RX family

How to use AES Cryptography with TSIP

Introduction

Trusted Secure IP (TSIP) is a comprehensive security library that bundles many cryptography algorithms. This application note is targeted for AES cryptography implementation on RX family device using TSIP. It provides sample projects which includes BSP configuration, TSIP configuration, and then runs TSIP API to perform AES encryption and decryption in ECB and CBC mode.

The project includes pre-wrapped UFPK as an example for easy demonstration, but it can be replaced with user wrapped UFPK from Renesas DLM server. Please do not use pre-wrapped UFPK in your product.

Objectives

The purpose of this application note is to provide a sample implementation of the AES 128 cryptography in ECB and CBC mode using Renesas TSIP library. The smart configurator is used to add and configure TSIP and other peripheral. The provided example code runs on Renesas Starter Kit+ for RX65N (RX65N RSK) and RX72N Envision Kit. However, it can easily be ported to any Renesas microcontroller which has TSIP.

Target Device

The following is a list of devices that are currently supported by this example program:

RX65N: R5F565NE

RX72N: R5F572NN

Operating Environment

Operation was confirmed in the following environments. The version of Integrated Development Environment and Toolchains are same to the version which is described in the supplication note RX Family TSIP (Trusted Secure IP) Module Firmware Integration Technology (R20AN0371).

Integrated Development Environment	e^2 studio
Toolchains	CCRX Compiler
Target Board	Renesas Starter Kit+ for RX65N-2MB (P/N: RTK50565N2CxxxxxBE) RX72N Envision Kit (P/N: RTKRX72N0CxxxxxBJ)
Debuggers	E2 Lite emulator
TSIP	The version Included in this package
Tera-Term	Version 4.104

Contents

1. Overview

1.1 AES Algorithm

The AES algorithm is a symmetric block cipher that can encrypt and decrypt information. Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext. The NIST initially announced this as a process for the selection of a new encryption algorithm to replace the DES(*1) algorithm.

The Feistel and SPN(*2) structures are representative structures for block-cipher algorithms. A SPN structure has been adopted for the AES algorithm. In the SPN structure, substitution and permutation are repeatedly applied (round processing). Randomization by the SPN structure is typically more efficient than that by the Feistel structure, which means that fewer rounds of processing are required and processing is faster overall.

*1) DES: The Data Encryption Standard (DES) algorithm is a symmetric block cipher using cipher keys. This standard was adopted as the US government standard in 1977.

*2) SPN: The Substitution Permutation Network (SPN) is a type of block cipher architecture that is adopted for the AES algorithm.

The AES was released by the National Institute of Standards and Technology (NIST) in November 2001. There are currently three different versions of AES; all of them have a block length of 128 bits, whereas key length allowed to be 128, 192 or 256 bits. The larger the key size used, the more difficult it is to break the algorithm and obtain the encrypted data. For the purposes of this application note, only a key length of 128 bits is discussed.

In the AES algorithm, 128, 192, or 256-bit cipher keys are used to generate a 128-bit ciphertext. A different key called round key is used in each round, and is generated from the given 128, 192, or 256-bit cipher key.

1.2 Block Cipher Modes of Operation

The NIST SP 800-38A includes several modes of block cipher to generate ciphertext blocks by encrypting plaintext blocks. The project explained in this document supports ECB and CBC modes.

- ECB (Electronic Code Book) Mode: The ciphertext is created by simply encrypting each block of plaintext with the cipher key to form the corresponding block of ciphertext.

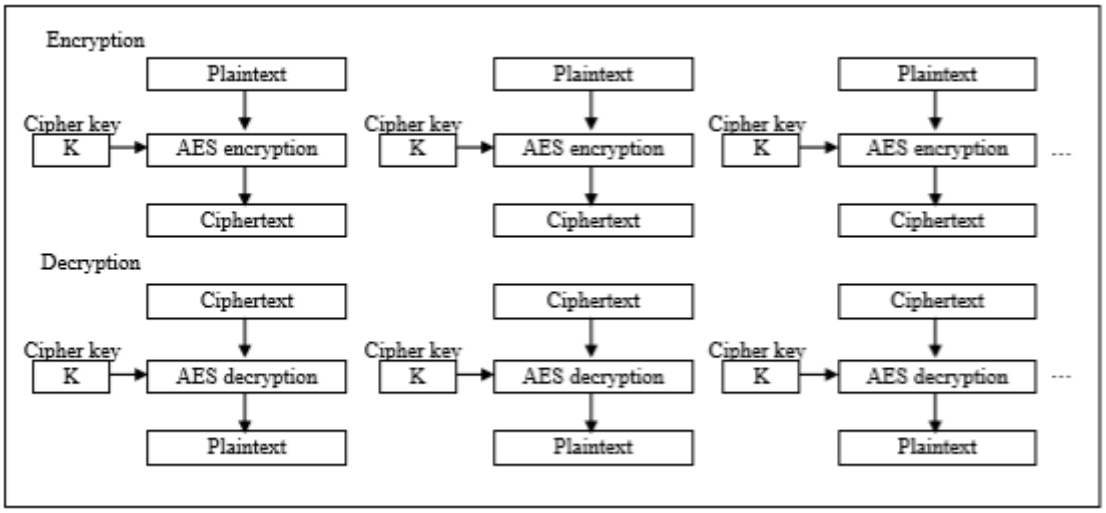


Figure 1-1 Electronic Code Book (ECB)

- CBC (Cipher Block Chaining) Mode: Each block of plaintext to be encrypted is XORed with the previous block of ciphertext. The result is then encrypted with the cipher key to create the corresponding block of ciphertext. Since there is no previous block of ciphertext for the first block, an initialization vector (IV) is used in place of the previous block of ciphertext block.

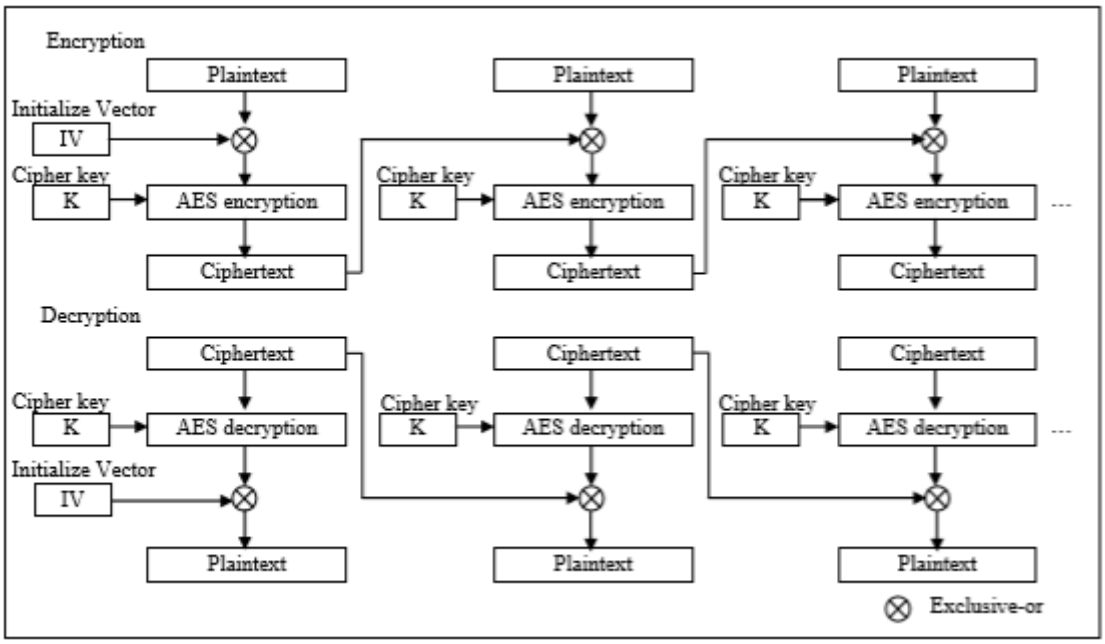


Figure 1-2 Cipher Block Chaining (CBC)

2. TSIP Overview

2.1 Definition of Terms

Terms used in this document are defined below.

Table 2-1 Terminology

Term	Description
user key	The key that the user uses as input to cryptographic functions on the device. Generated by the user.
encrypted key	Key information generated by AES128-encrypting, MAC-adding and the user key using a UFPK. Generated by Security Key Management Tool.
Wrapped Key	Data consisting of key information, such as the user key, that has been converted into a form that is usable by the TSIP driver. Generated by TSIP.
UFPK	The key needed to generate an encrypted key from a user key. Generated by the user.
hidden root key (HRK)	Key that exists only internally in the TSIP and in a secure room at Renesas such as DLM server.
DLM server (https://dlm.renesas.com/)	The Renesas key management server. “DLM server” is short for “Device Lifecycle Management server.” It is used for UFPK wrapping.

2.2 Trusted Secure IP (TSIP)

The Trusted Secure IP (TSIP) block within the RX family creates a secure area inside the MCU by monitoring for unauthorized access attempts. It ensures that the encryption engine and encryption key can be utilized safely. The encryption key, the most important element in reliable and secure encryption, is linked to a unique ID and stored in the flash memory in a safe, undecipherable format.

Each TSIP devices include a safe area, which holds: an encryption engine, storage for raw keys, and a hidden root key, used to encrypt keys.

TSIP hardware generates wrapped key from encrypted user key inside the TSIP which is device-specific, and tied to a unique ID. Hence, the key from one device will not work on a different device. The TSIP driver software allows applications access to the TSIP hardware.

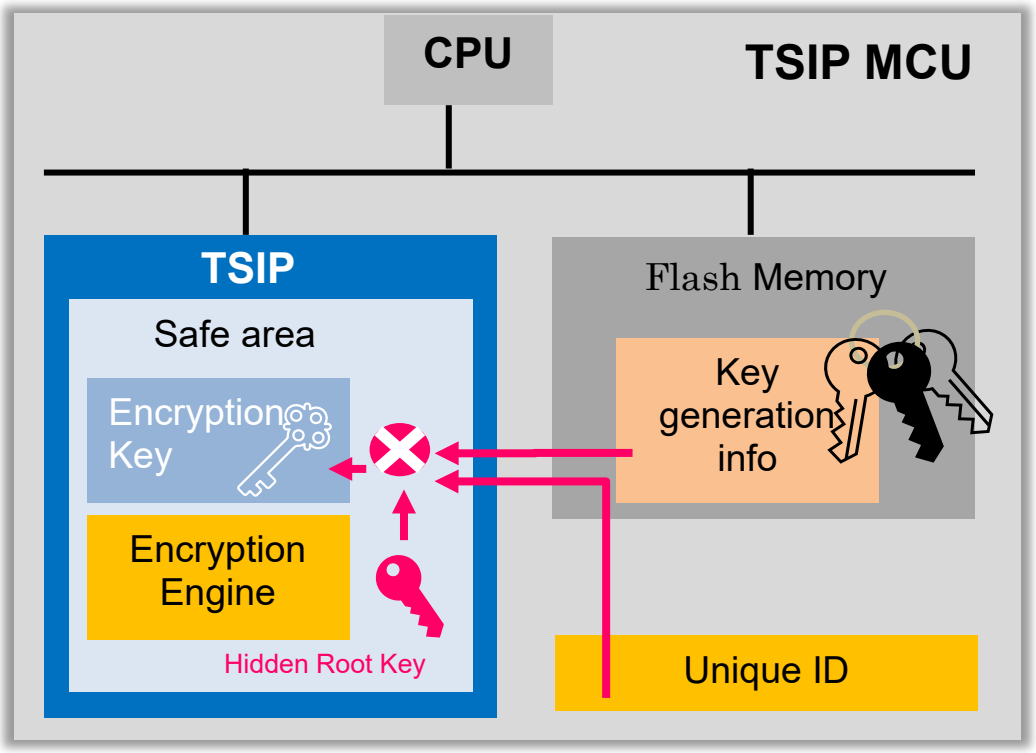


Figure 2-1 TSIP Hardware

2.2.1 Trusted Secure IP driver Package

The TSIP driver binary can be download from Renesas website. The complete source code package can be requested by contacting local FAE. This includes TSIP drivers, application note RX Family TSIP (Trusted Secure IP) Module Firmware Integration Technology, sample code. The directory structure of the package is described in “1.3 Structure of Product Files” of the above application note.

2.2.2 TSIP Installation

The TSIP driver is provided as a FIT module that needs to be downloaded into the FIT module directory of the IDE before it can be used with the e^2 stduio RX smart configurator for project creation. The TSIP FIT module is included in the FITModules folder of the TSIP package and need to be extracted and copied to the FIT module installation directory.

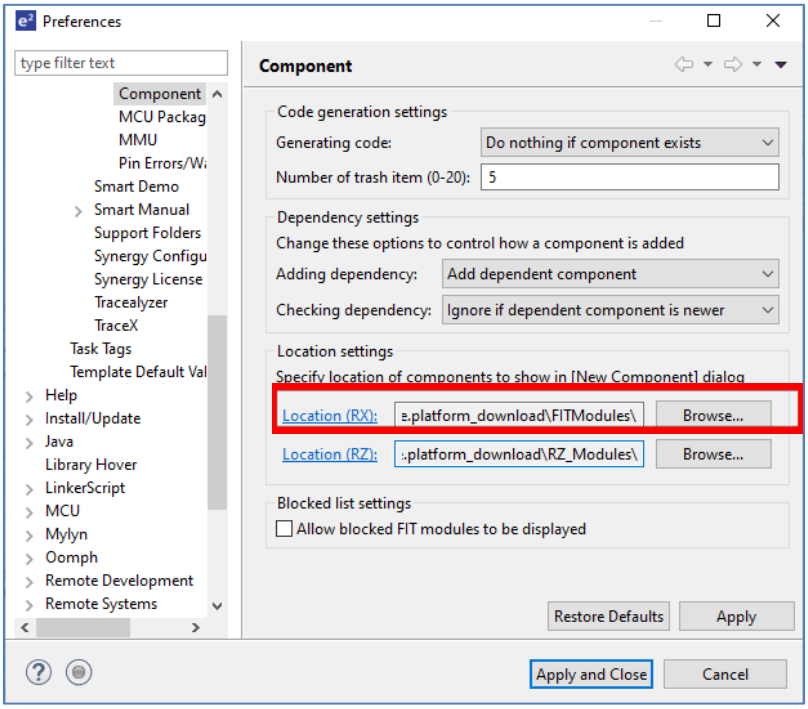
The following three files from the TSIP package need to be copied into the FIT module installation folder.

1. r_tsip_rx_v1.xx.xml

2. r_tsip_rx_v1.xx.zip

3. r_tsip_rx_v1.xx_extend.mdf

The FIT module installation folder location can be found in the “Component” tab as shown in the figure below in e^2 studio inside the Smart Configurator wizard.

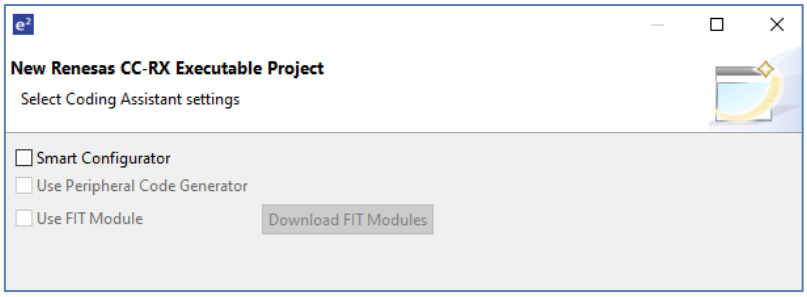


3. AES Cryptography Project and Configuration

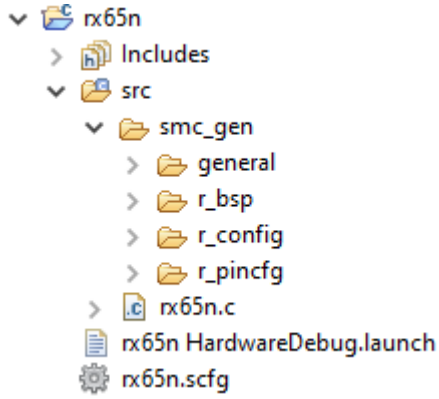
3.1 Creating Project with TSIP

The easiest way to create a project with TSIP is by using e^2 studio project wizard and then use the smart configurator to add the required peripheral.

- In e^2 studio, go to “File->New->C/C++ project”.
- Select “Renesas RX->Renesas CC-RX C/C++ Executable Project.
- Enter the project name and directory.
- Select Compiler and target device (For RX65N RSK: R5F565NEHDFC). Make sure device selected has the security feature to use the TSIP. For instance, RX65N Family R5F565NEHDFC supported HW Encryption function
- Put a check mark on the “Smart configurator” to use the code generator feature that allows adding the FIT module driver.



- Click finish to complete the project creation. The sample project will look like this with *.scfg file for the smart configurator.

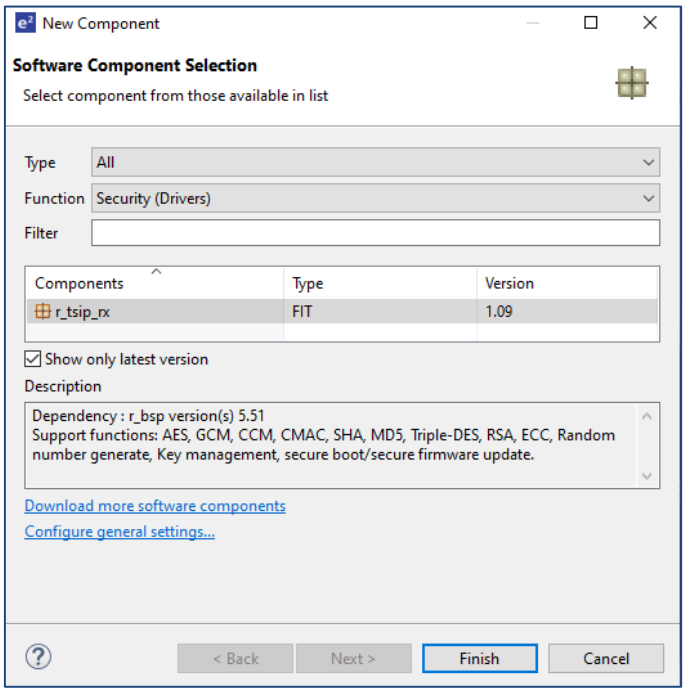


3.2 Adding TSIP FIT module

The project wizard only creates basic project with bsp. The TSIP driver and other necessary peripheral drivers needed for the application code can be added using smart configurator.

The TSIP driver has dependency on the Flash FIT module (r_flash_rx) to store wrapped keys in the data flash and this needs to be added along with the TSIP driver. The Flash FIT module includes API to program, erase, write and blank check the Code and/or Data flash. Any other FIT module needed for the application project can also be added using the smart configurator.

The smart configurator can be started by double clicking *.scfg file in the project from “Project Explorer” window. By going to “Component” tab and then clicking add button allows adding any FIT module. The component can be filtering by the function needed for the application. The example below shows TSIP driver (r_tsip_rx) addition by filtering the function by the “Security (Drivers)”. Using same process, Flash FIT module can be added.



The demo code communicates through the terminal using UART interface. Hence, SCI FIT module (r_sci_rx) and ByteQ FIT module (Required for SCI driver) is also added to the project. The ByteQ module allocates a queue control block in an Open() call to handle indexing for buffers provided by the user application. It also includes functions to put and get data from the buffer queue, inspect the number of bytes available or in use, and to flush the queue. There is no limit to the number of queues supported, and queue control blocks can be allocated statically at build time or dynamically at run time. The RX65N RSK board uses SCI channel 8 and the RX72N Envision Kit board uses SCI channel 2 for transmission and reception.

3.3 BSP configuration

By default, BSP disables security feature and needs to be enabled in the bsp configuration file generated by the e^2 studio project wizard to use the TSIP. The security configuration can be found in the bsp configuration file “r_bsp_config_h .

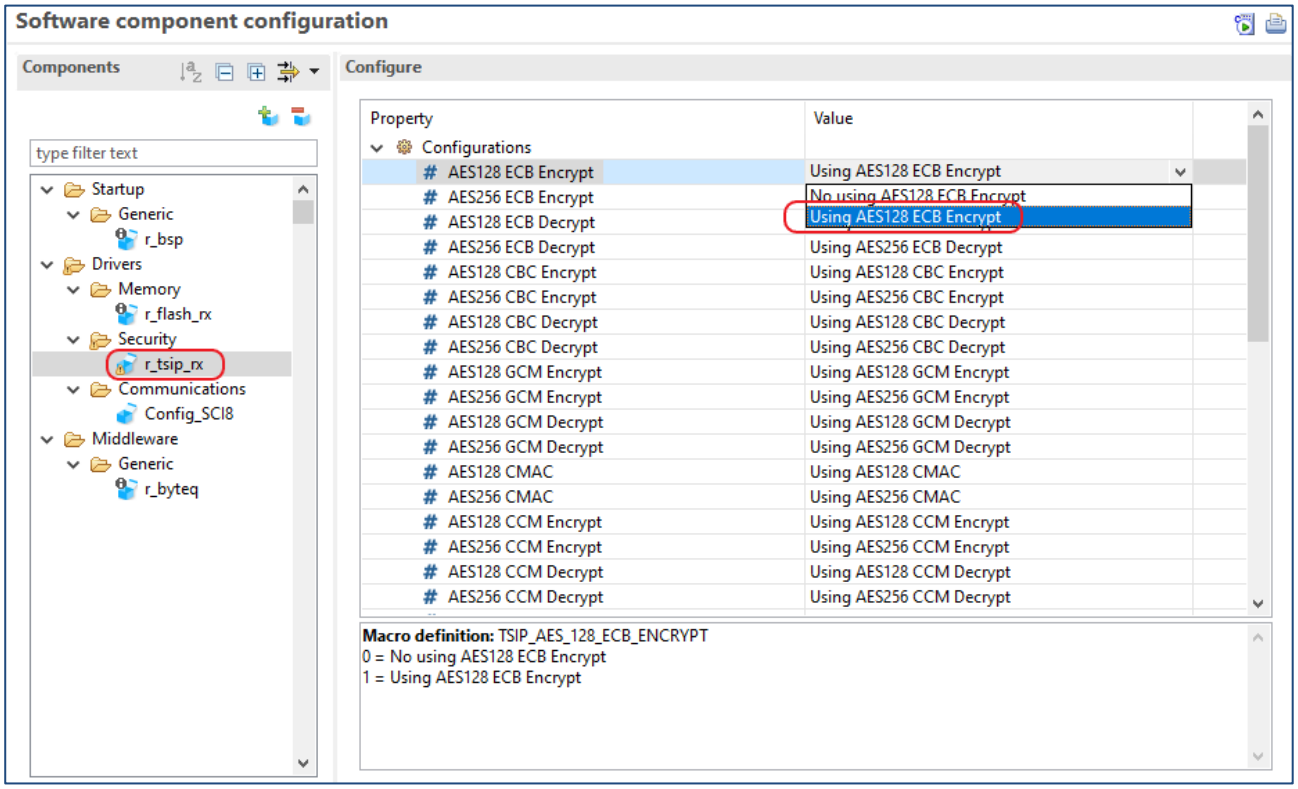
Make sure to set the below macro as described below before using TSIP API function. Please set the macro manually although the configuration file is not edited manually originally.

Table 3-1 The setting value to bsp configuration file

Using MCU	The macro to set	The value to set
RX65N	BSP_CFG_MCU_PART_ENCRYPTION_INCLUDED	true
RX72N	BSP_CFG_MCU_PART_FUNCTION	0x11

3.4 TSIP driver configuration for AES usage

By default, TSIP FIT module addition enables all the features from the TSIP. This includes all the cryptography, hashing, random number generation etc. So, when “Code Generator” generates the code, it generates driver for all the Cryptography apart from the AES function that is needed for the application. So, for application that requires AES cryptography only, the AES component can be enabled, and all other components can be marked as “No using…” from the drop-down menu for each component in the smart configurator as below to reduce memory usage.
Note that this function is available only the source code package.



3.5 Key Wrapping and Wrapped Key

TSIP driver does not accept raw user key as input, hence key needs to be wrapped and encrypted. The method to convert user key into wrapped key and to write in RX MCU is explained below.

To use the user key more securely, use the Renesas DLM server (https://dlm.renesas.com/) and the Security Key Management Tool to wrap the user key used by the TSIP driver according to the following procedure.

1. First, prepare two keys, a user key and a UFPK. The user key is the key used for cryptographic processing, and the UFPK is the key used to wrap the user key.
2. Next, generate the wrapped UFPK with wrapping the raw UFPK on the Renesas DLM server. The UFPK file which format is acceptable by Renesas DLM server can be generated by Security Key Management Tool. Please refer User's Manual of Security Key Management Tool and application note RX Family TSIP (Trusted Secure IP) Module Firmware Integration Technology to see how to generate the UFPK file. And please refer the FAQ at the top of the DLM server web site to see how to use the DLM server.
3. Then enter the wrapped UFPK, the raw UFPK, and the user key into the Security Key Management Tool to output the wrapped UFPK and the user key wrapped with the UFPK (encrypted key). Implement these keys to the encryption key files (key_data.c and key_data.h). Please see the User's Manual of Security Key Management Tool and application note RX Family TSIP (Trusted Secure IP) Module Firmware Integration Technology to see more detail.

The created encryption key files (key_data.c and key_data.h) must be included in the project. The advantage of using a wrapped key is that the user can guarantee the security of the raw user key because the program does not need to include the raw user key.

Each TSIP API requires "Wrapped Key ". This should be generated in the first step before other APIs are called. Wrapped Key is generated by the Wrapped Key generation API based on the wrapped UFPK and encrypted key included in key_data.c and key_data.h. The generated Wrapped Key can be installed in the data flash area (0x00100000), etc.

The key associated with the Unique ID is used to generate the Wrapped Key. Since the Unique ID is a device-specific value, the Wrapped Key will be different for each device even if it is generated from the same user key. Therefore, even if the Wrapped Key is read from the device by an attacker, it is not available on other devices and does not pose a security threat to the system.

The sample project uses following user key and UFPK. For a quick demonstration, the key_data.c and key_data.h are pre-generated by using the Renesas DLM server and the Security Key Management Tool and has been added in the project.

Table 3-2 Keys to be used

Key Type	Raw Key (hexadecimal)
User key (AES-128 bit)	11111111222222223333333344444444
UFPK	22222222222222222222222222222222111111111111111111111111111111111

4. Demo Project

The demo program executes user command input in the prompt on terminal window.

4.1 Demo Commands

Table 4-1 shows the list of the user command supported by the demo. It is to note that the commands are case sensitive.

Table 4-1 List of Commands

Command	Description
ecbdemo <plain text>	Encrypt 16-byte hexadecimal input plain text and output Cypher text in ECB mode
cbcdemo <plain text>	Encrypt 16-byte hexadecimal input plain text and output Cypher text in CBC mode
function	Test the AES ECB/CBC APIs with sample data and outputs the result

4.2 List of Files and Folder

Table 4-2 list the files and folders in the project. The peripheral drivers are generated from the smart configurator.

Table 4-2 List of Files and Folders

Folder Name Name	File/Folder Name	Description
src/genkey	euk_aes128.c euk_aes128.h	Key data generated from Security Key Management Tool
src/smc_gen	• general • r_bsp • r_byteq • r_config • r_flash_rx • r_pincfg • r_sci_rx • r_tsip_rx	Code generated from Smart Configurator • Includes BSP • Peripheral drivers • Pin Configuration • BSP and Peripheral Configuration
src	main.c	main function
src	key_data.c key_data.h	Wrapped User Key and UFPK generated from Security Key Management Tool
src	tsip_sample_aes.c	Sample program for AES function test
src	secure_boot.c secure_boot.h	Sample program for generating AES wrapped key and installing it in the data flash
src	tsip_sample.c tsip_sample.h	utility functions for outputting the data

4.3 List of TSIP API Functions used for AES implementation

The code uses following TSIP API function for AES 128 in CBC and ECB mode implementation. These functions are associated with TSIP initialization, wrapped key generation, and encrypting and decrypting the input data in CBC and ECB mode.

Table 4-3 API function for AES

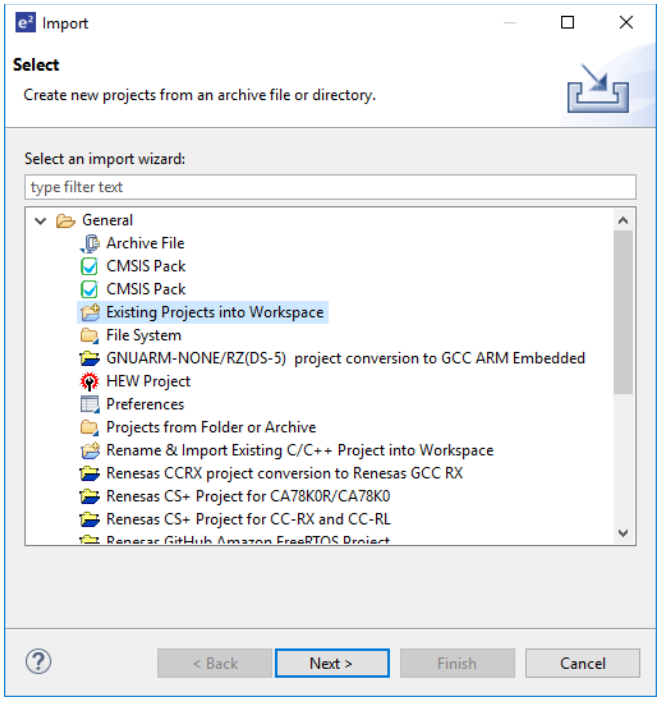
Type	API	Description
API functions for TSIP initialization	R_TSIP_Open	Enables TSIP functionality.
	R_TSIP_Close	Disables TSIP functionality
API for generating user wrapped key data	R_TSIP_GenerateAes128KeyIndex	Generates a 128-bit AES user wrapped key.
API for ECB mode encryption and decryption	R_TSIP_Aes128EcbEncryptInit	Prepares to encrypt data in AES128-ECB mode using a 128-bit AES user wrapped key.
	R_TSIP_Aes128EcbEncryptUpdate	Encrypts data in AES128-ECB mode.
	R_TSIP_Aes128EcbEncryptFinal	Performs final processing for encryption in AES128-ECB mode.
	R_TSIP_Aes128EcbDecryptInit	Prepares to decrypt data in AES128-ECB mode using a 128-bit AES user wrapped key.
	R_TSIP_Aes128EcbDecryptUpdate	Decrypts data in AES128-ECB mode.
	R_TSIP_Aes128EcbDecryptFinal	Performs final processing for decryption in AES128-ECB mode
API for CBC mode encryption and decryption	R_TSIP_Aes128CbcEncryptInit	Prepares to encrypt data in AES128-CBC mode using a 128-bit AES user wrapped key.
	R_TSIP_Aes128CbcEncryptUpdate	Encrypts data in AES128-CBC mode.
	R_TSIP_Aes128CbcEncryptFinal	Performs final processing for encryption in AES128-CBC mode.
	R_TSIP_Aes128CbcDecryptInit	Prepares to decrypt data in AES128-CBC mode using a 128-bit AES user wrapped key.
	R_TSIP_Aes128CbcDecryptUpdate	Decrypts data in AES128-CBC mode
	R_TSIP_Aes128CbcDecryptFinal	Performs final processing for to decryption in AES128-CBC mode.

5. Build and Run the Project

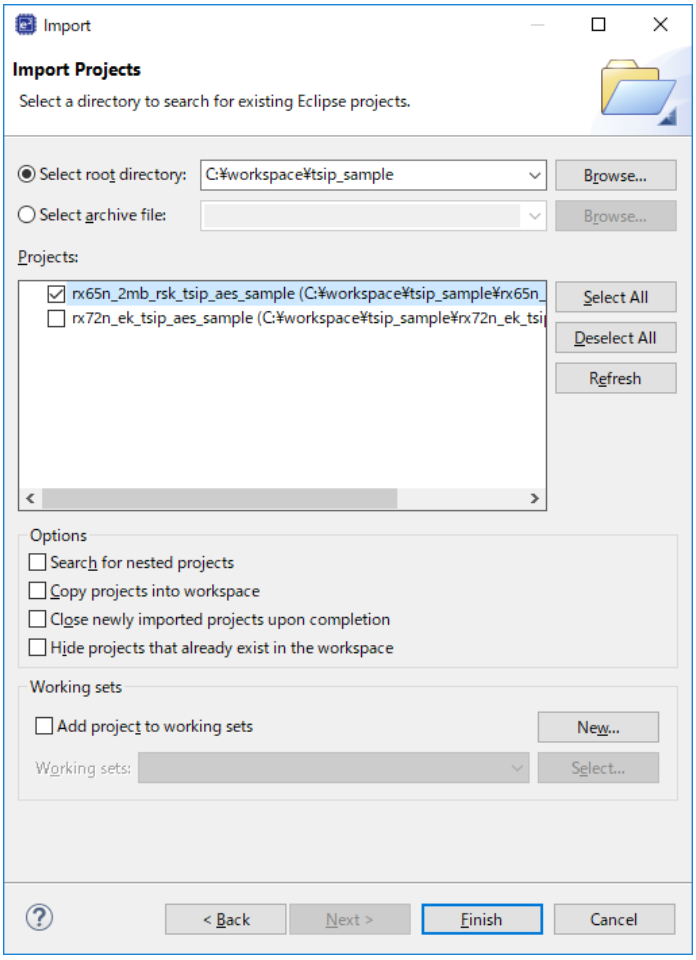
In this section, the procedure to use RX65N RSK is explained. Please interpret the project name to the project name which is actually used when other RX family device is used.

5.1 Import and Build the Project

Open e^2 studio to a workspace of your choosing and click File > Import. Then select General > Existing Projects into Workspace as shown below.



Select “Select root directry”, click “Browse…” button and navigate to the directory where projects are downloaded. Select the rx65n_2mb_rsk_tsip_aes_sample, ensure your screen looks like below before clicking finish. This will import the “rx65n_2mb_rsk_tsip_aes_sample project” into your workspace. Now build the project “Project->Build Project” .


Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
C:¥workspace¥tsip_sample
Browse...
Select archive file:
Browse...
Projects:
rx65n_2mb_rsk_tsip_aes_sample (C:¥workspace¥tsip_sample¥rx65n_
rx72n_ek_tsip_aes_sample (C:¥workspace¥tsip_sample¥rx72n_ek_tsi
Select All
Deselect All
Refresh
Options
Search for nested projects
Copy projects into workspace
Close newly imported projects upon completion
Hide projects that already exist in the workspace
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Finish
Cancel

5.2 Setting up the hardware

Connect the hardware as shown in the picture below so that the code can be download using E1 or E2Lite emulator. To prevent shorts, ensure power is disconnected before making any physical changes to the board. Also, when RX65N RSK is used, make sure SW4-1 is OFF to have the MCU in the Single chip mode. When RX72N Envision Kit is used, make sure SW1-2 is OFF to set emulator use enabled.

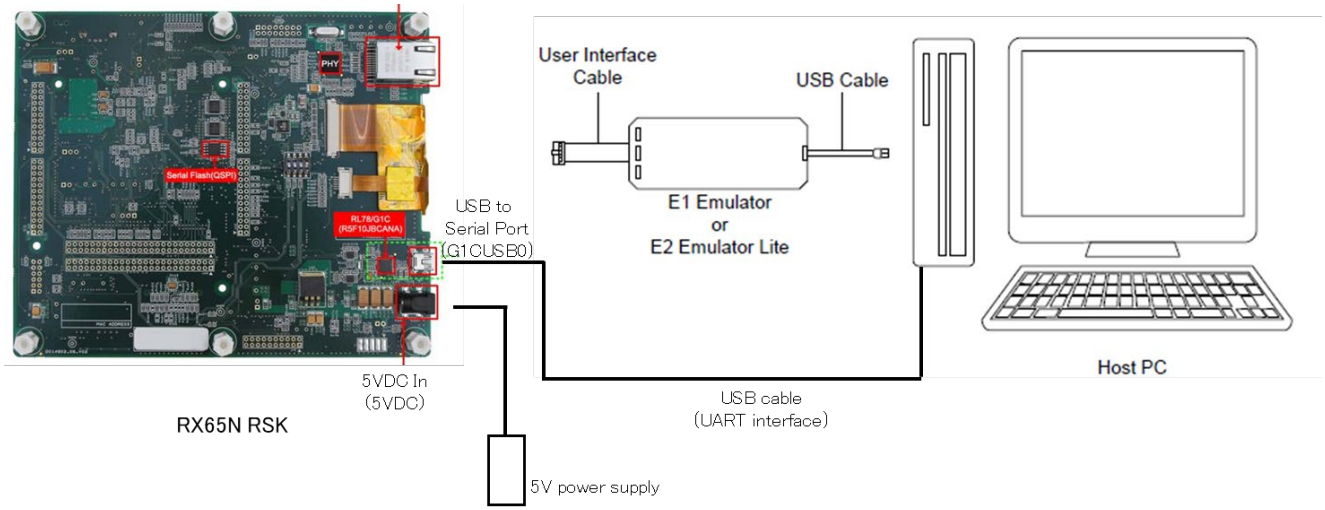


Figure 5-1 Connection Diagram of RX65N RSK

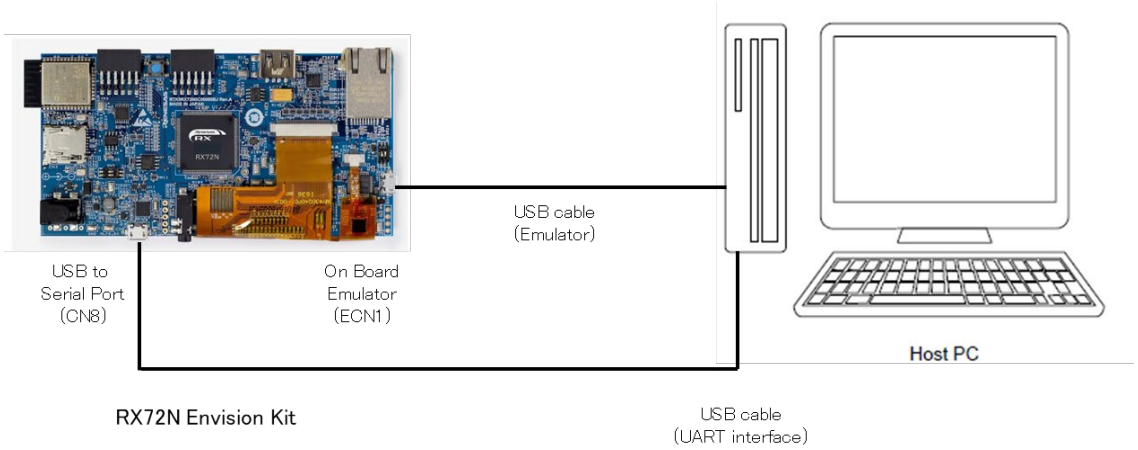


Figure 5-2 Connection Diagram of RX72N Envision Kit

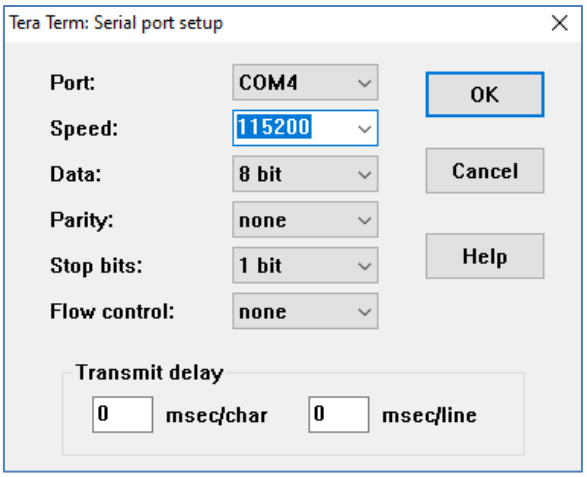
5.2.1 Terminal Software Communication

Connect USB cable to PC for UART communication with Terminal software. Configure Tera-term for “Serial” communication. First, set New-line of Transmit to “CR” in Terminal Setup. Next, set up serial port as below:

Baud rate: 115200

Parity: None

Flow Control: None

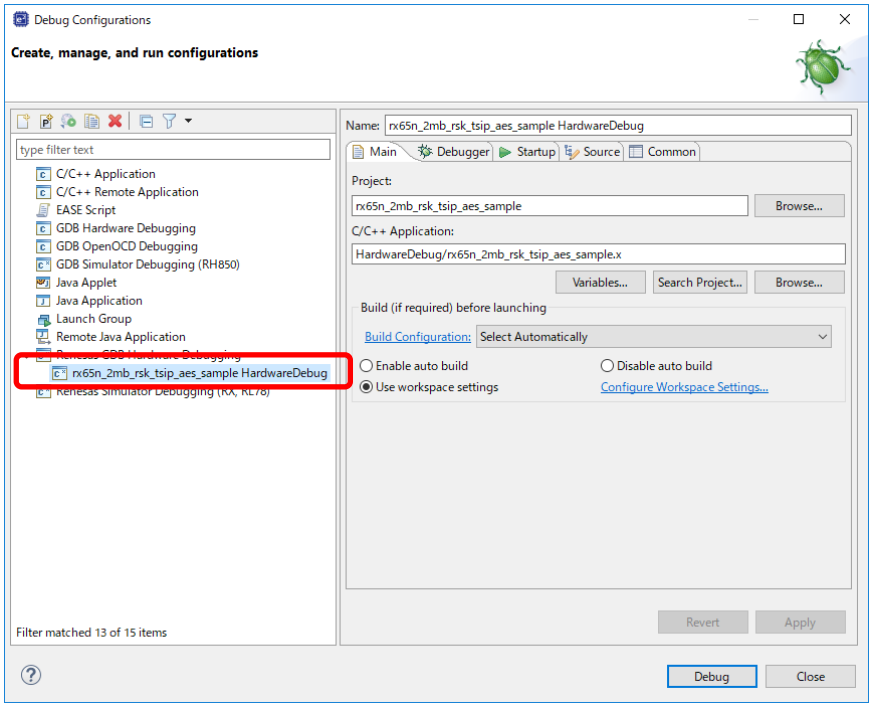

Tera Term: Serial port setup
Port:
COM4
Speed:
115200
Data:
8 bit
Parity:
none
Stop bits:
1 bit
Flow control:
none
OK
Cancel
Help
Transmit delay
0
msec/char
0
msec/line

5.2.2 Download Program

The program can be downloaded with using whether e^2 studio or Renesas Flash Programmer.

5.2.2.1 Download Program using e^2 studio

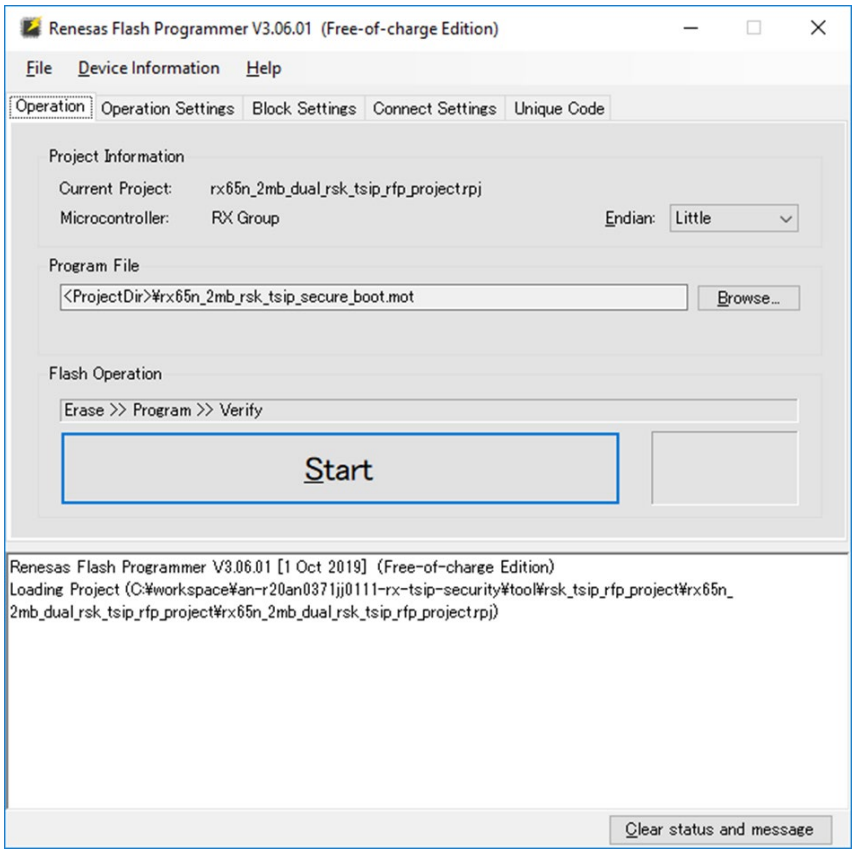
After hardware setup is complete, the project can be download by clicking “Run-> Debug Configuration and selecting “rx65n_2mb_rsk_tsip_aes_sample HardwareDebug” and then click “Debug” button. This will start the download process.



5.2.2.2 Download Program using Renesas Flash Programmer

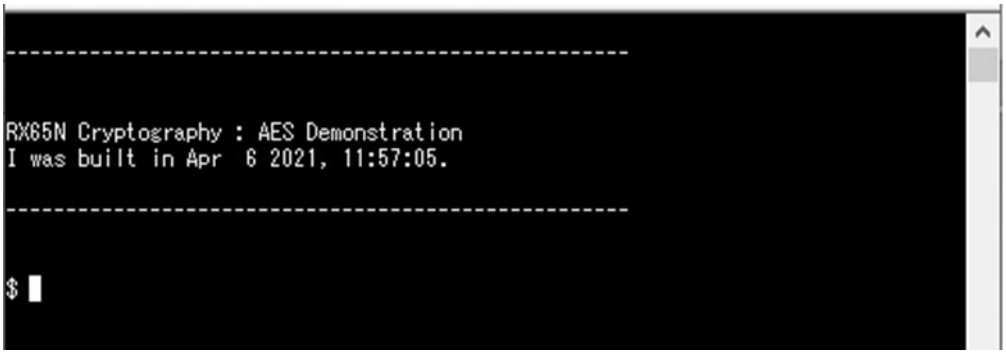
The code can also run in the standalone mode which does not use a debugger. In this case, please use Renesas Flash Programmer (RFP). Please refer below web site to see how to use the RFP.

https://www.renesas.com/rfp



5.3 Run the Program

In order to run the program from e^2 studio, click “Resume” button. At first click, it will break at main function. Click on “Resume” button to run the program. The following output will be shown on the Tera-term:



- The $ prompt is ready to accept the user command. In the prompt, enter any of the following commands.
  1. ecbdemo <plain text>
  2. cbcdemo <plain text>
  3. function

- If command is entered correctly, the output will be shown on the terminal window as below:
  1. $ ecbdemo aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
     Cipher text (output data) which is generated by encrypting the plain text (input data) in AES128-ECB mode with using user key shown in Table 3-2 is displayed. Note that the Wrapped Key is different for each device.

  2. $ cbcdemo aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
     Cipher text (output data) which is generated by encrypting the plain text (input data) in AES128-CBC mode with using user key shown in Table 3-2 and initial vector (=12345678fedcba0955555555aaaaaaaa) is displayed. Note that the Wrapped Key is different for each device.

3. $ function
   Test results of AES ECB/CBC API functions with using sample data are displayed.

```
$ function
R_TSIP_Open() returns:                          TSIP_SUCCESS
R_TSIP_Aes128EcbEncryptInit() returns:          TSIP_SUCCESS
R_TSIP_Aes128EcbEncryptUpdate() returns:        TSIP_SUCCESS
R_TSIP_Aes128EcbEncryptFinal() returns:         TSIP_SUCCESS
aes128_ecb_enc_input:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbccccccccccccccccc
ccccccccccccccccddddddddddddddddddddddddddddddddd
aes128_ecb_enc_output:
f4136afc2a9df52b31c447f4d13a78b4c1c759e6e44a6de1d23e6efb5c48fb55dcdb81139b2a6ca3
14ca50c7e4fa1a179310094c38bd4e2c93ea63670bd92cb3
aes128_ecb_enc_expect:
f4136afc2a9df52b31c447f4d13a78b4c1c759e6e44a6de1d23e6efb5c48fb55dcdb81139b2a6ca3
14ca50c7e4fa1a179310094c38bd4e2c93ea63670bd92cb3

R_TSIP_Aes128EcbDecryptInit() returns:          TSIP_SUCCESS
R_TSIP_Aes128EcbDecryptUpdate() returns:        TSIP_SUCCESS
R_TSIP_Aes128EcbDecryptFinal() returns:         TSIP_SUCCESS
aes128_ecb_dec_input:
f4136afc2a9df52b31c447f4d13a78b4c1c759e6e44a6de1d23e6efb5c48fb55dcdb81139b2a6ca3
14ca50c7e4fa1a179310094c38bd4e2c93ea63670bd92cb3
aes128_ecb_dec_output:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbccccccccccccccccc
ccccccccccccccccddddddddddddddddddddddddddddddddd
aes128_ecb_dec_expect:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbccccccccccccccccc
ccccccccccccccccddddddddddddddddddddddddddddddddd

R_TSIP_Aes128CbcEncryptInit() returns:          TSIP_SUCCESS
R_TSIP_Aes128CbcEncryptUpdate() returns:        TSIP_SUCCESS
R_TSIP_Aes128CbcEncryptFinal() returns:         TSIP_SUCCESS
aes128_cbc_enc_input:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbccccccccccccccccc
ccccccccccccccccddddddddddddddddddddddddddddddddd
aes128_cbc_enc_output:
f5c065e3cfdb49f4d62c1011baf001851d70ad50b9ca17af41cec50fe152e8db1a8bc5d62baf765e
d3d064a3109cf05ccd78f719397e05ae2341cb816a2764fa
aes128_cbc_enc_expect:
f5c065e3cfdb49f4d62c1011baf001851d70ad50b9ca17af41cec50fe152e8db1a8bc5d62baf765e
d3d064a3109cf05ccd78f719397e05ae2341cb816a2764fa

R_TSIP_Aes128CbcDecryptInit() returns:          TSIP_SUCCESS
R_TSIP_Aes128CbcDecryptUpdate() returns:        TSIP_SUCCESS
R_TSIP_Aes128CbcDecryptFinal() returns:         TSIP_SUCCESS
aes128_cbc_dec_input:
f5c065e3cfdb49f4d62c1011baf001851d70ad50b9ca17af41cec50fe152e8db1a8bc5d62baf765e
d3d064a3109cf05ccd78f719397e05ae2341cb816a2764fa
aes128_cbc_dec_output:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbccccccccccccccccc
ccccccccccccccccddddddddddddddddddddddddddddddddd
aes128_cbc_dec_expect:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbccccccccccccccccc
ccccccccccccccccddddddddddddddddddddddddddddddddd

R_TSIP_Close() returns:                         TSIP_SUCCESS
```

4. All above command is case sensitive and output error if it does not match. The below output is when $ fun is pressed.

```
$ fun
command format error.

$
```

6. Using Security Key Management Tool

You can use Security Key Management Tool to generate encrypted user keys. Security Key Management Tool is also available in a command line interface (CLI) version that can easily be deployed in settings such as the production line in a factory.

Security Key Management Tool

https://www.renesas.com/software-tool/security-key-management-tool

Refer to the user's manual for detailed instructions on how to use Security Key Management Tool.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar 31, 2021	–	First release
1.01	Jun 30, 2021	- 24-25	Add explanation about RX72N Envision Kit Add section 6
1.02	Nov 30, 2023	-	Change the key wrapping tool to Security Key Management Tool

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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

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/.

Trademarks

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