# RL78 Family CS+
# Debugging Using Hot Plug-in Function

R20AN0248EJ0400
Rev.4.00
Dec. 20, 2023

## Introduction

This document describes the procedures of debugging using the hot plug-in function provided in some RL78 family microcontrollers such as RL78/Fxx (RL78/F12 MCU does not support hot plug-in function.).

Tool settings and connection procedures when using CS+ for CA,CX, Renesas Flash Programmer, E1 emulator, hot-plug adapter (R0E000010ACB00), and CPU board (QB-R5F10BMG-TB) are described in detail in the order of program creation -> building -> debugging steps.

When using CS+ for CC, the hot plug-in method is same, so please refer to this document. But note that only the hot plug-in initialization function is different.

Remarks 1: For operations of CS+, refer to the CS+ User's Manual.
2: For information regarding the circuit for connecting the E1 to the user system and regarding the user resources to be used, refer to " Emulator User's Manual" (document number: R20UT1994).
3: For operations of the Renesas Flash Programmer, refer to the Renesas Flash Programmer User's Manual.

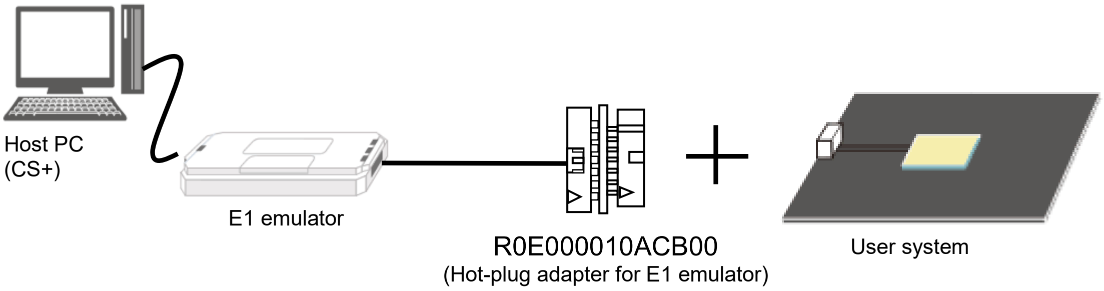


Figure 1 Overview of System

## Contents

1. Overview of Debugging Using Hot Plug-in Function .... 3
1.1 Features of Hot Plug-in Function .... 3
1.2 Debugging after Hot Plug-in Connection .... 4

2. Debugging Procedures Using Hot Plug-in Function .... 5
2.1 Overview of Operation Procedures .... 5
2.2 Actual Operations Using Hot Plug-in Function .... 6
2.2.1 Creating a Program that Allows Hot Plug-in Connection .... 6
2.2.2 Executing Build Process .... 14
2.2.3 Writing and Executing Program .... 18
2.2.4 Hot Plug-in Connection .... 19
2.2.5 Debugging .... 23
2.2.6 Disconnecting from User System .... 26

3. Notes .... 27
3.1 Note on Debug DTC Operating Clock .... 27
3.2 Note on DTC Suspending Instructions .... 27
3.3 Note on Data Access Event .... 27
3.4 Note on Access to a 32-Bit or Longer Variable .... 27
3.5 Notes on Standby Mode .... 28
3.6 Notes on Reset .... 28
3.7 Note on RAM Usage .... 28

# 1. Overview of Debugging Using Hot Plug-in Function

## 1.1 Features of Hot Plug-in Function

The hot plug-in function allows the debugger to be connected during user system operation with "program execution continued", "no reset applied", and "program contents not modified".
When a failure is found during inspection or after shipment of a user system, it can be debugged with the failure status retained through the hot plug-in function.

The following shows the main features of the hot plug-in function.

►Feature 1: The failure status can be retained.

When the debugger is connected,

- program execution can continue,
- no reset is generated, and
- program contents do not need to be modified.

►Feature 2: The program is protected by the security function.

- ID code authentication is done when the debugger is connected.

►Feature 3: Failures can be analyzed while operation continues.

- After the debugger is connected, RAM monitoring using the DTC is possible without stopping the CPU operation.

►Feature 4: Detailed analysis is possible.

- Forced break can be used.
- After a break, software break or event break can be specified.

Note: The hot plug-in function is aimed at failure analysis. Use the conventional on-chip debugging function in early stages of user program development.

## 1.2 Debugging after Hot Plug-in Connection

After hot plug-in connection of the debugger to the microcontroller that is operating, debugging can be done with the CPU operation continued through the RRM or DMM using the DTC dedicated to debugging.

The conventional RL78 microcontrollers use the CPU to access memory for debugging, but integrating the debug DTC on the chip enables memory access for debugging without using the CPU.

The following areas can be accessed through RRM/DMM using the DTC.

[Readable/writable resources]

- RAM
- SFRs

[Read-only resources]

- Data flash (only when reading is enabled)
- Mirror area
- General registers

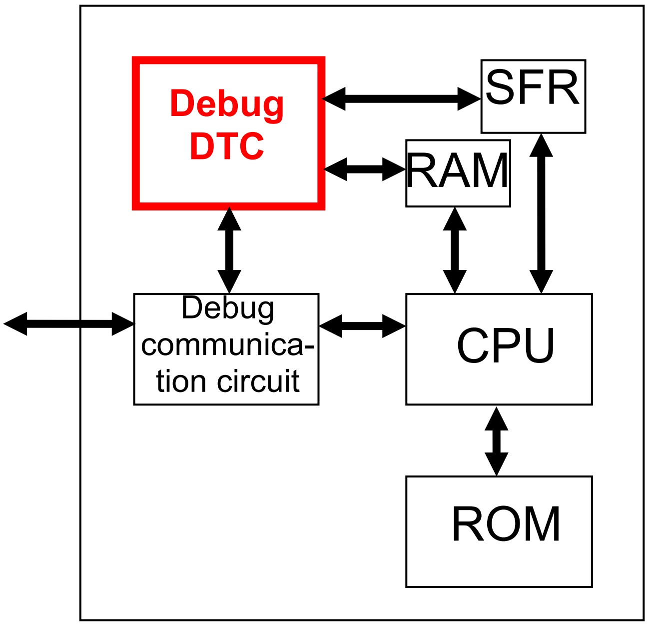


Figure 2 Concept of Debug DTC

[Description of terms]

- DTC: Data transfer controller. This transfers data between memory units without CPU intervention.
- RRM: Realtime RAM monitor. This reads data from RAM or SFRs without stopping CPU operation.
- DMM: Dynamic memory modification. This writes to RAM or SFRs without stopping CPU operation.

## 2. Debugging Procedures Using Hot Plug-in Function

This section describes the procedures for debugging using the hot plug-in function.

### 2.1 Overview of Operation Procedures

The following shows the operation procedures.
Install the CS+ before starting the procedures.

**1. Creating a program**
To use the hot plug-in function, the hot plug-in initialization function should be executed in the user program.
A later section describes how to create a user program including the hot plug-in initialization function.

**2. Building**
To use the hot plug-in function, user RAM should be allocated and the on-chip debugging option byte should be set up. A later section describes how to make these settings through build options.

**3. Writing and executing the program**
Write the hex file through the Renesas Flash Programmer and execute it on the user system.

**4. Connecting through hot plug-in function**
Execute the hot plug-in function through the E1 emulator.

**5. Debugging**
How to use the debugging functions while the debugger is connected through the hot plug-in function is described in a later section.

## 2.2 Actual Operations Using Hot Plug-in Function

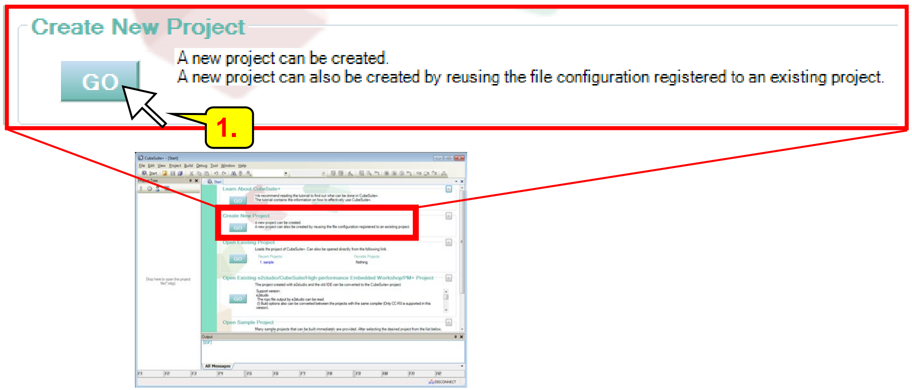
### 2.2.1 Creating a Program that Allows Hot Plug-in Connection

(1) Starting CS+

Select [Start] → [Programs] → [Renesas Electronics CS+] → [CS+ for CA,CX] to start CS+.

(2) Creating a Project

- Create a new project from the start panel.

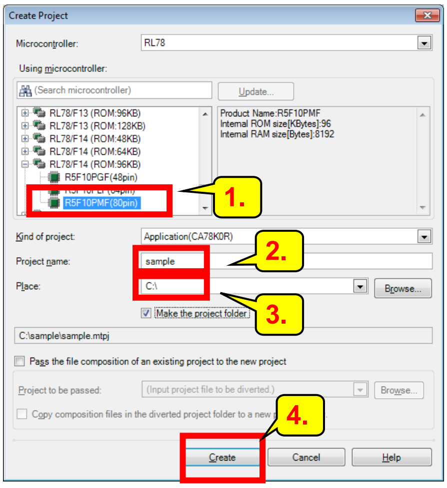


Create New Project

A new project can be created.
A new project can also be created by reusing the file configuration registered to an existing project.

| | |
|---|---|
| 1. Start creation | Click the [GO] button under "Create New Project". |

- Make detailed settings for the project.



Create Project

Microcontroller: RL78

Using microcontroller:

RL78/F13 (ROM:96KB)
RL78/F13 (ROM:128KB)
RL78/F14 (ROM:48KB)
RL78/F14 (ROM:64KB)
RL78/F14 (ROM:96KB)
R5F10PGF(48pin)
R5F10PMF(80pin)

Product Name:R5F10PMF
Internal ROM size[KBytes]:96
Internal RAM size[Bytes]:8192

Kind of project: Application(CA78K0R)

Project name: sample

Place: C:\

☑ Make the project folder

C:\sample\sample.mtpj

☐ Pass the file composition of an existing project to the new project

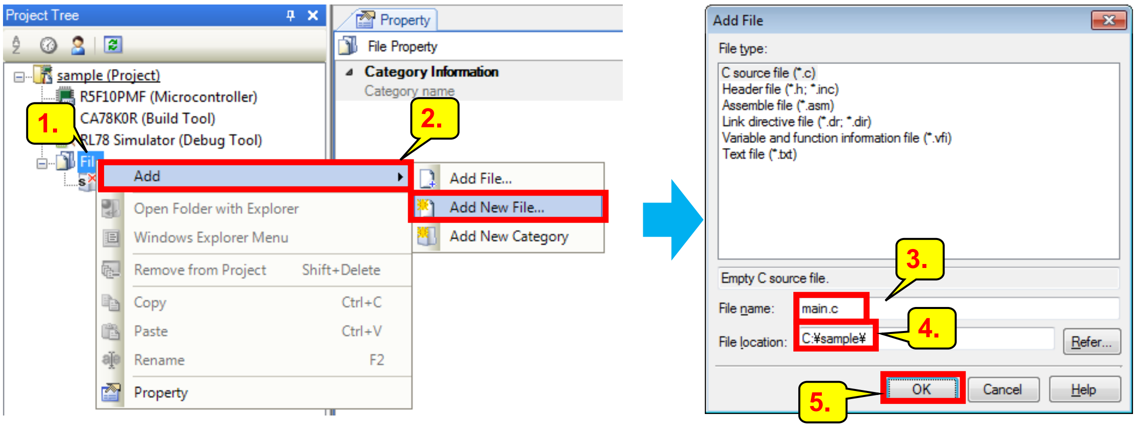
Project to be passed: (Input project file to be diverted.)

☐ Copy composition files in the diverted project folder to a new

| | | |
|---|---|---|
| 1. | Specify [Using microcontroller] | Select the name of the device to be used. In this example, "R5F10PMF(80pin)" is selected. |
| 2. | Specify [Project name] | A desired name can be specified. In this example, "sample" is specified. |
| 3. | Specify [Place] | A desired place can be specified. In this example, "C:¥" is specified. |
| 4. | Create a project | Click the [Create] button. |

## (3) Creating a User Program

- Create the main.c file.

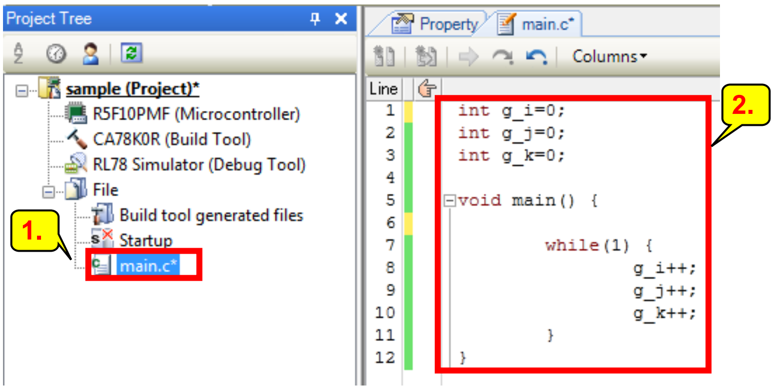


File name: main.c

File location: C:¥sample¥

| 1. Open the context menu for "File" | Select "File" in the project tree and then right-click it. |
|---|---|
| 2. Add a new file | Select [Add] → [Add New File]. |
| 3. Specify [File name] | A desired name can be specified. In this example, "main.c" is specified. |
| 4. Specify [File location] | A desired location can be specified. In this example, "C:¥sample¥" is specified. |
| 5. Create a file | Click the [OK] button. |

- Edit the main.c file.



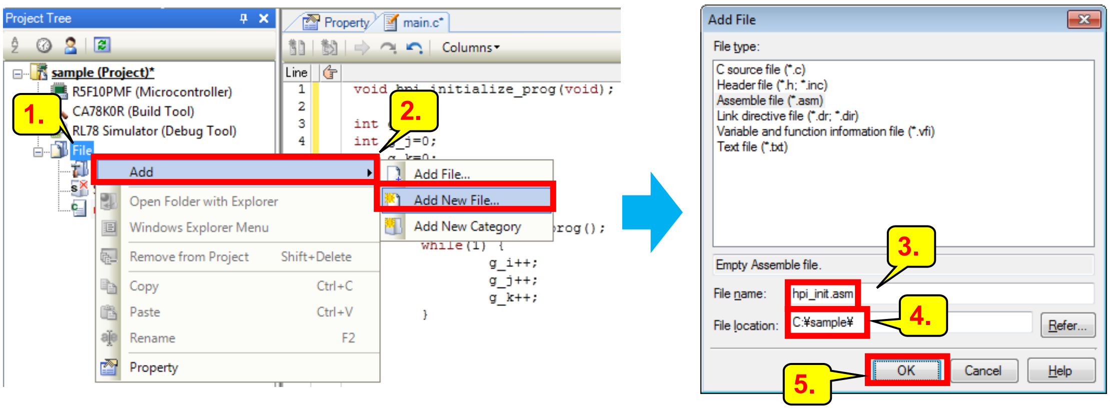
```
int g_i=0;
int g_j=0;
int g_k=0;

void main() {

        while(1) {
                g_i++;
                g_j++;
                g_k++;
        }
}
```

| 1. Open "main.c" | Double-click "main.c" in the project tree. |
|---|---|
| 2. Edit the main function | Copy and paste the following to the main.c file. |

```
int g_i=0;
int g_j=0;
int g_k=0;

void main() {
        while(1) {
                g_i++;
                g_j++;
                g_k++;
        }
}
```

## (4) Creating the Hot Plug-in Initialization Function

- Create a file for defining the hot plug-in initialization function (hpi_initialize_prog).
  Add the hpi_init.asm file to the project.

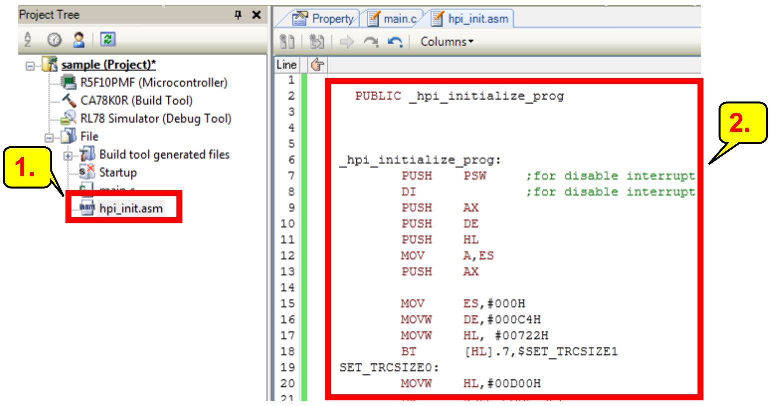


File name: hpi_init.asm

File location: C:¥sample¥

| | |
|---|---|
| 1. Open the context menu for "File" | Select "File" in the project tree and then right-click it. |
| 2. Add a new file | Select [Add] → [Add New File]. |
| 3. Specify [File name] | A desired name can be specified. In this example, "hpi_init.asm" is specified. |
| 4. Specify [File location] | A desired location can be specified. In this example, "C:¥sample¥" is specified. |
| 5. Create a file | Click the [OK] button. |

- Edit the hot plug-in initialization function (hpi_initialize_prog).



```
  PUBLIC _hpi_initialize_prog



_hpi_initialize_prog:
        PUSH    PSW     ;for disable interrupt
        DI              ;for disable interrupt
        PUSH    AX
        PUSH    DE
        PUSH    HL
        MOV     A,ES
        PUSH    AX

        MOV     ES,#000H
        MOVW    DE,#000C4H
        MOVW    HL, #00722H
        BT      [HL].7,$SET_TRCSIZE1
SET_TRCSIZE0:
        MOVW    HL,#00D00H
```

| | |
|---|---|
| 1. Open "hpi_init.asm" | Double-click "hpi_init.asm" in the project tree. |
| 2. Edit the hot plug-in initialization function | Copy and paste the contents on the next page to the hpi_init.asm file. |

Note: The following shows the hot plug-in initialization function, which initializes and enables the DTC dedicated to debugging. Copy this function to the user program without change.

[How to Use Initialization Function]

- Call the function with the earliest possible timing after RAM initialization is completed. Execution of the initialization function should be completed within 400 ms after the CPU reset is canceled. If execution is not completed within this period, the debugger will hang when a pin reset or a POC reset occurs after hot plug-in connection.
- Call this function only once after a reset.
- As SFRs are manipulated in the initialization function, disable the SFR guard function before calling this function.
- In this initialization function, interrupts are disabled until the execution of the function is completed. When interrupts need to be enabled, delete three lines including "for disable interrupt" as comments. (When interrupts are enabled, take care of the time until the execution of the initialization function is completed and the stack size to be used.)

CS+ for CA,CX (RL78/F1x)

```
  PUBLIC _hpi_initialize_prog

_hpi_initialize_prog:
          PUSH      PSW       ;for disable interrupt
          DI                  ;for disable interrupt
          PUSH      AX
          PUSH      DE
          PUSH      HL
          MOV       A,ES
          PUSH      AX

          MOV       ES,#000H
          MOVW      DE,#000C4H
          MOVW      HL, #00722H
          BT        [HL].7,$SET_TRCSIZE1
SET_TRCSIZE0:
          MOVW      HL,#00D00H
          BR        $SEC_CODE_SET
SET_TRCSIZE1:
          MOVW      HL,#00E00H
SEC_CHK:
          MOV       A,!0723H
          BT        A.0, $DTC_DES_SET

SEC_CODE_SET:
          MOV       A,ES:[DE]
          MOV       [HL],A
          INCW      DE
          INCW      HL
          MOVW      AX,DE
          CMPW      AX,#000CEH
          SKZ
          BR        $SEC_CODE_SET
DTC_DES_SET:
          MOV       ES,#00FH
          MOV       L,#00EH
          MOVW      AX,#0FFADH
          MOVW      [HL],AX
          MOV       L,#010H
DTC_SET:
          MOV       [HL+00H],#013H
          MOV       [HL+01H],#001H
          MOV       [HL+02H],#00AH
          MOV       [HL+03H],#00AH
          MOV       [HL+04H],#000H
          MOV       A,H
          MOV       [HL+05H],A
          MOV       [HL+06H],#023H
          MOV       [HL+07H],#007H
          MOV       [HL+08H],#000H
          MOV       [HL+09H],#001H
          MOV       [HL+0AH],#001H
```

```
            MOV     [HL+0BH],#001H
            MOV     [HL+0CH],#023H
            MOV     [HL+0DH],#007H
            MOV     [HL+0EH],#0ADH
            MOV     [HL+0FH],#0FFH
            MOV     [HL+10H],#011H
            MOV     [HL+11H],#001H
            MOV     [HL+12H],#010H
            MOV     [HL+13H],#010H
            MOV     [HL+14H],#0ADH
            MOV     [HL+15H],#0FFH
            MOV     [HL+16H],#000H
            MOV     A,H
            MOV     [HL+17H],A
            MOV     [HL+18H],#003H
            MOV     [HL+19H],#001H
            MOV     [HL+1AH],#010H
            MOV     [HL+1BH],#010H
            MOV     [HL+1CH],#000H
            MOV     A,H
            MOV     [HL+1DH],A
            MOV     [HL+1EH],#0ADH
            MOV     [HL+1FH],#0FFH
            SET1    DTCEN

            POP     AX
            MOV     ES,A
            POP     HL
            POP     DE
            POP     AX
            POP     PSW         ;for disable interrupt

            RET
            END
```

### CS+ for CC (RL78/F1x)

```
. PUBLIC _hpi_initialize_prog

_hpi_initialize_prog:
            PUSH PSW            ;for disable interrupt
            DI                  ;for disable interrupt
            PUSH AX
            PUSH DE
            PUSH HL
            MOV A,ES
            PUSH AX
            MOV ES,#0x000
            MOVW DE,#0x000C4
            MOVW HL, #0x00722
            BT [HL].7,$SET_TRCSIZE1
            SET_TRCSIZE0:
            MOVW HL,#0x00D00
            BR $SEC_CODE_SET
SET_TRCSIZE1:
            MOVW HL,#0x00E00
SEC_CHK:
            MOV A,!0x0723
            BT A.0, $DTC_DES_SET
SEC_CODE_SET:
            MOV A,ES:[DE]
            MOV [HL],A
            INCW DE
            INCW HL
            MOVW AX,DE
            CMPW AX,#0x000CE
            SKZ
            BR $SEC_CODE_SET
DTC_DES_SET:
            MOV ES,#0x00F
            MOV L,#0x00E
            MOVW AX,#0x0FFAD
            MOVW [HL],AX
            MOV L,#0x010
DTC_SET:
            MOV [HL+0x00],#0x013
            MOV [HL+0x01],#0x001
            MOV [HL+0x02],#0x00A
            MOV [HL+0x03],#0x00A
            MOV [HL+0x04],#0x000
            MOV A,H
            MOV [HL+0x05],A
            MOV [HL+0x06],#0x023
            MOV [HL+0x07],#0x007
            MOV [HL+0x08],#0x000
```

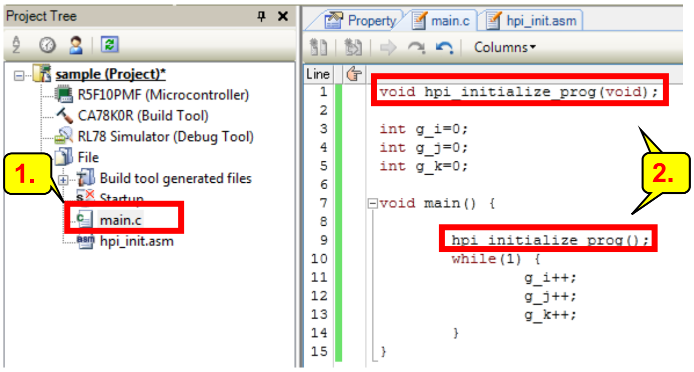
```
                MOV [HL+0x09],#0x001
                MOV [HL+0x0A],#0x001
                MOV [HL+0x0B],#0x001
                MOV [HL+0x0C],#0x023
                MOV [HL+0x0D],#0x007
                MOV [HL+0x0E],#0x0AD
                MOV [HL+0x0F],#0x0FF
                MOV [HL+0x10],#0x011
                MOV [HL+0x11],#0x001
                MOV [HL+0x12],#0x010
                MOV [HL+0x13],#0x010
                MOV [HL+0x14],#0x0AD
                MOV [HL+0x15],#0x0FF
                MOV [HL+0x16],#0x000
                MOV A,H
                MOV [HL+0x17],A
                MOV [HL+0x18],#0x003
                MOV [HL+0x19],#0x001
                MOV [HL+0x1A],#0x010
                MOV [HL+0x1B],#0x010
                MOV [HL+0x1C],#0x000
                MOV A,H
                MOV [HL+0x1D],A
                MOV [HL+0x1E],#0x0AD
                MOV [HL+0x1F],#0x0FF
                SET1 !DTCEN
                POP AX
                MOV ES,A
                POP HL
                POP DE
                POP AX
                POP PSW                 ;for disable interrupt
                RET
```

CS+ for CC (RL78/F2x)

```
.PUBLIC _hpi_initialize_prog_f2x

_hpi_initialize_prog_f2x:
                PUSH PSW                ;for disable interrupt
                DI                      ;for disable interrupt
                PUSH AX
                PUSH DE
                PUSH HL
                MOV A,ES
                PUSH AX
                MOV ES,#0x000
                MOVW DE,#0x000C4
                MOV A,ES:[DE]
                BT A.2,$GO_SETHPI
                BR $!NOHPISET_RETURN
GO_SETHPI:
                MOVW DE,#0x000C6
                MOVW HL, #0x00722
                BT [HL].7,$SET_TRCSIZE1
SET_TRCSIZE0:
                MOVW HL,#0x00D00
                BR $SEC_CHK
SET_TRCSIZE1:
                MOVW HL,#0x00E00
SEC_CHK:
                MOV A,!0x0723
                BT A.0, $DTC_DES_SET
SEC_CODE_SET:
                MOV A,ES:[DE]
                MOV [HL],A
                INCW DE
                INCW HL
                MOVW AX,DE
                CMPW AX,#0x000D6
                SKZ
                BR $SEC_CODE_SET
                BR $DTC_SET_SUB
DTC_DES_SET:
                MOV ES,#0x00F
                MOV L,#0x00E
                MOVW AX,#0x0FFAD
                MOVW [HL],AX
DTC_SET_SUB:
                MOV L,#0x010
DTC_SET:
                MOV [HL+0x00],#0x013
                MOV [HL+0x01],#0x001
                MOV [HL+0x02],#0x010
```

```
            MOV [HL+0x03],#0x010
            MOV [HL+0x04],#0x000
            MOV A,H
            MOV [HL+0x05],A
            MOV [HL+0x06],#0x023
            MOV [HL+0x07],#0x007
            MOV [HL+0x08],#0x000
            MOV [HL+0x09],#0x001
            MOV [HL+0x0A],#0x001
            MOV [HL+0x0B],#0x001
            MOV [HL+0x0C],#0x023
            MOV [HL+0x0D],#0x007
            MOV [HL+0x0E],#0x0AD
            MOV [HL+0x0F],#0x0FF
            MOV [HL+0x10],#0x011
            MOV [HL+0x11],#0x001
            MOV [HL+0x12],#0x010
            MOV [HL+0x13],#0x010
            MOV [HL+0x14],#0x0AD
            MOV [HL+0x15],#0x0FF
            MOV [HL+0x16],#0x000
            MOV A,H
            MOV [HL+0x17],A
            MOV [HL+0x18],#0x003
            MOV [HL+0x19],#0x001
            MOV [HL+0x1A],#0x010
            MOV [HL+0x1B],#0x010
            MOV [HL+0x1C],#0x000
            MOV A,H
            MOV [HL+0x1D],A
            MOV [HL+0x1E],#0x0AD
            MOV [HL+0x1F],#0x0FF
            SET1 !DTCEN
NOHPISET_RETURN:
            POP AX
            MOV ES,A
            POP HL
            POP DE
            POP AX
            POP PSW                 ;for disable interrupt
            RET
```

- Add the lines for calling the hot plug-in initialization function (hpi_initialize_prog) to the main function.



```
void hpi_initialize_prog(void);

int g_i=0;
int g_j=0;
int g_k=0;

void main() {

        hpi_initialize_prog();
        while(1) {
                g_i++;
                g_j++;
                g_k++;
        }
}
```

| | |
|---|---|
| 1. Open "main.c" | Double-click "main.c" in the project tree. |
| 2. Write the hot plug-in initialization function | Add the following to the main.c file. |

```
void hpi_initialize_prog(void);

int g_i=0;
int g_j=0;
int g_k=0;

void main() {

        hpi_initialize_prog();
        while(1) {
                g_i++;
                g_j++;
                g_k++;
        }
}
```

**Add this line.**

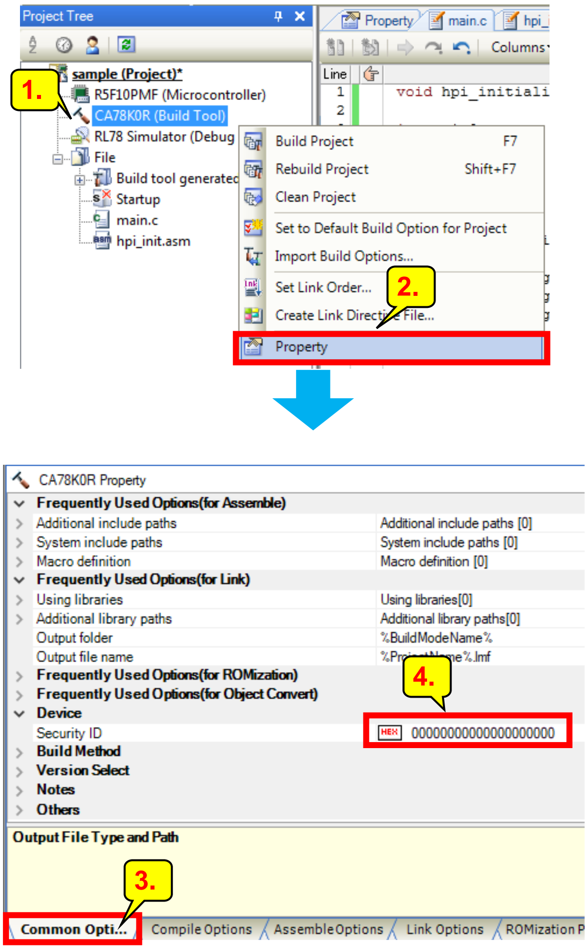
**Add this line.**

### 2.2.2 Executing Build Process

#### (1) Specifying the Security ID

- Specify the security ID.

  Use the [Common Options] tab in the property panel for the build tool.

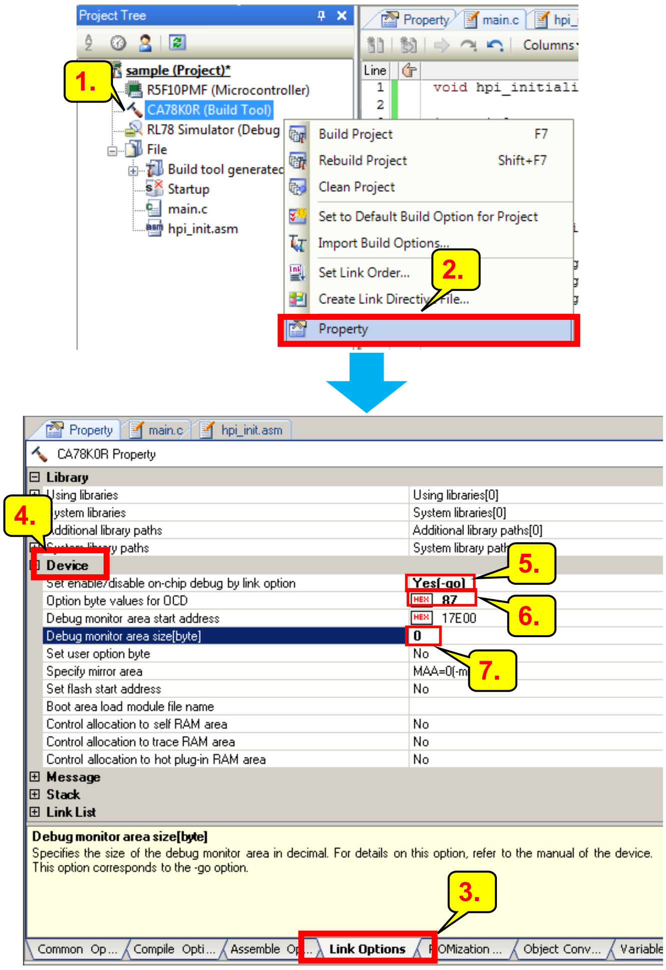


Frequently Used Options(for Assemble)
Additional include paths: Additional include paths [0]
System include paths: System include paths [0]
Macro definition: Macro definition [0]
Frequently Used Options(for Link)
Using libraries: Using libraries[0]
Additional library paths: Additional library paths[0]
Output folder: %BuildModeName%
Output file name: %ProjectName%.lmf
Frequently Used Options(for ROMization)
Frequently Used Options(for Object Convert)
Device
Security ID: 00000000000000000000
Build Method
Version Select
Notes
Others

| | |
|---|---|
| 1. Open the context menu for "CA78K0R (Build Tool)" | Select "CA78K0R (Build Tool)" in the project tree and then right-click it. |
| 2. Open the property panel | Select [Property] from the context menu. |
| 3. Open the [Common Options] tab | Select the [Common Options] tab in the main panel. |
| 4. Specify [Security ID] | Specify [Security ID] under "Device". A desired ID can be specified. In this example, "00000000000000000000" is specified. |

### (2) Specifying the Option Bytes

- Specify the option byte for on-chip debugging.
  Use the "Link Options" tab in the property panel for the build tool.



Library
- Using libraries: Using libraries[0]
- System libraries: System libraries[0]
- Additional library paths: Additional library paths[0]
- System library paths: System library path

Device
- Set enable/disable on-chip debug by link option: Yes(-go)
- Option byte values for OCD: HEX 87
- Debug monitor area start address: HEX 17E00
- Debug monitor area size[byte]: 0
- Set user option byte: No
- Specify mirror area: MAA=0(-m
- Set flash start address: No
- Boot area load module file name:
- Control allocation to self RAM area: No
- Control allocation to trace RAM area: No
- Control allocation to hot plug-in RAM area: No

**Debug monitor area size[byte]**
Specifies the size of the debug monitor area in decimal. For details on this option, refer to the manual of the device.
This option corresponds to the -go option.

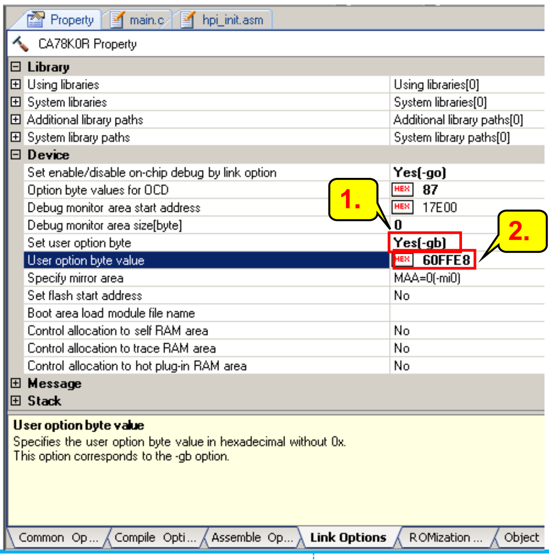
| | |
|---|---|
| 1. Open the context menu for "CA78K0R (Build Tool)" | Select "CA78K0R (Build Tool)" in the project tree and then right-click it. |
| 2. Open the property panel | Select "Property" from the context menu. |
| 3. Open the [Link Options] tab | Select the [Link Options] tab in the main panel. |
| 4. Expand "Device" | Expand "Device". |
| 5. [Set enable/disable on-chip debug by link option] | Select "Yes". |
| 6. [Option bye values for OCD] | Specify "87". |
| 7. [Debug monitor area size] | Specify "0".<br>Note: Specify an appropriate size when normal on-chip debugging is done without using the hot plug-in function. |

[Option Byte Setting for On-chip Debugging]

When using the hot plug-in function, be sure to set the option byte for on-chip debugging to 0x87. The hot plug-in is detected using the low-speed on-chip oscillator. When the above value is specified, the low-speed on-chip oscillator operates and cannot be stopped from the user program. However, only in standby mode, it can be stopped through register settings.

- Specify the user option byte.

  Use the [Link Options] tab in the property panel for the build tool.



| | |
|---|---|
| **Library** | |
| Using libraries | Using libraries[0] |
| System libraries | System libraries[0] |
| Additional library paths | Additional library paths[0] |
| System library paths | System library paths[0] |
| **Device** | |
| Set enable/disable on-chip debug by link option | Yes(-go) |
| Option byte values for OCD | 87 |
| Debug monitor area start address | 17E00 |
| Debug monitor area size[byte] | 0 |
| Set user option byte | Yes(-gb) |
| User option byte value | 60FFE8 |
| Specify mirror area | MAA=0(-mi0) |
| Set flash start address | No |
| Boot area load module file name | |
| Control allocation to self RAM area | No |
| Control allocation to trace RAM area | No |
| Control allocation to hot plug-in RAM area | No |
| **Message** | |
| **Stack** | |

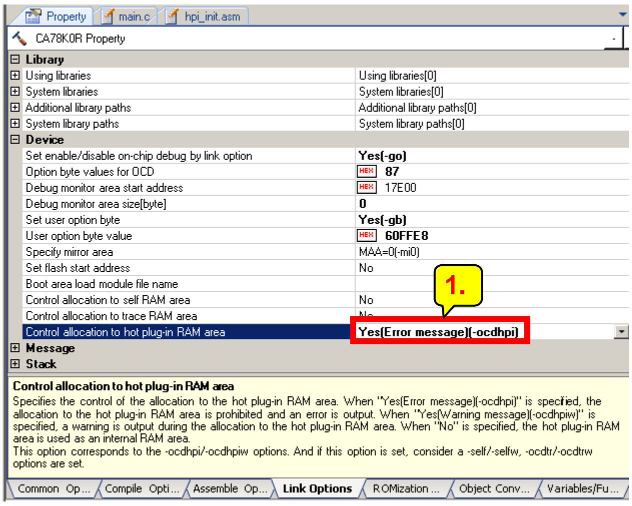
**User option byte value**
Specifies the user option byte value in hexadecimal without 0x.
This option corresponds to the -gb option.

| | |
|---|---|
| 1. [Set user option byte] | Select "Yes". |
| 2. [User option byte value] | A desired value can be specified. In this example, "60FFE8" is specified. |

### (3) Allocating RAM for Hot Plug-in

- Use the [Link Options] tab in the property panel for the build tool to prohibit the user program from using the RAM area that is to be used for hot plug-in.

  Note: For the product types that do not need RAM allocation to the hot plug-in function, the following item will not appear in the CS+ display. Skip this setting and move to the next step.



| | |
|---|---|
| **Library** | |
| Using libraries | Using libraries[0] |
| System libraries | System libraries[0] |
| Additional library paths | Additional library paths[0] |
| System library paths | System library paths[0] |
| **Device** | |
| Set enable/disable on-chip debug by link option | Yes(-go) |
| Option byte values for OCD | 87 |
| Debug monitor area start address | 17E00 |
| Debug monitor area size[byte] | 0 |
| Set user option byte | Yes(-gb) |
| User option byte value | 60FFE8 |
| Specify mirror area | MAA=0(-mi0) |
| Set flash start address | No |
| Boot area load module file name | |
| Control allocation to self RAM area | No |
| Control allocation to trace RAM area | No |
| Control allocation to hot plug-in RAM area | Yes(Error message)(-ocdhpi) |
| **Message** | |
| **Stack** | |

**Control allocation to hot plug-in RAM area**
Specifies the control of the allocation to the hot plug-in RAM area. When "Yes(Error message)(-ocdhpi)" is specified, the allocation to the hot plug-in RAM area is prohibited and an error is output. When "Yes(Warning message)(-ocdhpiw)" is specified, a warning is output during the allocation to the hot plug-in RAM area. When "No" is specified, the hot plug-in RAM area is used as an internal RAM area.
This option corresponds to the -ocdhpi/-ocdhpiw options. And if this option is set, consider a -self/-selfw, -ocdtr/-ocdtrw options are set.

| | |
|---|---|
| 1. [Control allocation to hot plug-in RAM area] | Select "Yes(Error message)". |

### (4) Executing the Build Process

- Settings of edit and build tool options have been completed. Execute the build process.

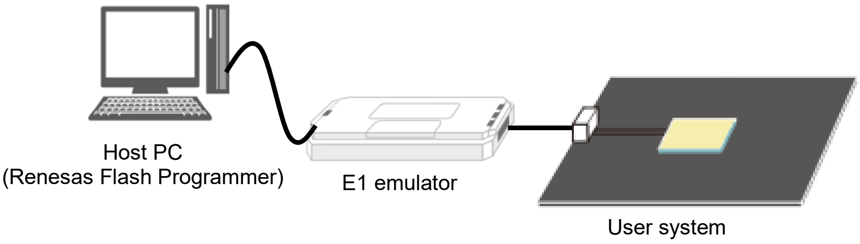
| | |
|---|---|
| 1. Execute rebuild process | Click the [Rebuild] menu button. |

### 2.2.3 Writing and Executing Program

(1) Writing a Program

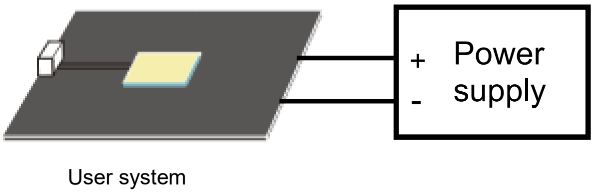
- Write the created hex file to the target device through the Renesas Flash Programmer.

 For usage of the Renesas Flash Programmer, refer to the user's manual that can be found by selecting [Start] → [Programs] → [Renesas Electronics Utilities] → [Programming Tools] → [Renesas Flash Programmer Vx.xx].



(2) Executing the Program

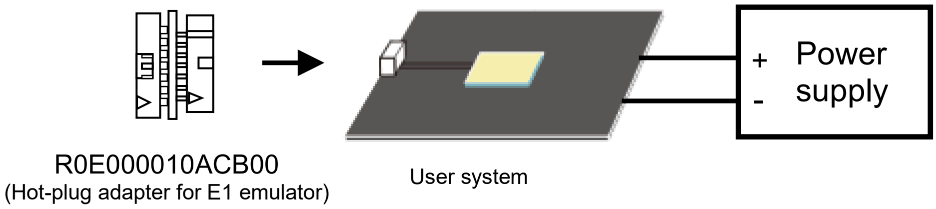
- Supply power to the user system and execute the program on the actual device.



### 2.2.4 Hot Plug-in Connection

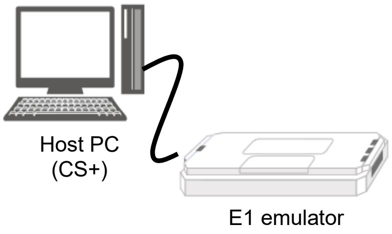
(1) Connecting the Hot-Plug Adapter (R0E000010ACB00) to the User System

- Be sure to connect the R0E000010ACB0 to the user system first.
  The R0E000010ACB00 has a mechanism that ensures that connection to the E1 emulator begins with the GND line.



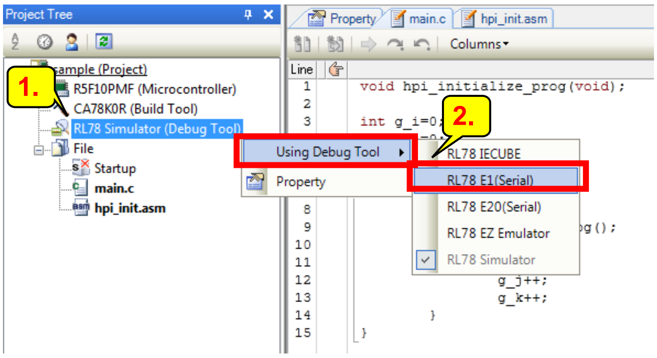
(2) Connecting the E1 Emulator to the Host PC

- Connect the E1 to the host PC.



(3) Making Debugging Function Settings

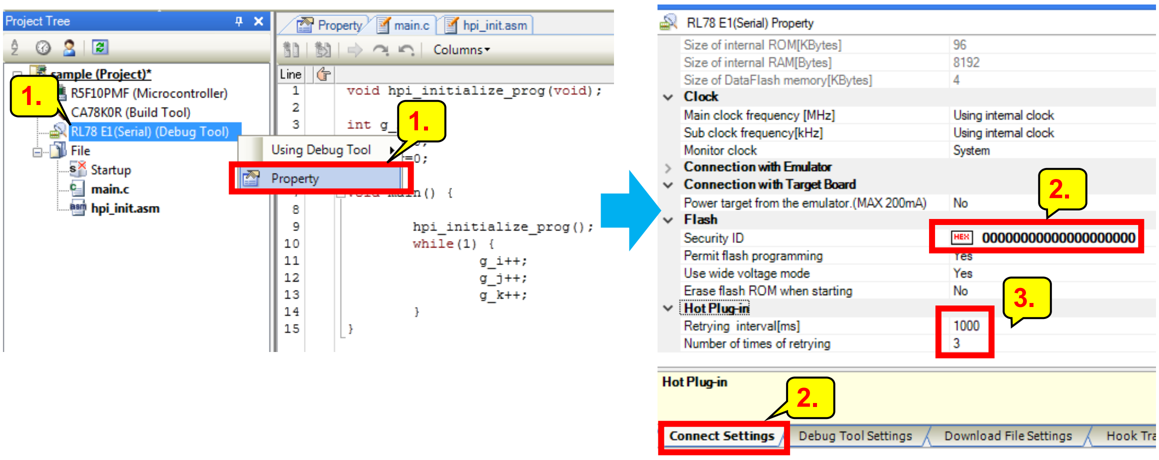
- Select a debug tool in CS+.



```
void hpi_initialize_prog(void);

int g_i=0;
```

| | |
|---|---|
| 1. Open the context menu for "Debug Tool" | Select "Debug Tool" in the project tree and then right-click it. |
| 2. Select a debug tool | Select [Using Debug Tool] → [RL78 E1(Serial)]. |

- Specify the security ID for authentication of E1 emulator connection. In addition, specify retry settings for the hot plug-in function.



```
void hpi_initialize_prog(void);

int g_
      =0;

         n() {

        hpi_initialize_prog();
        while(1) {
                g_i++;
                g_j++;
                g_k++;
        }
}
```

RL78 E1(Serial) Property

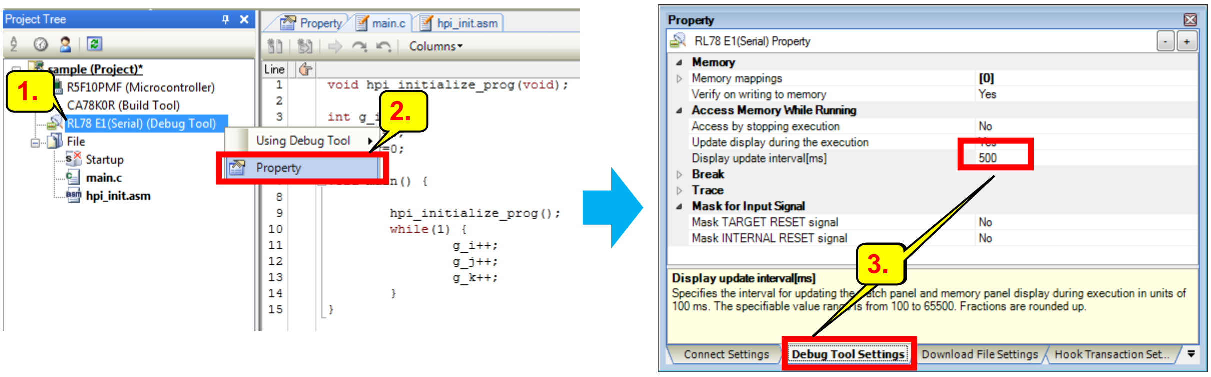
| | |
|---|---|
| Size of internal ROM[KBytes] | 96 |
| Size of internal RAM[Bytes] | 8192 |
| Size of DataFlash memory[KBytes] | 4 |
| **Clock** | |
| Main clock frequency [MHz] | Using internal clock |
| Sub clock frequency[kHz] | Using internal clock |
| Monitor clock | System |
| **Connection with Emulator** | |
| **Connection with Target Board** | |
| Power target from the emulator.(MAX 200mA) | No |
| **Flash** | |
| Security ID | HEX 00000000000000000000 |
| Permit flash programming | Yes |
| Use wide voltage mode | Yes |
| Erase flash ROM when starting | No |
| **Hot Plug-in** | |
| Retrying interval[ms] | 1000 |
| Number of times of retrying | 3 |

| | |
|---|---|
| 1. Open the property panel for "Debug Tool" | Select "Debug Tool" in the project tree and then right-click it. |
| 2. Enter the security ID for authentication. | Specify [Security ID] under "Flash" in the [Connect Settings] tab.<br>Enter the same value as that specified as a common option for the build tool. In this example, enter "00000000000000000000". |
| 3. Make retry settings for hot plug-in | Hot plug-in connection may fail when the low-speed on-chip oscillator stops due to, for example, the CPU being in the STOP mode. For such a case, specify the interval and number of hot plug-in retries.<br>Since the program in this example does not stop the low-speed on-chip oscillator, these settings are left as the default values (1000 ms and three times). |

[Retry Settings for Hot Plug-in]

- For the retrying interval, enter the longest period for which the STOP mode continues in the program. Note that the debugger cannot be operated during the retry processing (retrying interval × number of retries).
  Therefore, be careful not to make the retry processing period extremely long.

- Specify the RAM monitoring interval.



RL78 E1(Serial) Property

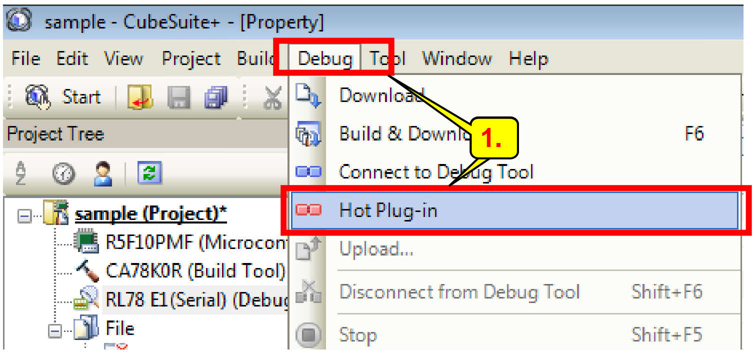
| | |
|---|---|
| **Memory** | |
| Memory mappings | [0] |
| Verify on writing to memory | Yes |
| **Access Memory While Running** | |
| Access by stopping execution | No |
| Update display during the execution | Yes |
| Display update interval[ms] | 500 |
| **Break** | |
| **Trace** | |
| **Mask for Input Signal** | |
| Mask TARGET RESET signal | No |
| Mask INTERNAL RESET signal | No |

**Display update interval[ms]**
Specifies the interval for updating the watch panel and memory panel display during execution in units of 100 ms. The specifiable value range is from 100 to 65500. Fractions are rounded up.

| | |
|---|---|
| 1. Open the context menu for "Debug Tool" | Select "Debug Tool" in the project tree and then right-click it. |
| 2. Open the property panel | Select [Property]. |
| 3. Specify [Display update interval] | Specify [Display update interval] under "Access Memory While Running" in the [Debug Tool Settings] tab.<br>A value of 100 ms or greater can be specified.<br>In this example, "500" is specified. |

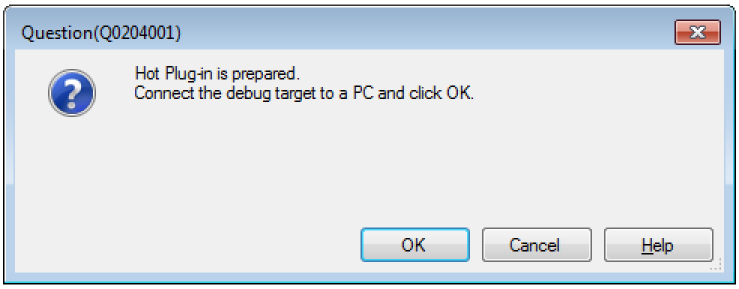
(4) Executing Hot Plug-in Connection

- Prepare for hot plug-in connection.



| | |
|---|---|
| 1. Shift the debugger to the hot plug-in preparation state | Select [Hot Plug-in] from the [Debug] menu. |

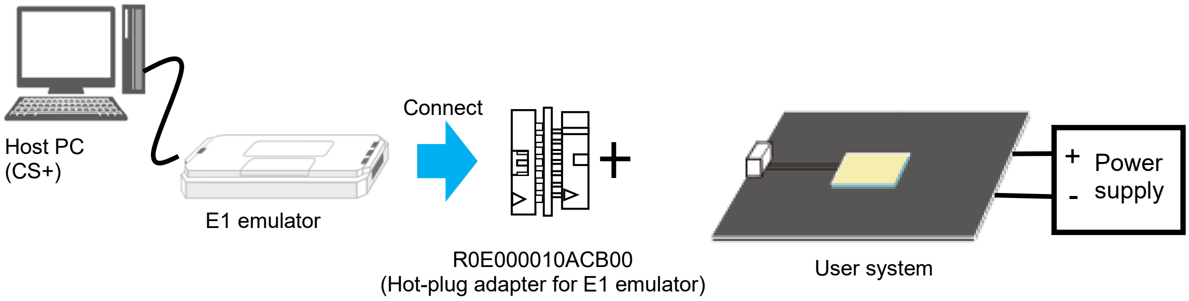
- Check that the following dialog appears.


Question(Q0204001)

Hot Plug-in is prepared.
Connect the debug target to a PC and click OK.

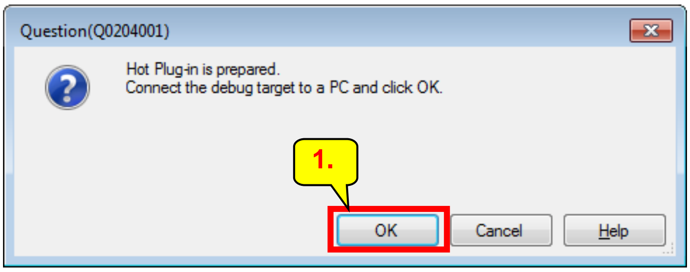
- Connecting the E1 emulator to the hot-plug adapter (R0E000010ACB00)

  Connect the E1 to the R0E000010ACB00. Connect the E1 emulator and hot-plug adapter while the connectors are correctly aligned on both sides.



- Hot plug-in connection

  After connecting the emulator, click the [OK] button on the following box.



Hot Plug-in is prepared.
Connect the debug target to a PC and click OK.

| | |
|---|---|
| 1. Hot plug-in connection | Click the [OK] button. |

[Unavailable Functions after Hot Plug-in Connection]

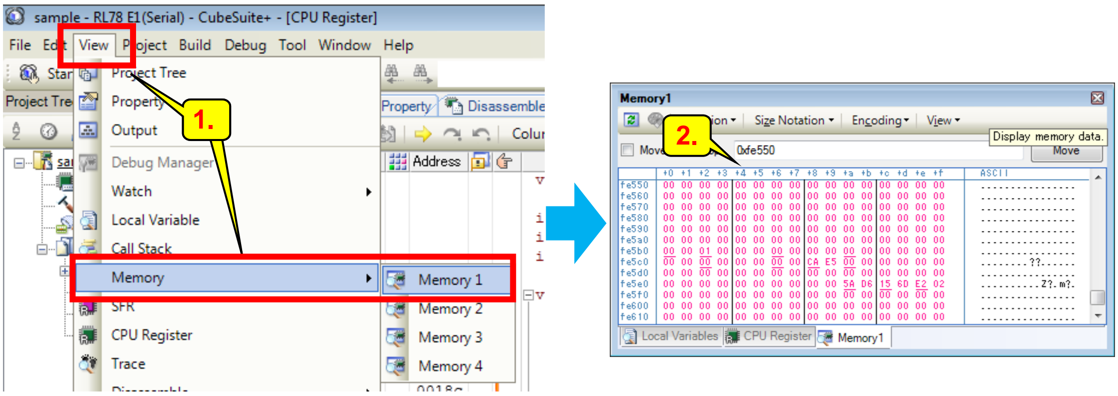
The following functions cannot be used after hot plug-in connection. To enable them, execute a forced break after hot plug-in connection.

- Trace function
- Input signal masking function
- Event break function/sequential break function
- Software break function
- Settings regarding whether to operate or stop emulation of peripheral timer and serial modules while execution is stopped

### 2.2.5 Debugging

#### (1) Memory Display Update Function

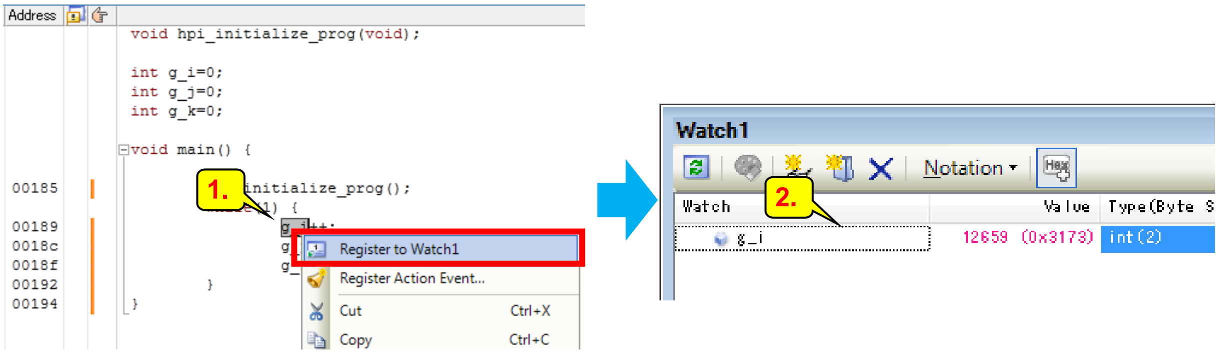
- Open the memory panel.



| | |
|---|---|
| 1. Open the memory panel | Open the [View] menu and select [Memory] → [Memory 1]. |
| 2. Check the memory panel where the RAM monitor function works | The values on the memory panel are displayed in pink and the display is updated periodically. |

#### (2) Displaying Global Variables

- Register a variable from the source.



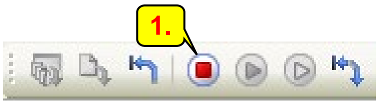
```
        void hpi_initialize_prog(void);

        int g_i=0;
        int g_j=0;
        int g_k=0;

        void main() {
00185                   initialize_prog();
                      (1) {
00189                         g_i++;
0018c                         g
0018f                         g_
00192               }
00194   }
```

g_i 12659 (0x3173) int(2)

| | |
|---|---|
| 1. Register a global variable in the watch panel | Drag and select "g_i" in the main.c source and right-click it. Then, select [Register to Watch1". |
| 2. Check the variable registered in the watch panel | The value of the variable is updated periodically. |

### (3) Forced Break

- Break execution of the program.

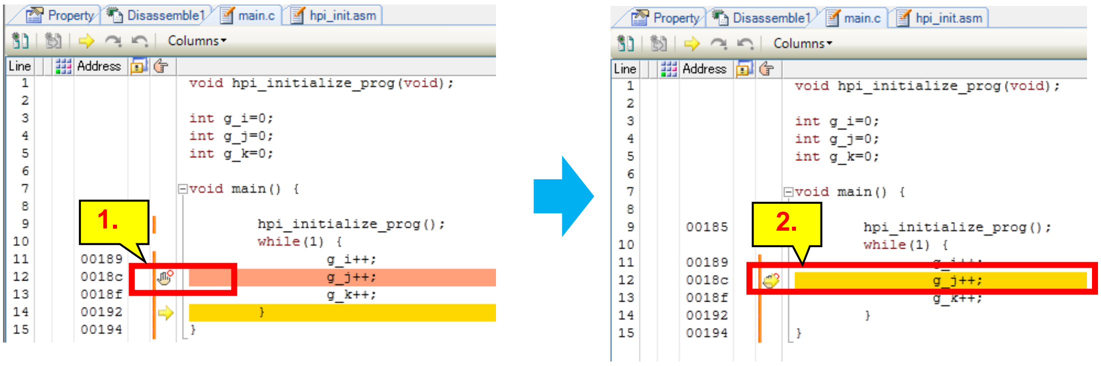


| | |
|---|---|
| 1. Forced break of the program | Click the "Stop" button on the menu bar. |

### (4) Debugging Functions after Break

- Software break

During program execution, a break occurs where a breakpoint is specified.



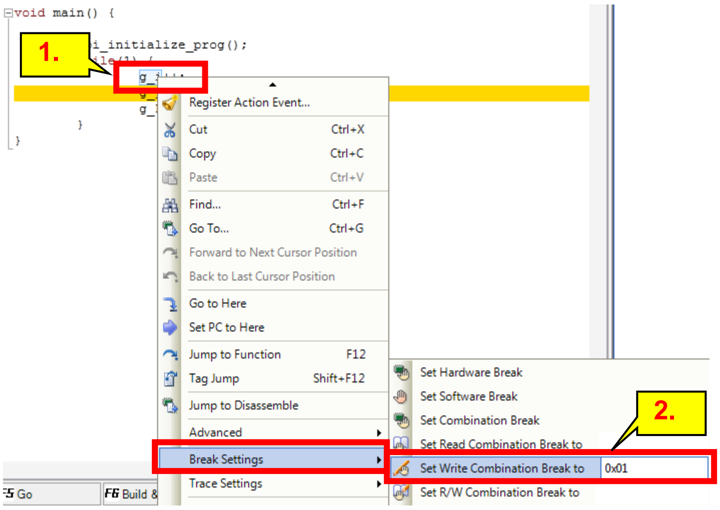
```
                   void hpi_initialize_prog(void);

                   int g_i=0;
                   int g_j=0;
                   int g_k=0;

                   void main() {

                           hpi_initialize_prog();
                           while(1) {
00189                              g_i++;
0018c                              g_j++;
0018f                              g_k++;
00192                      }
00194              }
```

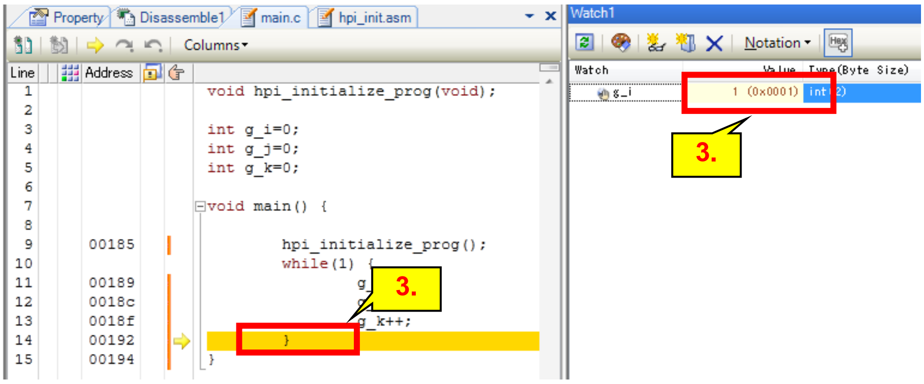
```
                   void hpi_initialize_prog(void);

                   int g_i=0;
                   int g_j=0;
                   int g_k=0;

                   void main() {

00185                      hpi_initialize_prog();
                           while(1) {
00189                              g_i++;
0018c                              g_j++;
0018f                              g_k++;
00192                      }
00194              }
```

| | |
|---|---|
| 1. Specify a software break | Click the break setting column for the "g_j++;" line on the main.c source display. |
| 2. Check occurrence of the software break | Execute the program and check that the specified software break occurs. |

- Access break

This function generates a break when a specified value is written to, read from, or written to/read from a variable.





```
void hpi_initialize_prog(void);

int g_i=0;
int g_j=0;
int g_k=0;

void main() {

00185        hpi_initialize_prog();
             while(1) {
00189                g
0018c                g
0018f                g_k++;
00192        }
00194 }
```

| | |
|---|---|
| 1. Open the context menu | Drag and select "g_i" on the main.c source and right-click it. |
| 2. Specify an access break | Select [Break Settings] → [Set Write Combination Break to] and specify the value that generates a break. A desired value can be specified. In this example, specify "0x01". |
| 3. Check occurrence of the access break | When an access break occurs, execution of the program will stop and the accessed address and value will be displayed. |

### 2.2.6 Disconnecting from User System

#### (1) Disconnecting the Debug Tool

Disconnect the debug tool.

Notes:
1. Before disconnection, stop program execution.
2. When the debug tool is disconnected, a reset is applied to the microcontroller. After disconnection, the reset state continues as long as the emulator is connected.
3. To use the hot plug-in function again after disconnection, disconnect the E1 from the user system, reset the microcontroller, and then execute hot plug-in connection.

| | |
|---|---|
| 1. Disconnect the debug tool | Break execution of the program and click the [Disconnect] button on the menu bar. |

#### (2) Turning off the User System

- Power off the user system.

#### (3) Disconnecting the E1 Emulator

- Disconnect the USB cable from the E1 emulator and the host PC.
- Then, disconnect the E1 emulator from the user system.

#### (4) Terminating the CS+

- Terminate CS+.

# 3. Notes

## 3.1 Note on Debug DTC Operating Clock

Supply of the clock for operating the debug DTC is specified through the DTCEN bit in the same way as the clock for the normal DTC. Therefore, do not clear the DTCEN bit to 0 before using the hot plug-in function or RRM/DMM using the DTC.

## 3.2 Note on DTC Suspending Instructions

Since the hot plug-in function and RRM/DMM using the DTC are implemented by using the DTC, when multiple DTC suspending instructions continue, start of the DTC is suspended. If the start of the DTC continues to be suspended, hot plug-in connection or RRM/DMM will fail.

[DTC suspending instructions]

a) Unconditional branch instructions
b) Call/return instructions
c) Conditional branch instructions
d) Instructions for read access to the code flash area
e) Bit manipulation instructions for IFxx, MKxx, PRxx, and PSW and 8-bit manipulation instructions including the ES register as an operand

Especially when an infinite loop processing is specified in the C language, note that the code is expanded to the assembly language as follows.

```
void main (void){
       while(1){}
}
```

```
_main:
       br main
```

In this case, modify the code so that DTC suspending instructions do not continue; for example, insert a NOP instruction in the infinite loop as follows.

```
void main (void){
       while(1){
              NOP()
       ;
       }
}
```

```
_main:
       nop
       br main
```

## 3.3 Note on Data Access Event

When RRM/DMM using the DTC is used, if a data access event is specified for a target variable of RRM/DMM or for an SFR, the following events will occur at RRM/DMM access.

- Event break (including sequential break)
- Trace start/stop event

When a data access event is specified for a variable or an SFR, do not use RRM/DMM using the DTC for that variable or SFR.

## 3.4 Note on Access to a 32-Bit or Longer Variable

The maximum data size in the DTC used in RRM/DMM using the DTC is 16 bits.

Therefore, when reading from or writing to a 32-bit or longer variable conflicts with writing to the variable by the CPU, the read or written value may be wrong.

For example, when a 32-bit variable is read, if the CPU writes to the variable between the upper 16-bit reading and lower 16-bit reading, the data before and after writing are read as the upper 16 bits and the lower 16 bits, respectively.

## 3.5 Notes on Standby Mode

Pay attention to the following notes on standby mode.

- Before hot plug-in connection
  When the MCU enters the standby mode before connection and either the low-speed on-chip oscillator or DTC operating clock stops, hot plug-in connection is disabled.
  When hot plug-in connection is disabled due to standby mode, use the retry settings.
- After hot plug-in connection
  When the MCU enters the STOP mode after hot plug-in connection and the source of the DTC operating clock is not the high-speed on-chip oscillator, RRM/DMM using the DTC is suspended while the MCU is in the STOP mode. (RRM/DMM using the DTC is resumed when the STOP mode is canceled.)
  Note that the debugger uses the time specified for [Retrying interval] in the retry settings as the maximum period for suspension of RRM/DMM using the DTC. (If the STOP mode continues for the retrying interval, an error will occur.)

## 3.6 Notes on Reset

Pay attention to the following notes on reset.

- Reset by the POR circuit while program execution is stopped
  After hot plug-in connection, if a reset is applied by the POR circuit while program execution is stopped, the user program runs for about 900 ms after the reset occurs and then stops.
  * This is because the user program can be stopped only after execution of the hot plug-in initialization program is completed.
- Reset by the POR circuit or pin reset during program execution
  After hot plug-in connection, if a reset by the POR circuit or a pin reset occurs during program execution, the following settings become disabled.
  - Hardware break settings (events are disabled)
  - Trace function (events are disabled and tracing is stopped)
  - Masking of input signals (all mask settings are disabled)

  When software break is used, if a reset by the POR circuit or a pin reset occurs and the instruction where a software breakpoint is set is executed before hot plug-in connection is done again, a reset due to execution of the 0xFF code will occur.
  Note that even after hot plug-in connection, the pin reset masking function can be used after a forced break is done. Therefore, if the above-described behavior is a problem, use the pin reset masking function instead.
- Internal reset during program execution (except for the reset by the POR circuit and pin reset)
  After hot plug-in connection, if an internal reset occurs during program execution, RRM/DMM using the DTC will stop. After the time specified as the retrying interval has passed, RRM/DMM using the DTC will resume.

## 3.7 Note on RAM Usage

In some device types, an appropriate RAM area should be allocated to use the hot plug-in connection, RRM/DMM using the DTC, or trace function. For details, refer to the hardware manual(On-chip Trace) for each MCU.

## Website and Support

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

Inquiries
https://www.renesas.com/us/en/contact-us

## Revision History

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 2.00 | Aug. 20, 2013 | — | English version created from Japanese revision 2.00. |
| 3.00 | Aug. 26, 2016 | — | Changed CubeSuite+ to CS+ |
| | | 1 | Addition of RL78/F15. |
| | | 10 | Addition of the hot plug-in initialization function for “CS+ for CC”. |
| 4.00 | Dec. 20, 2023 | 11 | Addition of the hot plug-in initialization function for “RL78/F2x”. |

# 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

## 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.