Renesas Synergy™ Platform

Synergy Device Identity Application

Introduction

This application note describes IoT security in general and offers a brief introduction to the security features offered by the Synergy MCU, including different key generation options. The application example provided in this package uses the SCE module to generate a unique hardware-based device identity that will be securely stored in the internal flash using Security MPU and FAW.

This application note enables you to effectively use the Synergy SCE modules in your own design. Upon completion of this guide, you will be able to add the SCE Synergy Software Package (SSP) modules to your own design, configure them correctly for the target application, and write code using the included application example code as a reference and efficient starting point. References to more detailed API descriptions, and other application projects that demonstrate more advance uses of the module, are in the *Synergy Software Package (SSP) User's Manual*, and serve as a valuable resource in creating more complex designs.

Currently, the Synergy Device Identity Application is implemented and tested on the PK-S5D9/AE-CLOUD2 kit. Support for other Synergy kits/MCUs will be provided in upcoming releases.

Required Resources

To build and run the Synergy Device Identity Application example, you need the following resources:

Development tools and software

- ISDE v6.2.1 or later (renesassynergy.com/devtools/e2studio), or IAR Embedded Workbench® for Renesas Synergy™ v8.23.1 or later (renesassynergy.com/devtools/ewsyn)
- Synergy Software Package (SSP) v1.5.0 or later (renesassynergy.com/software/ssp), Synergy Standalone Configurator (SSC) 6_2_1_R20180629 or later (renesassynergy.com/devtools/ssc)
- SEGGER J-link® USB driver (renesassynergy.com/devtools/jlink)

Hardware

- Renesas Synergy PK-S5D9 kit (renesassynergy.com/kits/pk-s5d9) or AE-Cloud2 kit
- Test PC running Windows 7/10 OS
- Two Micro USB cables

Prerequisites and Intended Audience

This application note assumes you have some experience with the Renesas ISDE and Synergy Software Package (SSP). Before you perform the procedures in this application note, follow the procedure in the *SSP User Manual* to build and run the Blinky project. Doing so enables you to become familiar with the and the SSP, and validates that the debug connection to your board functions properly. In addition, this application note assumes that you have some knowledge of cryptography and Synergy Crypto Engine features.

The intended audience are users who want to develop applications with SCE modules using Renesas Synergy™ S5D9, S5D5, and S5D3 MCU groups.

Contents

1. Introduction to IoT Security

This section provides an overview of IoT Security (in general) and covers the different aspects of the security features offered by Synergy MCUs.

1.1 Overview

A generic IoT (Internet of Things) environment consists of the following:

- IoT Devices
- Cloud Server
- Device Management services
- Certificate Authority (CA)

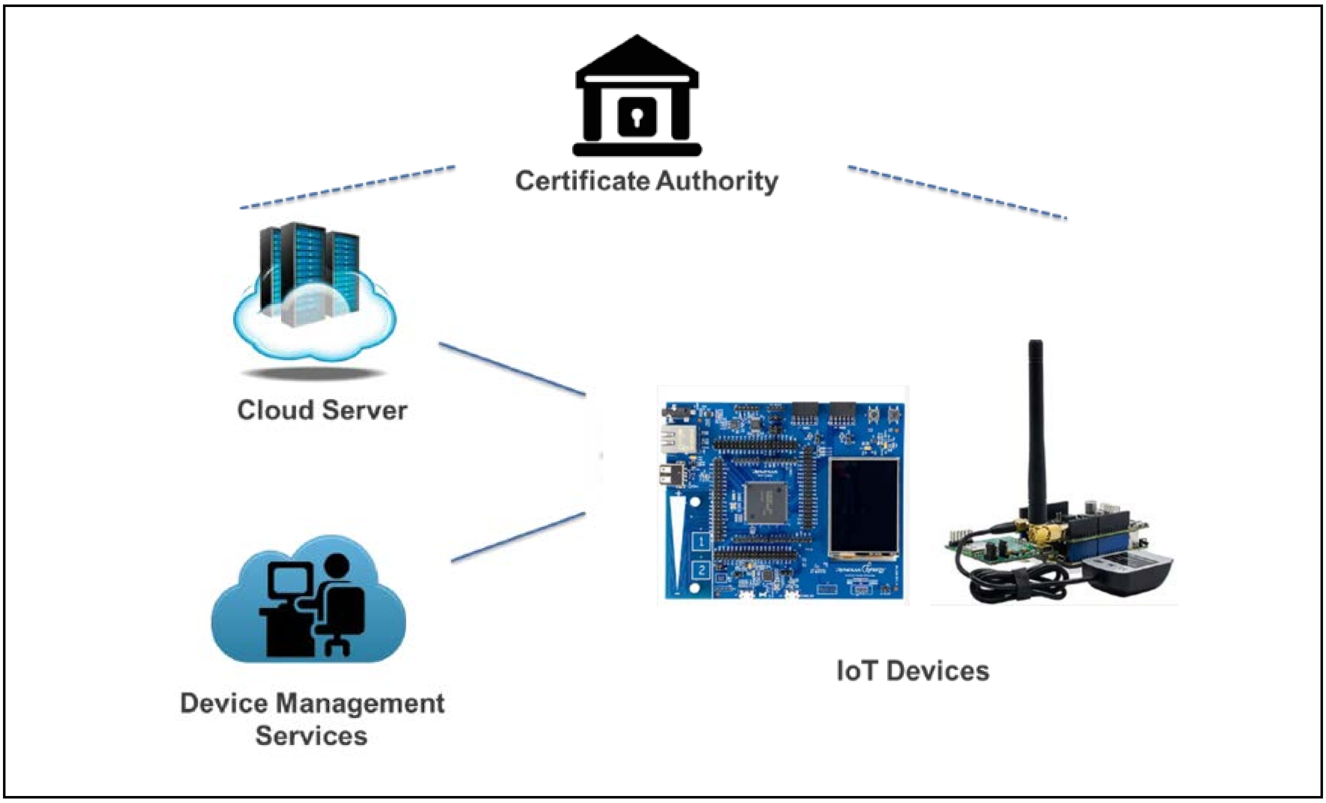


Figure 1. IoT Environment Overview

IoT Devices

IoT devices constitute the network of physical devices embedded with software, sensors, and connectivity which enables these devices to connect, collect, and exchange data. Devices can be in a secure or non-secure location, but all are increasingly vulnerable to attack.

Cloud Server

A portal providing everyday services and access to those devices. Typically located in a highly secure and controlled data center.

Device Management Services

Device Management services offers a comprehensive suite of IoT device management capabilities to enable IoT customers of any size to have complete control over their devices and data. This includes (but is not limited to):

- Application Security
  Keys/certificates identifying the Cloud Server
- Device Management Security
  - Keys/certificates identifying each unique IoT Device
  - Keys/certificates identifying Device Management Services Server

— Initial firmware deployment, and subsequent firmware updates. Firmware contains a signature verifying its authenticity, and may be encrypted.

Certificate Authority (CA)
An authorized and trusted entity that issues certificates as a service is commonly referred to as CA. Certificates are used to give identity to public keys and thus the devices that contain those keys. The process by which a certificate is generated for a key is a well-defined process that is part of your security scheme. A Certificate Authority can be public or private. If your devices are managed in a tight eco-system (for example, devices for Industrial setting), the CA will likely be private. If your devices are distributed through a consumer channel where the services and hardware are likely to be provided by different vendors (for example, surveillance cameras, thermostats, home security systems, and so forth.), the CA will likely be a public CA.

1.2 Importance of Device Identity in an IoT Ecosystem

With an establishment of a strong device identity, each IoT device can uniquely be identified and authenticated when they are connected to ensure secure and encrypted communication between other devices, services and users.

Strong device identity addresses core IoT security requirements:

- Trust
  When a device connects to the network, it must authenticate and establish trust between other devices, services and users. Once trust is established, devices, users and services can securely communicate and exchange encrypted data and information.
- Privacy
  As more IoT devices connect, more data is generated, collected, and shared. This data can include personal, sensitive, and financial information that must be kept private and secured – often under regulatory compliance. A device identity can ensure authentication, identification when the IoT devices are connected to one another.
- Integrity
  Device integrity applies to both the devices and data being transmitted within the IoT ecosystem. The integrity of a device starts with proving it is what it says it is. With a strong unique device identity, it can be ensured that the devices are legitimate – reducing counterfeit products and protecting a company's brand. Data integrity is an often-overlooked requirement, but connected devices and systems rely on the authenticity and reliability of the information being transmitted.

1.3 Synergy Hardware Security Features

Synergy MCUs provides a hardware root-of-trust via protected memory (read/write) which can only be accessed by firmware modules located in secured memory segments. Protected memory features offered by Synergy devices can be used for storing the secure boot code and device certificate/keys amongst other sensitive data which are vital for device identity application.

1.3.1 ARM MPU

The ARM MPU is a memory protection function for the CPU that can be used to define memory access permissions (that is, privileged access only or full access) and memory attributes (that is, buffer-able, cacheable) for different memory regions.

- ARM MPU can support up to eight programmable memory regions, each with their own programmable starting addresses, size and settings.
- Using a Memory Protection Unit (MPU) can protect applications from many potential errors, ranging from undetected programming errors to errors introduced by system or hardware failures.

Configuration details of the ARM MPU is outside the scope of this application project. For more information for the ARM MPU operation and functionality, please refer to the Synergy MCU Hardware User's Manual.

1.3.2 Bus Master MPU

Bus master MPU monitors the addresses accessed by the bus masters in the entire MCU address space (0000 0000h to FFFF FFFFh). The access control information, consisting of read and write permissions, can be independently set for up to 32 regions. The bus master MPU monitors access to each region based on these settings.

- If access to a protected region is detected, the bus master MPU generates a reset or a non-maskable interrupt.
- There are three Bus master MPU groups
  - DMA Bus (group A)
  - ETHER Bus (group B)
  - GPX Bus (group C)
- The secure MPU regions are protected from the bus masters.

Configuration details of the Bus Master MPU are outside the scope of this application project. For more information on the Bus Master MPU operation and functionality, please refer to the *Synergy MCU Hardware User's Manual*.

1.3.3 Bus Slave MPU

Bus slave MPU monitors access to the bus slave functions, such as flash or SRAM. The function can be accessed from four bus masters, the CPU, and bus master MPU groups A, B, and C as follows:

- Bus slave MPU has a separate protection register for each of the four bus masters, with independent access protection control, consisting of read and write permission.
- If access to a protected region is detected, the bus slave MPU generates a reset or a non-maskable interrupt, and can store the bus error address, bus error status, and error access status.

Configuration details of the Bus Master MPU is outside the scope of this application project. For more information for the Bus Master MPU operation and functionality, please refer to the Synergy MCU Hardware User's Manual.

1.3.4 Security MPU

Synergy MCUs incorporate a security MPU with four secured regions. The 4 secure regions include individual areas in Code flash, SRAM, and two security function regions. These regions can only be accessed by "secure code". Secure regions are protected from unauthorized accesses by:

- A non-secure program
- Additional bus masters, such as the DMA, DTC
- Debugger interface

This mechanism allows untrusted code to exist and operate alongside trusted code. The security MPU settings are stored in flash and it is activated before fetching the reset vector.

For more detailed information on the Security MPU, please refer to the Synergy MCU Hardware User's Manual.

1.3.5 FAW (Flash Access Window)

The Flash Access Window registers are used to set the code flash address range, which can be erased/programmed. The addresses that are outside this range, referred to as outside the Flash Access Window, cannot be modified once programmed. This feature is used to prevent the device identity (keys/cert) from being erased or reprogrammed.

The example application project provided along with this package includes code reference to configure the Flash Access Window using APIs provided by SSP for the device identity use case. User can also refer to Secure data in the rest application project for more use cases on FAW configuration.

For more detailed information on FAW, see the SSP UM link in the reference section.

Note: The FAW is set to the area of flash that can be written, so the area of memory that is LOCKED is the inverse of the FAW address range.

FSPR (One Time Programmable Setting)

The FAW and MPU registers can be permanently set using the FSPR bit. This bit is one-time programmable and so must only be set once all the settings are confirmed and the device is ready to leave the production floor.

In use cases where the Renesas Secure Boot Manager is used, the FSPR settings are done in the boot loader code. In this case, the application needs for allocating memory regions to securely store device certificates/keys should be considered as part of Secure boot loader design.

It is important that this bit is set to prevent the FAW and MPU registers themselves from being modified in the field. For example, when using the Security MPU, the FAW and FSPR bit must be set to lock the MPU settings.

1.3.6 Secure Crypto Engine Module

The Secure Crypto Engine (SCE) is a Synergy hardware peripheral that provides several security features, NIST certified algorithms, and support for primitive cryptographic algorithms.

There are 3 variants of the security hardware peripherals available in Synergy MCUs, as shown in the following figure.

- SCE 7
- SCE 5
- Security and Encryption

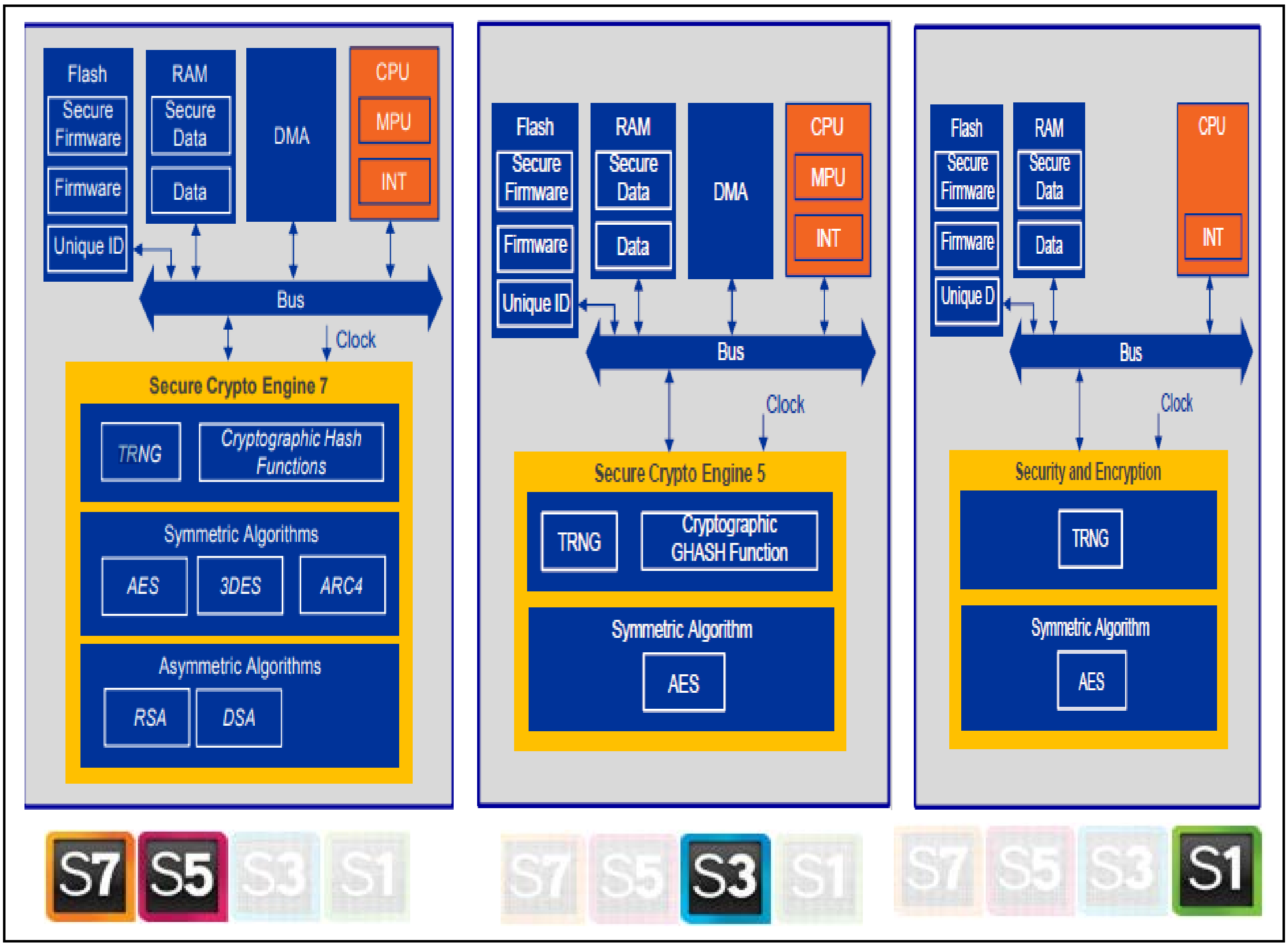


Figure 2. Security Hardware Peripherals Available in Synergy MCUs

The SCE engine provided by Synergy devices is used by this example application project in the following areas:

- Generate ECC key pairs (public and wrapped private key)
- Sign the challenge string using the ECC private key

2. Overview of Key Generation in Synergy MCUs

2.1 Key Wrapping

Device keys generated inside the Synergy MCU using the SCE hardware module can be either in plain text or wrapped.

Plaintext Keys

Plaintext refers to information or data in an unencrypted, or unprotected form that is readable by either a human or a machine, can be used without the need for any special processing. It can be used on other systems and must be protected when they are private keys.

Wrapped Keys

A wrapped key is a key that has been encrypted by the SCE, using a method that involves the MCU's unique ID. Because it requires the MCU's unique ID to unwrap the key, the key can only be unwrapped by the same MCU that wrapped it. Therefore, key wrapping on Synergy MCUs is considered secure, as a wrapped key can only be used on the Synergy MCU on which it was generated, and it cannot be used outside of that MCU.

Wrapped keys provide the following advantages:

- A wrapped key can only be used on the Synergy MCU on which it was generated.
- It cannot be moved to another Synergy MCU. If moved to another Synergy device, the original key cannot be recovered from the wrapped key.

2.2 Key Generation in the Device

This is the common use case where the device specific key is generated inside the Synergy MCU using the SCE module. To generate the device key using the Synergy Software Package (SSP), the SF Crypto Key framework module is used.

The SF_CRYPTO_KEY Framework provides a high-level API and is implemented as the SF_CRYPTO_KEY SSP module. The SF_CRYPTO_KEY Framework provides cryptographic key generation services through the Secure Cryptographic Engine (SCE) HAL module, which in turn drives the SCE IP on the device.

SF CRYPTO KEY Framework Module Features

The following key types can be generated using the services of the SF_CRYPTO_KEY module using SCE7 hardware:

- RSA 2048-bit, 1024-bit plain text/raw keys in standard and Chinese Reminder Theorem (CRT) format
- RSA 2048-bit, 1024-bit standard format wrapped private keys (public keys in plain text)
- AES 128-bit, 192-bit and 256-bit wrapped keys for ECB, CBC, CTR and GCM chaining modes
- AES 128-bit and 256-bit wrapped keys for XTS chaining mode
- ECC 192-bit, 256-bit plain-text/raw public keys and wrapped private keys.

In this application, crypto HAL module is used to generate wrapped ECC keys. The reason the crypto HAL module is used instead of SF_CRYPTO_KEY framework is that the device identity application can be used in non-RTOS use cases as well such as the Secure Boot Manager provided by Renesas.

2.3 Key Injection from Secure Infrastructure

Key Injection is a security feature that is meant for use cases where a key is generated external to the MCU device in a secure facility and then injected into the MCU. In general, the RSA key generation takes more time when generated inside the MCU compared to if it was generated external to the device (through a PC tool) inside a secure facility.

If hardware-based unique identity is not a requirement for the application, and/or if it is necessary to securely inject customer-specific keys, key injection can be utilized. This process will be covered by a separate Application Project.

3. Device Identity Design Overview

In this section we will explain how Synergy hardware and software features are integrated to create a unique device identity for each device.

Key Generation

The first step in creating a device identity is key generation. The keys can be either generated inside the Synergy MCU or they can be generated outside in a secure facility and injected into the Synergy device. Each methodology has its pros and cons on their approach. Based on the customer use case, the decision must be made.

Certificate Authority (CA)

Once the device keys are generated/injected, we need an entity that issues digital certificates. A CA can be either public or private CA located in the Cloud or in an on-premises CA (local CA), which would typically be hosted on a secure server.

Securing Device Identity

Once the device identity is created and programmed on the Synergy device, it must be securely stored to prevent being stolen or corrupted. This can be achieved by using the Security MPU and FAW features offered by Synergy MCU. The features configure a portion of internal code flash as secure code and data regions. The secure code region contains API functions that are only authorized to work on the secure data region. The secure data region contains key information such as device certificates. This section cannot be accessed or modified by any un-secure code running on the Synergy MCU.

The Security MPU settings are locked using the FAW feature (using one-time programmable FPSR bit) before leaving the secure facility (programming center) to prevent them from being modified.

4. Device Identity Application Example

4.1 Overview

This example application project demonstrates the Renesas Synergy™ device identity application using the onboard Synergy SCE modules. For demonstration purposes, this application uses a local Certificate Authority (CA) running on a windows PC to generate a signing key and root CA that will be used to sign the device certificate. This project will be later upgraded to use a third-party CA to generate the device certificate. USB-CDC is used as the primary communication interface between the PK-S5D9/AE-Cloud2 kit and the host console application running on the Windows PC.

4.2 Software Architecture Overview

The following figure shows the overall software architecture of the Synergy device identity application project.

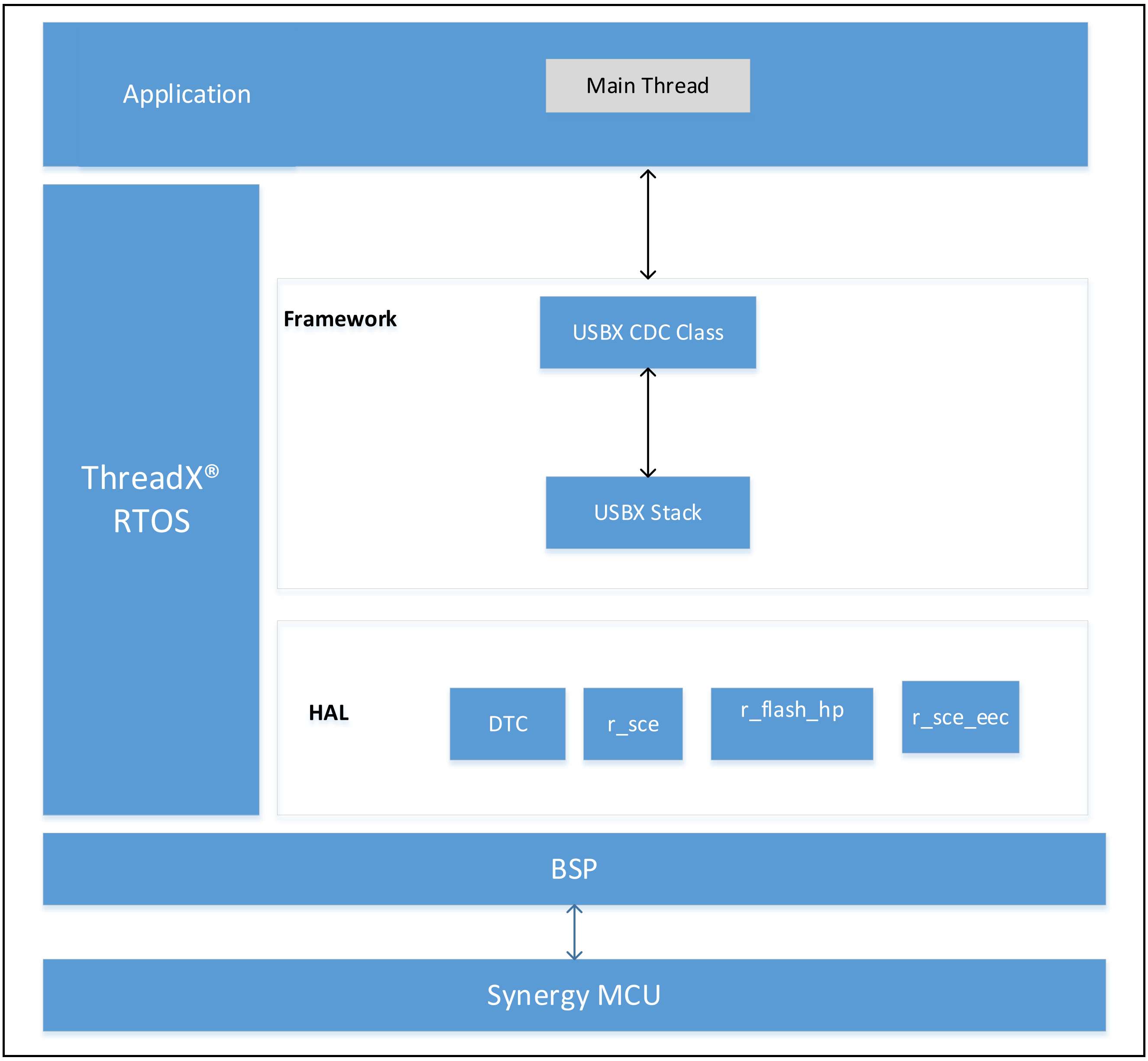


Figure 3. Synergy Device Identity Application Software Architecture

The major software components of this application are:

- USBX CDC Device framework
- r_sce crypto driver
- r_flash_hp flash driver

The application contains the following thread:

- Main Thread

Main Thread

This is the main control thread which handles the following functions:

1. Incoming/outgoing USB data using the USBX CDC Device framework.
2. Decodes the command and calls the appropriate command handler functions, which in turn handle the corresponding command functionalities.

The following commands are handled by the Main Thread:

- WRAPPER_KEY_REQUEST
- WRAPPED_KEY_CERT_PROGRAM
- WRAPPED_KEY_CHALLENGE_RESP

WRAPPER_KEY_REQUEST

This command is handled by the following API function: `handleHrkKeyCreation ()`

This function handles the key generation using SSP Crypto modules. This application supports ECC Key pair generation. The public/private key pairs generated in the application are wrapped. Once the key pair is generated, the public key is sent to the host application to be used for the device certificate.

The wrapped private key is stored internally in the data flash that will later be used for signing the challenge response.

WRAPPED_KEY_CERT_PROGRAM

This command is handled by the following API function: `handleHrkCertProgram ()`

This function handles programming the device certificate received from the host application into the secure region of the internal code flash.

WRAPPED_KEY_CHALLENGE_RESP

This command is handled by the following API function: `handleHrkCertChallengeResp ()`

The intention of this challenge request is to allow the target to prove its ownership of the device private key for the corresponding public key being certified.

This function handles the challenge response request sent by the host application. Once the request is received, it signs the string sent as part of the request using the private key generated as part of WRAPPER_KEY_REQUEST command. The signed string is sent back to the host application for verification. Once the host application receives the signed string, it verifies the signature using the device public key extracted from the device certificate. When the signature validation is successful, the host application will send the device certificate to the device to be stored securely using Security MPU and FAW.

4.3 Operational Overflow

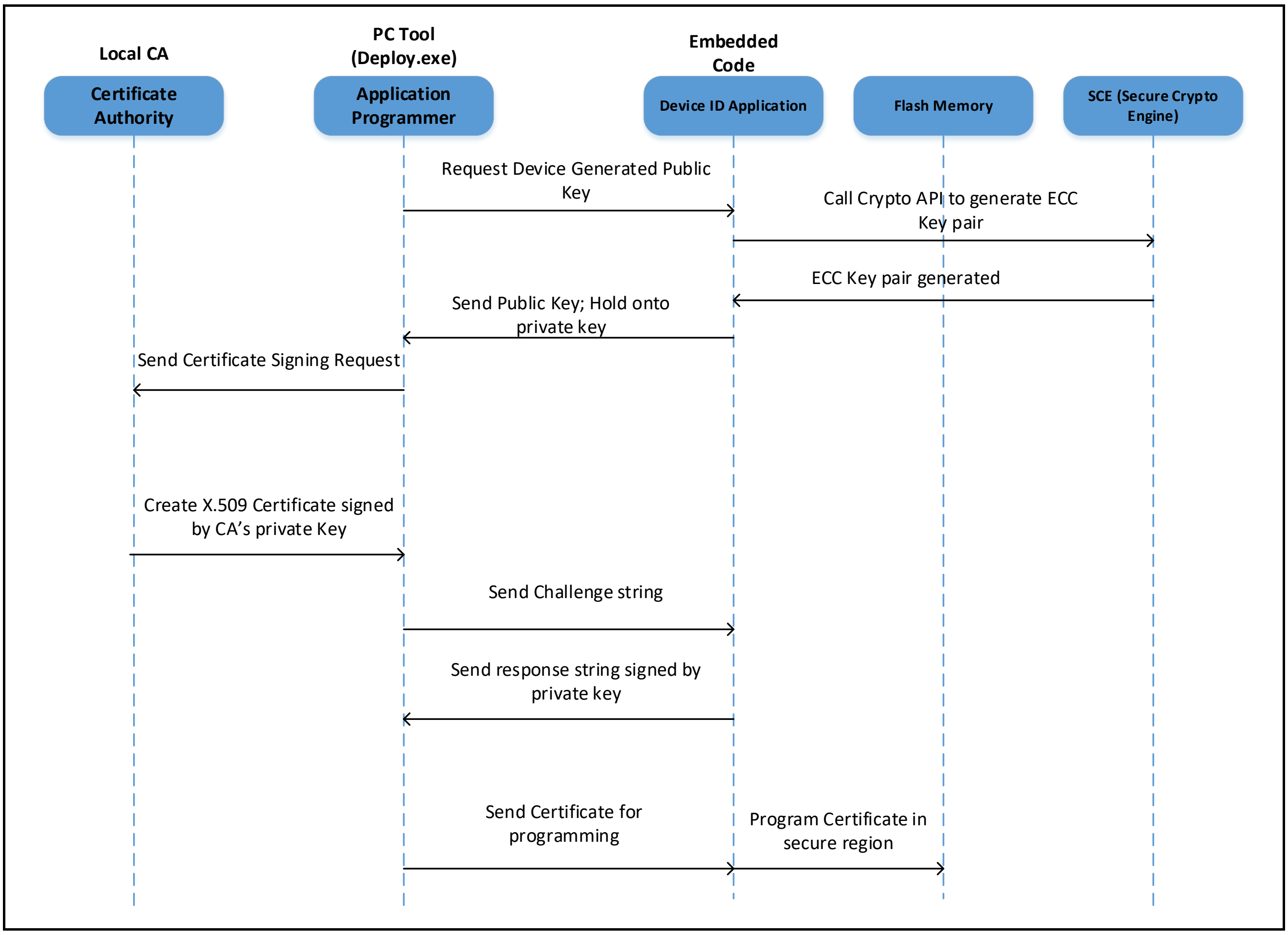

Local CA
Certificate Authority
PC Tool (Deploy.exe)
Application Programmer
Embedded Code
Device ID Application
Flash Memory
SCE (Secure Crypto Engine)
Request Device Generated Public Key
Call Crypto API to generate ECC Key pair
ECC Key pair generated
Send Public Key; Hold onto private key
Send Certificate Signing Request
Create X.509 Certificate signed by CA's private Key
Send Challenge string
Send response string signed by private key
Send Certificate for programming
Program Certificate in secure region

Figure 4. Operational Overflow

This application project consists of two software projects:

- Embedded project running on the PK-S5D9/AE-Cloud2 kit
- Host application running on Windows (7/10) PC.

On powering up the PK-S5D9/AE-Cloud2 kit, the embedded software initializes the platform and the underlying USB CDC stack that is used for communication with the host application running on Windows PC. At end of initialization, the embedded software is waiting for the USB device connect event. Once the user connects the kit to the Windows PC through USB cable, the USB device detection event is processed and the USB CDC instance is created. At this stage, the embedded software is waiting for the commands from the host application.

When the user runs the host utility on the Windows PC, it scans the available COM ports and opens the port to which the PK-S5D9/AE-Cloud2 kit is connected. Once the COM port is opened successfully, it generates a signing key and root CA certificate that will later be used to sign the device certificate. Now, the host application generates the **WRAPPER_KEY_REQUEST** command and sends it to the kit. On receiving this request, the embedded code running on the target kit generates device key pairs, and sends out the public key to the host application. On the host application, it receives the public key from the device, generates a device certificate (signed by CA's signing key).

Before issuing the device certificate, the host application issues a challenge string to the device to prove that the device owns the private key. The embedded software, on receiving the challenge string, signs it using its private key and sends it back to the host application. The host application validates the signature using the device public key and if the validation is successful, the device certificate will be sent to the PK-S5D9/AE-Cloud2 kit to be securely stored using the Security MPU and FAW on the Synergy MCU.

4.4 Securely Storing Device Identity

The two unique device identities created as part of this application are as follows:

- Wrapped ECC private key
- Device certificate

These two device identities need to be securely stored inside the Synergy MCU using the security MPU and FAW to avoid being accessed and modified. The private key generated as part of this application is already wrapped, so we can skip the step to securely store the device key. However, in some cases, users prefer to also store the wrapped key in a secure location to avoid being misused in the device. This can be done using the same steps used to store the device certificate.

Following is the memory map of the current device identity application project.

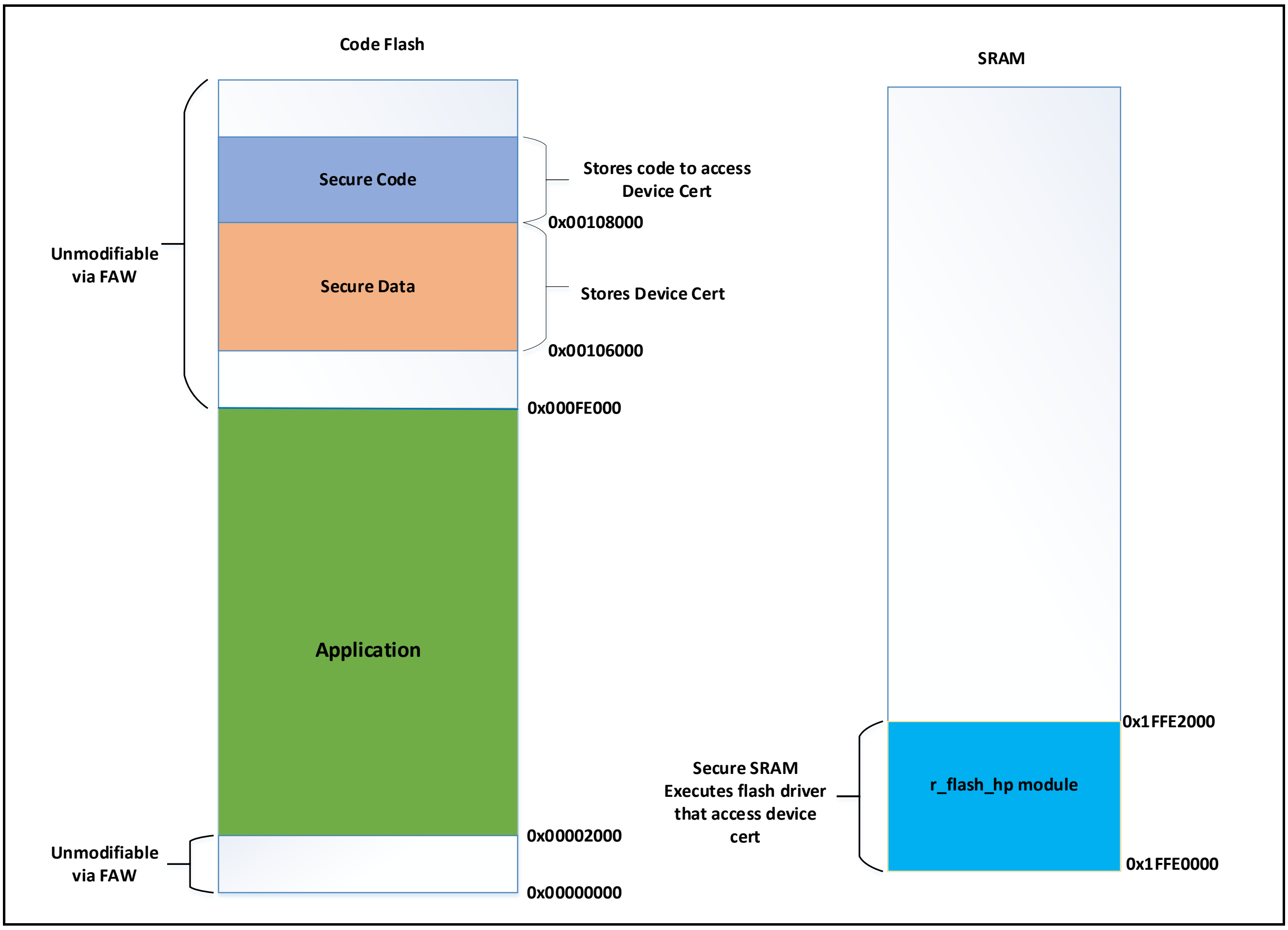


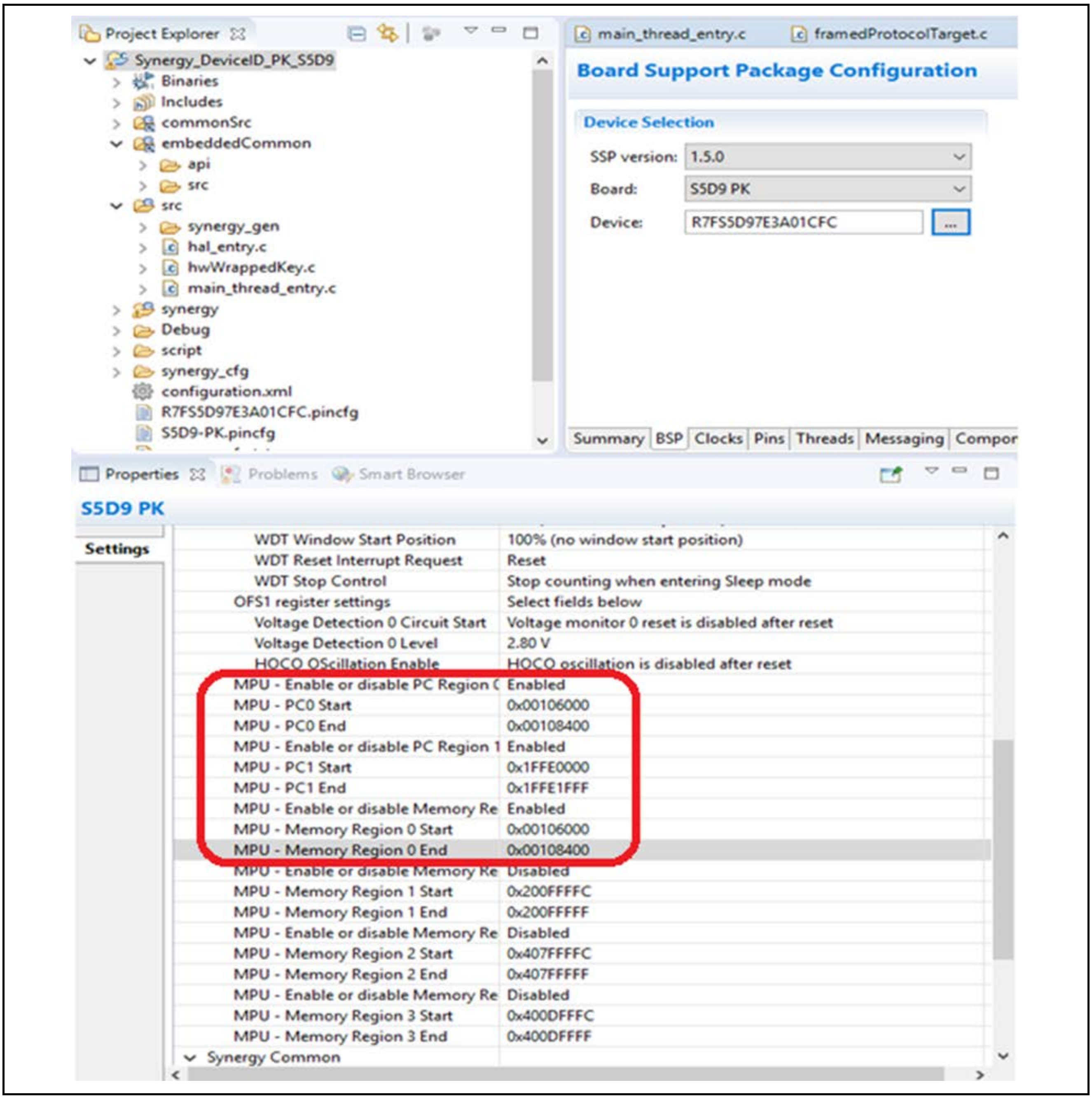
Figure 5. Memory Map used in the Application Project

The left side of Figure 5 shows the memory layout of the code flash in this application project. The green color denotes the area where the application resides, orange color denotes the area reserved for storing the device certificate, and blue color denotes the area reserved for the secure code region that accesses the secure data region.

As shown in the preceding memory map, the address region between 0x0 to 0x2000 contains the Security MPU settings and the address region from 0xFE000 (block aligned address boundary) contains secure data and secure code that needs to be protected from being erased by the user application. To achieve this, the FAW is configured to protect the memory regions from 0x2000 to 0xFE000. By setting the FAW start address to 0x2000 and end address to 0xFE000, the sections outside these address regions are protected from being modified. See the flash_FAW_Set() API found in Synergy_Device_Identity_Solution\embedded\common\src\framedProtocolTarget.c file, which implements the FAW settings.

The right side of Figure 5 shows the SRAM region memory map in this application project. The SSP flash driver module (r_flash_hp) is mapped to the bottom portion of the SRAM that has been configured as a secured SRAM region. This is needed to allow the flash driver to erase and program the secure data region while loading the device certificate.

See the linker script associated with the example application project to see how these secured data/code/SRAM regions are allocated and mapped. Also see the Synergy configurator settings as shown in Figure 6, used in the application project for details on how to configure a region in internal code flash as a secure region. The user can use this example as a reference in their design when allocating a specific region in code flash/SRAM as a secured location for storing the device identities and other application specific modules such as protected IP or 3rd party licensed FW, and so forth.



SSP version: 1.5.0

Board: S5D9 PK

Device: R7FS5D97E3A01CFC

WDT Window Start Position	100% (no window start position)
WDT Reset Interrupt Request	Reset
WDT Stop Control	Stop counting when entering Sleep mode
OFS1 register settings	Select fields below
Voltage Detection 0 Circuit Start	Voltage monitor 0 reset is disabled after reset
Voltage Detection 0 Level	2.80 V
HOCO OScillation Enable	HOCO oscillation is disabled after reset
MPU - Enable or disable PC Region (	Enabled
MPU - PC0 Start	0x00106000
MPU - PC0 End	0x00108400
MPU - Enable or disable PC Region 1	Enabled
MPU - PC1 Start	0x1FFE0000
MPU - PC1 End	0x1FFE1FFF
MPU - Enable or disable Memory Re	Enabled
MPU - Memory Region 0 Start	0x00106000
MPU - Memory Region 0 End	0x00108400
MPU - Enable or disable Memory Re	Disabled
MPU - Memory Region 1 Start	0x200FFFFC
MPU - Memory Region 1 End	0x200FFFFF
MPU - Enable or disable Memory Re	Disabled
MPU - Memory Region 2 Start	0x407FFFFC
MPU - Memory Region 2 End	0x407FFFFF
MPU - Enable or disable Memory Re	Disabled
MPU - Memory Region 3 Start	0x400DFFFC
MPU - Memory Region 3 End	0x400DFFFF

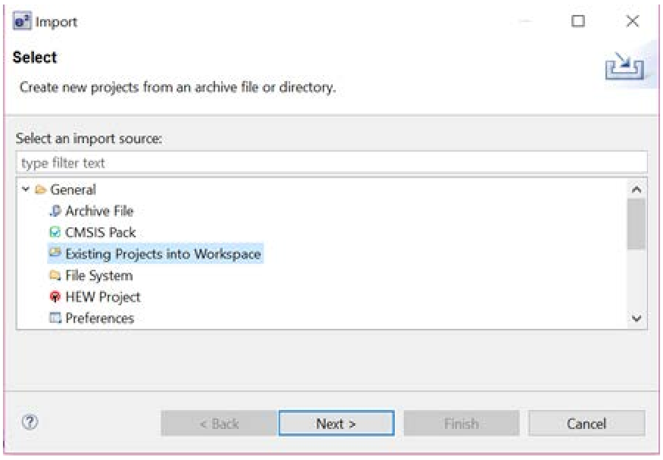
Figure 6. Security MPU settings using ISDE Configurator

5. Running the Device Identity Application Example

5.1 Importing, Building, and Running the Project

The embedded projects are included in the folder **Synergy_Device_Identity_Solution\embedded**. The following instructions will show the user how to import these projects in their workspace.

In ISDE, select **File** -> **Import…** -> **Existing Projects into Workspace** and browse to the above folder in the **Select Root Directory** section:



Import

Select

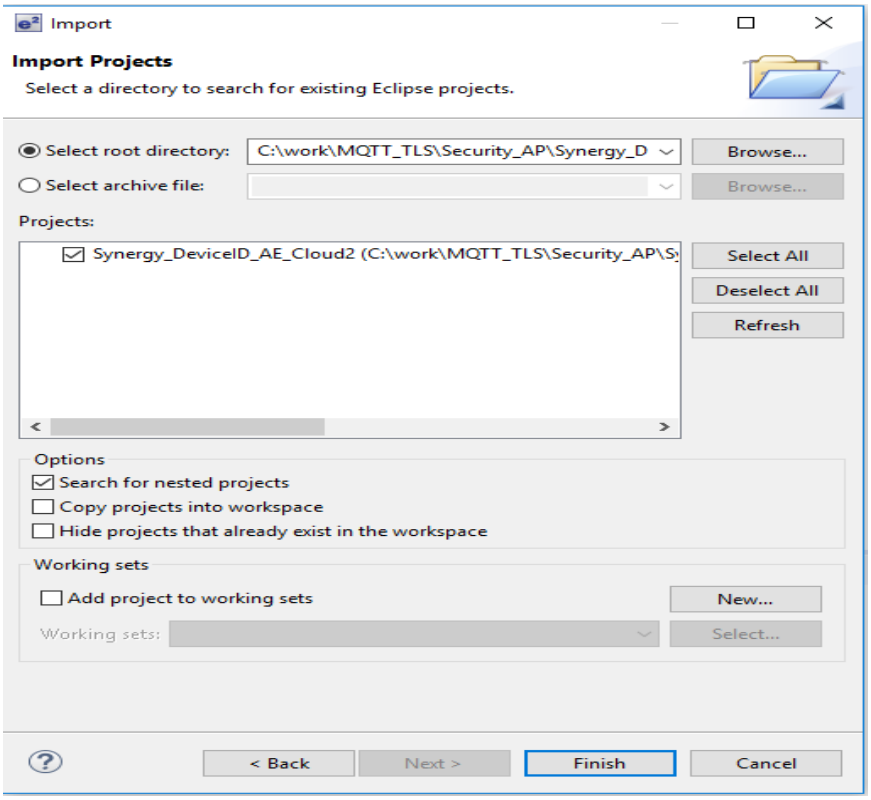
Create new projects from an archive file or directory.

Select an import source:

type filter text

Figure 7. Importing the Project

Select to import all projects as shown in the following figure. **DO NOT CHECK the "Copy projects into workspace" box.**



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:\work\MQTT_TLS\Security_AP\Synergy_D

○ Select archive file:

Projects:

☑ Synergy_DeviceID_AE_Cloud2 (C:\work\MQTT_TLS\Security_AP\S

Options

☑ Search for nested projects

□ Copy projects into workspace

□ Hide projects that already exist in the workspace

Working sets

□ Add project to working sets

Working sets:

Figure 8. Selection for Importing All Projects

5.2 Powering up the Board

To connect power to the board, use the following instructions.

1. Connect the micro USB end of the supplied USB cable to the PK-S5D9 board J19 connector (DEBUG_USB) or AE-Cloud2 board J6 connector (DEBUG_USB).

Note: The kit contains a SEGGER J-Link® On-board (OB). J-Link provides full debug and programming for the PK-S5D9/AE-Cloud2 board.

2. Connect the other end of the USB cable to the USB port on your workstation.

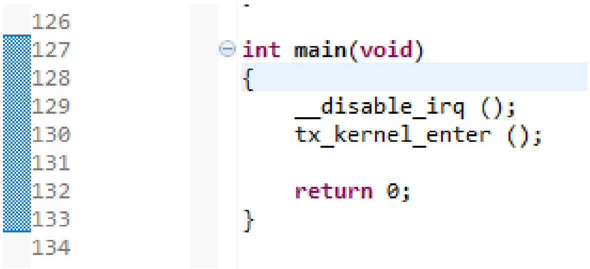
5.3 Debugging using ITM printf

In the embedded application project given in this package, there are number of `printf()` statements to output information about the system. Rather than using the **Renesas Debug Console** in that uses semi-hosted `printf` that is intrusive into the real-time execution of the Synergy processor, the project is set up to use `printf` via SWO. The output of the SWO `printf` can be captured and viewed directly within using the **Live Trace Console** window. However, the performance of the **Live Trace Console** is poor for capturing `printf` output, as every packet (that is, character) that is captured from SWO is time stamped and saved into a trace file in the project. Instead, the SWO Viewer that is shipped with the J-Link software tools can be used to display the output of the SWO `printf` with much better performance.

5.4 Verifying the Demonstration

At this stage, it is assumed that the you followed the instructions in section 5.1 to import, build, and load the application project into the target kit. If not, go back to section 5.1 and follow the steps before moving further in this section.

Now start a debug session for the project and run to `main()`.

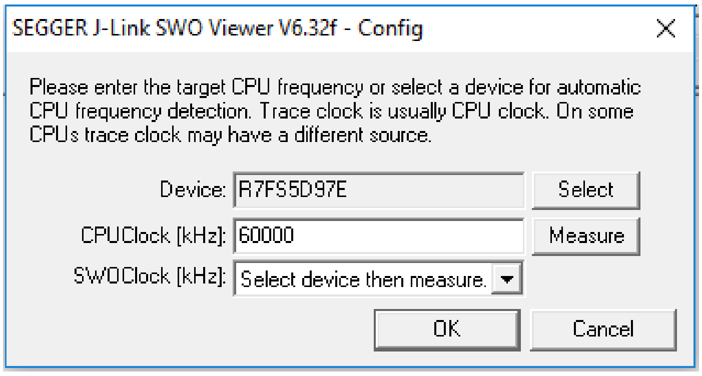


```
int main(void)
{
    __disable_irq ();
    tx_kernel_enter ();

    return 0;
}
```

In the configuration dialog box, choose the device as **R7FS5D97E**, click the **Measure** button, then replace the CPU clock (kHz) with 60000

Note: The J-Link tool used in this application is v6.32f. It is recommended to use J-Link tool v6.32f or greater.



SEGGER J-Link SWO Viewer V6.32f - Config

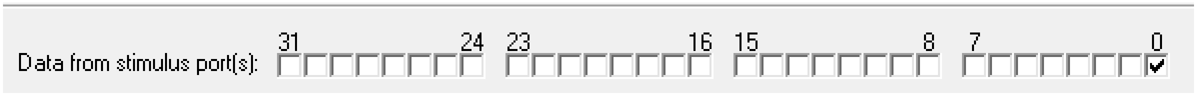
Please enter the target CPU frequency or select a device for automatic CPU frequency detection. Trace clock is usually CPU clock. On some CPUs trace clock may have a different source.

Device: R7FS5D97E

CPUClock [kHz]: 60000

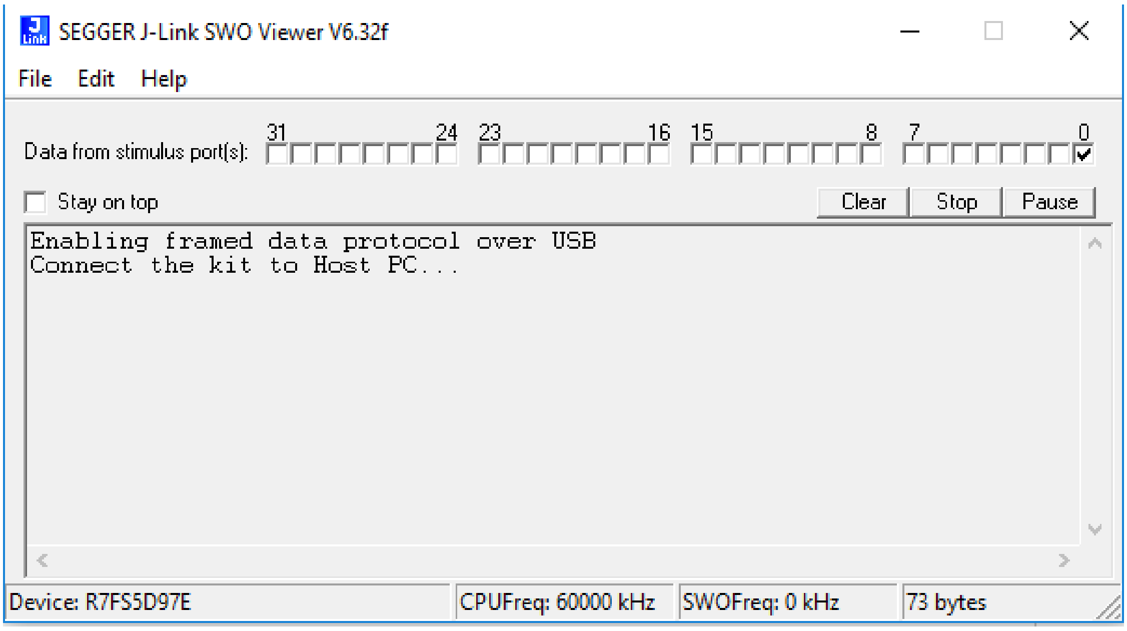
SWOClock [kHz]: Select device then measure.

Click **OK**. Ensure port 0 is selected as the data source.



Data from stimulus port(s): 31 □□□□□□□□ 24 23 □□□□□□□□ 16 15 □□□□□□□□ 8 7 □□□□□□□☑ 0

Resume execution of the project. The output from the SWO `printf` can be seen in the SWO viewer.



```
Enabling framed data protocol over USB
Connect the kit to Host PC...
```

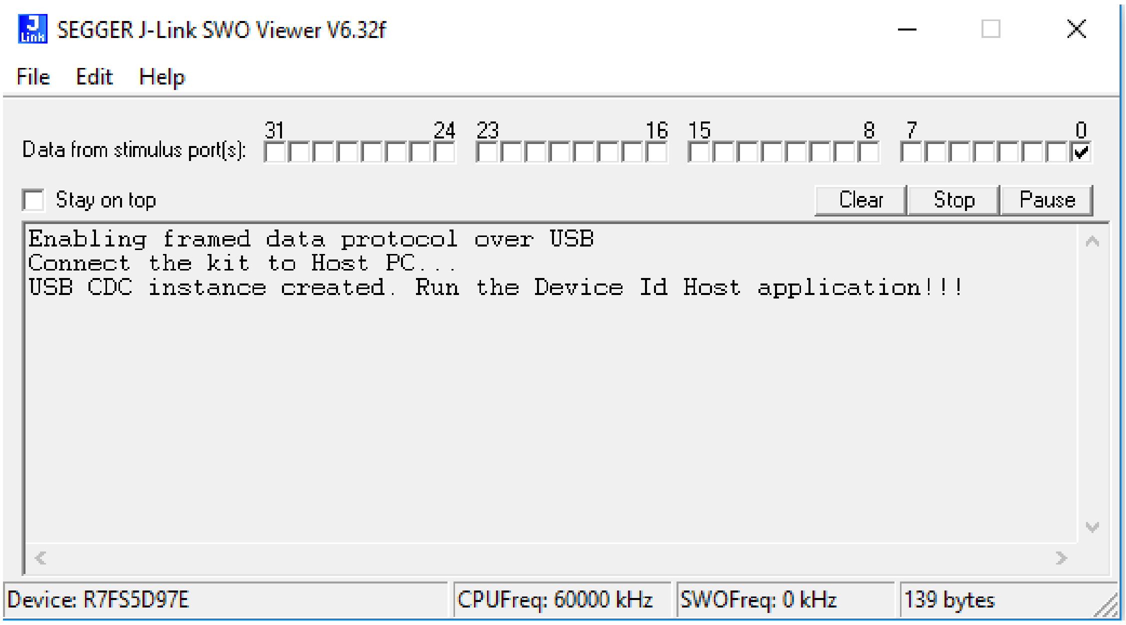
Figure 9. Welcome Message on SWO Viewer

Now, connect one end of a micro USB cable to the USB device port on the target kit and the other end to your window PC. Follow the instructions mentioned in link below to load the Synergy USB CDC Communication driver for the Windows (7/10) PC:

https://www.renesas.com/en-us/products/synergy/software/add-ons/usb-cdc-drivers.html

The target kit will show up as the **Synergy USB Communication Port** in the device manager. Make a note of the COM port number of the target kit from the device manager.

In the SWO viewer, the user will notice the following messages on the console indicating that the USB CDC instance is created and it is ready for the host application.

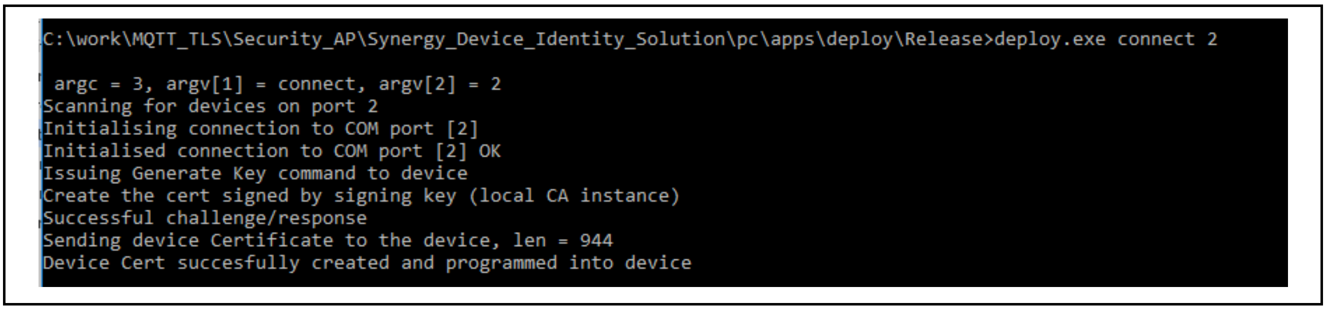


```
Enabling framed data protocol over USB
Connect the kit to Host PC...
USB CDC instance created. Run the Device Id Host application!!!
```

Now, run the host application on the windows PC. To run the application, open the command window in your windows PC and navigate to the folder where this application project is stored. The deploy.exe file will be located under `Synergy_Device_Identity_Solution\pc\apps\deploy\Release` directory.

To run the host application, type the following command on the command window as shown below.

```
deploy.exe connect <COM port Number>
```

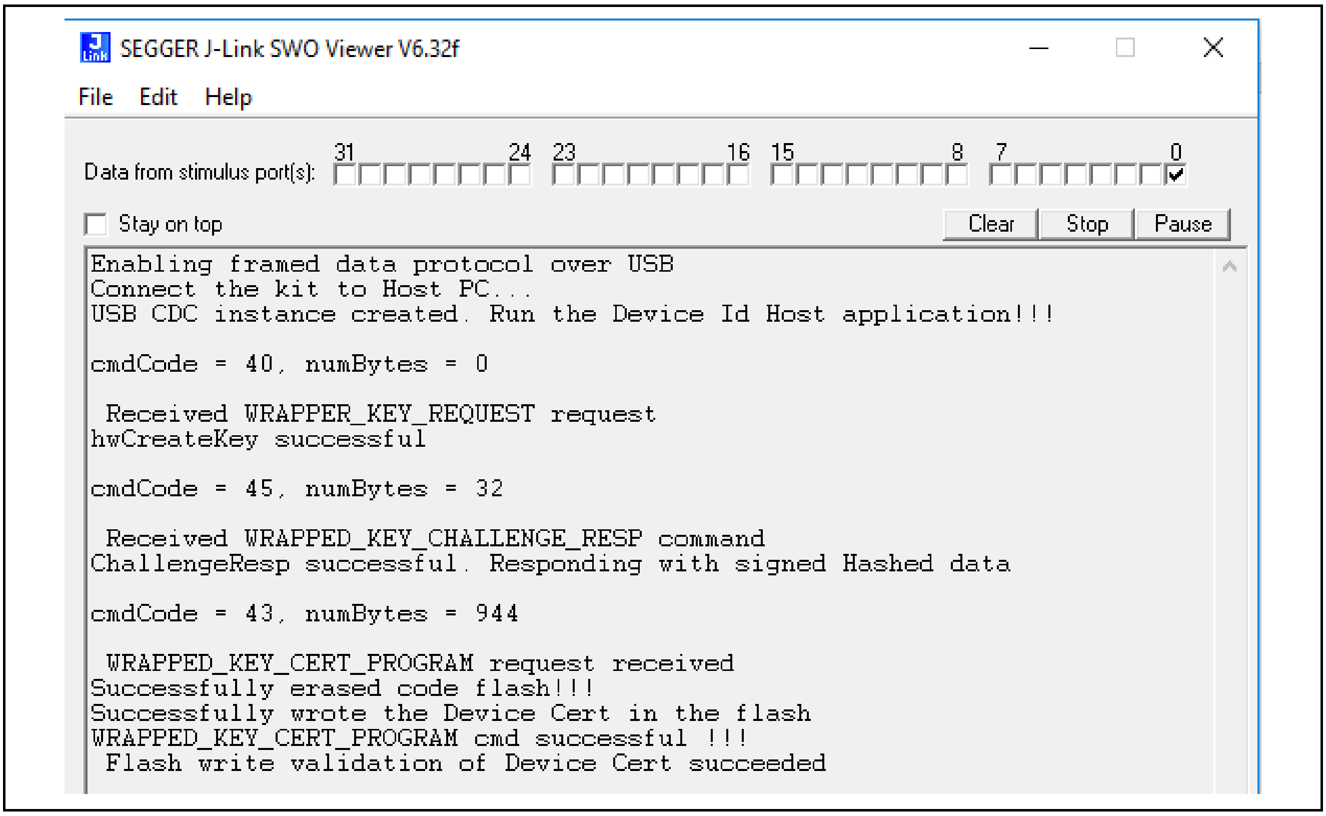


```
C:\work\MQTT_TLS\Security_AP\Synergy_Device_Identity_Solution\pc\apps\deploy\Release>deploy.exe connect 2

 argc = 3, argv[1] = connect, argv[2] = 2
Scanning for devices on port 2
Initialising connection to COM port [2]
Initialised connection to COM port [2] OK
Issuing Generate Key command to device
Create the cert signed by signing key (local CA instance)
Successful challenge/response
Sending device Certificate to the device, len = 944
Device Cert succesfully created and programmed into device
```

Figure 10. Host application console messages

You will notice the console messages from the host application as indicated above and the console message on the SWO viewer as shown below.



```
Enabling framed data protocol over USB
Connect the kit to Host PC...
USB CDC instance created. Run the Device Id Host application!!!

cmdCode = 40, numBytes = 0

 Received WRAPPER_KEY_REQUEST request
hwCreateKey successful

cmdCode = 45, numBytes = 32

 Received WRAPPED_KEY_CHALLENGE_RESP command
ChallengeResp successful. Responding with signed Hashed data

cmdCode = 43, numBytes = 944

 WRAPPED_KEY_CERT_PROGRAM request received
Successfully erased code flash!!!
Successfully wrote the Device Cert in the flash
WRAPPED_KEY_CERT_PROGRAM cmd successful !!!
 Flash write validation of Device Cert succeeded
```

Figure 11. Console Messages on Target Kit

At this stage, the host application starts communicating with the target through the USB-CDC communication interface. User application does the following tasks as shown in section 4.3:

1. Scans for the USB COM port provided by the user. If found, it opens the serial connection.
2. On successful serial connection, it issues the Generate Key command to the target kit.
3. On the device side, the ECC key pairs are generated using SCE crypto modules. The public key is sent back to the host application.
4. The host application creates root CA and signing key to be used to sign the device certificate at the latter stage.
5. Generates a challenge/response string and sends it to the target kit.
6. On successful challenge/response, the host application will sign and send the device certificate to the device.
7. The device certificate will be securely stored in the internal code flash and protected by Security MPU and FAW.

6. Device Identity Application Next Steps

This section provides a high-level overview of phase 2 features planned for device identity application.

Support for 3rd party CA
In phase 1, the application project uses local CA running on a windows PC to create a self-signed root certificate, generate device certificate which is signed by the local CA's signing key. In phase 2, we are planning to update the application project to use a standard 3rd party CA to generate the device certificate.

As part of this feature, we are planning to create a generic interface that can allow plugging in another CA seamlessly.

Support for interface to cloud providers (AWS, Google, Azure)
Current phase of Synergy Cloud connectivity solutions for AWS/Google Cloud/Azure uses the device keys and certificates generated by the Cloud CA. These projects will be updated to use the Renesas supported 3rd party CA for generating device identities.

Support for interface to SBM
After adding the support for standard 3rd party CA, this feature will be integrated with the existing Synergy Secure Boot Manager.

Support for Key injection
Key Injection is a security feature that is meant for use cases where a key is generated external to the MCU device in a secure facility and then injected into the MCU. This support will be documented separately.

7. References

- SSP User Manual (renesassynergy.com/ssp)
- Secure Boot Manager Application Project (https://www.renesas.com/us/en/software/D6002619.html)
- Secure Data-at-rest application

8. Known Issues and Limitations

The host application is tested only on Windows (7/10) PC.

9. Appendix

9.1 Glossary

Term	Meaning
Certificate Authority (CA)	An entity that issues digital certificates according to policy based rules. A CA could be public or private, located in the Cloud, or in the case of an on-premises CA, typically hosted on a secure appliance.
Device Certificate	Certificate uniquely identifying an individual Synergy device. It is digitally signed, asserting that the certificate comes from a known source and has not been modified, and that the device is trusted (Has the Secure Boot Manager installed along with trusted application firmware).
Root of Trust	Roots of trust are highly-reliable hardware, firmware, and software components that perform specific, critical security functions. (https://csrc.nist.gov/projects/hardware-roots-of-trust)
SCE	Secure Crypto Engine – A module in the MCU that provides for efficient, low-power cryptographic acceleration TRNG (True Random Number Generation) and creation and isolation of isolating cryptographic keys.
PKI	Public Key Infrastructure – A set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates, which are typically used to manage secure identity via public key cryptography.
Key Pair	Asymmetric keys are generated in pairs – a public and private key. The private key is held in secret by only one party and can be used to assert that party's identity. The public key is freely distributed and is uniquely associated with the private key.

Term	Meaning
Secure Code	A function or group of functions that reside in a secure region of internal flash, as defined and enforced by the MPUs. These secure functions can access both secure data and non-secure data regions.
Non-Secure code	A function or group of functions which resides in a non-secure region of internal flash. These non-secure codes cannot access the secure region. They can only access the non-secure region.
HRK	Hidden Root Key. This is a unique key stored inside the Synergy MCU.
SBM	Secure Boot Manager, the collection of software that allows the secure download, booting and update of the application binaries on the Synergy MCU.
Challenge String	Randomly generated string at the host application. This string is used by the host application to validate the ownership of the private key by the target.
Unique ID	An identification value, unique to each individual Synergy MCU, that is stored inside the Synergy MCU. The unique ID is used by the SCE when it wraps a key.
Challenge Response String	The response to the challenge string. The Challenge Response String is the signature of the challenge data as created by signing the Challenge String with the receiver's private key.

Website and Support

Visit the following vanity URLs to learn about key elements of the Synergy Platform, download components and related documentation, and get support.

Synergy Software	www.renesas.com/synergy/software
Synergy Software Package	www.renesas.com/synergy/ssp
Software add-ons	www.renesas.com/synergy/addons
Software glossary	www.renesas.com/synergy/softwareglossary
Development tools	www.renesas.com/synergy/tools
Synergy Hardware	www.renesas.com/synergy/hardware
Microcontrollers	www.renesas.com/synergy/mcus
MCU glossary	www.renesas.com/synergy/mcuglossary
Parametric search	www.renesas.com/synergy/parametric
Kits	www.renesas.com/synergy/kits
Synergy Solutions Gallery	www.renesas.com/synergy/solutionsgallery
Partner projects	www.renesas.com/synergy/partnerprojects
Application projects	www.renesas.com/synergy/applicationprojects
Self-service support resources:	
Documentation	www.renesas.com/synergy/docs
Knowledgebase	www.renesas.com/synergy/knowledgebase
Forums	www.renesas.com/synergy/forum
Training	www.renesas.com/synergy/training
Videos	www.renesas.com/synergy/videos
Chat and web ticket	www.renesas.com/synergy/resourcelibrary

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct.24.19	—	First release document
1.01	Mar.27.19	1	Specified developers of applications with SCE modules using Renesas Synergy™ S5D9, S5D5, and S5D3 MCUs as audience.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.