

RX Family

Tamper Detection Method Utilizing Existing Peripheral Functions

Introduction

This application note explains a method for detecting tampering from malicious third parties by using application with existing peripheral functions such as the Data Transfer Controller (DTC), Event Link Controller (ELC), and Serial Communication Interface (SCI).

The target devices of this application note are as follows. When adapting the content of this application note to RX family devices other than the tested RX66N group, please adjust it according to the specifications of the target microcontrollers and perform thorough evaluations.

RX family equipped with DTC, ELC, TMR, MTU3a, DOC, SCI, and TSIP (with built-in TRNG).

Target Device

Supported Devices

RX66N group

Contents

1. Overview	4
1.1 Background	4
1.2 Tamper Detection Function	4
1.3 System Configuration	6
1.4 Memory Map	7
1.5 LED Status Indication	8
1.6 Definition of Sequence	9
1.6.1 MTU0.TGRA Settings	9
1.7 Notes	10
1.7.1 Implementation Notes	10
1.7.2 Notes on Operating Conditions	10
1.7.3 Notes on modules used	10
2. Operational Check Conditions	11
3. Operating Environment	12
3.1 Connection Method	12
3.2 Writing Method	13
4. System Description	16
4.1 Tamper Detection Operation Description	16
4.1.1 DTC Transfer Operation	18
4.1.2 Bit Length Alignment for Random Numbers	19
4.1.3 Tamper Detection Erasure	20
4.2 1 Sequence Operation Details	21
4.2.1 Behavior from Power-On to the Start of the First Sequence Transfer	23
4.2.2 Details of Operations During Tamper Events	23
5. Software Description	24
5.1 Configuration	24
5.1.1 File Configuration	24
5.1.2 List of Variables	24
5.1.3 List of Functions	25
5.1.4 Function Specifications	26
5.2 Flowchart	30
5.2.1 Overall Operation Flow	30
5.2.2 Main Function Initialization and ActiveTAMP Start Flow	31
5.2.3 Flow of Tamper Detection Processing	32
5.2.4 Flow When the Timer Detects Two or More Mismatches	34
5.2.5 Communication Error Interrupt Processing	34
5.2.6 Sensitive Data Deletion Function	35
5.3 Footprint	36
5.4 Notes	36
5.4.1 Notes on Stopping Tamper Detection	36

6. Reference Documents36

Revision History37

1. Overview

1.1 Background

In today's era of IoT, there are concerns that malicious third parties may target IoT devices for unauthorized modifications or tampering with internal data.

One of the measures to prevent this is the tamper detection function, which detects modifications and tampering of the hardware itself, thereby enhancing security. By utilizing the existing peripheral functions of the RX family, it is possible to implement tamper detection.

1.2 Tamper Detection Function

In this sample program detect tamper which sends a random number and looped back and received externally. Then compares these numbers to monitor for the presence of a tamper.

This way is different to other standard tamper detection by voltage level.

Additionally, communication errors such as framing, parity, and overflow errors caused by disconnections in the external loopback signal are detected as tampering. Once the tampering is detected, the internal SRAM and backup registers are erased to prevent the leakage of confidential information, such as billing data and personal information.

Figure 1-1 and Figure 1-2 show the level monitoring system and the configuration diagram of this sample program. As an example, a switch (SW) is installed inside the enclosure to detect its opening and closing.

Figure 1-3 illustrates the method of tamper detection using level monitoring. Tamper detection occurs when a change in the level of the TAMPn signal is detected. This method also detects the opening and closing of the enclosure.

Figure 1-4 and Figure 1-5 illustrate the tamper detection method implemented in this sample program. In this sample program, tamper detection is not based on level monitoring; instead, it occurs when the received random numbers do not match a specified number of times or when a communication error occurs.

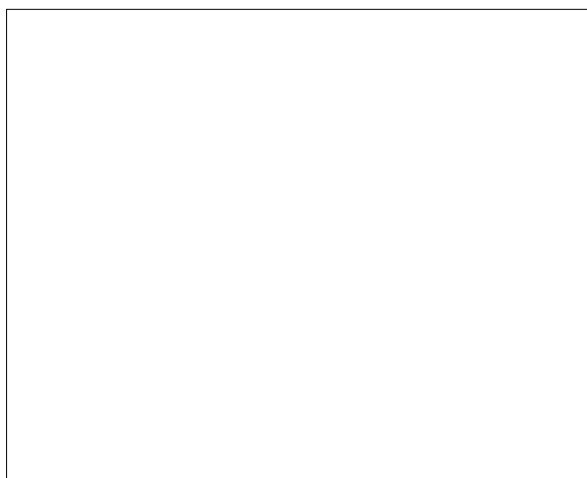


Figure 1-1 Level Monitoring System Configuration Diagram (n=0 to 2)

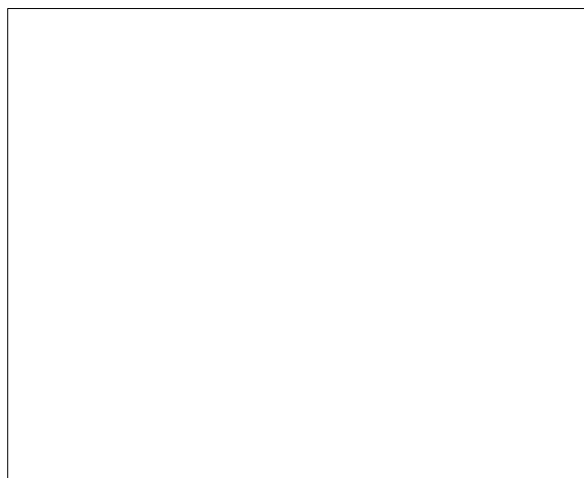


Figure 1-2 System Configuration Diagram of This Sample Program

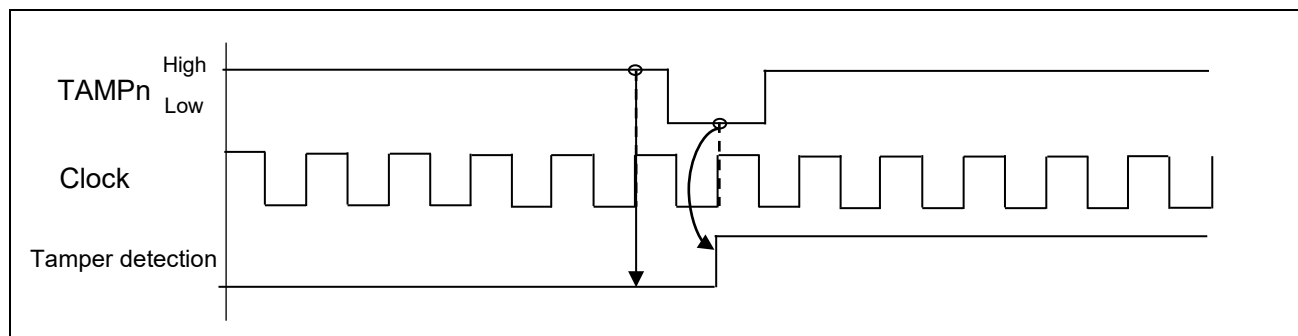


Figure 1-3 Tamper Detection Through Level Monitoring

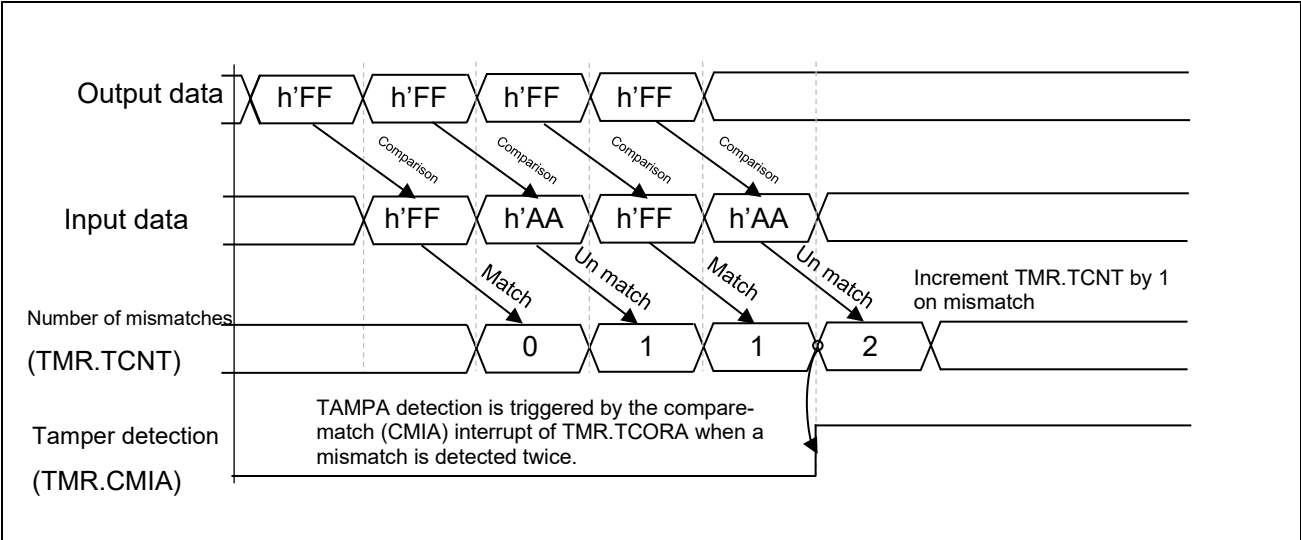


Figure 1-4 Tamper detection by this sample program (tamper detection after 2 discrepancies)

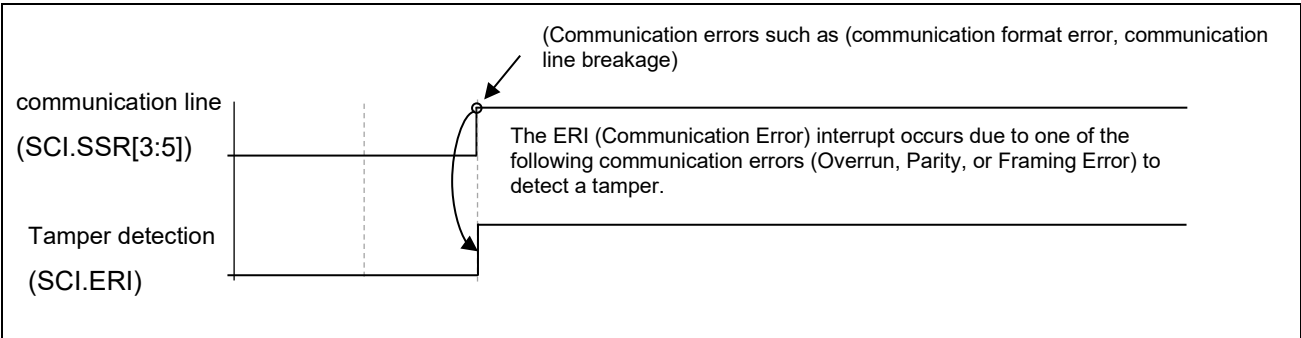


Figure 1-5 Tamper detection by this sample program (in case of communication error)

Table 1-1 summarizes the tampers that can be detected by this sample code.

Table 1-1 Detectable tamper summary

Tamper	Detection Factors
- Opening and closing the enclosure - Broken Communication Line on the Board	- Communication error - Mismatch in Transmitted /Received Counts
Data tampering of communication lines by probes	"The number of mismatches in the comparison of transmitted and received data is two or more. - Mismatch in Transmitted /Received Counts

1.3 System Configuration

Figure 1-6 shows the system configuration diagram for tamper detection using the RX66N, and Table 1-1 lists the peripheral functions used.

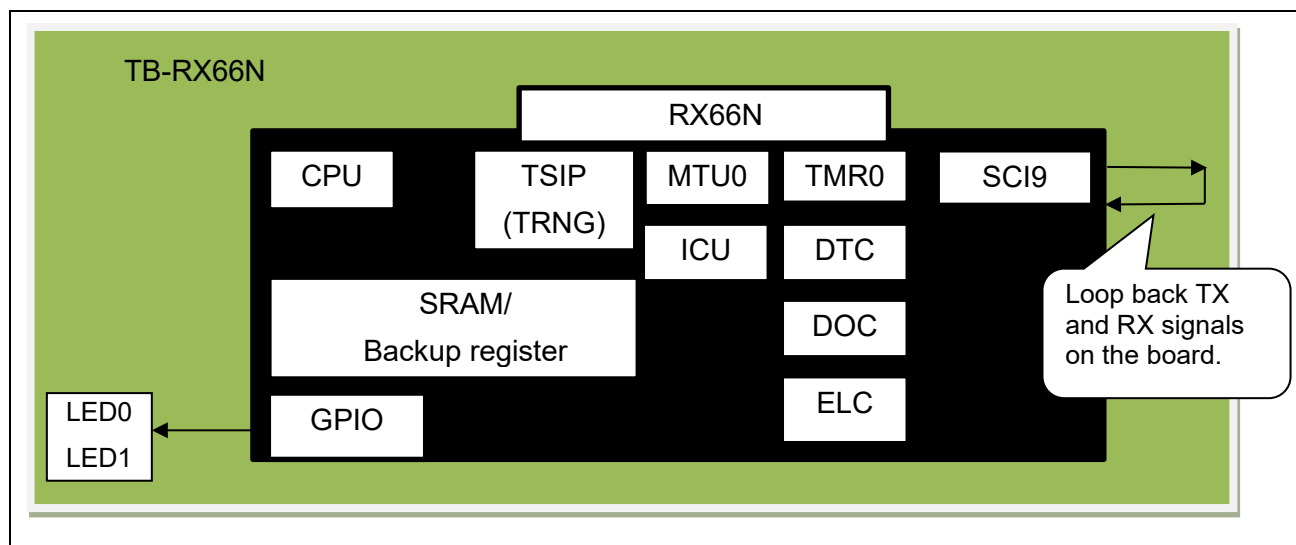


Figure 1-6 System configuration diagram

Table 1-2 Peripheral Functions Used and Applications

Peripheral Functions	Application
MTU0	Manages the sequences described later.
ICU	Manages various interrupt requests from peripheral modules and generates interrupt requests for the CPU or transfer requests to the DTC.
TRNG in TSIP	Generates random numbers. The random numbers are written to SRAM by the CPU.
SCI9	Transmitted random numbers and receive them through external loopback.
DTC	Random numbers generated by the TRNG are transferred from SRAM to SCI9. The random numbers received by SCI9 are then transferred to the DOC. The remaining transfer count of the random numbers is also counted.
DOC	Transmitted random numbers are compared with random numbers received to determine a match or mismatch.
ELC	Event link the DOC mismatch signal to TMR0.
TMR0	Count the number of DOC mismatch signals through the ELC
GPIO	Turn on LED0 and LED1/Turn off LED0 and LED1

1.4 Memory Map

Figure 1-7 shows a memory map that represents the space used for storing sensitive information and for sending and receiving random numbers.

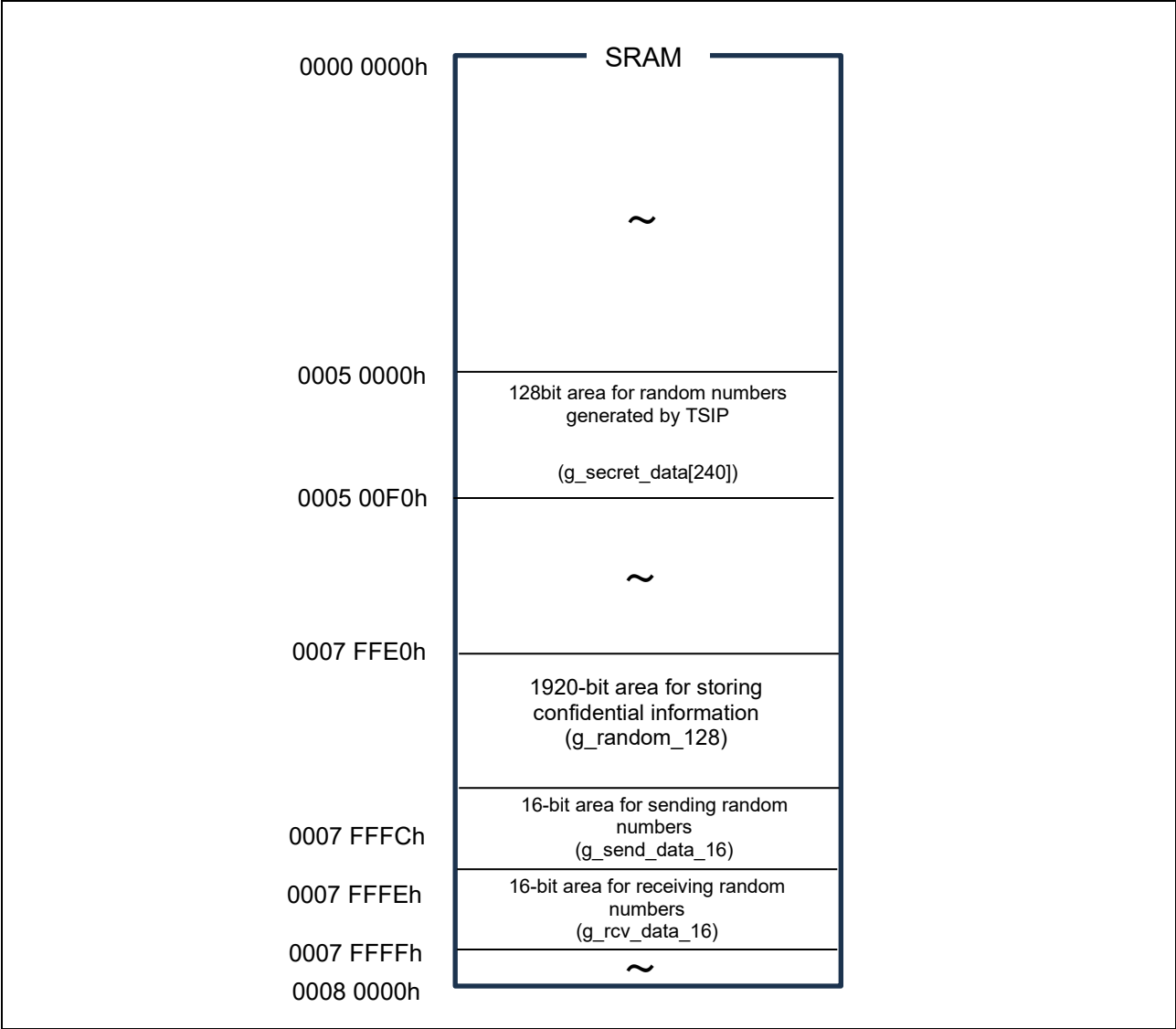


Figure 1-7 Memory Map

1.5 LED Status Indication

Table 1-3 shows the status of LED0 and LED1 for tamper detection.

Table 1-3 LED Status for Tamper Detection

LED0	LED1	Status
OFF	OFF	Normal (No tamper detection)
OFF	ON	Detected the Tamper The number of discrepancies in sent/received data comparison results is 2 or more (Data tampering)
ON	OFF	Detected the Tamper Disconnection (enclosure damage or board modification)
ON	ON	No setting

1.6 Definition of Sequence

The sequence is a period from the generation of the first random number to the start of the next transfer. The sequence consists of tamper detection time and idle time, and the next sequence is initiated by a compare match on MTU0.TGRA.

This enables the sequence operation to occur at the configured cycle.

For details on the tamper detection operation, please refer to Section 4.1: Tamper Detection Operation Description.

For information regarding idle time, please refer to Section 1.6.1: MTU0.TGRA Settings.

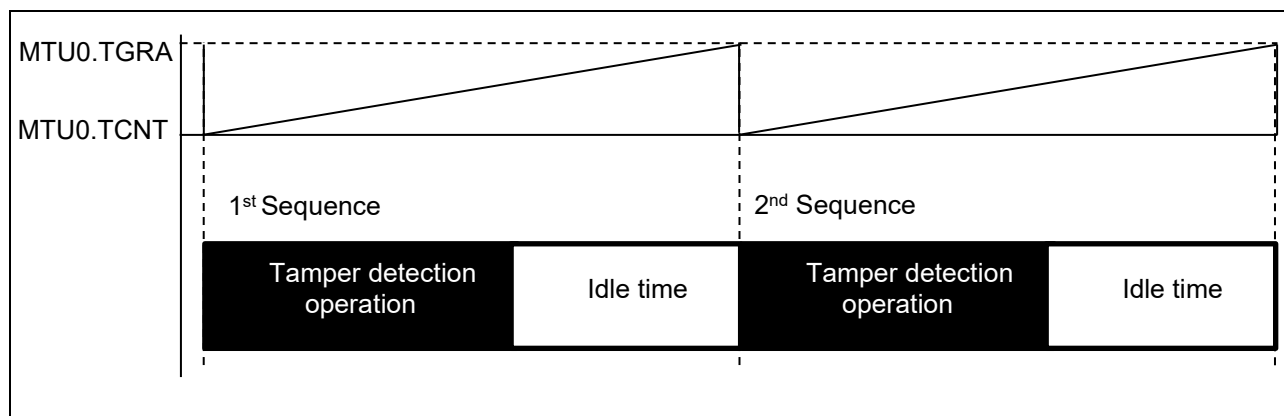


Figure 1-8 Sequence Definition

1.6.1 MTU0.TGRA Settings

The sequence contains the tamper detection operation time and idle time.

The tamper detection operation time can be calculated as $5.6 \text{ [}\mu\text{s]} + (1000000 / \text{baud rate [bps]}) * 10 * 4 \text{ [}\mu\text{s]}$.

The idle time can be set by the user to any duration. After calculating the tamper detection time, please add any desired duration chosen by the user to set the total time in MTU0.TGRA.

Example: When the baud rate is set to 9600 bps and the desired idle time is 100 ns, the tamper detection operation time, calculated using the above formula, is 4172.26 μs . Therefore, to ensure that the total time in MTU0.TGRA is greater than 4272.36 μs , please set it to the sum of these two values.

This formula does not account for the time required to complete reception. Therefore, in practice, please adjust the formula to consider the time it takes to receive the data.

1.7 Notes

1.7.1 Implementation Notes

The sample code specifies that if the external folding is severed, it will remove all confidential information in any case. Before opening or closing the enclosure for regular maintenance, please ensure that you upload any confidential information to the cloud and take specific steps to disable tamper detection.

For information on how to stop tamper detection, Please refer **5.1.4 Notes on Stopping Tamper Detection**.

1.7.2 Notes on Operating Conditions

During the operation of this sample program, please be cautious of short circuits between the communication lines, power supply, and ground, as well as noise on the communication lines, as these may be detected as tampering.

1.7.3 Notes on modules used.

When using this sample program, please note that the DOC has only one channel and cannot be used within the user program. However, other modules can utilize any channels that are not used by this sample program.

2. Operational Check Conditions

The sample program in this application note has been tested under the following conditions.

Table 2-1 Operating Environment

Item	Contents
MCU used	R5F566NNHDFP (Equipped with Target Board for RX66N)
Operating Frequency	Main Clock: Stopped (using HOCO) PLL : 240MHz (HOCO x 1/1 x 15) HOCO : 16MHz LOCO : Stop System clock (ICLK) : 120MHz (PLL x 1/2) Peripheral module clock A (PCLKA) : 120MHz (PLL x 1/2) Peripheral module clock B (PCLKB) : 60MHz (PLL x 1/4) Peripheral module clock C (PCLKC) : 60MHz (PLL x 1/4) Peripheral module clock D (PCLKD) : 60MHz (PLL x 1/4) Flash IF Clock (FCLK) : 60MHz (PLL x 1/4)
Operating Voltage	3.3V
Integrated Development Environment	Renesas Electronics e ² studio Version 2024-07.0
C Compiler ^{Note}	Renesas Electronics C/C++ Compiler Package for RX Family V3.06.00
RX Smart Configurator	V2.22.0
Board Support Package (r_bsp)	V7.50
Endian	little Endian
Operation Mode	Single Chip Mode
Processor Mode	Supervisor Mode
Sample Code Version	V1.00
Evaltion Board	Target Board for RX66N (Type Name : RTK5RX66N0C00000BJ)
Emulator	E2-Lite

Note If the version of the toolchain (C compiler) specified for this project is not installed, the toolchain will remain unselected, resulting in an error. If an error occurs, please check the toolchain selection status in the project settings screen.

For setup instructions, please refer to FAQ 3000404.

[FAQ 3000404: When building an imported project, a "Program 'make' not found in PATH" error occurs \(e² studio\)](#)

3. Operating Environment

The sample program project for this application needs to be written to the Target Board for RX66N by connecting it to the development PC via USB.

In this sample program, the on-chip debugging emulator on the Target Board for RX66N will be used. Table 3-1 shows the setup environment for writing the sample program, and Figure 3-1 illustrates the connection method.

Table 3-1 Sample Program Writing Environment

Item	Contents
Development Environment	- Development Windows PC - e² studio Version 2024-07.0
Debugging Tool	E2 Lite(RX)
Board	Target Board for RX66N (Type Name : RTK5RX66N0C00000BJ)
Cable	- USB Cable (type-A ⇔ type micro B) - Jumper Cable (Male ⇔ Male)

3.1 Connection Method

- Connect a jumper cable (male to male) to pins 9 and 10 of the Pmod connector (CN1).
- Connect the micro-B side of the USB function cable to the USB connector on the Target Board for RX66N.
- Connect the type-A side of this USB cable to a USB port on the development PC. This will connect you to the on-chip debugging emulator.

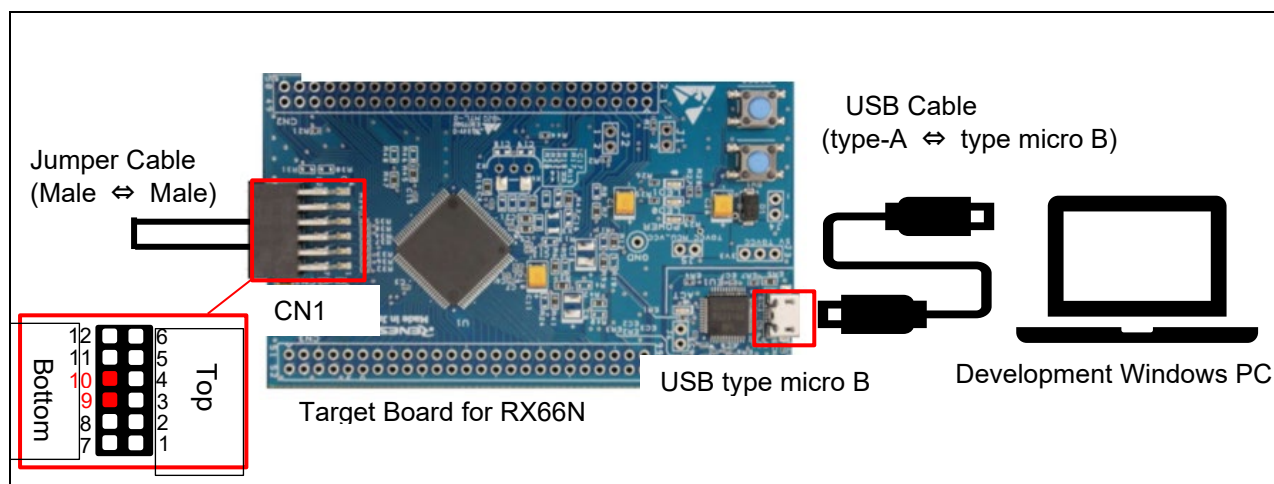


Figure 3-1 Target Board for RX66N Connection Diagram

3.2 Writing Method

Once the connection is complete, follow the steps below to write the project.

a. Import the project.

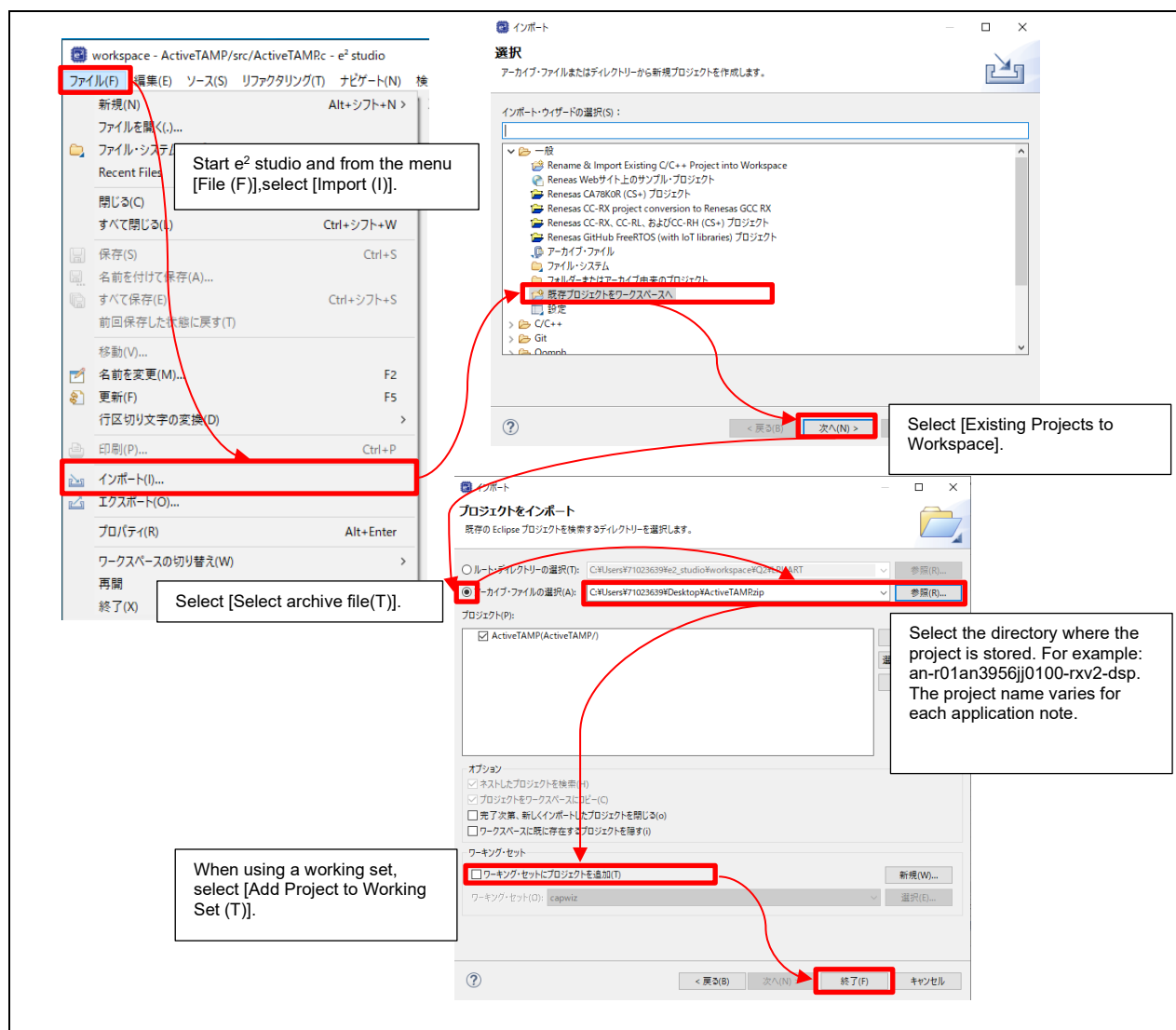


Figure 3-2 Import the project

- b. After building the project, press the debug button in e² studio, which will start the writing process.

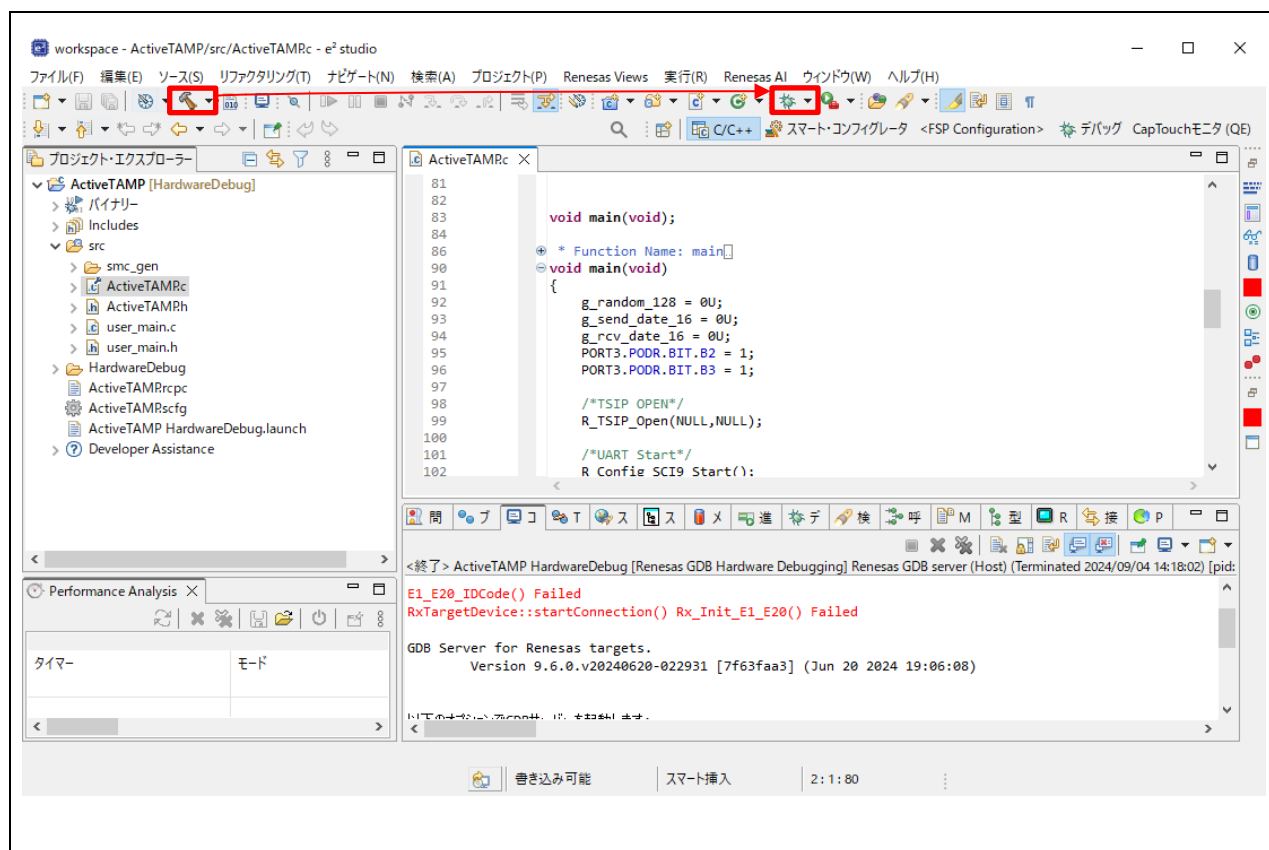


Figure 3-3 Build and Debug

- c. After the writing process in b, it will automatically start in FINE boot mode. Press 'Exit' to disconnect e² studio and the on-chip debugging emulator. and the on-chip debugging emulator.

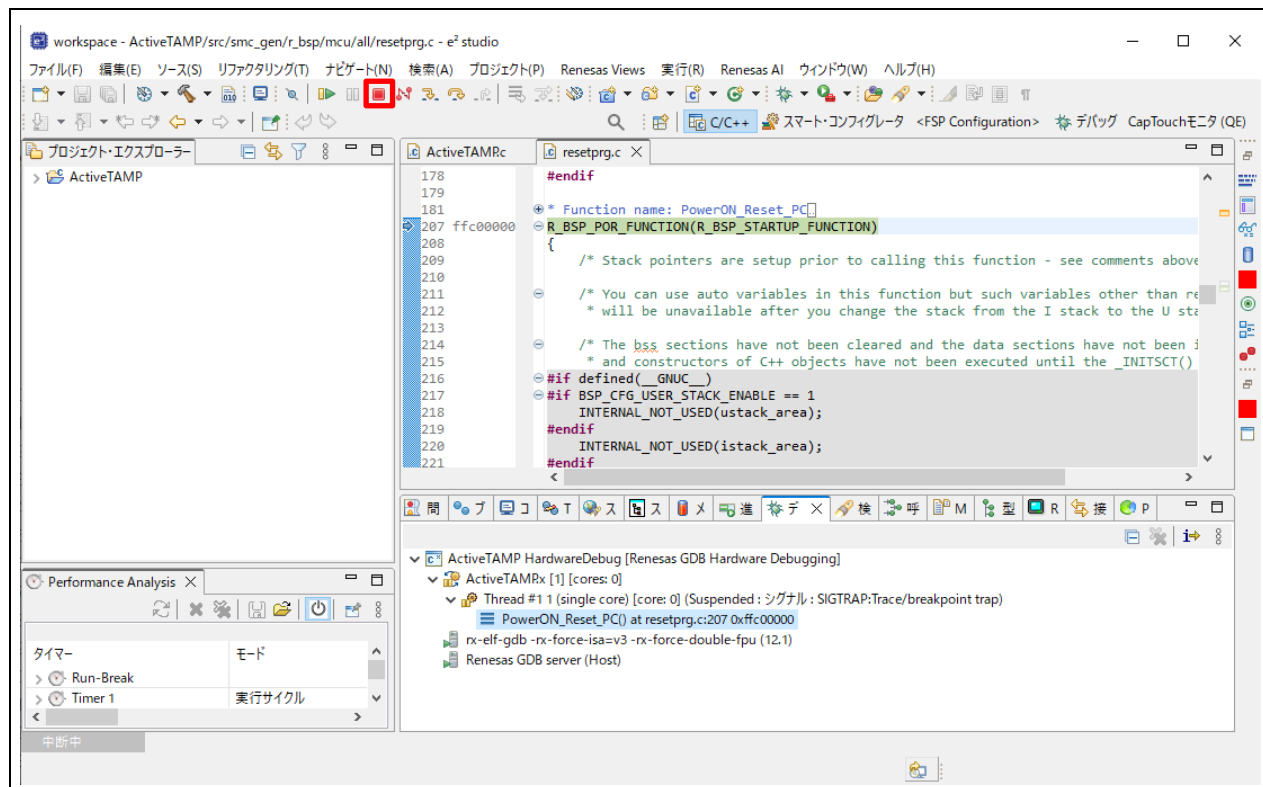


Figure 3-4 Emulator disconnection

- d. When the emulator reset header (J6) is connected, it will start in single-chip mode without requiring a debugger, executing the programmed project.

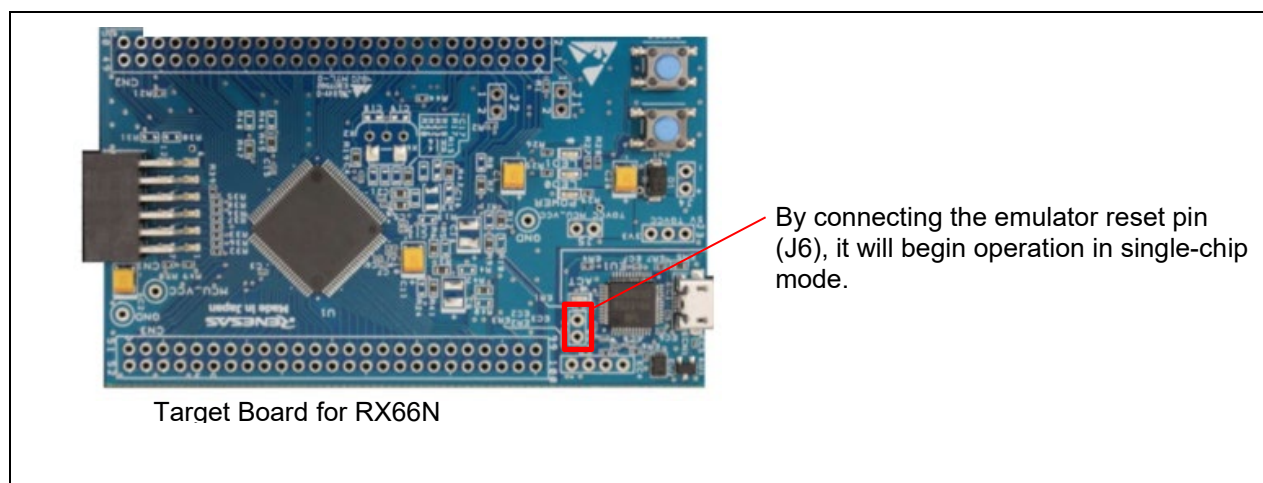


Figure 3-5 Operating Mode Switch

4. System Description

4.1 Tamper Detection Operation Description

Figure 4-1 shows the data flow (random numbers) in this sample program. Table 4-1 provides details on the operation when no tampering has occurred, Table 4-2 describes the operation when tampering is detected due to transmitted or received data alteration, and Table 4-3 details the operation when tampering is detected due to a communication error.

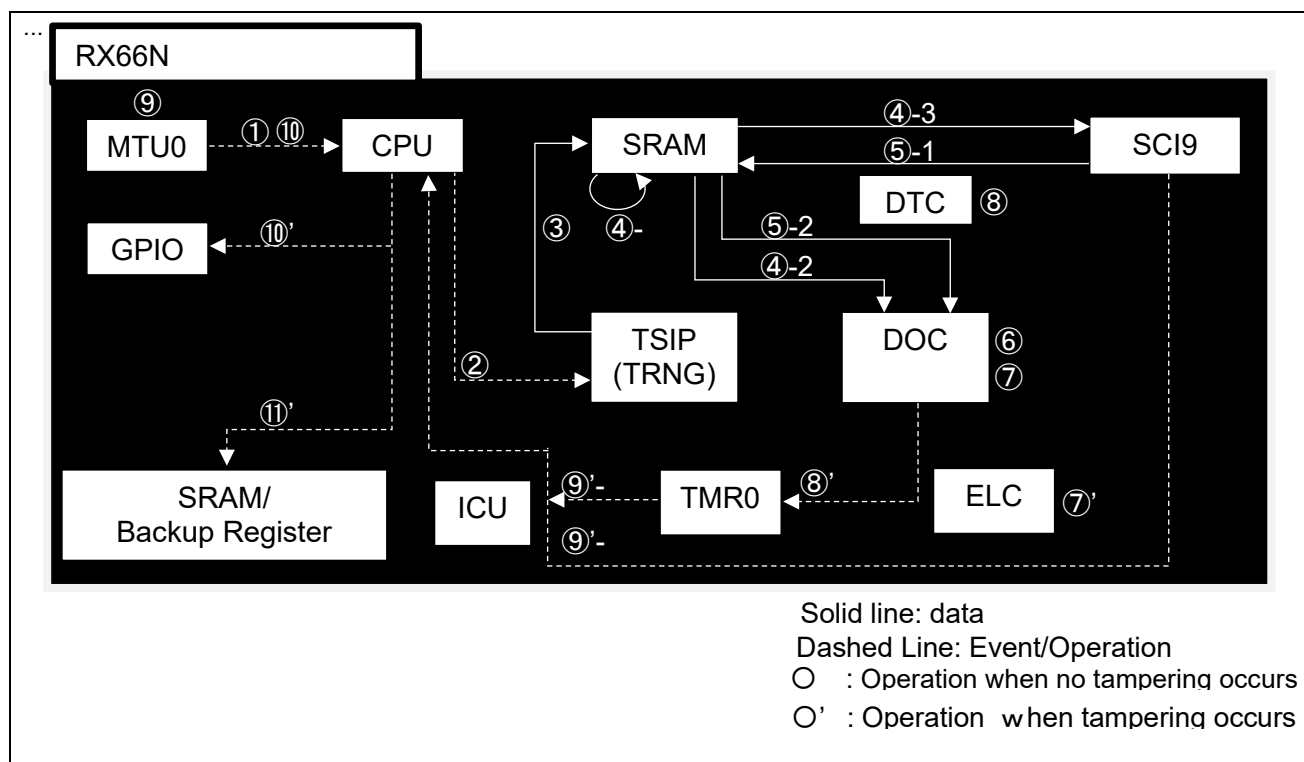


Figure 4-1 Tamper Detection Operation

Table 4-1 Tamper Detection Operation Details (When No Tampering Occurs)

Operation	Description
1 Start of Sequence	The sequence will start with the MTU0.TGRA compare match interrupt.
2 Random Number Generation	A random number generation command will be issued to the TRNG within the TSIP.
3 Random number storage	The random numbers generated by 2 will be stored in SRAM by the CPU.
4-1 DTC Transfer (Transmitted Data)	A part of the random numbers stored in 3 is transferred from SRAM to SRAM by DTC normal transfer.
4-2 DTC Transfer (Transmit Data)	The random numbers transferred in 4-1 will be transferred from SRAM to DOC.DODSR using the chain transfer from 4-1.
4-3 DTC Transfer (Transmitted Data)	A portion of the random numbers transferred in 4-1 will be transferred from SRAM to SCI9.TDR using the chain transfer in 4-2.
5-1 DTC Transfer (Received Data)	The random numbers received will be transferred from SCI9.RDR to SRAM using the normal transfer of the DTC.
5-2 DTC Transfer (Transmitted Data)	The random numbers transferred in 5-1 will be transferred from SRAM to DOC.DODIR using the chain transfer from 5-1.
6 Comparison of Transmitted and Received Random Numbers	The random numbers transferred 4-2 are compared to the random numbers transferred in 5-2 by the DOC.
7 When matched	Starting the transfer of the next random number.
8 Repeat	Repeat steps 4 to 7-1 a total of four times.
9 Idle time	We will wait for the idle time.

10 End of sequence	The sequence will end with the MTU0.TGRA compare match interrupt. (Return to 1)
--------------------	--

Table 4-2 Tamper Detection Operation Details (When Tamper Detection Occurs Due to Data Alteration in Transmitted and Received Data)

Operation	Description
1 to 6: Same as when no tamper has occurred.	-
7' In the Case of mismatch	If the comparison results of 6 is a mismatch, the event link signal will be output, and the transfer of the next random number will start.
8 Count of mismatches	The 7' event signal increments to the TMR0.TCNT (mismatch count).
9'-1 When there are two mismatches, TMR0 interrupt is triggered.	When the value of TMR0.TCNT reaches 2, TMR0 raises the TCORA compare match interrupt to the CPU.
10' Turn on LED1	When the interrupt from 9-2 occurs, the CPU will turn on LED1.
11' Deletion of confidential information	After step 10, the CPU will delete (clear to zero) the confidential information in SRAM and the contents of the backup registers, and then call the user program in case of an error.

Table 4-3 Tamper Detection Operation Details (When Tamper Detection Occurs Due to Communication Errors)

Operation	Description
1 to 6: Same as when tampering has not occurred.	-
9'-2 Communication error Interrupt occurred	When a communication error is detected, SCI9 raises a communication error interrupt to the CPU.
10' Turn on LED0	When the interrupt of 9-1 occurs, the CPU turns on LED0
11' Deletion of confidential information	After step 10, the CPU will delete (clear to zero) the confidential information in SRAM and the contents of the backup registers, and then call the user program in case of an error.

4.1.1 DTC Transfer Operation

In this sample program, DTC transfers are conducted due to software factors, transmission data empty factors, and reception data full factors.

From here on, the DTCs for each activation factor will be referred to as (Software) DTC, (Transmission) DTC, and (Reception) DTC.

Table 4-4 provides an overview of the DTC operations for each activation factor. The numbers in the table correspond to the numbers in Figure 4-1.

Please refer to Section 4.1.2 'Random Number Bit-Length Alignment,' for details on the random numbers to be transferred.

Table 4-4 Activation Factors and DTC Transfer Operations

Operation Overview	Name	Activation Factors
4-1 Transfers 8 bits of 128-bit random numbers from SRAM to SRAM	(Soft)DTC	Software
4-2 In the chain transfer described above, the random numbers transferred earlier will be sent from SRAM to DOC.	(Soft)DTC	No chain transfer
4-3 In the above chain transfer, the data will be sent from SRAM to SCI9.TDR.	(Soft)DTC	Same as above
4-3 The transfer will be performed from SRAM to SCI9.TDR.	(Transmission)DTC	Transmitted data empty
5-1 The random number in SCI.RDR will be transferred to SRAM.	(Reception)DTC	Received data full
5-2 In the above chain transfer, the data will be sent from SRAM to DODIR.	(Reception)DTC	No chain transfer

4.1.2 Bit Length Alignment for Random Numbers

The random numbers generated by the TRNG are 128 bits long, while those transmitted via SCI are 8 bits long, and those compared by the DOC are 16 bits long. Therefore, it is necessary to align the bit lengths when exchanging these random numbers.

In this sample program, the random numbers are aligned to a length of 16 bits for compatibility with the DOC for comparison.

Figure 4-2 provides an overview of the method for aligning the bit lengths of random numbers, while Table 4-5 presents the details of this alignment method.

The numbers within the figures are linked to the numbers in Figure 4-1.

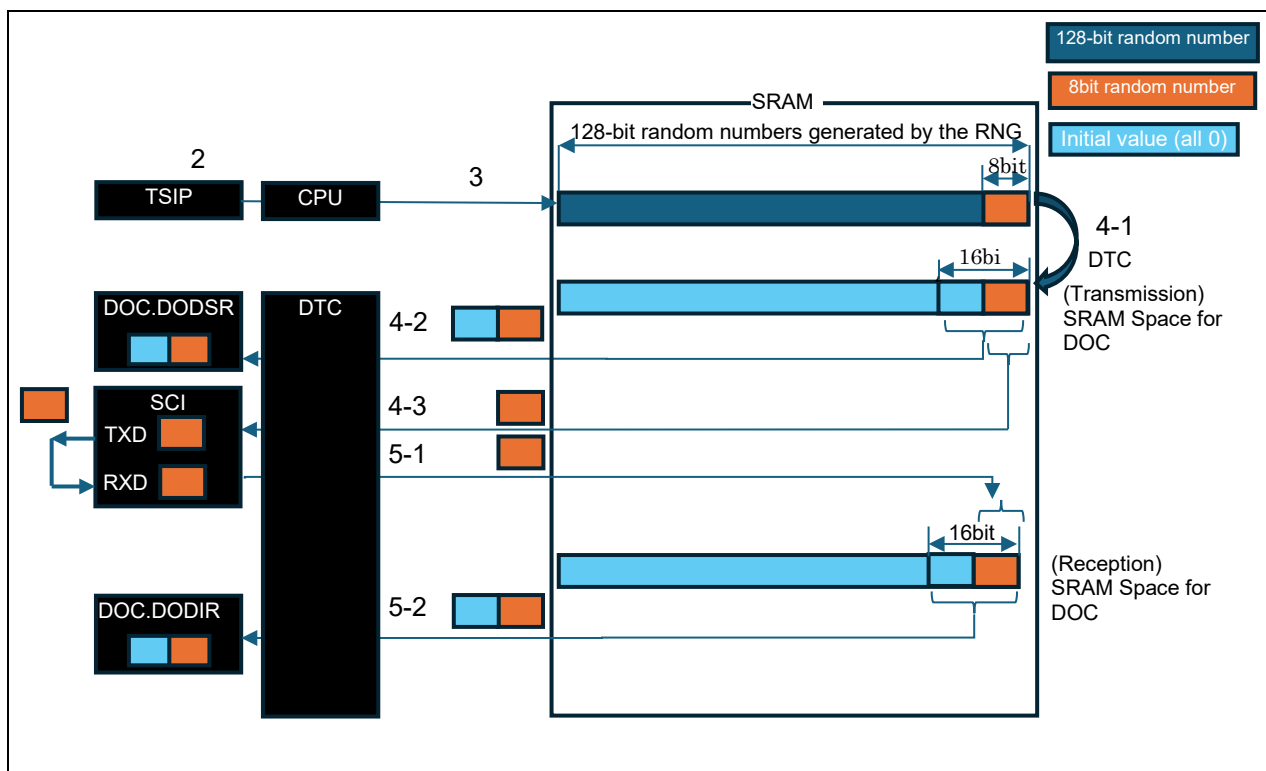


Figure 4-2 Overview of Random Number Bit Length Alignment

Table 4-5 Detailed Method for Random Number Bit Length Alignment

Operation	Description
2 Random Number Generation	Random numbers (128 bits long) are generated by the TRNG within TSIP.
3 Random number storage	The 128-bit random number generated in 2 will be stored in SRAM by the CPU.
4-1 16-bit Random Numbers	The least significant 8 bits of the 128-bit random number stored in 3 will be transferred to the lower section of the (transmit) DOC SRAM space, aligning it with the initial value to form a 16-bit random number.
4-2 Storing Transmitted Random Numbers in DOC	The 16-bit random number aligned in 4-1 will transfer to DOC.DODSR via DTC.
4-3 Transfer of Transmitted Random Numbers	Transfer the lower 8 bits of the 16-bit random number created in step 4-1 to SCI9.TDR via DTC.
5-1 Transfer of Received Random Numbers	Transfer the received 8-bit random number to the lower part of the (receive) DOC-designated SRAM space and align it to 16 bits by combining it with the initial value.
5-2 Storing Received Random Numbers in DOC	The 16-bit random number aligned in 5-1 will be transferred to DOC.DODIR via DTC.

4.1.3 Tamper Detection Erasure

Upon tamper detection, confidential information stored in SRAM at addresses 0005 0000h to 0005 00F0h, as well as in the backup registers at addresses 0008 C2A0h to 0008 C2BFh, will be erased.

After the erasure, this sample program enters an infinite loop. Additionally, the user can be informed of the erasure of confidential information due to tamper detection via a communication error (overrun, framing, or parity error) interrupt or the TMR0 TCORA compare match interrupt.

4.2 1 Sequence Operation Details

This sample program minimizes CPU processing to reduce the load on the user program as much as possible, allowing tamper detection to occur in the background. The operation of this sample program from power-on to the completion of a single sequence is outlined below.

1. Initialize each peripheral function (TSIP, SCI9, TMR0, ELC, DOC, MTU0, DTC).
2. Set the value of MTU0.TGRA to MTU0.TCNT, and then activate each peripheral function in the following order: TSIP, SCI9, TMR0, ELC, DOC, MTU0.
3. Setting the MTU0.TSTRA.CST0 bit to 1b starts the counting process of MTU0.
4. When the value of the MTU0.TCNT register matches the value of the MTU0.TGRA register, the MTU0.TCNT register is cleared, and simultaneously, a MTU0.TGRA compare match interrupt is generated.
5. In the MTU0.TGRA compare match interrupt handling, random numbers are generated, and the DTC transfer count is verified, initialized, and enabled. After that, a value of 1 is set in SWINTR.SWINT to start the DTC transfer for software triggers.
6. Through the DTC transfer for software triggers, data is transferred from SRAM(A)* to SRAM(B)*, and then, using chain transfer, SRAM(B)* is sent to DOC.DODIR. Additionally, a portion of SRAM(B)* is transferred to SCI9.TDR via chain transfer. This initiates the DTC transfer for the transmit data empty condition, starting four transmissions via SCI9.
7. Upon receiving data from SCI9 due to self-looping of the SCI9 transmitted data, the DTC transfer for the receive data full condition is initiated, transferring the received data (SCI9.RDR) to SRAM(C)*.
8. In the chain transfer of (7), the received data is transferred to DOC.DODSR, where it is compared with the transmitted data.

9.

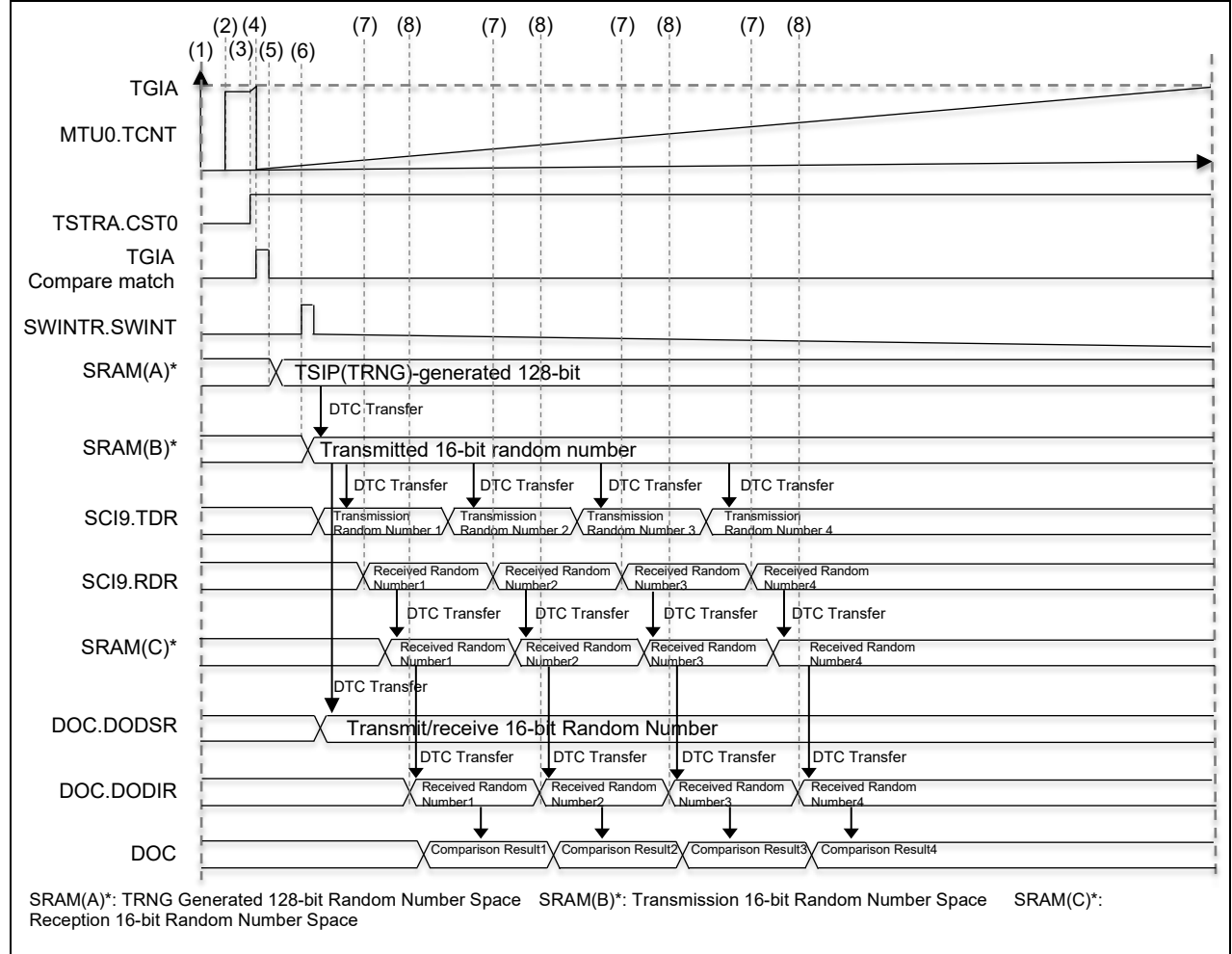


Figure 4-3 Sample Program Operation

4.2.1 Behavior from Power-On to the Start of the First Sequence Transfer

In this sample program, to minimize the time during which tamper detection cannot occur after power-on, tamper detection begins at the rising edge of the first clock cycle of the MTU0 source clock.

°

After the first sequence, the sequence will start at the period set in MTU0.TGRA.

4.2.2 Details of Operations During Tamper Events

If a tamper event occurs during the operation of the sequence, the following actions will be taken immediately.

1. The corresponding LED will be turned on to indicate the cause of the tamper event (random number tampering or communication error).
2. All peripheral functions used for tamper detection will be stopped.
3. All confidential information in RAM and backup registers will be deleted (zeroed out).
 4. The user-defined error handling function will be called.

5. Software Description

5.1 Configuration

5.1.1 File Configuration

Table 5-1 lists the files used in this sample program. Note that files containing source code generated directly by the Smart Configurator's code generation feature are excluded.

Table 5-1 Files used in the sample code

File name	Overview	Notes
ActiveTAMP.c	Main Process	This sample code performs the initial setup and start process for tamper detection.
ActiveTAMP.h	Header file of ActiveTAMP .c	-
user_main.c	User's main, Error handling	Called from ActiveTAMP.c. This sample program does not perform any processing.
user_main.h	Header file of user_main.c	-
Config_MTU0_user.c	Timer Interrupt Processing	-
Config_DTC_soft.c	DTC transfer setting processing	Software Factors
Config_DTC_rcv_user.c	DTC transfer setting processing	Software Factors
Config_DTC_send.c	DTC transfer setting processing	Factors for Empty Transmission Data
Config_DTC_send_user.c	DTC transfer setting processing	Factors for Empty Transmission Data
Config_DTC_rcv.c	DTC transfer setting processing	Factors for Full Reception Data
Config_DTC_soft_user.c	DTC transfer setting processing	Factors for Full Reception Data

5.1.2 List of Variables

Table 5-2 shows the global variables.

Table 5-2 List of Variables

Type	Variable Name	Description	Functions Used
uint32_t	g_random_128	SRAM area for storing 128-bit random numbers	main r_Config_MTU0_tgia0_interrupt
uint16_t	g_send_data_16	SRAM area for storing a 16-bit transmission random number	main r_Config_MTU0_tgia0_interrupt
uint16_t	g_rcv_data_16	Store a 16-bit reception random number	main
uint8_t	g_secret_data[240]	SRAM area for storing data that simulates confidential information	main
uint8_t	g_backup[32]	Backup register area for storing data that simulates confidential information	main

5.1.3 List of Functions

Table 5-3 lists the functions. Note that functions that use source code generated directly by the Smart Configurator's code generation feature are omitted.

Table 5-3 List of Functions

Function Name	Overview
main	Main Processing, Initial Setup for Tamper Detection Operation, Start Processing
erase_secret_data	Confidential Information Deletion Process
user_main_function	User Main Processing
user_err_function	User Error Handling
r_Config_MTU0_tgia0_interrupt	Start Process of Tamper Detection Sequence
R_Config_DTC_rcv_Remaining	Process to Retrieve Remaining Transfer Count of DTC for Full Reception Data Factors
R_Config_DTC_rcv_reset	Process to Reset DTC Transfer Count for Full Reception Data Factors
R_Config_DTC_rcv_Create_UserInit	Initial Setup Process for DTC Transfer Count for Full Reception Data Factors
R_Config_DTC_send_reset	Process to Reset DTC Transfer Count for Empty Transmission Data Factors
R_Config_DTC_send_Create_UserInit	Initial Setup Process for DTC Transfer Count for Empty Transmission Data Factors
R_Config_DTC_soft_reset	Process to Reset DTC Transfer Count for Software Factors
R_Config_DTC_soft_Create_UserInit	Initial Setup Process for DTC Transfer Count for Software Factors

5.1.4 Function Specifications

The function specifications of the sample code are shown.

main	
Overview	Main Processing, Initial Setup for Tamper Detection Operation, Start Processing
Header	None
Declaration	void main(void)
Description	After the initial setup, the tamper detection operation will be started, and the user main processing will be called.
Arguments	None
Return Value	None
Notes	-
erase_secret_data	
Overview	Confidential Information Deletion Process
Header	ActiveTAMP.h
Declaration	void erase_secret_data(void)
Description	Deletes (clears to 0) sensitive information from SRAM and backup registers and calls the user error handling routine.
Arguments	None
Return Value	None
Notes	After the process, it enters an infinite loop and does not perform any other operations.
user_main_function	
Overview	User Main Processing
Header	user_main.h
Declaration	void user_main_function(void)
Description	You can implement the user-defined main processing, which operates in parallel with tamper detection.
Arguments	None
Return Value	None
Notes	-

user_err_function	
Overview	User Error Handling
Header	user_main.h
Declaration	void user_err_function(void)
Description	You can implement the user-defined error handling that occurs when a tamper event is detected. This process is called only once, after the deletion of confidential information.
Arguments	None
Return Value	None
Notes	It is called by the erase_secret_data function.
r_Config_MTU0_tgia0_interrupt	
Overview	Timer Interrupt Handling
Header	r_Config_MTU0.h
Declaration	static void r_Config_MTU0_tgia0_interrupt(void)
Description	The start process of the tamper detection sequence is performed by the compare match interrupt handling of MTU0.TGIA.
Arguments	None
Return Value	None
Notes	-
R_Config_DTC_rcv_Remaining	
Overview	Process to retrieve the remaining transfer count of DTC for full reception data factors.
Header	R_Config_DTC_rcv.h
Declaration	int R_Config_DTC_rcv_Remaining(void)
Description	It returns the remaining transfer count for the full reception data factors of SCI9.
Arguments	None
Return Value	int dtc_transferdata_vector102[0].cra_crb
Notes	-
R_Config_DTC_rcv_reset	
Overview	Process to reset the remaining transfer count of DTC for full reception data factors.
Header	R_Config_DTC_rcv.h
Declaration	void R_Config_DTC_rcv_reset(void)
Description	The remaining transfer count of DTC for full reception data factors will be reset to 4.
Arguments	None
Return Value	None
Notes	-

R_Config_DTC_rcv_Create_UserInit	
Overview	Initial setup process for the DTC transfer count for full reception data factors.
Header	R_Config_DTC_rcv.h
Declaration	void R_Config_DTC_rcv_Create_UserInit(void)
Description	During initialization, the remaining transfer count of DTC for full reception data factors is initialized to 0.
Arguments	None
Return Value	None
Notes	-
R_Config_DTC_send_reset	
Overview	Process to reset the DTC transfer count for empty transmission data factors.
Header	R_Config_DTC_send.h
Declaration	void R_Config_DTC_send_reset(void)
Description	The remaining transfer count of DTC for empty transmission data factors will be reset to 3.
Arguments	None
Return Value	None
Notes	-
R_Config_DTC_send_Create_UserInit	
Overview	Initial setup process for the DTC transfer count for empty transmission data factors.
Header	R_Config_DTC_send.h
Declaration	void R_Config_DTC_send_Create_UserInit(void)
Description	During initialization, the remaining transfer count of DTC for empty transmission data factors is initialized to 0.
Arguments	None
Return Value	None
Notes	-
R_Config_DTC_soft_reset	
Overview	Process to reset the DTC transfer count for software factors.
Header	R_Config_DTC_soft.h
Declaration	void R_Config_DTC_soft_reset(void)
Description	The remaining transfer count of DTC for software factors will be reset to 1.
Arguments	None
Return Value	None
Notes	-

R_Config_DTC_soft_Create_UserInit

Overview	Initial setup process for the DTC transfer count for software factors.
Header	R_Config_DTC_soft.h
Declaration	void R_Config_DTC_soft_Create_UserInit(void)
Description	During initialization, the remaining transfer count of DTC for software factors is initialized to 0.
Arguments	None
Return Value	None
Notes	-

5.2 Flowchart

In the following flow, the initialization of each module is generated by the Smart Configurator using the `R_Config_<ModuleName>_Create` function. Additionally, the TSIP is generated by the `R_TSIP_Open` function.

5.2.1 Overall Operation Flow

Figure 5-1 illustrates the overall operation flow of this sample program.

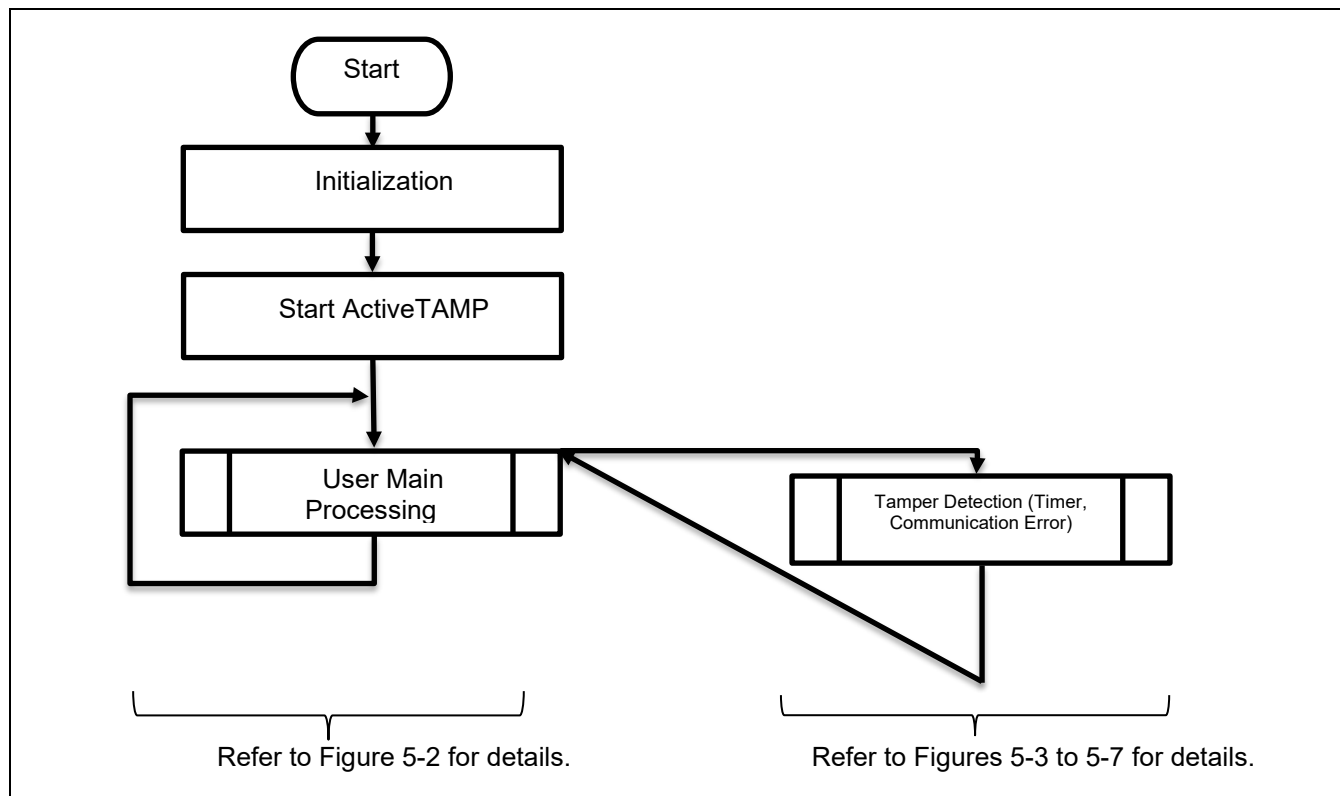


Figure 5-1 Overall Operation Flow

5.2.2 Main Function Initialization and ActiveTAMP Start Flow

Figure 5-2 illustrates the flow of the initialization of the main function and the start of ActiveTAMP.

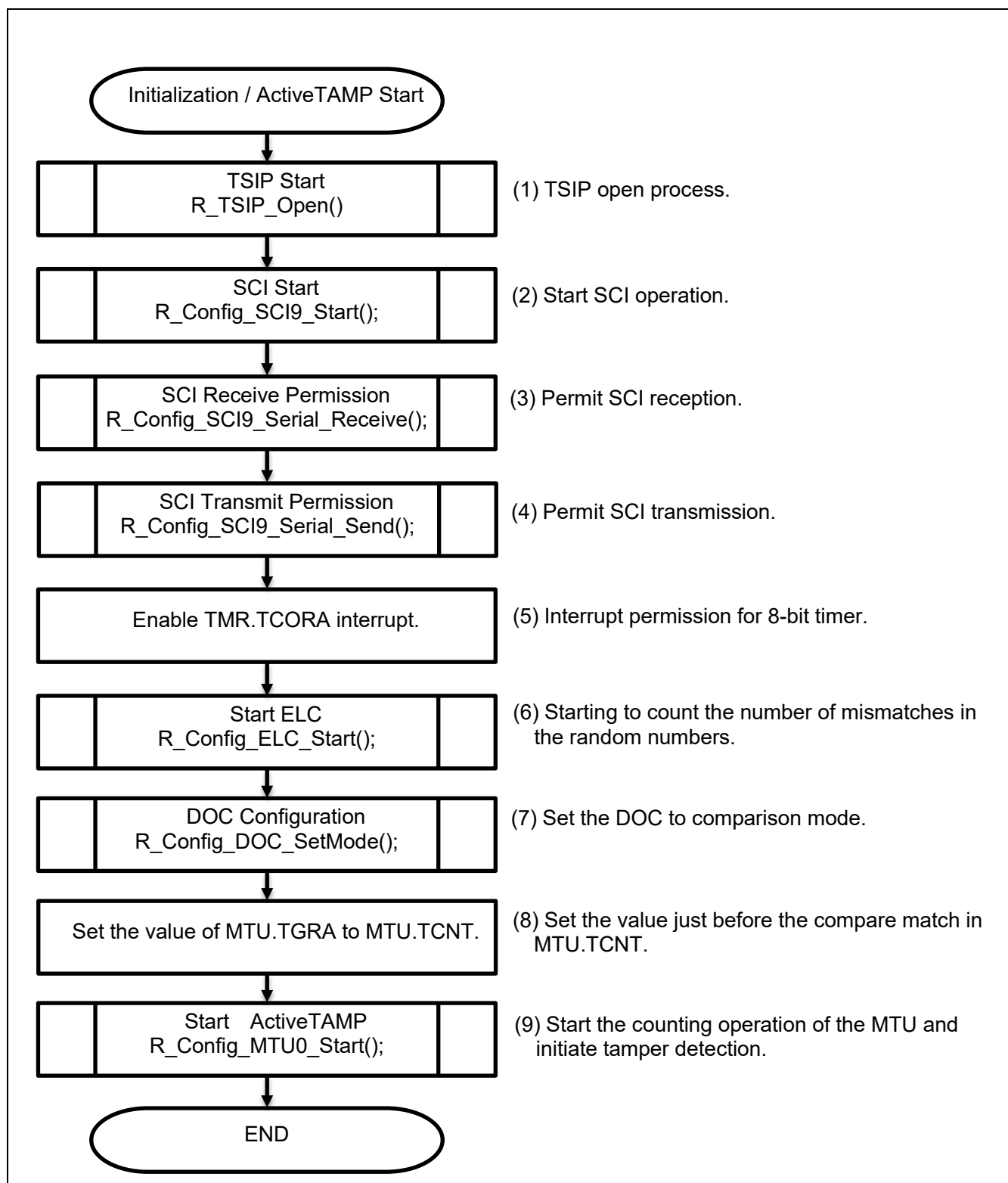


Figure 5-2 Flow of Main Function Initialization and ActiveTAMP Start

5.2.3 Flow of Tamper Detection Processing

Figure 5-3 illustrates the overview flow of tamper detection interrupt processing by the timer, while Figure 5-4 presents the detailed flow.

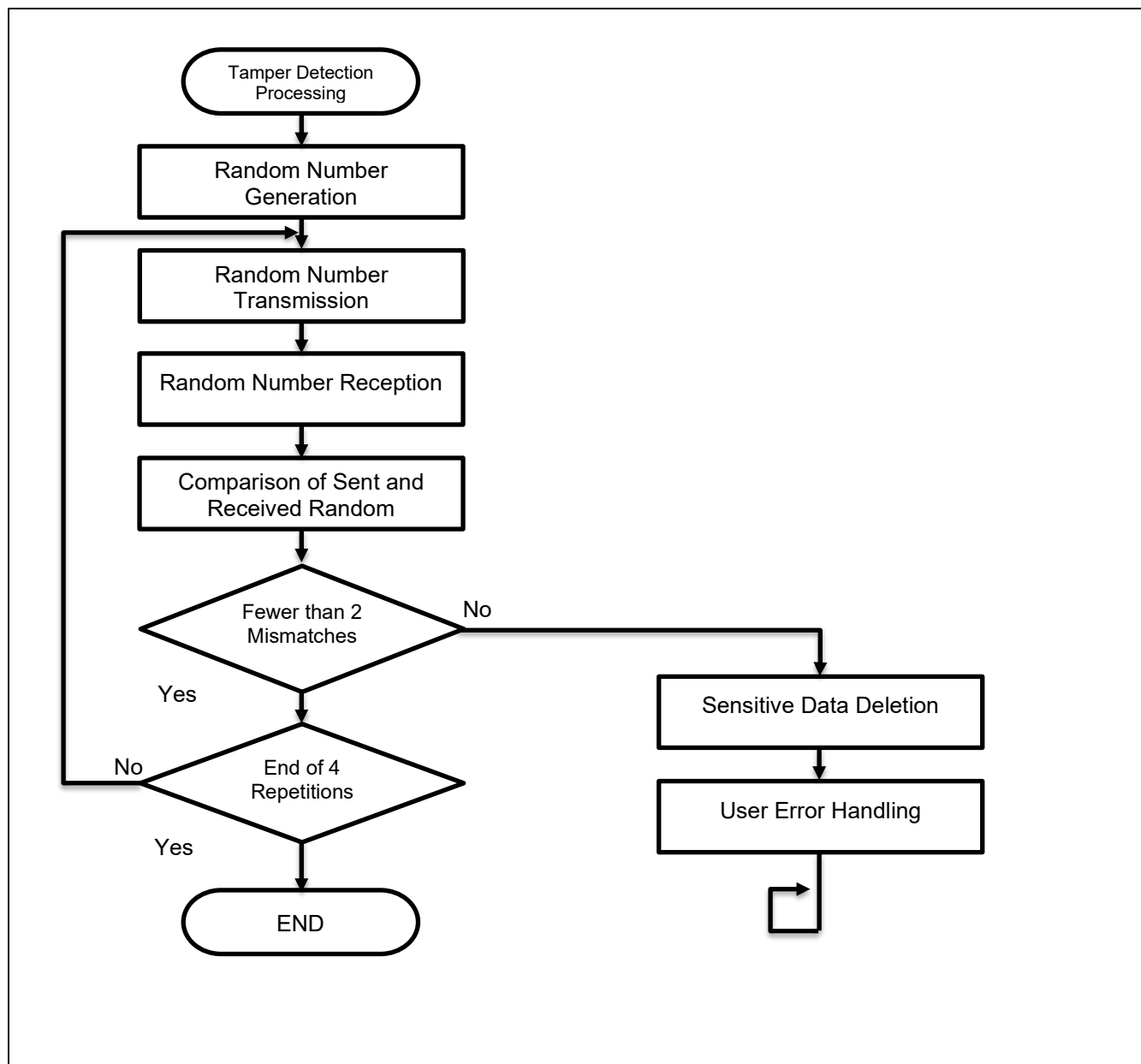
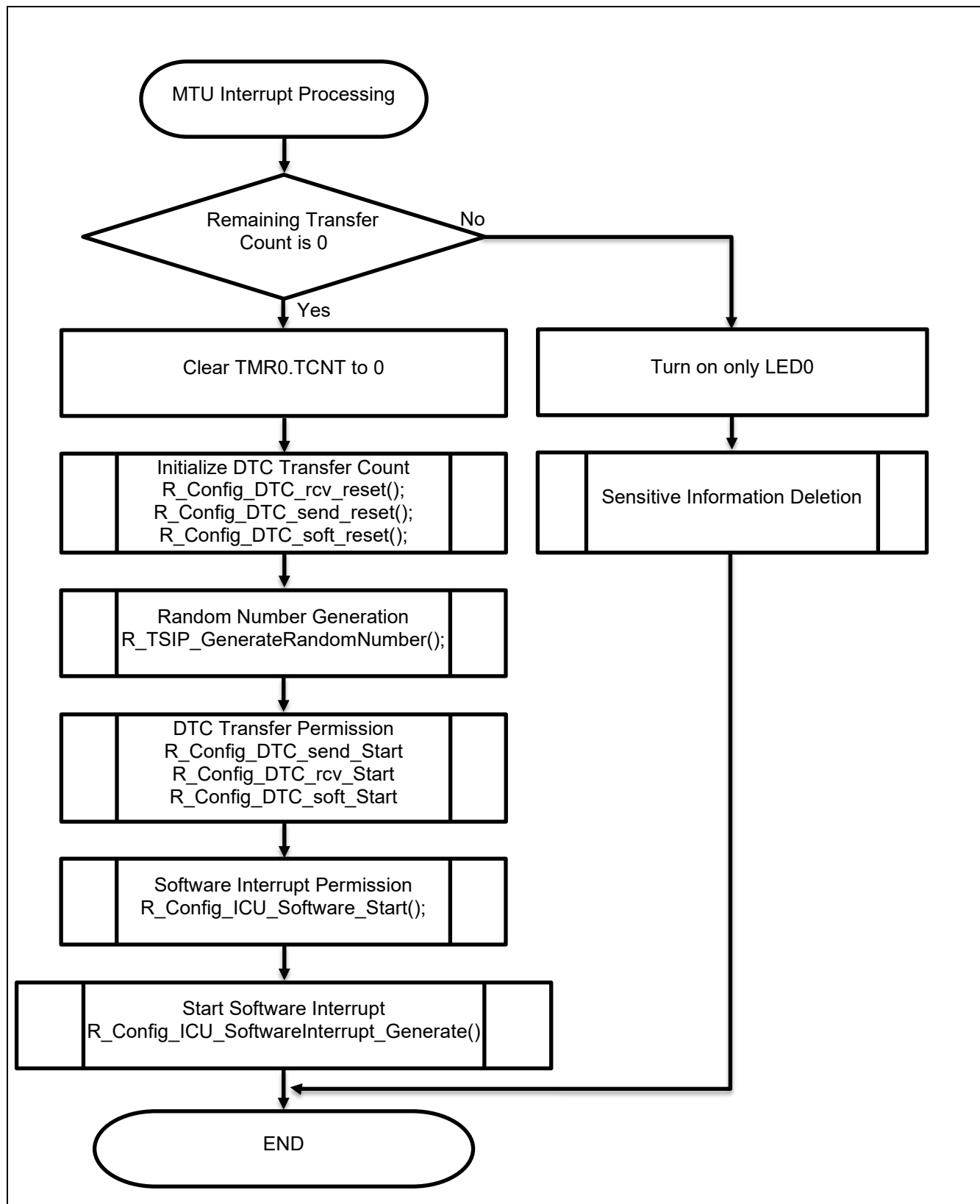


Figure 5-3 Tamper Detection Interrupt Processing by Time Overview Flow

**Figure 5-4** Timer-Based Tamper Detection Interrupt Processing Detailed Flow

5.2.4 Flow When the Timer Detects Two or More Mismatches

Figure 5-5 shows the flow when the timer detects two or more mismatches.

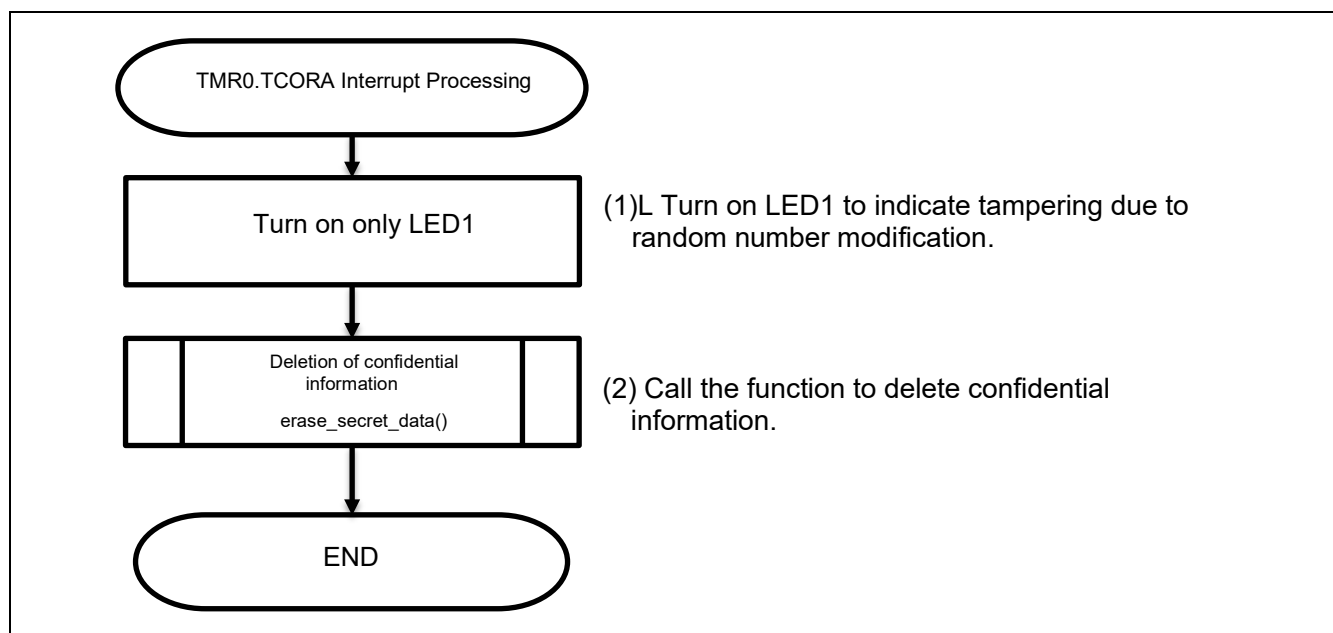


Figure 5-5 Interrupt Processing Flow for Timer When Comparison Mismatches are Two or More

5.2.5 Communication Error Interrupt Processing

Figure 5-6 illustrates the interrupt processing due to communication errors.

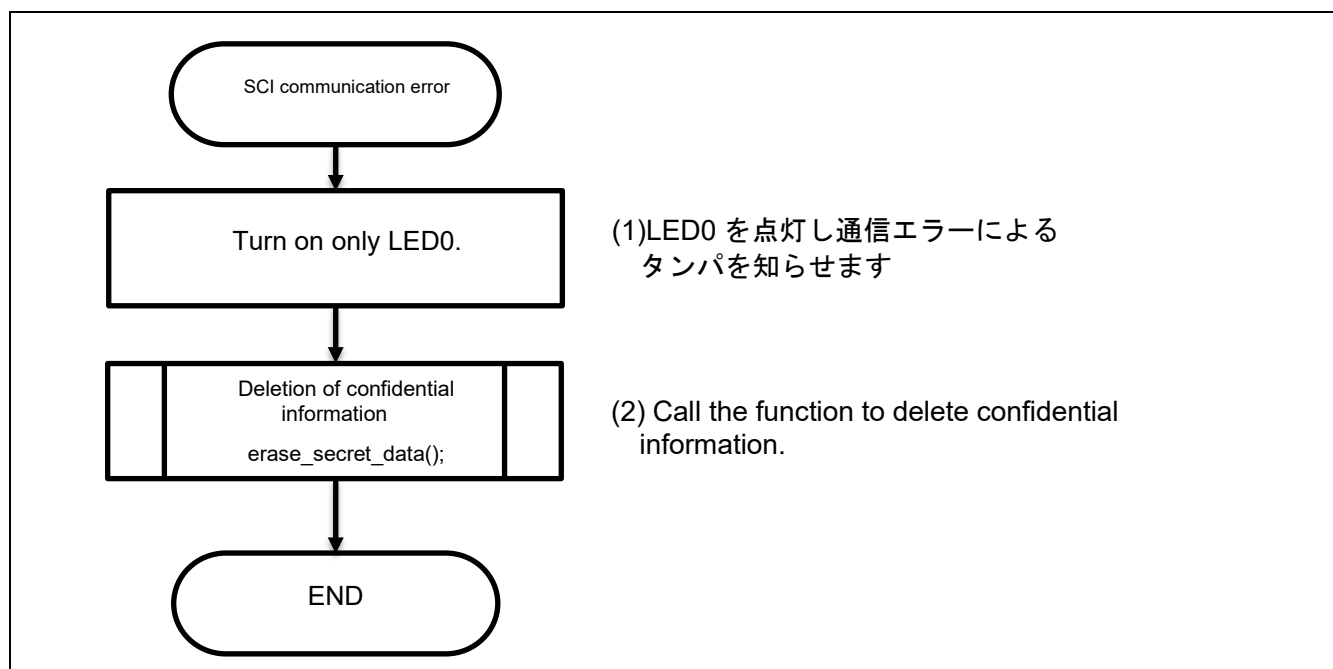


Figure 5-6 Interrupt Handling Flow for Communication Errors

5.2.6 Sensitive Data Deletion Function

Figure 5-7 shows the process of sensitive data deletion during tamper occurrence.

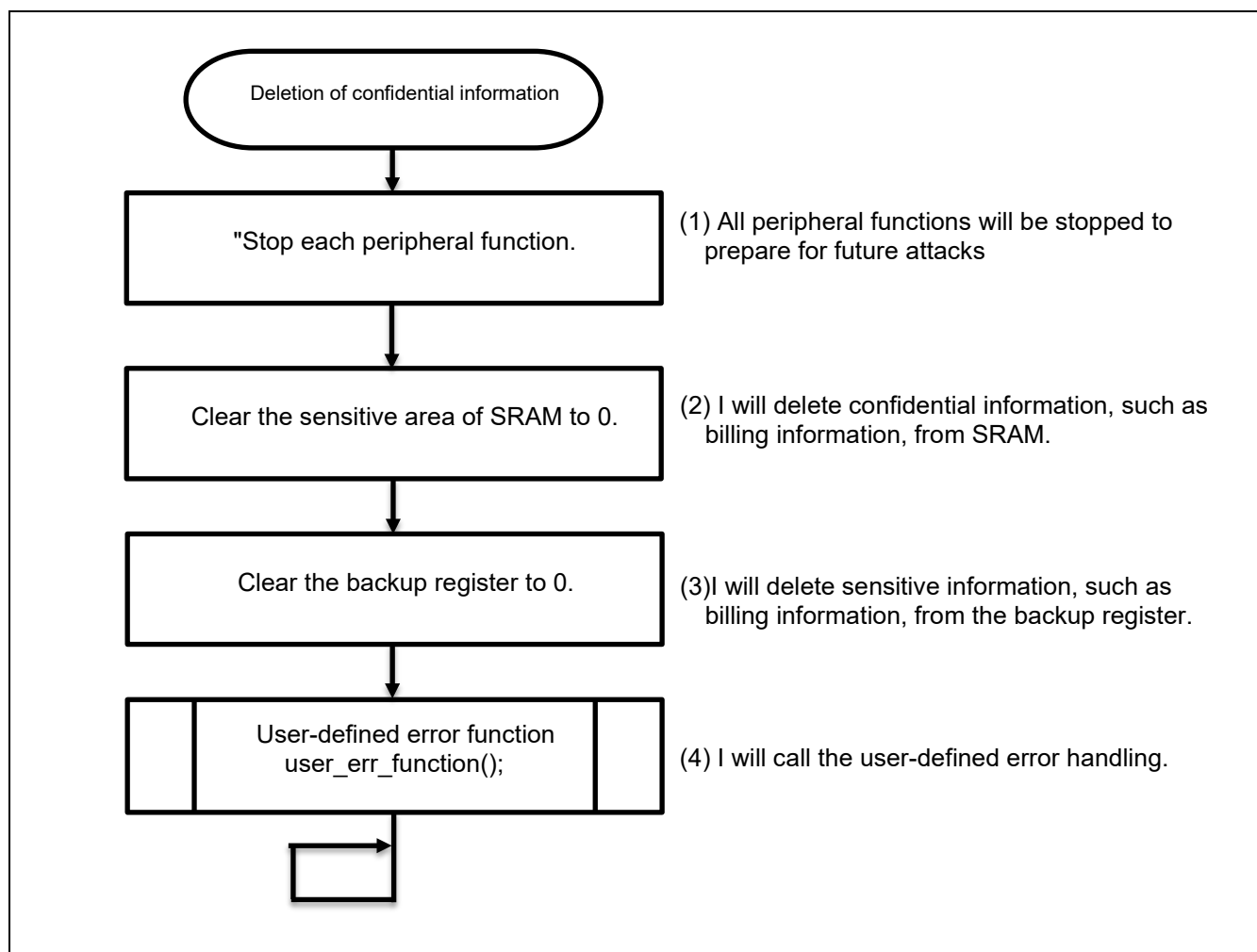


Figure 5-7 Confidential Information Deletion Flow

5.3 Footprint

Table 5-4 shows the footprint of this sample program.

Table 5-4 Footprint of the Sample Program

Program	Size
TSIP driver	245 kB
Sample Code (excluding TSIP)	10 kB

Note that the footprint of the TSIP-Lite driver installed on the RX100 and RX200 series is approximately 55kB. Therefore, this sample program can also be used with the RX100 and RX200 series.

5.4 Notes

5.4.1 Notes on Stopping Tamper Detection

Tamper detection is initiated by a compare match trigger on MTU0.TGRA. Therefore, to stop the detection operation, please disable the counting operation of MTU0.

When stopping peripheral functions other than MTU0, please first confirm that MTU0 has stopped. Then, verify that there is no transmission or reception occurring on SCI9 before stopping the other peripherals.

6. Reference Documents

[RX66N Group User's Manual Hardware \(renesas.com\)](#)

[Target Board for RX66N User's Manual Rev.1.00 \(renesas.com\)](#)

[RX Family TSIP \(Trusted Secure IP\) Module Firmware Integration Technology Rev.1.21 \(renesas.com\)](#)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	January 10, 2025	-	Initial release

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

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/.