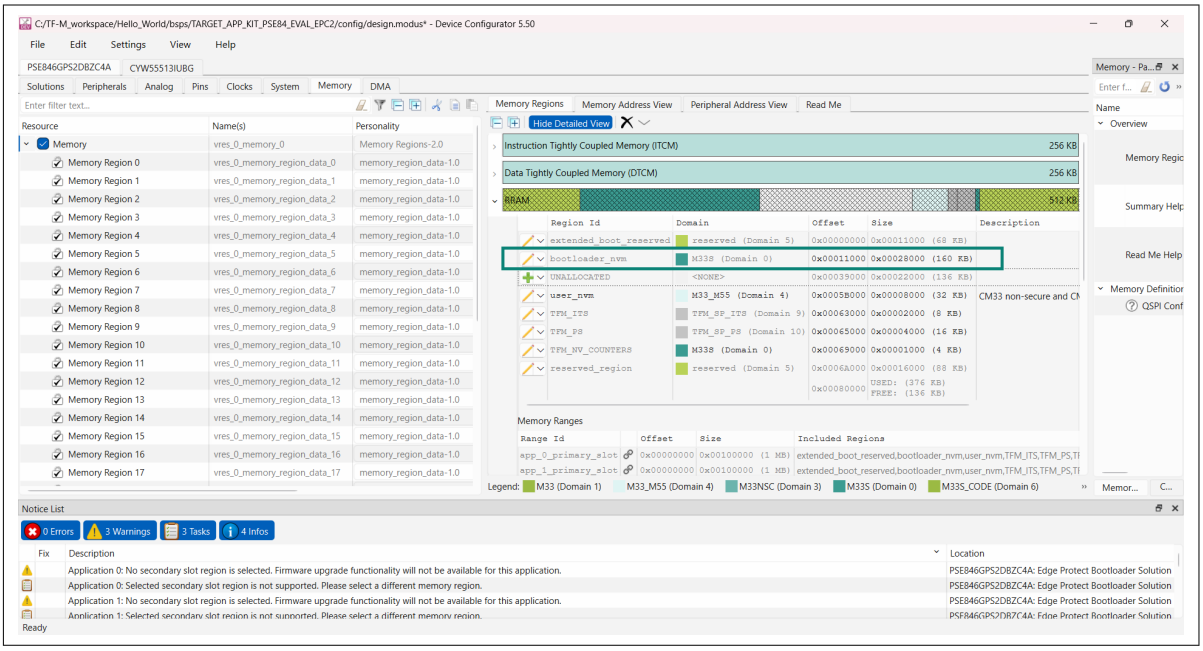

Figure 26 Place bootload_nvm region in RRAM

4. Open Device Configurator, navigate to Memory tab and create secondary slot memory regions for image update. The secondary slots should be placed in external flash memory and should have the same size as their corresponding primary slots

3 Add TF-M to an existing application

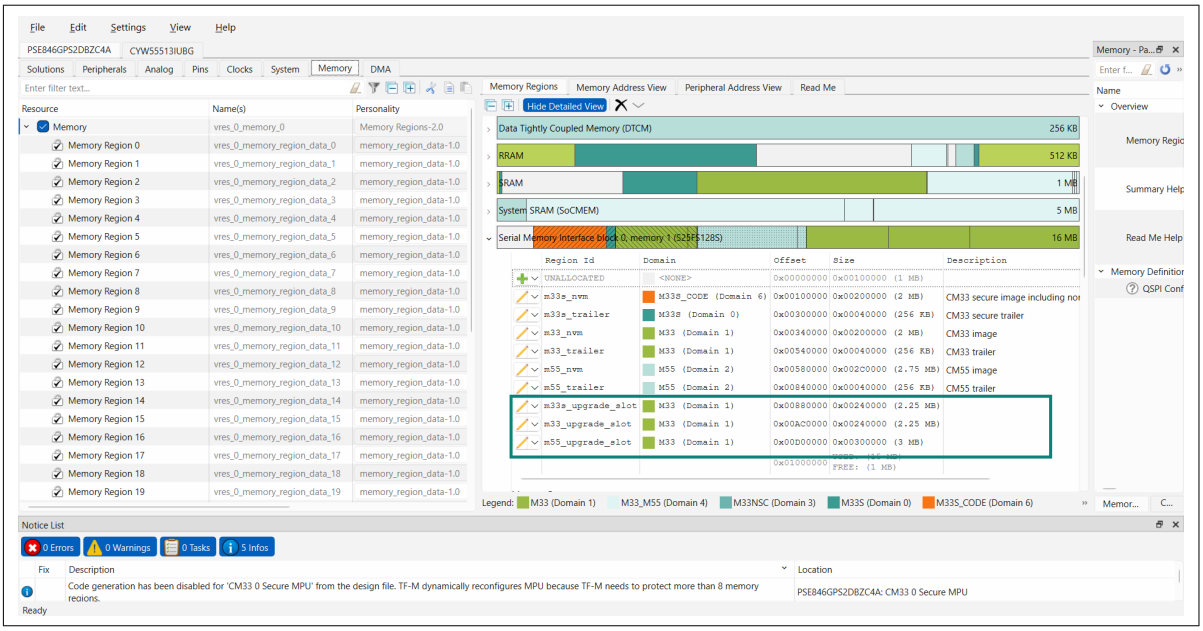

Figure 27 Add secondary slots

5. Navigate back to Edge Protect Solution tab and update slot information for all three images

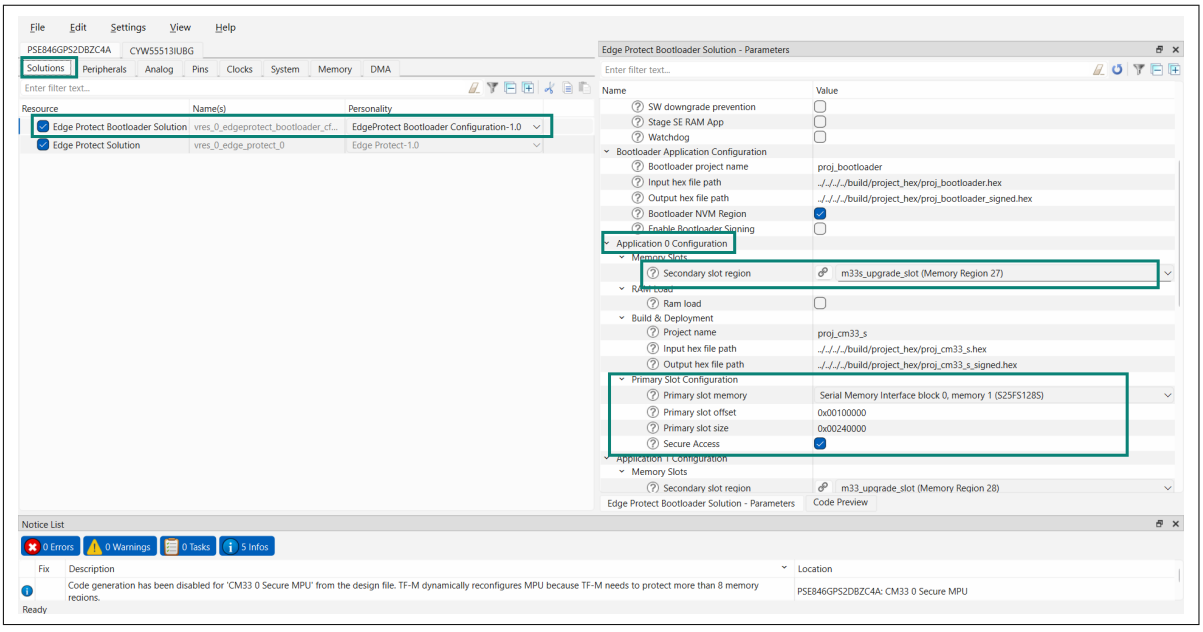

Figure 28 Application 0 configuration

3 Add TF-M to an existing application

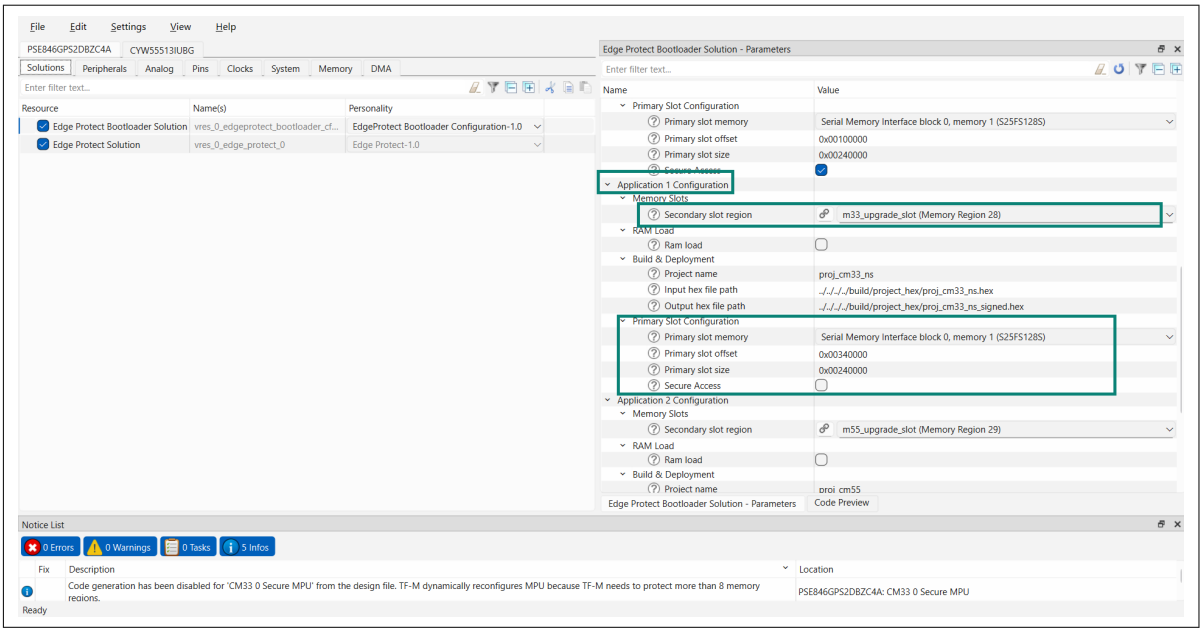

Figure 29 Application 1 configuration

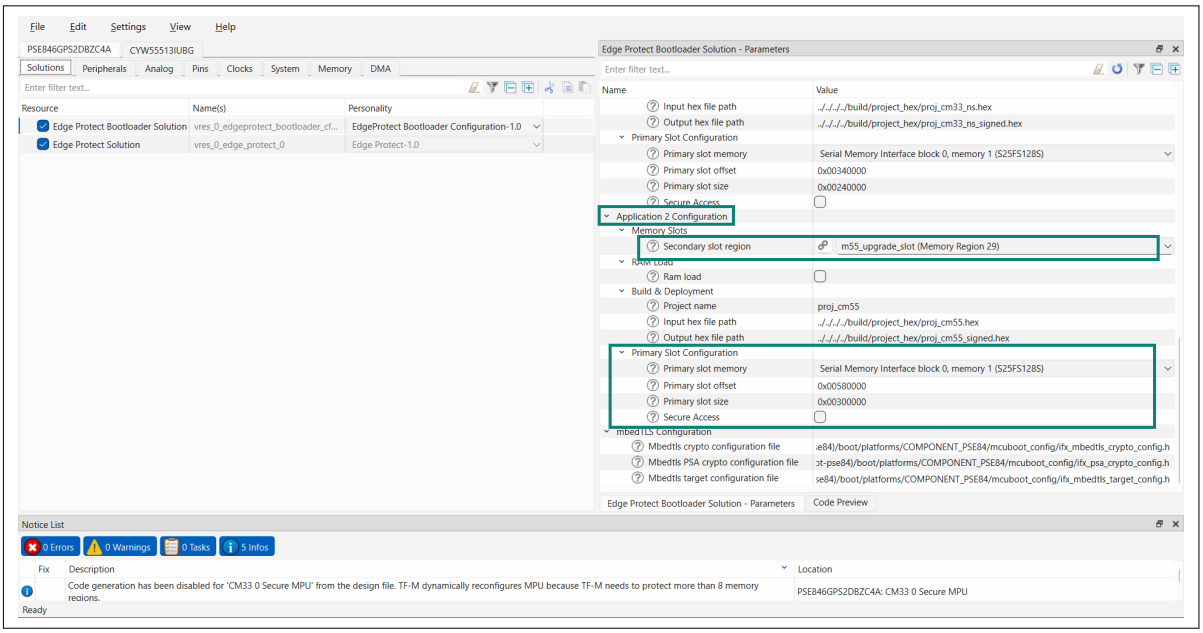

Figure 30 Application 2 configuration

6. Save the changes to generate required files for the application and close the Device Configurator

3.4 Add TF-M interface to use TF-M services

To use the secure services provided by TF-M, the interface must be initialized to enable communication between the NSPE and SPE. The initialization function for the NSPE interface, `tfm_ns_interface_init()`, must be called by the CM33 NSPE. This can be achieved by including the `tfm_ns_interface.h` and `os_wrapper/`

`common.h` header files in the M33 NSPE code. The below code snippet shows how to initialize the TF-M interface on the M33 NSPE. Once the Interface is initialized, both M33 NSPE and M55 NSPE can use TF-M services.

```
#include "tfm_ns_interface.h"
#include "os_wrapper/common.h"


int main(void)
{
    cy_rslt_t result;
    uint32_t rslt;

    /* Device initialization */
    result = cybsp_init();
    ...
    ...

    /* Initialize TF-M interface */
    rslt = tfm_ns_interface_init();
    if(rslt != OS_WRAPPER_SUCCESS)
    {
        CY_ASSERT(0);
    }

    /* User application */
    ...
    ...
}
```

***Note**: The security aware PDL is integrated as part of Platform partition. To use these PDL APIs in your application, TF-M interface must be first initialized.*

4 TF-M architecture

Figure 31 and Figure 32 provide a high-level view of the TF-M for PSOC™ Edge devices. They show where each component resides, runs, and how it interfaces with other components.

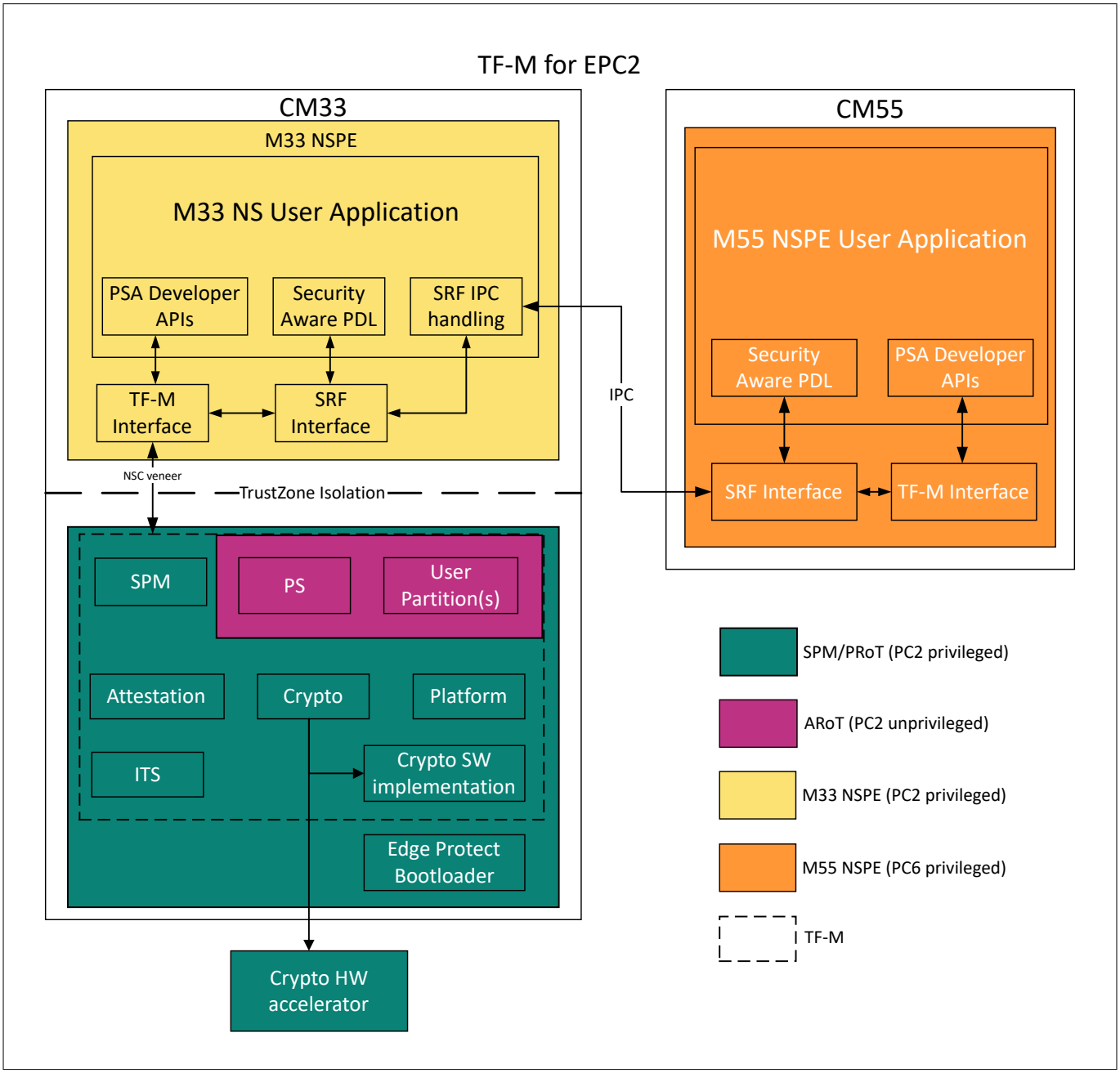

Figure 31 PSOC™ Edge EPC2 TF-M architecture

4 TF-M architecture

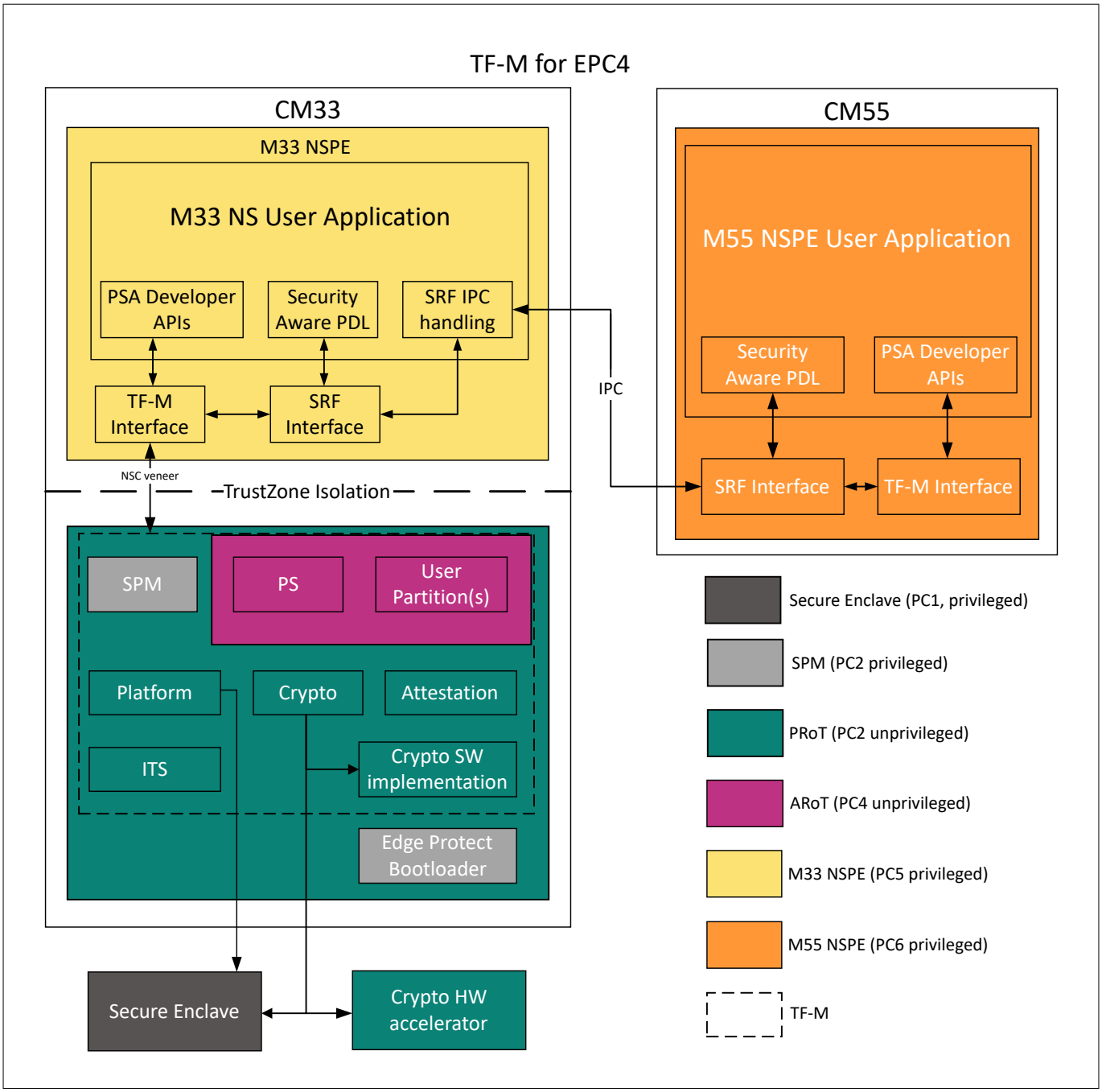

Figure 32 PSOC™ Edge EPC4 TF-M architecture

PSOC™ Edge is a CM33 + CM55 dual-core MCU. The CM33 implements the TrustZone® extension, while CM55 does not have TrustZone® implemented and therefore runs in non-secure (NS) mode. TF-M runs in the SPE on the CM33 CPU. The SPE is further partitioned into two domains: PSA Root of Trust (RoT) and the application RoT.

The PSA RoT consists of the following components:

- Edge Protect Bootloader (not part of TF-M)
- TF-M Secure Partition Manager (SPM)
- TF-M PSA RoT partitions
  - Crypto
  - FWU
  - Platform

- Attestation
- ITS

The application RoT domain houses the following:

- Protected storage partition
- User-defined partition (optional): This is where user-defined services can be implemented in a new partition as part of the application RoT

These partitions can be enabled/disabled from Edge Protect Configurator as per application needs. The Edge Protect Configurator also provides an option to create new user defined application RoT partitions. See Edge Protect Configurator user guide for more information.

***Note**: Firmware update partition is not supported with Infineon's port of TF-M. The non-secure application must write the upgrade images in secondary slots using Security aware SMIF Peripheral Driver Library (PDL) and the EPB shall perform the swap or overwrite operation to upgrade the image.*

The SPM and secure services are implemented in separate partitions. The partitions execute in privilege levels as described in Table 1 and Table 2. The SPM is a core component of TF-M which manages the isolation boundaries between these partitions. SPM sets up the execution environment for other secure partitions and NSPEs then transfers control to M33 NSPE. When NSPE requests for secure service, SPM manages the state transitions and schedules the execution of pending requests from NSPEs.

Infineon also provides Secure Request Framework (SRF) to facilitates secure access to certain system resources from the NSPE with security aware Peripheral Driver Library (PDL). In the Infineon port of TF-M, the SRF is seamlessly integrated into TF-M interface. Out of the box, SRF provides support for serial memory interface (SMIF), real time clock (RTC), system power management (SysPM), and system clock (SysClk) modules. The support for these security aware PDL APIs are offered as service by Platform partition of TF-M. Comprehensive list of supported APIs can be found within the respective PDL and SRF documentation. Additional PDL APIs and user-defined functions can be incorporated into the SRF as per specific use case requirements. Refer to the SRF documentation for more details on integration and configuration process.

4.1 Intercommunication between SPE and NSPE

The PSOC™ Edge device has NSPE on both CM33 and CM55 CPU and the TF-M NSPE interface is available on both. The communication between the M33 NSPE and CM33 SPE (TF-M) is facilitated through non-secure callable (NSC) function calls. Non-secure callable functions contain the special secure gateway (SG) instruction placed in the NSC memory. When the NSC function call is performed from the NS software, it executes the SG instruction that performs the state change from NS to S and the control is transferred to TF-M, where the secure partition manager (SPM) handles the scheduling of requested services. The SRF is integrated with the TF-M interface. As a result, when the M33 NSPE invokes a security-aware PDL API, SRF encapsulates the request as a call to the Platform partition of TF-M. This call is then handled by the SPM, which is responsible for managing and executing secure service requests.

The M55 NSPE utilizes Inter-Processor Communication (IPC) to communicate with M33 CPU. When the M55 NSPE initiates a call for TF-M secure service or a security aware PDL, the TF-M M55 interface encapsulates the request into an SRF request which is sent via IPC. This IPC request is then received by the M33 CPU and handled by M33 NSPE. The SRF on M33 NSPE then forwards the request to the TF-M interface. The request is further processed by the SPM within TF-M, and a response is sent back to the M55 NSPE through the M33 NSPE SRF using IPC.

If the M33 NSPE operates with an RTOS, two dedicated threads are created: one for receiving SRF IPC requests and another for processing those IPC requests. These threads are created during device initialization by cybsp_init(). However, in a bare-metal M33 NSPE system, the handling of SRF IPC communication (between M33 NSPE and M55 NSPE) must be implemented by the user application. The mtb_srf_ipc_receive_request()

and mtb_srf_ipc_process_pending_request() APIs can be used for SRF IPC handling in a bare-metal M33 NSPE system as shown below. See SRF API documentation for more information.

```
int main(void)
{
    /* Device initialization */
    result = cybsp_init();
    ...
    ...

    for(;;)
    {
        /* User application code */
        ...
        ...
        ...

        srf_result = mtb_srf_ipc_receive_request(&cybsp_mtb_srf_relay_context,
                                                 IPC_TIMEOUT);
        if(srf_result != CY_RSLT_SUCCESS)
        {
            CY_ASSERT(0);
        }
        srf_result =  mtb_srf_ipc_process_pending_request(&cybsp_mtb_srf_relay_context);
        if(srf_result != CY_RSLT_SUCCESS)
        {
            CY_ASSERT(0);
        }
    }

}
```

4.2 Isolation in TF-M

The PSA Firmware Framework-M (PSA FF-M) specification defines three isolation levels to ensure data confidentiality. These three levels of isolation address the varying security needs of devices with varying capabilities.

- **Level 1 isolation:** The isolation boundary exists between the SPE and NSPE
- **Level 2 isolation:** The isolation boundary exists between the SPE and NSPE as well as between the PSA RoT and App RoT
- **Level 3 isolation:** The isolation boundary exists between the SPE and NSPE, between the PSA RoT and App RoT, and between each App RoT partition

The PSOC™ Edge EPC2 device is PSA L2 compliant and supports isolation level 2 (L2) and isolation level 3 (L3), with L2 being the default isolation level. The EPC4 device supports both isolation level 2 (L2) and isolation level 3 (L3), with L3 configured as the default isolation level. The default isolation level can be changed from Edge Protect Configurator.

To establish these isolation boundaries between domains effectively, SPM uses protection mechanisms such as TrustZone®, memory protection unit (MPU), memory protection controller (MPC), and peripheral protection controller (PPC). TrustZone® is a core component of the CM33 CPU that partitions the address space of the

entire device using the implementation defined attribution unit (IDAU) and secure attribution unit (SAU) in to secure and non-secure regions.

In addition to TrustZone®, PSOC™ Edge implements Infineon's proprietary protection contexts (PC) for additional security. Protection context isolates shared resources in a system from different software components as well as from multiple masters. It acts like a pseudo-state, thereby adding an additional security attribute to the bus masters over the existing NS attribute of TrustZone®. PSOC™ Edge supports eight PCs with each software layer executed using a predefined PC. The PC and NS attribute is used by MPC and PPC to ensure isolation of memory and peripherals between different software components.

For granular access control of memories, TF-M utilizes MPC and MPU for the following:

- The MPC divides memory into fixed-size blocks and provides access control permissions for Read/Write/Non-secure access for each protection context to ensure data confidentiality among partitions
- The MPU controls code execution allowance and access to memory by unprivileged code

Table 1 and Table 2 summarize the execution privilege level of each firmware component.

Table 1 Privilege level of EPC2 software component

Software component	NS attribute	PC	Priv attribute
Edge Protect Bootloader	S	PC2	Priv
TF-M core (SPM)	S	PC2	Priv
PSA RoT	S	PC2	Priv
App RoT	S	PC2	Unpriv
M33 NSPE	NS	PC2	Priv
M55 NS application	NS	PC6	Priv

Table 2 Privilege level of EPC4 software component

Software component	NS attribute	PC	Priv attribute
SE RT services	S	PC1	Priv
Edge Protect Bootloader	S	PC2	Priv
TF-M core (SPM)	S	PC2	Priv
PSA RoT	S	PC2	Unpriv
App RoT	S	PC4	Unpriv
M33 NSPE	NS	PC5	Priv
M55 NS application	NS	PC6	Priv

TF-M isolates peripherals using the PPC. The PPC provides access restrictions of peripherals among software components based on attributes such as non-secure, privileged, and protection context.

The protection attributes for MPC (read/write/non-secure/PC) and PPC (non-secure/privileged/PC) are grouped together and abstracted in a protection domain. For example, the M33S (secure) protection domain defines secure read and write access to memory and secure privileged access to peripherals (PPC regions) for bus masters with PC2 and PC7. The Device Configurator allows users to create up to 16 protection domains. Each memory region and PPC region in the device is assigned a protection domain through the Device Configurator. During the device initialization, the MPC and PPC are configured such that the memory regions and peripherals have appropriate access permissions as per their assigned protection domain.

The Edge Protect Configurator provides the necessary default protection domains required for TF-M across both EPC2 and EPC4 devices. The Edge Protect Configurator works in conjunction with Device configurator to

assign these domains to memory regions associated with TF-M as well as to peripherals. The security permission for each protection domain can be viewed from Systems tab in Device Configurator. See Edge Protect Configurator User Guide and ModusToolbox™ Device Configurator user guide for more information.

4.3 Profiles in TF-M

The features of TF-M are highly customizable. To cater to devices with different capabilities and their use cases, TF-M provides a set of profiles. A profile is a group of configuration options used for building TF-M. TF-M provides four profiles; small, medium, and ARoT less and large. Table 3 table provides a brief summary of the features offered in each TF-M profile. Out of the four profiles, Infineon's port of TF-M for PSOC™ Edge device supports medium and large profiles with medium being the default for both EPC2 and EPC4. The default TF-M profile can be changed from Edge Protect Configurator.

Table 3 TF-M profiles

	Small (not supported)	**Medium**	**ARoT less (not supported)**	**Large**
Isolation level	Level 1	Level 2	Level 1	Level 3
Crypto capabilities	• Symmetric cipher suite • SHA 256 hashing	• Symmetric and asymmetric cipher suite • SHA 256 hashing	• Symmetric and asymmetric cipher suite • SHA 256 hashing	• Larger symmetric and asymmetric cipher suite • SHA 256/384/512 hashing
Services/partitions	• ITS • Initial Attestation (symmetric) • Crypto	• ITS • Initial Attestation (asymmetric) • PS • Crypto	• ITS • Initial Attestation (asymmetric) • FWU • Crypto	• ITS • Initial Attestation (asymmetric) • PS • Crypto • FWU (not supported)

The configuration options defined by a TF-M profile can be customized further as per the application's requirements. The following table compares the default configuration used by Infineon's port of TF-M for PSOC™ Edge EPC2 and EPC4 devices:

Table 4 PSOC™ Edge E84 EPC2 and EPC4 default TF-M configurations

Configuration	**EPC2**	**EPC4**
Profile	Medium	Medium
Isolation level	Level 2	Level 3

(table continues...)

Table 4 (continued) PSOC™ Edge E84 EPC2 and EPC4 default TF-M configurations

Configuration	EPC2	EPC4
Services/Partitions	• ITS • Initial attestation (asymmetric) • PS • Crypto • Platform	• ITS • Initial attestation (asymmetric with Secure Enclave) • PS • Crypto (with Secure Enclave) • Platform
NV counters location	RRAM	Secure Enclave

5 Services offered by TF-M

The following services are offered by TF-M to the SPE and NSPE:

5.1 Cryptography

TF-M provides a cryptography service as an implementation of the PSA crypto API in a PSA RoT secure partition within TF-M. This implementation utilizes the `MbedTLS` library. The crypto service can be used by applications running in the NSPE or by other services of TF-M (running in the SPE) through PSA crypto APIs.

TF-M can be configured and built for four different profiles, each providing a different level of functionality. These profiles offer the following cryptography capabilities:

- **Small:** Provides only symmetric cipher, SHA256, and HMAC (not supported on PSOC™ Edge)
- **Medium:** Provides symmetric and asymmetric cryptography (as recommended in (RFC 7925) and CoAP (RFC 7252)), along with SHA, HMAC, and asymmetric key algorithms for signing or verification, and public-key cryptography-based key exchange
- **ARoT less:** Same as medium profile (not supported on PSOC™ Edge)
- **Large:** Provides symmetric and asymmetric cryptography as defined in TLS 1.2, SHA, HMAC, asymmetric key algorithms for signing or verification, public-key cryptography-based key exchange, and key derivation

By default, TF-M for PSOC™ Edge EPC2 and EPC4 devices is built with the medium profile. The crypto implementation for the PSOC™ Edge EPC2 device uses the following:

- A crypto hardware accelerator for accelerated algorithms
- Software implementation from the Mbed TLS library for other algorithms

The selection to use the hardware accelerator or software implementation is defined at compile-time configuration. By default in EPC2 device, hardware accelerator support is enabled in TF-M. Accelerator support can be enabled or disabled by passing the following Make configuration in `proj_cm33_s/Makefile`

```
TFM_CONFIGURE_OPTIONS+= -DIFX_MBEDTLS_ACCELERATION_ENABLED:BOOL=ON/OFF
```

In EPC2 devices, the cryptography service uses ITS to store the persistent keys while the volatile keys are stored in an SRAM region, which is allocated to Crypto partition.

The crypto implementation for the EPC4 device uses the following:

- Secured enclave (SE) for robust security
- A crypto hardware accelerator for accelerated algorithms (PSA L2 certifiable)
- Software implementation from the Mbed TLS library for other algorithms

In EPC4 device, it is recommended to use the Secure Enclave for crypto operation and keep the crypto hardware accelerator disabled. If built with a crypto hardware accelerator, the resulting system cannot meet PSA L3/4 requirements. EPC4 devices use the key lifetime attribute to specify if the secured enclave or M33 software implementation can be used for crypto operations. If the SE is used for crypto operation then both persistent and volatile keys are stored inside the SE. The following code snippet shows an example of how to set a key attribute to use SE for crypto operation:

```
psa_set_key_lifetime(&key_attributes,
        PSA_KEY_LIFETIME_FROM_PERSISTENCE_AND_LOCATION(
                PSA_KEY_LIFETIME_VOLATILE,
                PSA_KEY_LOCATION_IFX_SE) );
```

For SHA256 hashing, two options are available with EPC4 device; SE or software implementation. Unlike other crypto operations, the SHA256 selection must be done at compile time, by default, SHA256 with SE is enabled.

This selection is done by setting the `IFX_USE_SW_SHA` Make configuration flag to OFF for SE implementation and ON for M33 software implementation:

```
TFM_CONFIGURE_OPTIONS+= -DIFX_USE_SW_SHA:BOOL=OFF/ON
```

For the SHA384/512 family, software implementation is used and SE support is not available. For the complete list of supported SE crypto operations, see the "SE_RT_Services" section in the PSOC™ Edge E84 MCU architecture reference manual.

In addition to the predefined profiles, TF-M offers the flexibility to customize the crypto suite by adding or removing specific algorithms that align with the application's requirements. This can be achieved by defining a customized crypto configuration header file and setting the `TFM_MBEDCRYPTO_PLATFORM_EXTRA_CONFIG_PATH` CMake variable with the path of the header file. The following snippet shows how the CMake variable can be set via `proj_cm33_s/Makefile`

```
TFM_CONFIGURE_OPTIONS+= -DTFM_MBEDCRYPTO_PLATFORM_EXTRA_CONFIG_PATH="Path to config file"
```

The PSOC™ Edge MCU: Trusted Firmware-M (TF-M) Crypto - EPC2 and PSOC™ Edge MCU: Trusted Firmware-M (TF-M) Crypto - EPC4 code examples show how to use crypto service of TF-M. The EPC2 code example utilizes the hardware accelerator and the EPC4 code example uses the hardened Secure Enclave to demonstrate cryptography service.

5.2 Secured storage

TF-M provides the following two services for secure storage of assets:

5.2.1 Internal trusted storage (ITS)

ITS is a PSA RoT service designed for storing the most security-critical device data, such as cryptographic keys, in the internal RRAM storage. In PSOC™ Edge, RRAM NVM is used to store the ITS assets. By default, ITS supports the storage of ten assets, with a maximum asset size of 512 bytes.

The PSOC™ Edge MCU: Basic Trusted Firmware-M application - EPC2 and PSOC™ Edge MCU: Basic Trusted Firmware-M application - EPC4 code examples demonstrate the use of ITS service to store assets from NSPE in secure memory.

5.2.2 Protected storage (PS)

PS is an application RoT service that offers encrypted storage of assets in the internal RRAM. PS uses AES-CCM based AEAD encryption to protect the assets. By default, PS supports the storage of ten assets, with a maximum asset size of 2 KB.

5.3 Initial attestation service

The initial attestation service serves as a key mechanism to prove the genuineness of the device. Upon request, TF-M can create an initial attestation token (IAT) that is compliant with the IETF entity attestation token (EAT) specification. The attestation token contains a list of claims that serve as evidence to prove the identity of the device. The following list of claims are provided in the attestation token:

- boot seed
- challenge
- client_id

- implementation_id
- instance_id
- profile_id
- security_lifecycle
- sw_components (for each image)
  - measurement_description
  - measurement_value
  - signer_id
  - sw_component_type
  - sw_component_version

The EPC2 device includes software component measurements for the following images:

- CM33 SPE (TF-M)
- CM33 NSPE
- CM55 NSPE

The EPC4 device includes software component measurements for the following images:

- SE RRAM boot
- SE RT services
- Extended boot (L1 boot)
- CM33 SPE (TF-M)
- CM33 NSPE
- CM55 NSPE

This list of claims is encoded in concise binary object representation (CBOR) format and signed by initial attestation private key (256-bit ECC key) as per the CBOR object signing and encryption (COSE) specification (RFC 8152). A certification verification service verifies the token with the initial attestation public key. In EPC2 device, the attestation token is generated locally on M33 SPE while in EPC4 device, the attestation token generation is carried out by SE.

The initial attestation service relies on the Edge Protect Bootloader for boot measurements of all images to generate the attestation token. To get boot measurements, boot measurements feature must be enabled. This can be enabled in the Edge Protect Bootloader Solution in Device Configurator. See AN237857 Edge Protect Bootloader for PSOC™ Edge for more information.

The PSOC™ Edge TF-M based initial attestation - EPC2 and PSOC™ Edge TF-M based initial attestation - EPC4 code examples show how to use an initial attestation service to generate an attestation token and how the IAK public key can be extracted from the device for token verification.

Note:
- *The signer id is an optional claim and is not included for SE ROM firmware, extended boot and SE RT service images*
- *The SE ROM firmware is the root of trust therefore the boot measurement (SHA256 hash) value of SE ROM firmware is not available*

5.4 Platform service

The platform service leverages the platform-specific features of the MCU and offers it to the NSPE via the platform partition of TF-M. TF-M for PSOC™ Edge offers the following platform services:

Non-volatile counters: TF-M offers up to three non-volatile counters of four bytes each to the NSPE. The NSPE can use the platform service to either read or increment the NV counters. Include the `tfm_platform_api.h`

header file in the NSPE project and use the `tfm_platform_nv_counter_increment()` and `tfm_platform_nv_counter_read()` APIs to use the service.

System reset: The registers required to reset the system are protected and cannot be accessed by the NSPE. The system reset service provides a way to the NSPE application to reset the platform. To reset the device, include the `tfm_platform_api.h` header file in the NSPE project and call the `tfm_platform_system_reset()` API function.

NSPE logging: The logging platform service can be used by the NSPE to log messages on the serial terminal using UART. This service can be quite useful when only one SCB of the MCU is available and both the TF-M and NSPE applications need to log. Before using NSPE logging, secure logging in TF-M must be enabled. To enable logging add the following Make configuration in secure (`proj_cm33_s`) project Makefile.

```
TFM_CONFIGURE_EXT_OPTIONS+= -DTFM_EXCEPTION_INFO_DUMP=ON -DPLATFORM_EXCEPTION_INFO=ON
-DIFX_FAULTS_INFO_DUMP=ON -DTFM_SPM_LOG_LEVEL=TFM_SPM_LOG_LEVEL_DEBUG
-DTFM_PARTITION_LOG_LEVEL=TFM_PARTITION_LOG_LEVEL_DEBUG
```

To use this service, the NSPE application must include the `ifx_platform_api.h` header and call the `ifx_platform_log_msg()` API.

Security aware PDL: The Infineon's port of TF-M has an SRF framework integrated in to TF-M as part of the platform partition. It provides security aware driver implementation of the following modules: SMIF, RTC, SysClk and SysPM. The supported APIs from these modules can be called from NSPE and their implementation is provided as part of platform partition in TF-M. See SRF and PDL documentation for complete list of supported APIs.

6 TF-M memory map

This section describes the memory configuration and usage associated with TF-M for the PSOC™ Edge device.

Figure 33 illustrates the default memory layout for basic Trusted Firmware-M (TF-M) based application, showing the division of various memories for the secure and non-secure processing environments for code, data, and shared regions.

6 TF-M memory map

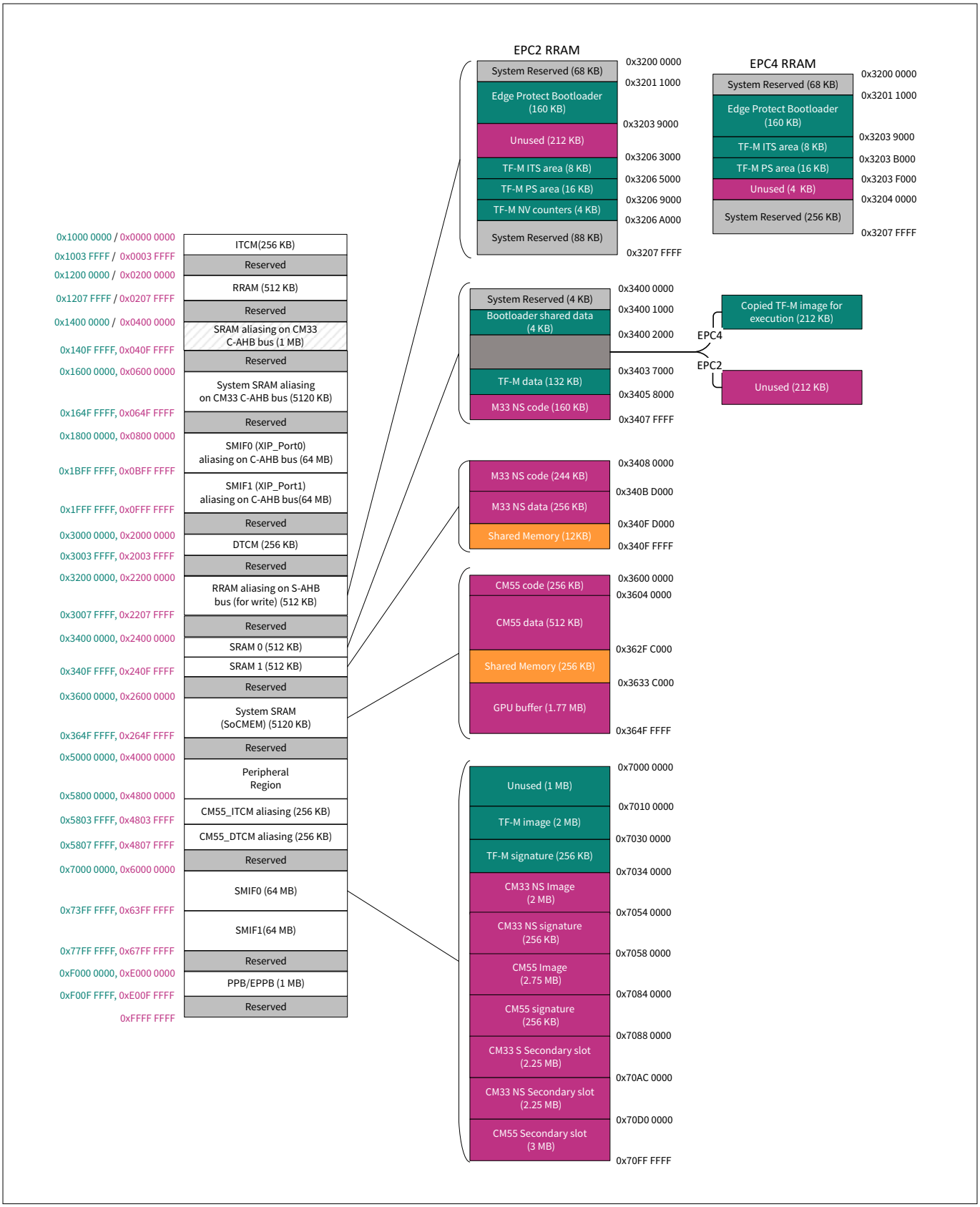

Figure 33 TF-M memory map

- SMIF interface
  - By default, the PSOC™ Edge Evaluation Kit has a 16 MB NOR flash interfaced with SMIF0 that has a sector size of 256 KB. The first 1 MB memory is unused
  - The next 2 MB is allocated for the TF-M image. If swap upgrade mode is used, the image trailer is required to be placed in a different sector than the image. Therefore, the TF-M image requires an additional 256 KB for the trailer
  - 2 MB and 2.75 MB slots are allocated for the CM33 NSPE image and CM55 image respectively, with the extra 256 KB required to place the image trailer on a new sector
- RRAM
  - The RRAM is used for placing extended boot, Edge Protect Bootloader, TF-M ITS region, TF-M NV counters (for EPC2 device), and TF-M protected storage. The remaining space is available to NS applications
- SRAM
  - The first 8 KB of SRAM0 is used for shared data by extended boot and the Edge Protect Bootloader, containing the boot measurements passed by the bootloader to the next firmware. This shared data is critical for the functioning of the Initial Attestation service
  - The next 212 KB is unused in EPC2 device. In EPC4 device it is used for TF-M execution
  - The next 132KB is used for TF-M data
  - The rest of the SRAM0 and SRAM1 are available to NS applications and shared data
- System SRAM (SoCMEM)
  - The SoCMEM is not used by TF-M and is entirely available to NS applications
- Instruction TCM (ITCM) and data TCM (DTCM)
  - The tightly-coupled memories (TCMs) are used only by the CM55 application and is not used by TF-M

6.1 Changing the TF-M memory map

The TF-M application memory map can be customized to satisfy the requirements of the use case. The default TF-M map can be changed by updating the memory sections in the memory tab of Device Configurator. The Device Configurator allows changing the size and location of memory regions as well as creating new regions and assigning protection domains to them. Refer to the memory tab section in Device Configurator User Guide for detailed instructions on modifying the memory map.

***Note**: When TF-M is executed from external flash (XIP), the PS storage area must be placed in RRAM. Writing to external flash while executing in XIP is not supported.*

7 Secure boot and image upgrade

By default, image authentication is disabled in both extended boot and Edge Protect Bootloader. Enabling secured boot enables the Chain of Trust (CoT) where each software is authenticated by the preceding software. In PSOC™ Edge MCU, the Root of Trust (RoT) is anchored in the Immutable boot firmware, which is executed by Secure Enclave (SE). To establish the full chain of trust:

- Enable secured boot in extended boot
- Enable secured boot in the Edge Protect Bootloader and sign the bootloader
- Enable image signing in PSOC™ Edge basic TF-M application

To enable secured boot in extended boot, the ownership of the device must be transferred to the OEM and the OEM policy must be updated, and the device must be provisioned with the new policy. See the "Enable secured boot by Extended Boot" section from AN237849 – Getting started with PSOC™ Edge security for guidance.

To enable secured boot in Edge Protect Bootloader, signing the bootloader and TF-M application, follow the instructions from AN237857 Edge Protect Bootloader for PSOC™ Edge.

The Edge Protect Bootloader is also responsible for updating the images. The Edge Protect Bootloader allows two image upgrade modes:

- **overwrite:**The Edge Protect Bootloader copies the image from secondary slot and overwrites the primary slot
- **swap:**The Edge Protect Bootloader uses a scratch area to swap the images from primary and secondary slot. The upgrade mode requires two extra memory regions, a scratch region for performing the image swapping operation and a swap status region to track the progress and state of the swap operation

Refer AN237857 Edge Protect Bootloader for PSOC™ Edge for detailed instructions on how to select upgrade mode.

References

Application notes

[1] Infineon Technologies AG: *Getting started with PSOC™ Edge E8 on ModusToolbox™ software*; Available online
[2] Infineon Technologies AG: *Edge Protect Bootloader for PSOC™ Edge*; Available online
[3] Infineon Technologies AG: *Getting started with PSOC™ Edge security*; Available online

Code examples

[4] Infineon Technologies AG: *PSOC™ Edge MCU: Basic Trusted Firmware-M (TF-M) application*; Available online
[5] Infineon Technologies AG: *PSOC™ Edge MCU: TF-M based Initial Attestation application*; Available online
[6] Infineon Technologies AG: *PSOC™ Edge MCU: Trusted Firmware-M (TF-M) Crypto application*; Available online
[7] Infineon Technologies AG: *PSOC™ Edge MCU: Basic Trusted Firmware-M (TF-M) application*; Available online
[8] Infineon Technologies AG: *PSOC™ Edge MCU: TF-M based Initial Attestation application*; Available online
[9] Infineon Technologies AG: *PSOC™ Edge MCU: Trusted Firmware-M (TF-M) Crypto application*; Available online

***Note**: For more documents on PSOC™ Edge E8 MCU, see the PSOC™ Edge E8 MCU product webpage.*

Glossary

Arm® TrustZone®

An Arm® technology used for isolating SPE and NSPE address space.

Edge Protect Bootloader (MCUboot)

An open-source secure bootloader library for 32-bit MCUs aiming to define a common infrastructure for the bootloader and system flash layout on microcontroller systems, and to provide a secure bootloader that enables easy software upgrade.

EPC2

Refers Edge Protect category 2 device which supports SESIP/PSA Level 2.

EPC4

Refers to Edge Protect category 4 device supports SESIP/PSA Level 3. EPC4 device features a Secured Enclave for secure processing and storage of sensitive data.

Extended boot

This is the first code executed by the CM33 during the secured boot process. It interprets the `OEM_POLICY` file and boots the first OEM code based on the policies in this file.

ITCM and DTCM

Instructions tightly-coupled memory (ITCM) and data tightly-coupled memory (DTCM) are volatile, low-latency memories that can be used by the processor for execution and data processing.

ModusToolbox™

It is an Eclipse-based embedded design platform for IoT designers that offers a single, coherent, and familiar design experience. The platform combines the industry's most deployed Wi-Fi and Bluetooth® technologies with the lowest power, most flexible MCUs, and best-in-class sensing capabilities.

MPC

Memory protection controller. This hardware block is used for access control to memory based on protection context (PC) and secure/non-secure (S/NS) attributes.

MPU

Memory Protection Unit. These hardware blocks are used to permit or restrict access to sections of memory or peripherals for CPU bus masters.

NSC

Non-secure callable (NSC) is a special type of secure region which is permitted to hold secure gateway (SG) instructions that allows state transition from non-secure to secure state.

NSPE

Non-secure processing environment. It is a space that is not isolated from other software running on the device. Data from the NSPE can be accessed by other software. Typically, the NSPE has limited access to memory and peripherals.

PC

Protection context. An Infineon proprietary access control mechanism that adds an additional layer of security over TrustZone®.

Glossary

PPC

Peripheral protection controller. These hardware blocks are used for isolating peripheral regions based on protection context (PC), S/NS attributes, and privileged/unprivileged attributes.

PSA

Arm® Platform Security Architecture. Arm® PSA is a comprehensive security framework developed to enhance the security of IoT devices by providing standardized practices, reference implementations, and certification processes.

PSA Certified

PSA Certified program offers a low-cost, time-saving way for implementing security. PSA Certified framework ensures security is designed into a device right from hardware to software.

PSOC™

It is a programmable, embedded design platform that includes a CPU, such as the 32-bit Arm® Cortex®-M0, with both analog and digital programmable blocks. This platform accelerates embedded system design with reliable, easy-to-use solutions such as touch sensing, and enables low-power designs.

SE

Isolated hardware block consisting of two CM0+ CPUs in lockstep architecture. The SE is responsible for handling sensitive data while ensuring confidentiality.

SPE

Secure processing environment. It is an isolated and protected space where data can be processed and stored securely. Data from the SPE is not accessible to the outside world but SPE has full access to the system's resources.

TF-M

Trusted Firmware-M. It is an open-source initiative by Arm® that implements the secure processing environment (SPE) for Armv8-M, Armv8.1-M architectures, and dual-core platforms.

TSBA-M

Arm® Trusted Base System Architecture. It is an architecture for designing and implementing secure devices which is part of Arm® PSA

Revision history

Document revision	Date	Description of changes
*D	2025-10-14	Release to web

Trademarks

The Bluetooth® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc., and any use of such marks by Infineon is under license.

PSOC™, formerly known as PSoC™, is a trademark of Infineon Technologies. Any references to PSoC™ in this document or others shall be deemed to refer to PSOC™.

Trademarks

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

Edition 2025-10-14

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-elp1717744370712

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.