

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

R20AN0248EJ0300  
Rev.3.00  
Aug. 26, 2016

#### Introduction

This document describes the procedures of debugging using the hot plug-in function provided in some RL78 family microcontrollers such as RL78/F13, RL78/F14, and RL78/F15.

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.

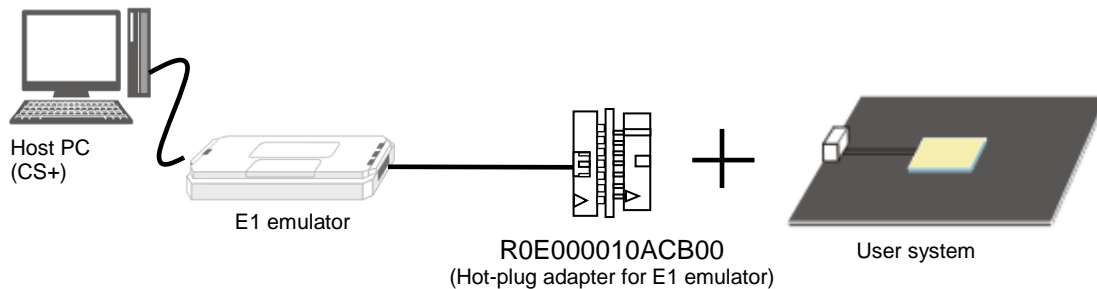


Figure 1 Overview of System

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 ..... 13
    - 2.2.3 Writing and Executing Program ..... 17
    - 2.2.4 Hot Plug-in Connection ..... 18
    - 2.2.5 Debugging ..... 22
    - 2.2.6 Disconnecting from User System..... 25
  
- 3. Notes ..... 26
  - 3.1 Note on Debug DTC Operating Clock..... 26
  - 3.2 Note on DTC Suspending Instructions..... 26
  - 3.3 Note on Data Access Event ..... 26
  - 3.4 Note on Access to a 32-Bit or Longer Variable..... 26
  - 3.5 Notes on Standby Mode..... 27
  - 3.6 Notes on Reset..... 27
  - 3.7 Note on RAM Usage ..... 27

### 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.

- Ten-byte 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

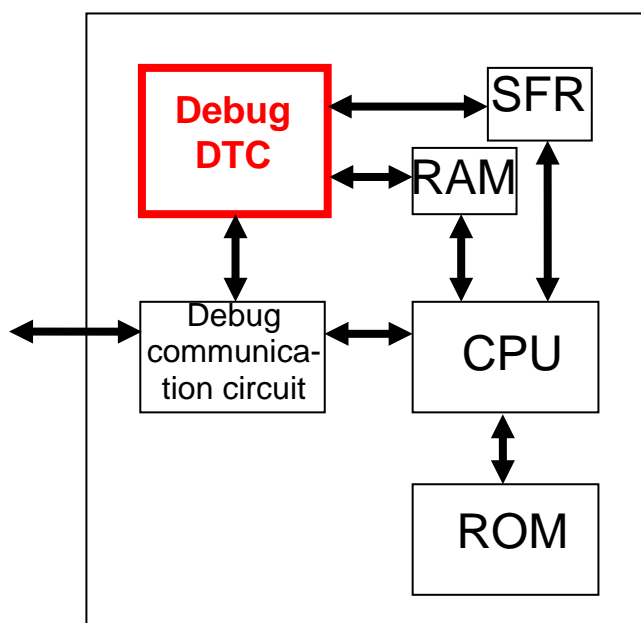


Figure 2 Concept of Debug DTC

[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

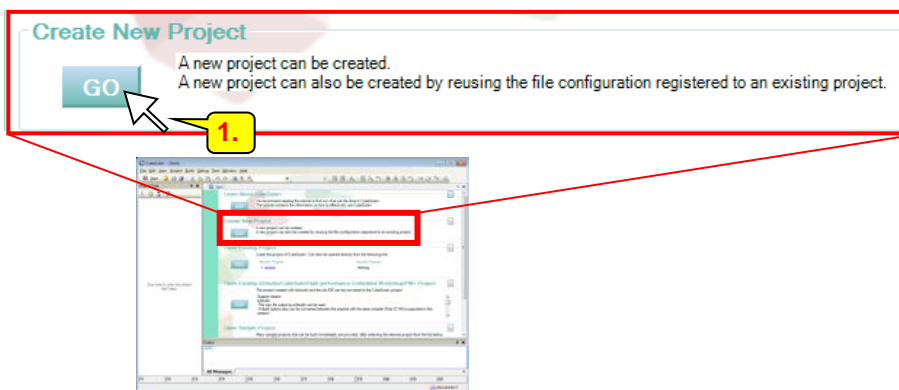
### 2.2.1 Creating a Program that Allows Hot Plug-in Connection

(1) Starting CS+

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

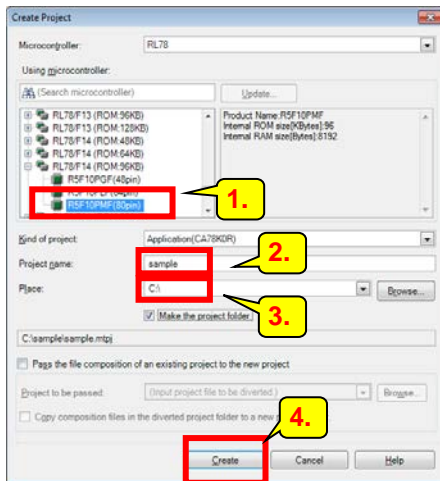
(2) Creating a Project

- Create a new project from the start panel.



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

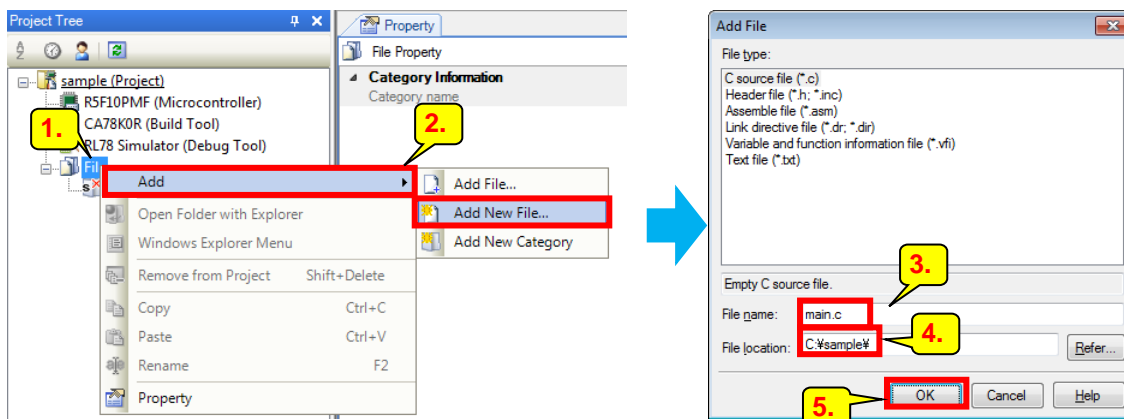
- Make detailed settings for the project.



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.

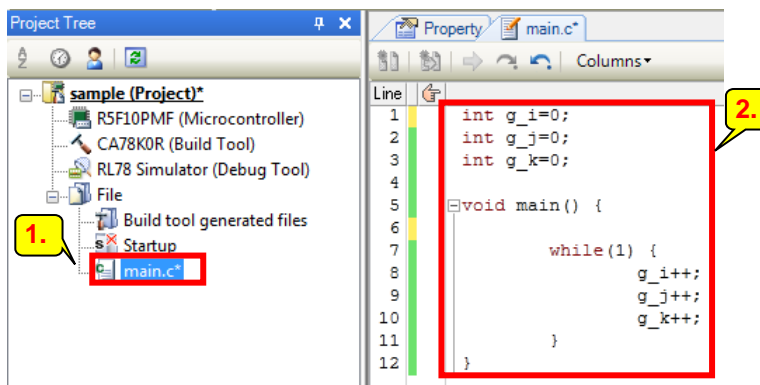
(3) Creating a User Program

- Create the main.c file.



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.

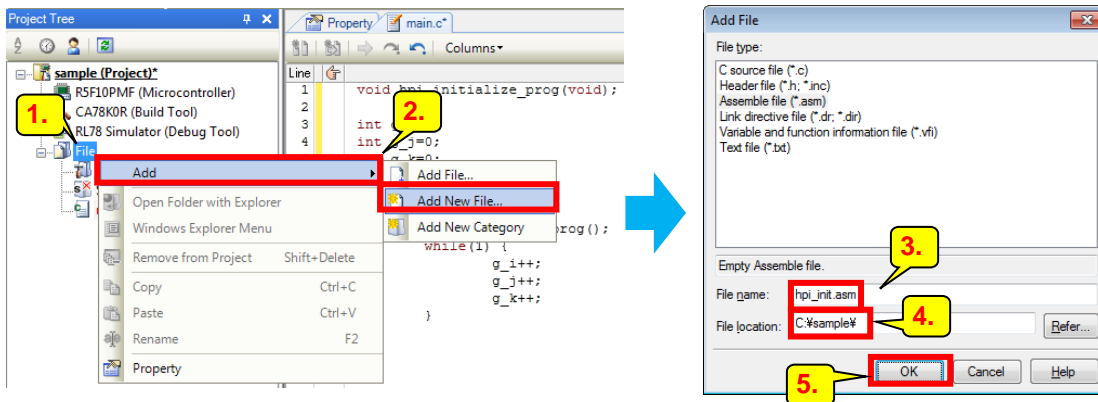


1. Open "main.c"	Double-click "main.c" in the project tree.
2. Edit the main function	Copy and paste the following to the main.c file.
<pre>int g_i=0; int g_j=0; int g_k=0;  void main() {     while(1) {         g_i++;         g_j++;         g_k++;     } }</pre>	

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

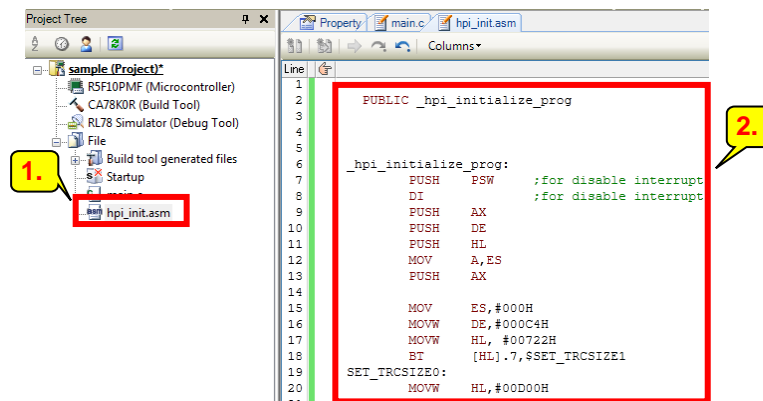
### (4) Creating the Hot Plug-in Initialization Function

- Create a file for defining the hot plug-in initialization function (hpi\_initialize\_prog).  
Add the hpi\_init.asm file to the project.



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).



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

```

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],#0023H
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

```

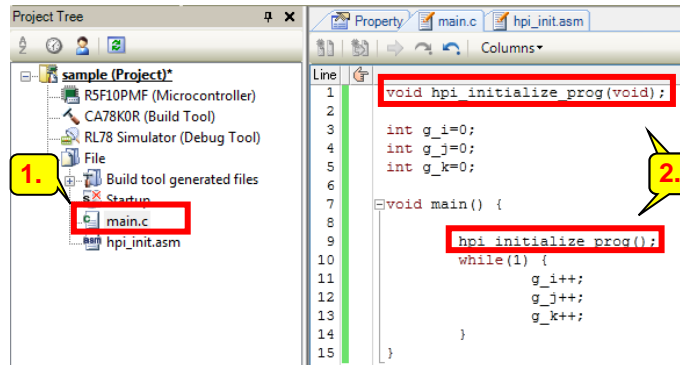
.PUBLIC _hpi_initialize_prog

_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],#0xFF
MOV [HL+0x10],#0x11
MOV [HL+0x11],#0x001
MOV [HL+0x12],#0x10
MOV [HL+0x13],#0x10
MOV [HL+0x14],#0x0AD
MOV [HL+0x15],#0xFF
MOV [HL+0x16],#0x000
MOV A,H
MOV [HL+0x17],A
MOV [HL+0x18],#0x003
MOV [HL+0x19],#0x001
MOV [HL+0x1A],#0x10
MOV [HL+0x1B],#0x10
MOV [HL+0x1C],#0x000
MOV A,H
MOV [HL+0x1D],A
MOV [HL+0x1E],#0x0AD
MOV [HL+0x1F],#0xFF
SET1 !DTCEN
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.



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.
<pre> 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++;     } } </pre>	

Add this line.

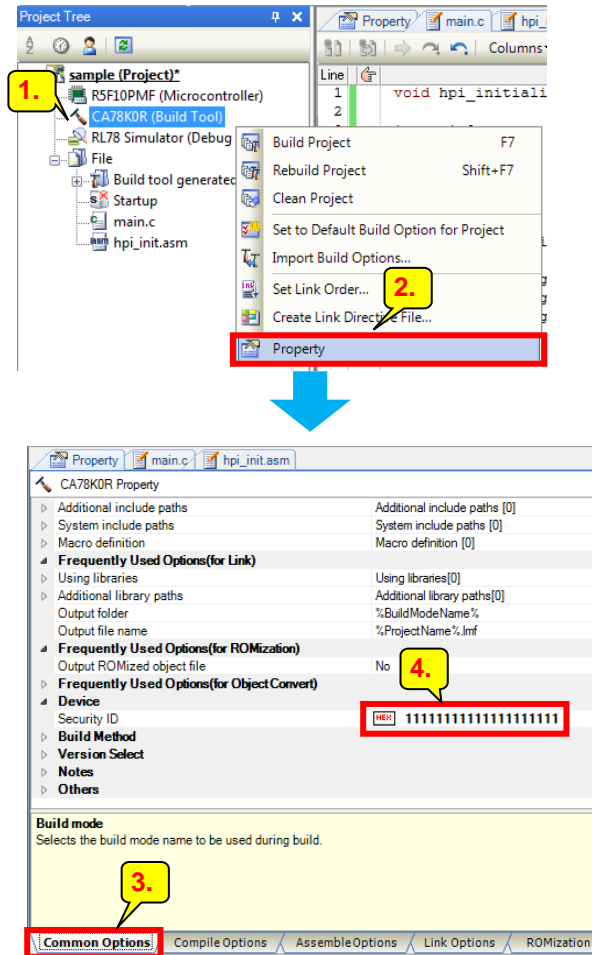
Add this line.

## 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.

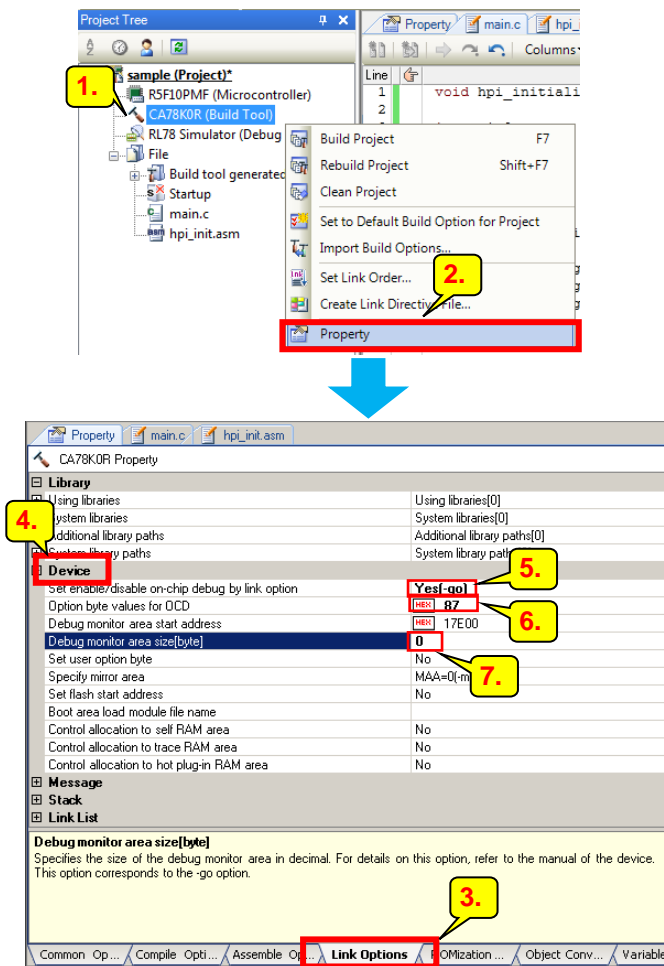


1. Open the context menu for "CA78K0R (Build Tool)"	Select "CA78K0R (Build Tool)" in the project tree and then right-click it.
2. Open the property panel	Select [Property] from the context menu.
3. Open the [Common Options] tab	Select the [Common Options] tab in the main panel.
4. Specify [Security ID]	Specify [Security ID] under "Device". A desired ID can be specified. In this example, "11111111111111111111" 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.



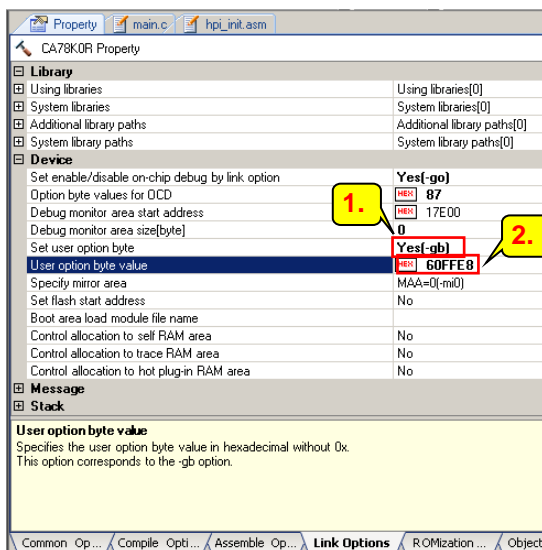
1. Open the context menu for "CA78K0R (Build Tool)"	Select "CA78K0R (Build Tool)" in the project tree and then right-click it.
2. Open the property panel	Select "Property" from the context menu.
3. Open the [Link Options] tab	Select the [Link Options] tab in the main panel.
4. Expand "Device"	Expand "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". 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.

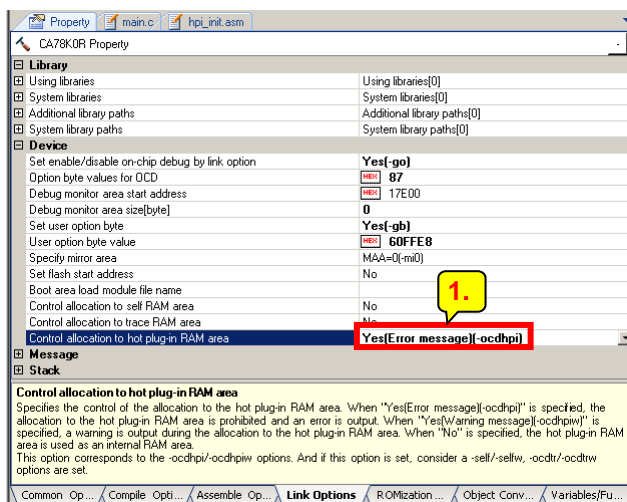


1. [Set user option byte]	Select "Yes".
2. [User option byte value]	A desired value can be specified. In this example, "60FFE8" is specified.

### (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 CubeSuite+ display. Skip this setting and move to the next step.



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.

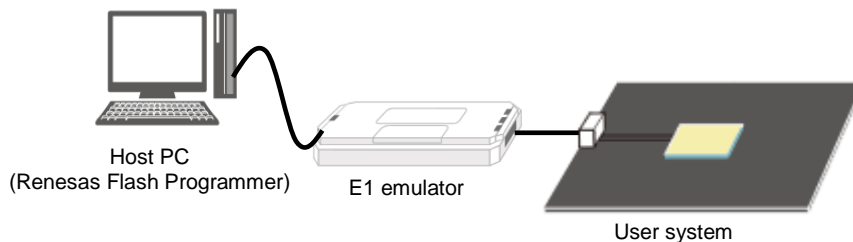


### 2.2.3 Writing and Executing Program

#### (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.

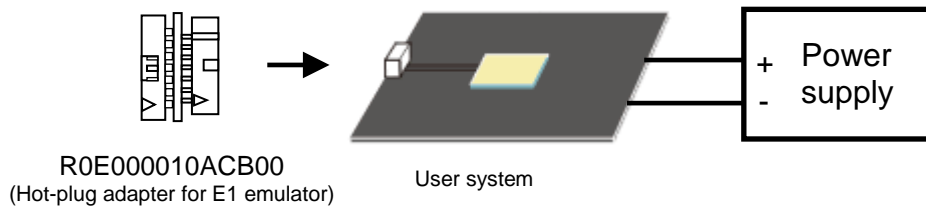


## 2.2.4 Hot Plug-in Connection

### (1) Connecting the Hot-Plug Adapter (R0E000010ACB0) to the User System

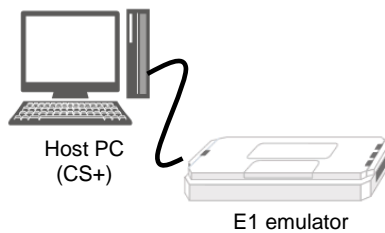
- Be sure to connect the R0E000010ACB0 to the user system first.

The R0E000010ACB0 has a mechanism that ensures that connection to the E1 emulator begins with the GND line.



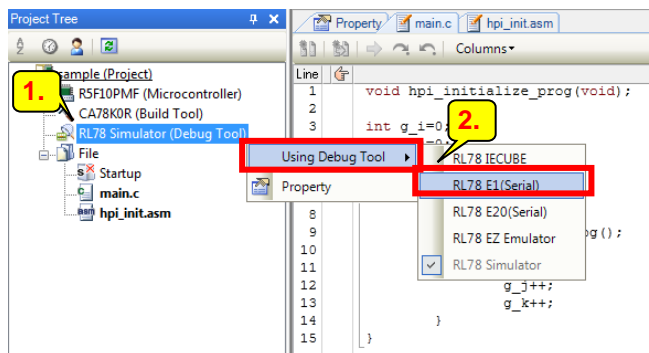
### (2) Connecting the E1 Emulator to the Host PC

- Connect the E1 to the host PC.



### (3) Making Debugging Function Settings

- Select a debug tool in CS+.



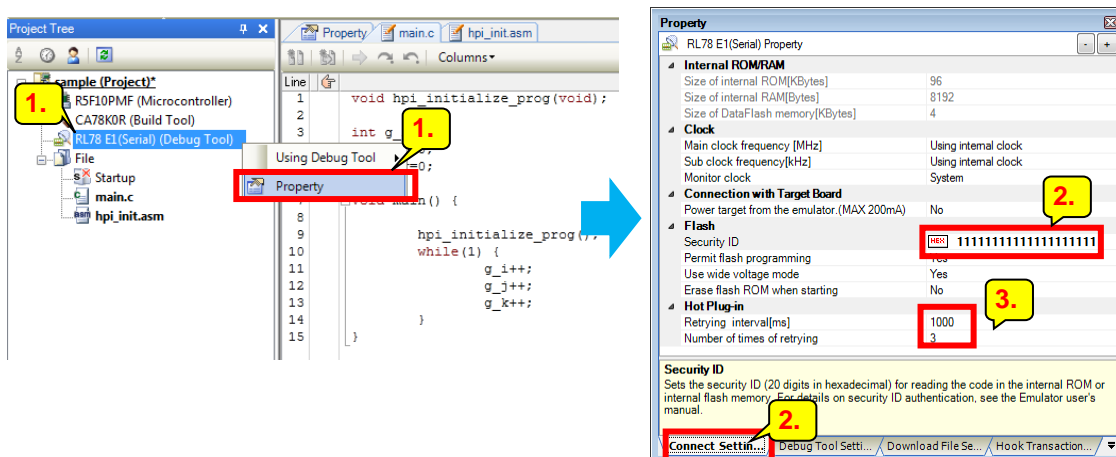
1. Open the context menu for "Debug Tool"

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.



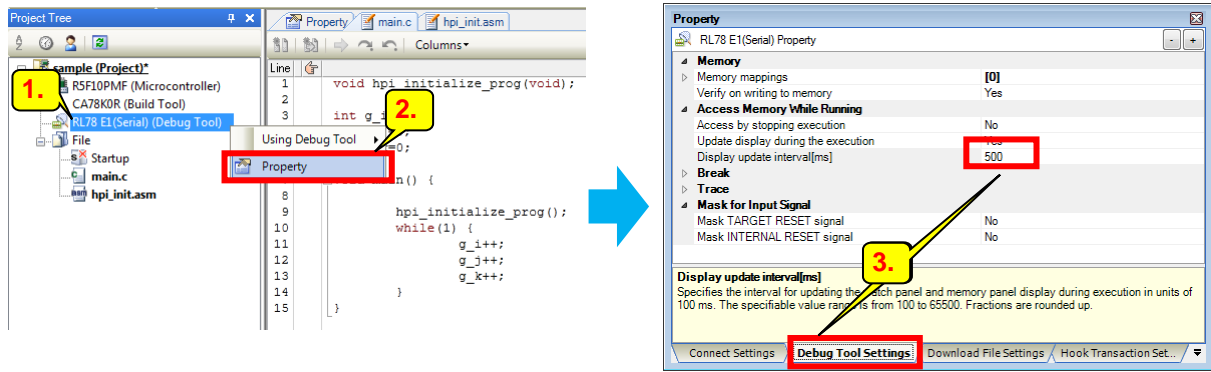
<p>1. Open the property panel for "Debug Tool"</p>	<p>Select "Debug Tool" in the project tree and then right-click it.</p>
<p>2. Enter the security ID for authentication.</p>	<p>Specify [Security ID] under "Flash" in the [Connect Settings] tab. Enter the same value as that specified as a common option for the build tool. In this example, enter "11111111111111111111".</p>
<p>3. Make retry settings for hot plug-in</p>	<p>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. 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).</p>

[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.

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

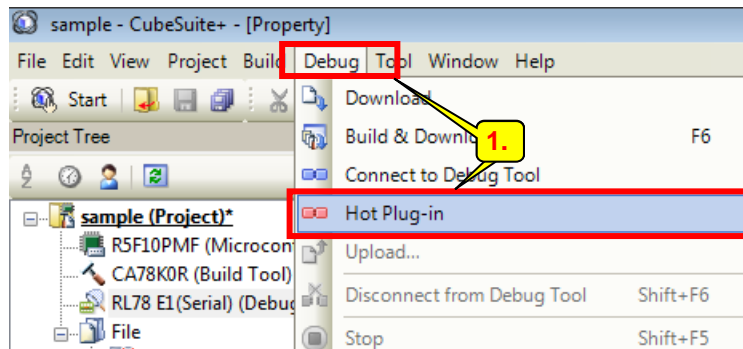
- Specify the RAM monitoring interval.



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. A value of 100 ms or greater can be specified. In this example, "500" is specified.

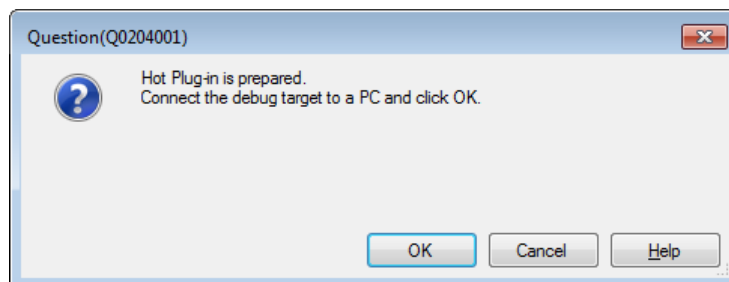
### (4) Executing Hot Plug-in Connection

- Prepare for hot plug-in connection.



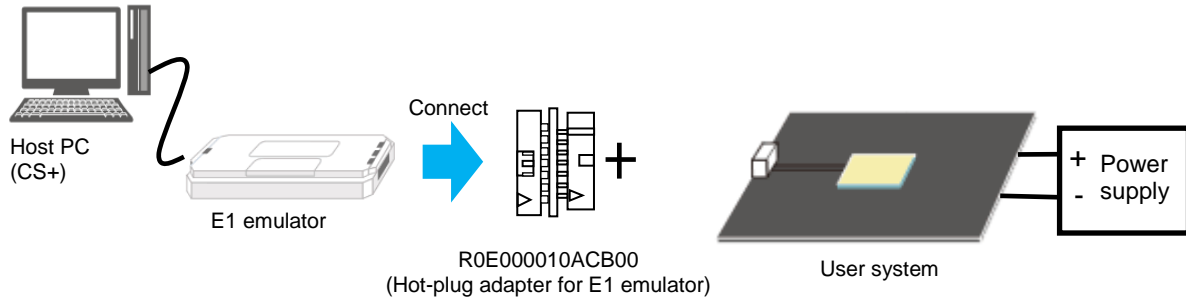
1. Shift the debugger to the hot plug-in preparation state	Select [Hot Plug-in] from the [Debug] menu.
--	---

- Check that the following dialog appears.



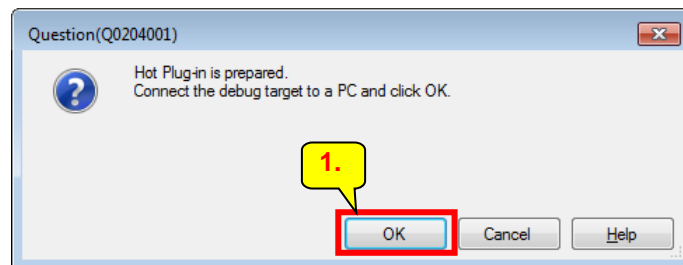
- Connecting the E1 emulator to the hot-plug adapter (R0E000010ACB00)

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.



1. Hot plug-in connection

Click the [OK] button.

### [Unavailable Functions after Hot Plug-in Connection]

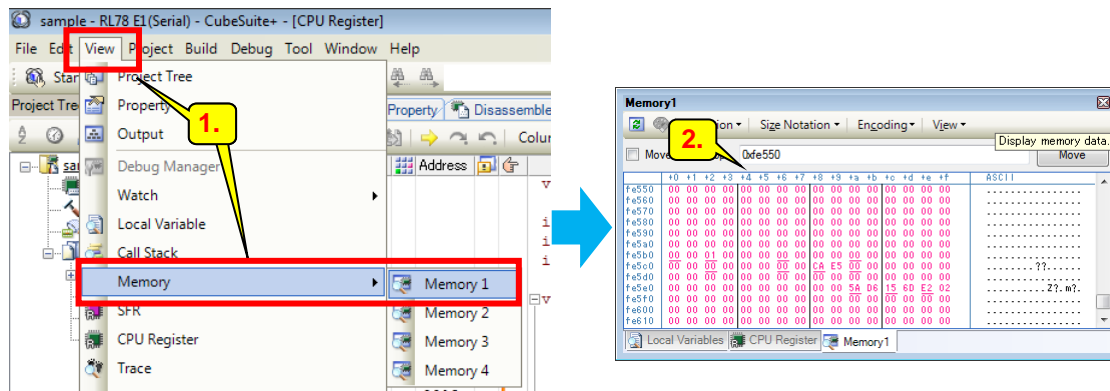
The following functions cannot be used after hot plug-in connection. To enable them, execute a forced break after hot plug-in connection.

- 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

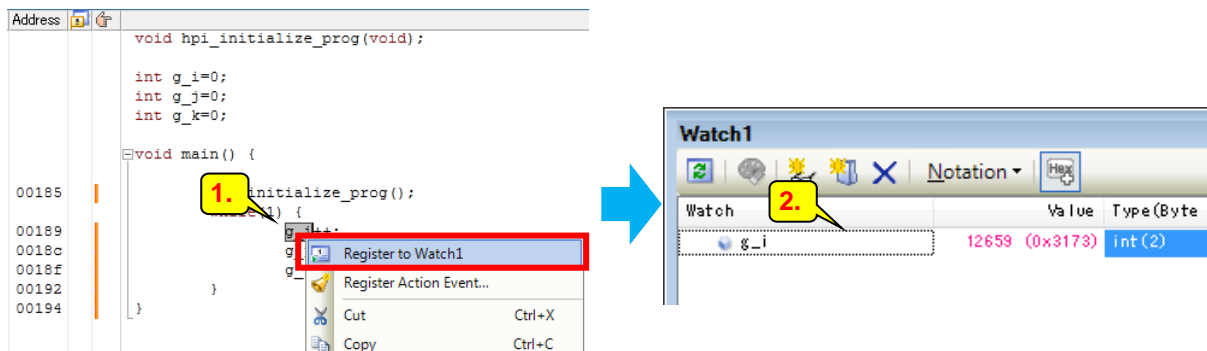
- Open the memory panel.



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.



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.

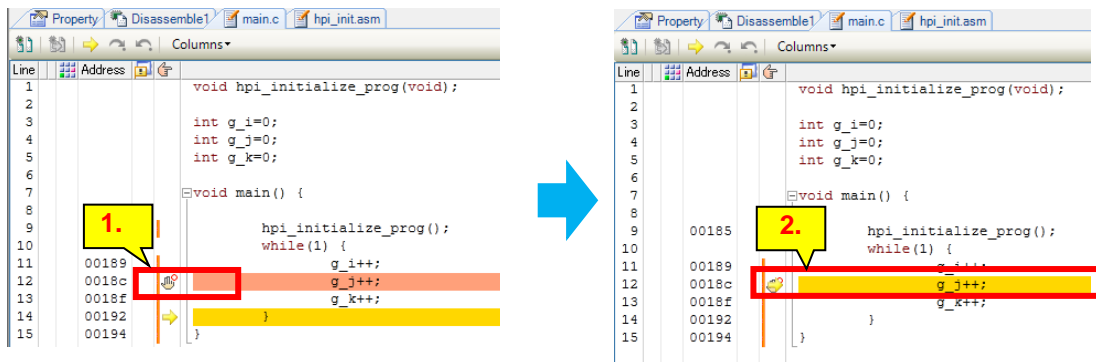


1. Forced break of the program	Click the "Stop" button on the menu bar.
--------------------------------	--

## (4) Debugging Functions after Break

- Software break

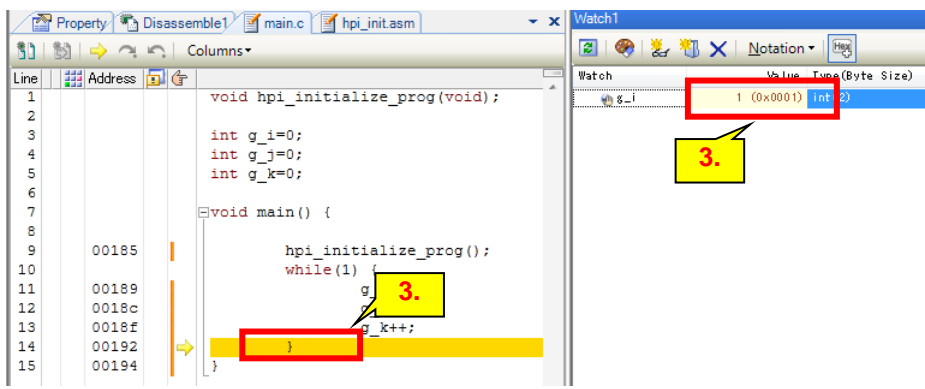
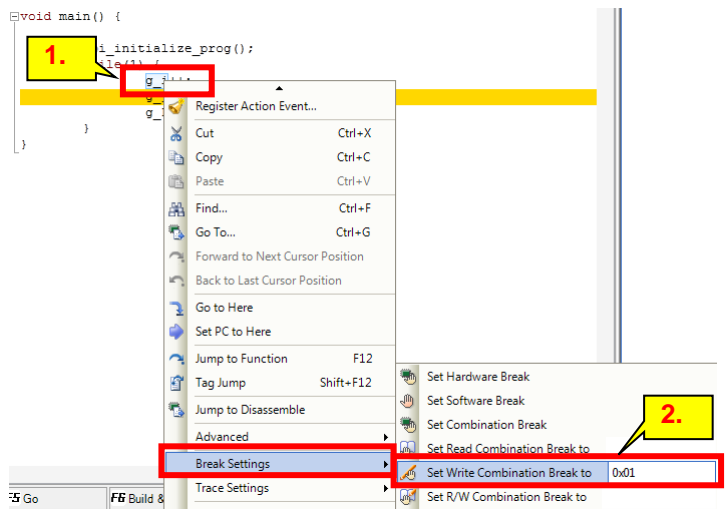
During program execution, a break occurs where a breakpoint is specified.



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.



<p>1. Open the context menu</p>	<p>Drag and select "g_i" on the main.c source and right-click it.</p>
<p>2. Specify an access break</p>	<p>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".</p>
<p>3. Check occurrence of the access break</p>	<p>When an access break occurs, execution of the program will stop and the accessed address and value will be displayed.</p>



### 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.

<pre>void main (void){     while(1){} }</pre>		<pre>_main:     br main</pre>
---	--	-------------------------------

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.

<pre>void main (void){     while(1){         NOP()     } }</pre>		<pre>_main:     nop     br main</pre>
--	---	---------------------------------------

#### 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/contact>

All trademarks and registered trademarks belong to their respective owners.

## 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”.

## 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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

### 2. Processing at Power-on

The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

### 3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

- The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

### 4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

### 5. Differences between Products

Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

- The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
  2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
  3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
  5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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; and industrial robots etc.  
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.  
Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
  6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
  7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or systems manufactured by you.
  8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
  9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
  10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
  11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
  12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.  
(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.



### SALES OFFICES

Renesas Electronics Corporation

<http://www.renesas.com>

Refer to "<http://www.renesas.com/>" for the latest and detailed information.

#### **Renesas Electronics America Inc.**

2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.  
Tel: +1-408-588-6000, Fax: +1-408-588-6130

#### **Renesas Electronics Canada Limited**

9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3  
Tel: +1-905-237-2004

#### **Renesas Electronics Europe Limited**

Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K.  
Tel: +44-1628-585-100, Fax: +44-1628-585-900

#### **Renesas Electronics Europe GmbH**

Arcadiastrasse 10, 40472 Düsseldorf, Germany  
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

#### **Renesas Electronics (China) Co., Ltd.**

Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China  
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

#### **Renesas Electronics (Shanghai) Co., Ltd.**

Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333  
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

#### **Renesas Electronics Hong Kong Limited**

Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong  
Tel: +852-2265-6688, Fax: +852 2886-9022

#### **Renesas Electronics Taiwan Co., Ltd.**

13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan  
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

#### **Renesas Electronics Singapore Pte. Ltd.**

80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949  
Tel: +65-6213-0200, Fax: +65-6213-0300

#### **Renesas Electronics Malaysia Sdn.Bhd.**

Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia  
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

#### **Renesas Electronics India Pvt. Ltd.**

No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India  
Tel: +91-80-67208700, Fax: +91-80-67208777

#### **Renesas Electronics Korea Co., Ltd.**

12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea  
Tel: +82-2-558-3737, Fax: +82-2-558-5141