User's Manual

E2 Emulator

Additional Document for User's Manual

(Setting up Interlocked Debugging of the DFP in an RH850/U2B-Series Device)

Supported Devices:

RH850 Family

RH850/U2B Series

DFP-Equipped Devices

Renesas Electronics
www.renesas.com

Rev.1.00 Jul 2022

Table of Contents

1. Outline

1.1 This manual

This manual explains how to set up the environment for interlocked debugging of an RH850/U2B-series device based on G4MH cores that also incorporates a DFP IP module (hereinafter referred to as the DFP) with both G4MH cores and the DFP as targets for debugging and to proceed with debugging by using an E2 emulator.

1.2 Configuration of manuals

This section describes documents that include information required for debugging an RH850/U2B device by using the E2 emulator.

The manuals of the E2 emulator consist of the E2 Emulator User's Manual and E2 Emulator Additional Document for User's Manual. Be sure to read both user's manuals before using the E2 emulator.

Name of Document	Document No.
E2 Emulator RTE0T00020KCE00000R User's Manual	R20UT3538E
E2 Emulator Additional Document for User's Manual (Notes on Connection of RH850/U2B Series)	R20UT5052E

In the interlocked debugging of G4MH cores and the DFP in a DFP-equipped RH850/U2B-series device, the CS+ integrated development environment (CS+ debugger) is required as the debugger for the G4MH cores. For use of the CS+ debugger, be sure to read its online help.

Name of Document	URL
CS+ online help	CS+ Online Help (renesas.com)

The interlocked debugging of G4MH cores and the DFP in a DFP-equipped RH850/U2B-series device requires the DR1000C debugger and IDE from NSITEXE for debugging of the DFP. To use the DR1000C debugger and IDE, be sure to read the following documents.

Name of Document
NSITEXE DR1000C Debugger User Manual
NSITEXE DR1000C IDE User Manual

2. System Configuration

This chapter describes the system configuration for interlocked debugging of the G4MH cores and DFP in a DFP-equipped RH850/U2B-series device.

2.1 Hardware configuration

Figure 2-1 and Table 2-1 show the hardware configuration for interlocked debugging of the G4MH cores and DFP in a DFP-equipped RH850/U2B-series device.

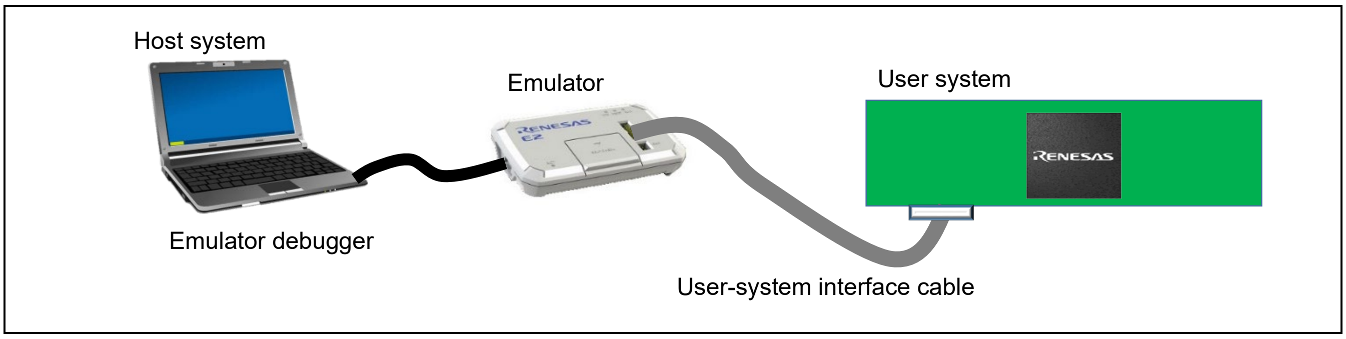


Figure 2-1 Hardware Configuration

Table 2-1 Hardware Configuration

Name	Product Name	Remarks
Emulator debugger	CS+ debugger from Renesas	—
	Debugger and IDE from NSITEXE	Included in the DR1000C SDK from NSITEXE.
Emulator	E2 emulator	—
User-system interface cable	User-system interface cable	Provided with the E2 emulator main unit.
Debugging interface	JTAG interface	LPD interface is not available.
User system	RH850/U2B Piggyback Board, etc.	Board having a DFP-equipped RH850/U2B-series device

2.2 Software configuration

Table 2-2 lists the software configuration for interlocked debugging of the G4MH cores and DFP in a DFP-equipped RH850/U2B-series device.

Table 2-2 Software Configuration

Name	Product Name	Remarks
Emulator debugger for the G4MH cores in the RH850 device	CS+ debugger from Renesas	—
Emulator debugger for the DFP in the RH850 device	Debugger and IDE from NSITEXE	Use the IDE and open OCD included in the DR1000C SDK from NSITEXE.
Virtual OS environment	Oracle VM VirtualBox	Use the Ubuntu 20.04 LTS environment to use the DR1000C debugger and IDE under Linux.

3. Setting up the Environment for Interlocked Debugging of the G4MH Cores and DFP

This chapter describes how to set up the environment for interlocked debugging of the G4MH cores and DFP.

3.1 Setting up RH850/U2B-series devices

For details on RH850/U2B-series devices, refer to the RH850/U2B Group User's Manual.

This section describes setting up of an RH850/U2B-series device and points for caution on interlocked debugging of the G4MH cores and DFP.

3.1.1 DFP-related option bytes

The following option byte must be set to debug the DFP installed in an RH850/U2B-series device.

- S_OPBT0: MPSELECT (Set the bit in DFP chain mode.)

The DFP states and reset vectors at the time power is supplied must be set with the following option bytes.

- OPBT38: DFP_init_boothart : 0000 0000_H (setting value)
- OPBT39: DFP_resetvec: 0030 0000_H (setting value)

3.1.2 Address map for the DFP installed in an RH850/U2B-series device

The following shows the address map for debugging the DFP installed in an RH850/U2B-series device.

Table 3-1 Address Map for the DFP

Start Address	End Address	Area
0030 0000_H	003F $FFFF_H$	External ROM
F000 0000_H	F000 $7FFF_H$	Control Core Unit (CCU) Local Memory
F004 0000_H	F004 $1FFF_H$	Scalar Processor Unit (SPU) 0 Local Memory
F004 4000_H	F004 $5FFF_H$	SPU 1 Local Memory
F004 8000_H	F004 $9FFF_H$	SPU 2 Local Memory
F004 $C000_H$	F004 $DFFF_H$	SPU 3 Local Memory
F005 0000_H	F005 $005F_H$	Mutual Exclusion Register Variable (MERV)
F010 0000_H	F013 $FFFF_H$	Vector Processor Unit (VPU) Local Memory (VLM)
F080 0000_H	F0BF $FFFF_H$	DFP Peripheral
F800 0000_H	F81F $FFFF_H$	DFP Debug
FE04 0000_H	FE07 $FFFF_H$	External RAM

3.2 Setting up the CS+ debugger

For details on the CS+ debugger, refer to the CS+ online help system. This section describes setting up of the CS+ debugger and points for caution on interlocked debugging of the G4MH cores and DFP.

3.2.1 Settings for interlocked debugging of the G4MH cores and DFP

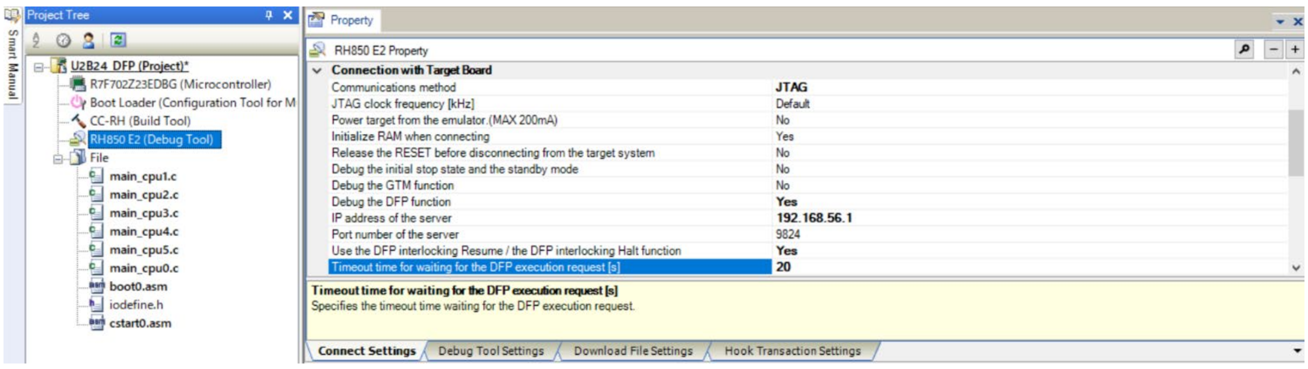
Property settings for the E2 emulator must be made in the CS+ debugger to debug the DFP installed in an RH850/U2B-series device. Additional settings are also required for interlocked debugging of the G4MH cores and DFP. Table 3-2 and Table 3-3 list the settings of the properties and Figure 3-1 and Figure 3-2 show the windows for reference.

Table 3-2 Settings of the Properties of the E2 Emulator in the CS+ Debugger for Debugging the DFP

Item	Setting
Communications method	JTAG
Debug the DFP function	Yes
IP address of the server	IP address of the GDB server specified by the open OCD Example: 192.168.56.1
Port number of the server	Port number of the GDB server specified by the open OCD Example: 9824

Table 3-3 Settings of the Properties of the E2 Emulator in the CS+ Debugger for Interlocked Debugging of the G4MH Cores and DFP

Item	Setting
Use the DFP interlocking Resume/the DFP interlocking Halt function	Yes
Timeout time for waiting for the DFP execution request [s]	Timeout time until issuing a request for executing the DFP after having issued a request for execution by the G4MH cores Example: 20
Stop emulation of peripherals when stopping	Yes

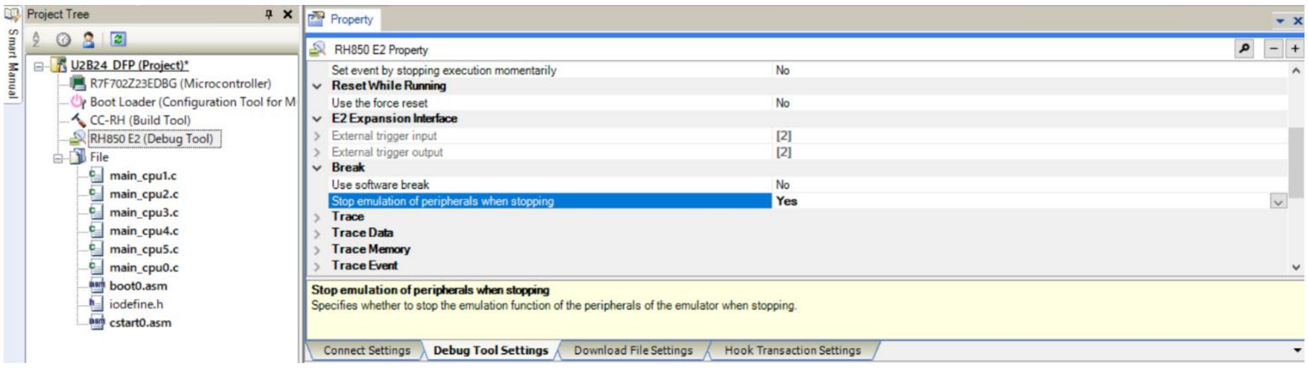


Connection with Target Board

Communications method	JTAG
JTAG clock frequency [kHz]	Default
Power target from the emulator.(MAX 200mA)	No
Initialize RAM when connecting	Yes
Release the RESET before disconnecting from the target system	No
Debug the initial stop state and the standby mode	No
Debug the GTM function	No
Debug the DFP function	Yes
IP address of the server	192.168.56.1
Port number of the server	9824
Use the DFP interlocking Resume / the DFP interlocking Halt function	Yes
Timeout time for waiting for the DFP execution request [s]	20

Timeout time for waiting for the DFP execution request [s]
Specifies the timeout time waiting for the DFP execution request.

Figure 3-1 Reference for Setting the Properties of the E2 Emulator in CS+ for Interlocked Debugging of the G4MH Cores and DFP ([Connect Settings] Tabbed Page)



Project Tree

- U2B24_DFP (Project)*
  - R7F702Z23EDBG (Microcontroller)
  - Boot Loader (Configuration Tool for M
  - CC-RH (Build Tool)
  - RH850 E2 (Debug Tool)
  - File
    - main_cpu1.c
    - main_cpu2.c
    - main_cpu3.c
    - main_cpu4.c
    - main_cpu5.c
    - main_cpu0.c
    - boot0.asm
    - iodefine.h
    - cstart0.asm

Property

RH850 E2 Property

Set event by stopping execution momentarily	No
Reset While Running	
Use the force reset	No
E2 Expansion Interface	
External trigger input	[2]
External trigger output	[2]
Break	
Use software break	No
Stop emulation of peripherals when stopping	Yes
Trace	
Trace Data	
Trace Memory	
Trace Event	

Stop emulation of peripherals when stopping
Specifies whether to stop the emulation function of the peripherals of the emulator when stopping.

Figure 3-2 Reference for Setting the Properties of the E2 Emulator in CS+ for Interlocked Debugging of the G4MH Cores and DFP ([Debug Tool Settings] Tabbed Page)

3.3 Setting up VirtualBox

Set up the DR1000C debugger and IDE for the virtual Linux running in VirtualBox to use the DR1000C debugger and IDE for debugging of the DFP under Linux.

3.3.1 Installing VirtualBox

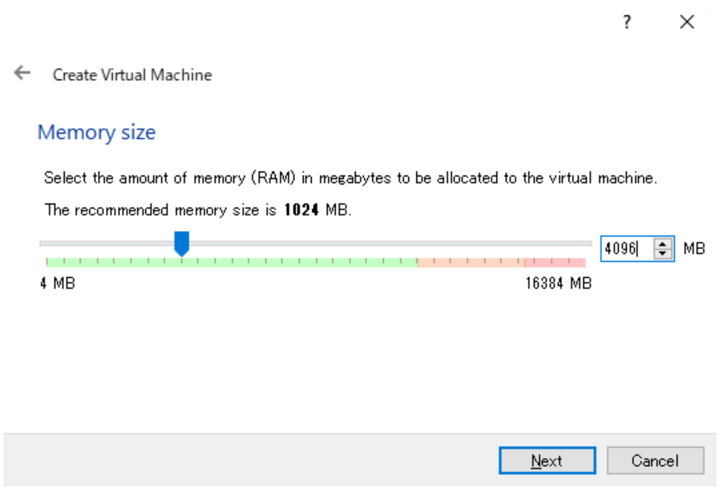
Download the installer for Oracle VM VirtualBox from the Oracle home page. Start the installer and proceed with standard installation according to the installer.

3.3.2 Starting VirtualBox and Setting the Virtual Hard Disk

Start VirtualBox. Select the [Tool] -> [Preference] -> [General] menu and specify the destination for storing the virtual hard disk (virtual HD) for the virtual environment. Then, create the virtual HD from the [Tool] -> [New] menu item. Table 3-4 lists the recommended settings and Figure 3-3 and Figure 3-4 show the windows for reference.

Table 3-4 Recommended Settings of the Virtual HD of VirtualBox for Interlocked Debugging of the G4MH Cores and DFP

Item	Setting
Memory size	4 GB (1 GB by default)
File size	30 GB (because the 10-GB default size is insufficient)
Hard disk file type	VHD
Storage on physical hard disk	Fixed size


Create Virtual Machine

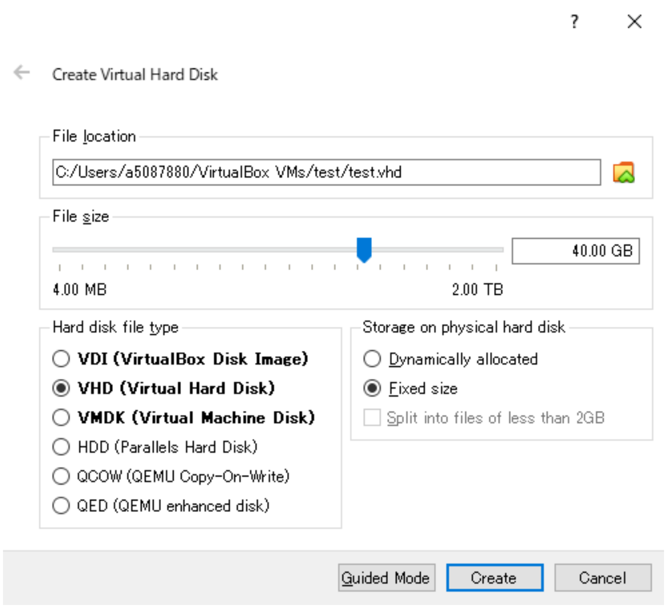
Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024** MB.

4096 MB

Figure 3-3 Reference for Recommended Settings of the Virtual HD of VirtualBox (Memory Size)


Create Virtual Hard Disk

File location: C:/Users/a5087880/VirtualBox VMs/test/test.vhd

File size: 40.00 GB (4.00 MB – 2.00 TB)

Hard disk file type:
- VDI (VirtualBox Disk Image)
- [✓ VHD (Virtual Hard Disk)]
- VMDK (Virtual Machine Disk)
- HDD (Parallels Hard Disk)
- QCOW (QEMU Copy-On-Write)
- QED (QEMU enhanced disk)

Storage on physical hard disk:
- Dynamically allocated
- [✓ Fixed size]
- Split into files of less than 2GB

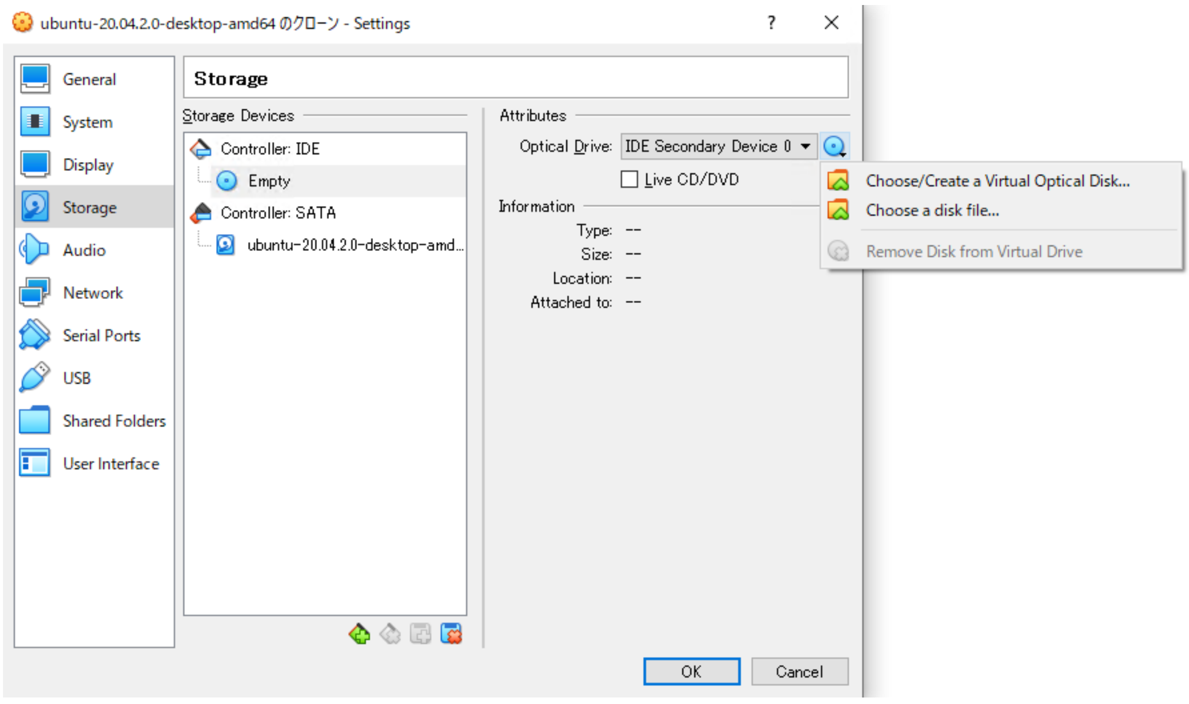
Figure 3-4 Reference for Recommended Settings of the Virtual HD of VirtualBox (Window for Expert Mode when Creating the Virtual HD)

3.3.3 Downloading the Ubuntu image file

Download the ISO image file for Ubuntu 20.04 LTS from the Ubuntu home page.

3.3.4 Preparing for creation of the virtual machine

Select the created virtual HD and [Settings] -> [Storage] and click on the [Empty] disk icon for the storage device. Click on the disk icon at the right of the [Optical Drive] drop-down list in the [Attributes] item and then [Choose a disk file…]. Mount the Ubuntu image file (iso) and click on the [OK] button. Figure 3-5 shows the window for reference.


ubuntu-20.04.2.0-desktop-amd64 のクローン - Settings

Storage

Storage Devices:
- Controller: IDE
  - Empty
- Controller: SATA
  - ubuntu-20.04.2.0-desktop-amd...

Attributes
Optical Drive: IDE Secondary Device 0
Live CD/DVD

Information
Type: --
Size: --
Location: --
Attached to: --

Choose/Create a Virtual Optical Disk...
Choose a disk file...
Remove Disk from Virtual Drive
OK
Cancel

Figure 3-5 Reference for Mounting an Image File on the VirtualBox Virtual Machine

3.3.5 Installing Ubuntu

Clicking on [Start] in VirtualBox starts the installation of Ubuntu. Select the language for the virtual environment and click on [Install Ubuntu]. Proceed with installation according to the installation window.

After installation, restart the virtual environment, enter the user name and password, and log in to the virtual environment. Then update the software and restart the virtual environment.

3.3.6 Configuring the virtual environment

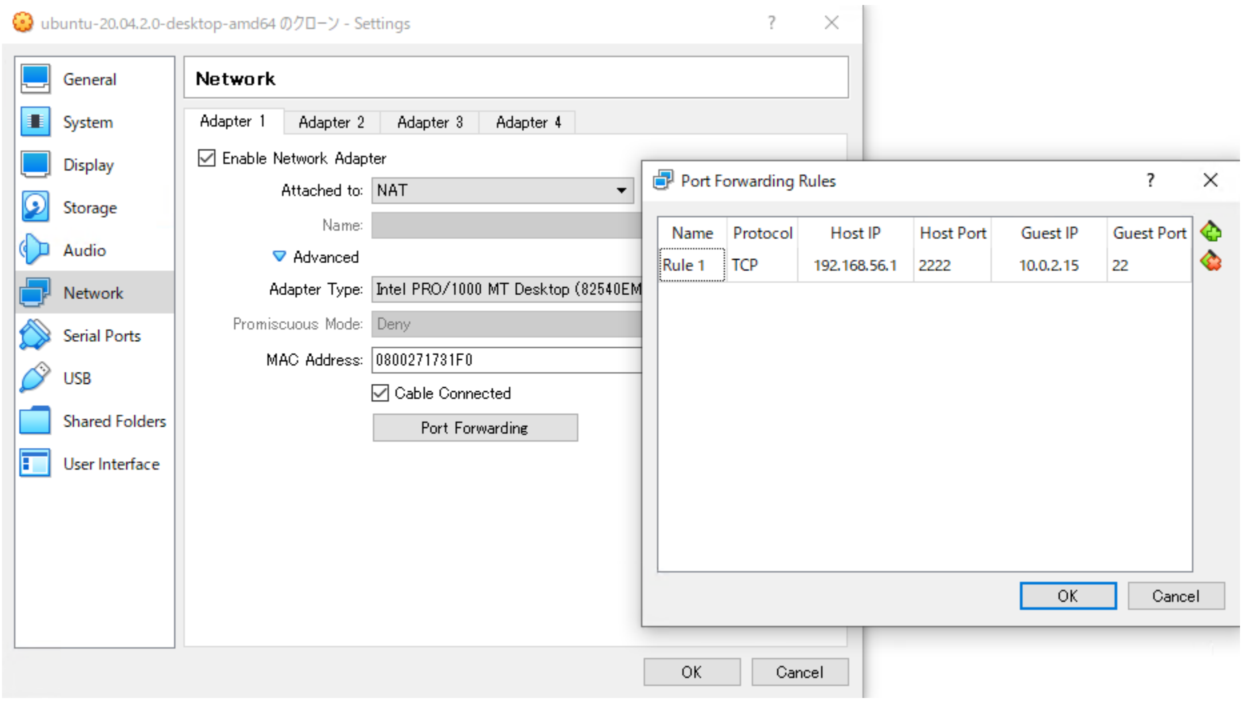
Select [Open Terminal] to open a terminal and install the build-essential software under Ubuntu; that is, make the use of GCC, MAKE, and GDB available.

3.3.7 Connecting a PC to the virtual machine

To connect a PC (Windows) to the virtual machine (Ubuntu), confirm the IP address of the PC and set an IP address for VirtualBox by clicking on [Settings] -> [Network] -> [Advanced] -> [Port Forwarding] -> [Adds new port forwarding rule.]. Table 3-5 lists the settings and Figure 3-6 shows the window for reference.

Table 3-5 Connecting the PC to the Virtual Machine

Item	Setting
Name	As desired
Protocol	TCP
Host IP	IPv4 address of the PC
Host port	Port number that does not coincide with the guest port number, a reserved port number, or the number of a port that is in use for another purpose
Guest IP	IPv4 address of the virtual machine
Guest port	Port number that does not coincide with a reserved port number or the number of a port that is in use for another purpose



Enable Network Adapter
Attached to: NAT
Name:
Advanced
Adapter Type: Intel PRO/1000 MT Desktop (82540EM
Promiscuous Mode: Deny
MAC Address: 0800271731F0
Cable Connected
Port Forwarding

Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
Rule 1	TCP	192.168.56.1	2222	10.0.2.15	22

Figure 3-6 Reference for Settings for Connecting the PC to the VirtualBox Virtual Machine

3.4 Setting up the open OCD

For details on the open OCD, refer to the DR1000C Debugger User Manual from NSITEXE. This section describes setting up of the open OCD and points for caution on interlocked debugging of the G4MH cores and DFP.

For the open OCD, use version 1.1.1 which has been pre-released as the next update.

3.4.1 Setting the IP addresses, port numbers and hart to debug

The IP addresses and port numbers set for the open OCD must match those specified in the CS+ debugger. If different IP addresses or port numbers are set, the DFP cannot be debugged.

The following shows an example of the open OCD configuration file when the DFP is to be debugged with five and nine instances of hart. If necessary, change the specification of the hart to be debugged. The hart ID for debugging can be selected from 0 and 16-31. However, the maximum number of harts that can be debugged at the same time is the number listed in 3.5.1. Also, be sure to debug hart0. In the example of the configuration file to be debugged with 9 harts, the difference from the setting to debug with 5 harts is described, so please refer to it when changing the specification of the hart to be debugged.

Open OCD configuration file (example of 5 harts debugging)

```
# Debug Adapter Configuration
interface remote_bitbang
remote_bitbang_host 192.168.56.1   ★ Specify [IP address of the server] which was specified in the CS+ debugger.
remote_bitbang_port 9824           ★ Specify [Port number of the server] which was specified in the CS+ debugger.

# Tap Declaration
set _CHIPNAME riscv
set _DAP_TAPID 0x100039df
set _ENDIAN little

jtag newtap $_CHIPNAME dap -irlen 5 -ircapture 0x01 -irmask 0x03 -expected-id $_DAP_TAPID

set _TARGETNAME    $_CHIPNAME.cpu0
set _TARGETNAME_16 $_CHIPNAME.cpu16
set _TARGETNAME_20 $_CHIPNAME.cpu20
set _TARGETNAME_24 $_CHIPNAME.cpu24
set _TARGETNAME_28 $_CHIPNAME.cpu28

target create $_TARGETNAME    riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 0 -rtos hwthread
target create $_TARGETNAME_16 riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 16
target create $_TARGETNAME_20 riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 20
target create $_TARGETNAME_24 riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 24
target create $_TARGETNAME_28 riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 28

target smp $_TARGETNAME $_TARGETNAME_16 $_TARGETNAME_20 $_TARGETNAME_24 $_TARGETNAME_28

$_TARGETNAME configure -event gdb-attach { halt }

riscv set_reset_timeout_sec   1200
riscv set_command_timeout_sec 1200
riscv set_enable_virt2phys off

puts "Before init"
# Server Configuration
gdb_port 3333                      ★ Port number for waiting for the connection with the GDB
gdb_report_data_abort enable
poll_period 500
init

# General Commands
bindto 0.0.0.0
```

```
targets $_TARGETNAME
```

Open OCD configuration file (example of 9 harts debugging)

```
# Debug Adapter Configuration
interface remote_bitbang
remote_bitbang_host 192.168.56.1   ★ Specify [IP address of the server] which was specified in the CS+ debugger.
remote_bitbang_port 9824            ★ Specify [Port number of the server] which was specified in the CS+ debugger.

# Tap Declaration
set _CHIPNAME riscv
set _DAP_TAPID 0x100039df
set _ENDIAN little

jtag newtap $_CHIPNAME dap -irlen 5 -ircapture 0x01 -irmask 0x03 -expected-id $_DAP_TAPID

set _TARGETNAME     $_CHIPNAME.cpu0
set _TARGETNAME_16 $_CHIPNAME.cpu16
set _TARGETNAME_17 $_CHIPNAME.cpu17   ★ Diff with 5 harts: Add hart17
set _TARGETNAME_20 $_CHIPNAME.cpu20
set _TARGETNAME_21 $_CHIPNAME.cpu21   ★ Diff with 5 harts: Add hart21
set _TARGETNAME_24 $_CHIPNAME.cpu24
set _TARGETNAME_25 $_CHIPNAME.cpu25   ★ Diff with 5 harts: Add hart25
set _TARGETNAME_28 $_CHIPNAME.cpu28
set _TARGETNAME_29 $_CHIPNAME.cpu29   ★ Diff with 5 harts: Add hart29

target create $_TARGETNAME     riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 0 -rtos hwthread
target create $_TARGETNAME_16 riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 16
target create $_TARGETNAME_17 riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 17   ★ Diff with 5 harts: Add hart17
target create $_TARGETNAME_20 riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 20
target create $_TARGETNAME_21 riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 21   ★ Diff with 5 harts: Add hart21
target create $_TARGETNAME_24 riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 24
target create $_TARGETNAME_25 riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 25   ★ Diff with 5 harts: Add hart25
target create $_TARGETNAME_28 riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 28
target create $_TARGETNAME_29 riscv -endian $_ENDIAN -chain-position $_CHIPNAME.dap -coreid 29   ★ Diff with 5 harts: Add hart29

target smp $_TARGETNAME $_TARGETNAME_16 $_TARGETNAME_20 $_TARGETNAME_24 $_TARGETNAME_28
$_TARGETNAME_17 $_TARGETNAME_21 $_TARGETNAME_25 $_TARGETNAME_29   ★ Diff with 5 harts: Add hart17, hart21, hart25, hart29

$_TARGETNAME configure -event gdb-attach { halt }
```

```
riscv set_reset_timeout_sec     1200
riscv set_command_timeout_sec 1200
riscv set_enable_virt2phys off

puts "Before init"
# Server Configuration
gdb_port 3333                          ★ Port number for waiting for the connection with the GDB
gdb_report_data_abort enable
poll_period 500
init

# General Commands
bindto 0.0.0.0
targets $_TARGETNAME
```

3.4.2 Starting the open OCD

Open [Terminal] in the virtual environment (Ubuntu) and run the following command.

```
$ openocd –f (Open OCD configuration file)
```

3.4.3 Point for caution on starting the open OCD

When the open OCD is started, confirm that the CS+ debugger starts debugging of the G4MH cores ([Connect to Debug Tool] is enabled). Starting the open OCD will fail if the CS+ debugger does not start debugging of the G4MH cores.

3.5 Setting up the NSI IDE

For details on the NSI IDE, refer to the DR1000C IDE User Manual from NSITEXE. This section describes setting up of the NSI IDE and points for caution on interlocked debugging of the G4MH cores and DFP.

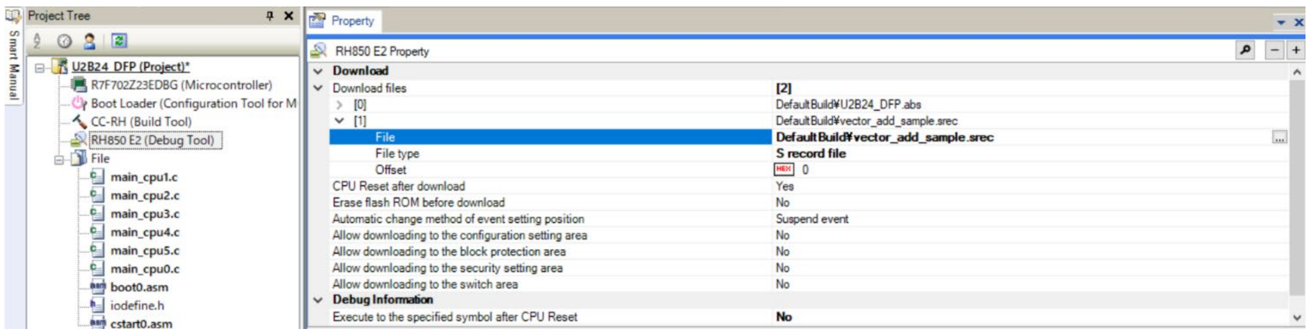
3.5.1 Number of instances of hart to be debugged

Up to nine instances of hart can be debugged when debugging the DFP installed in some RH850/U2B-series devices.

3.5.2 DFP program

When you create a new project for the NSI IDE, specify [RH850U2B] as the item for specifying the target board. When a newly created project is built, a DFP program (*.mot) file is generated. For the procedure for creating and building new projects from the NSI IDE, refer to the NSITEXE DR1000C IDE User Manual.

For interlocked debugging of a generated DFP program (*.mot), specify the file to be downloaded by clicking on [Project Tree] -> [RH850 E2 (Debug Tool)] -> [Download File Settings] -> [Download files] -> [[1]] in the CS+ debugger. Figure 3-7 shows the window for reference.



U2B24_DFP (Project)*
- R7F702Z23EDBG (Microcontroller)
- Boot Loader (Configuration Tool for M
- CC-RH (Build Tool)
- RH850 E2 (Debug Tool)
- File
  - main_cpu1.c
  - main_cpu2.c
  - main_cpu3.c
  - main_cpu4.c
  - main_cpu5.c
  - main_cpu0.c
  - boot0.asm
  - iodefine.h
  - cstart0.asm

RH850 E2 Property

Download	
Download files	[2]
[0]	DefaultBuild¥U2B24_DFP.abs
[1]	DefaultBuild¥vector_add_sample.srec
File	**DefaultBuild¥vector_add_sample.srec**
File type	**S record file**
Offset	HEX 0
CPU Reset after download	Yes
Erase flash ROM before download	No
Automatic change method of event setting position	Suspend event
Allow downloading to the configuration setting area	No
Allow downloading to the block protection area	No
Allow downloading to the security setting area	No
Allow downloading to the switch area	No
Debug Information	
Execute to the specified symbol after CPU Reset	**No**

Figure 3-7 Reference for Setting the Properties of the E2 Emulator in CS+ for Interlocked Debugging of the G4MH Cores and DFP ([Download File Settings] Tabbed Page)

3.5.3 Connection between the open OCD and GDB

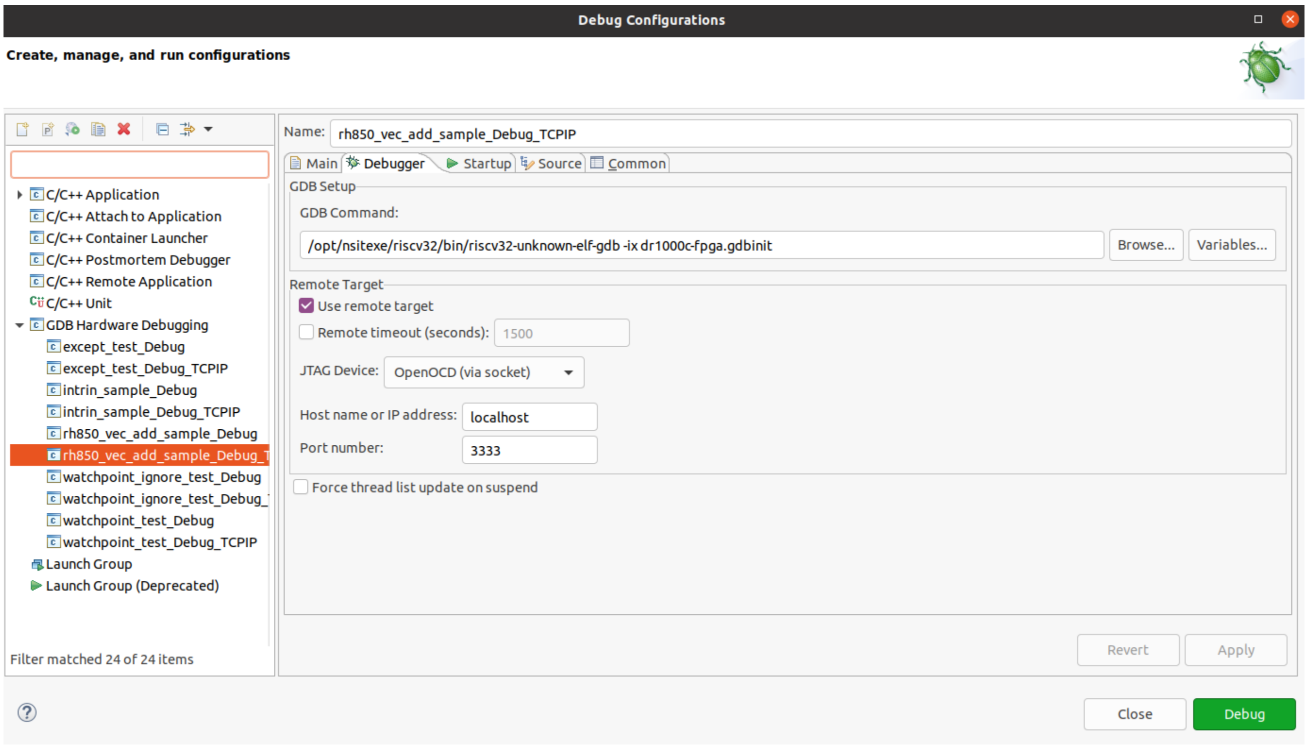
In the NSI IDE, connect the open OCD by specifying the IP address of the server on which the open OCD was started and the port number that is to wait for connection with the GDB.

Display the window with the following steps. Table 3-6 lists the settings and Figure 3-8 shows the window for reference.

(1) Right-click on the name of the target project in the [Project Explorer] view of the NSI IDE and click on [Debug As] -> [Debug Configurations…] to display the [Debug Configurations] window.
(2) Click on [GDB Hardware Debugging] -> [(project name)_Debug_TCPIP] -> [Debugger].

Table 3-6 Connecting the GDB to the Open OCD

Item	Setting
JTAG Device	Open OCD (via socket)
Host name or IP address	IP address of the server on which the open OCD was started. If the server is the same as that on which the NSI IDE was started, “local host” can also be specified.
Port number	Port number that waits for connection with the GDB specified by the open OCD


Debug Configurations
Create, manage, and run configurations
C/C++ Application
C/C++ Attach to Application
C/C++ Container Launcher
C/C++ Postmortem Debugger
C/C++ Remote Application
C/C++ Unit
GDB Hardware Debugging
except_test_Debug
except_test_Debug_TCPIP
intrin_sample_Debug
intrin_sample_Debug_TCPIP
rh850_vec_add_sample_Debug
rh850_vec_add_sample_Debug_T
watchpoint_ignore_test_Debug
watchpoint_ignore_test_Debug_
watchpoint_test_Debug
watchpoint_test_Debug_TCPIP
Launch Group
Launch Group (Deprecated)
Filter matched 24 of 24 items
Main
Debugger
Startup
Source
Common
GDB Setup
Browse...
Variables...
Remote Target
Revert
Apply
Close
Debug

Name: rh850_vec_add_sample_Debug_TCPIP

GDB Command: /opt/nsitexe/riscv32/bin/riscv32-unknown-elf-gdb -ix dr1000c-fpga.gdbinit

[✓] Use remote target
[ ] Remote timeout (seconds): 1500
JTAG Device: OpenOCD (via socket)
Host name or IP address: localhost
Port number: 3333
[ ] Force thread list update on suspend

Figure 3-8 Reference for Settings of [Debug Configurations] on the NSI IDE ([Debugger] Tabbed Page)

3.5.4 Starting debugging of the DFP

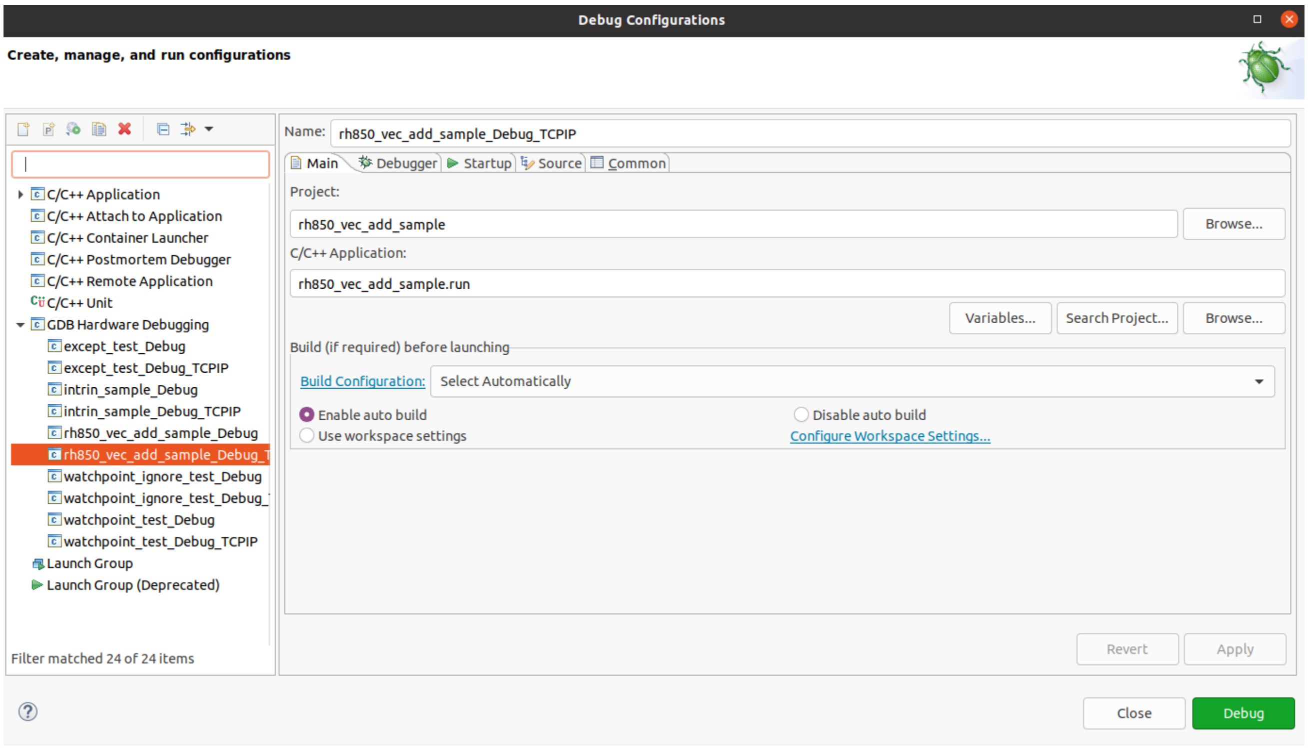
After the settings for section 3.5.3,

Connection between the open OCD and GDB, have been made, clicking on [Debug] in the lower left of the [Debug Configurations] window starts debugging.

Confirm the settings in the window before starting debugging. Display the [Main] window by clicking on [GDB Hardware Debugging] -> [(project name)_Debug_TCPIP] in the [Debug Configurations] window. Table 3-7 lists the settings and Figure 3-9 shows the window for reference.

Table 3-7 Setting for Debugging of a Project

Item	Setting
C/C++ Application	ELF file. The NSI IDE generates a .run file as an ELF file. The .run file is generated during the generation of a DFP program. For this setting, select a .run file rather than a .mot file.


Debug Configurations

Create, manage, and run configurations

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Container Launcher
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- C/C++ Unit
- GDB Hardware Debugging
  - except_test_Debug
  - except_test_Debug_TCPIP
  - intrin_sample_Debug
  - intrin_sample_Debug_TCPIP
  - rh850_vec_add_sample_Debug
  - rh850_vec_add_sample_Debug_T
  - watchpoint_ignore_test_Debug
  - watchpoint_ignore_test_Debug_
  - watchpoint_test_Debug
  - watchpoint_test_Debug_TCPIP
- Launch Group
- Launch Group (Deprecated)

Filter matched 24 of 24 items

Name: rh850_vec_add_sample_Debug_TCPIP

Main | Debugger | Startup | Source | Common

Project: rh850_vec_add_sample

C/C++ Application: rh850_vec_add_sample.run

Build (if required) before launching

Build Configuration: Select Automatically

[● Enable auto build] ○ Disable auto build ○ Use workspace settings

Configure Workspace Settings...

Figure 3-9 Reference for Settings of [Debug Configurations] of the NSI IDE ([Main] Tabbed Page)

3.6 Procedure for establishing a system for interlocked debugging of the G4MH cores and DFP

This section describes the procedure for establishing an environment for interlocked debugging of the G4MH cores and DFP.

(1) Make option byte settings for the DFP-equipped RH850/U2B-series device (see section 3.1.1).
(2) Set up interlocked debugging of the G4MH cores and DFP in the CS+ debugger (see section 3.2.1).
(3) Specify the DFP program to be debugged in the CS+ debugger (see section 3.5.2).
(4) Select [Connect to Debug Tool] from the [Debug] menu in the CS+ debugger to start debugging the G4MH cores.
(5) Start the Ubuntu environment in VirtualBox to provide a Linux environment for using the DR1000C debugger and IDE.
(6) Set the same IP address and port number of the open OCD for the DR1000C debugger and IDE as the settings in the CS+ debugger (see section 3.4.1).
(7) Start the open OCD for the DR1000C debugger and IDE (see section 3.4.2).
(8) Start the NSI IDE for the DR1000C debugger and IDE and set the connection between the open OCD and GDB (see section 0).
(9) Start debugging of the DFP in the NSI IDE for the DR1000C debugger and IDE (see section 0).

Debugging of the G4MH cores and DFP is now enabled in the CS+ debugger and NSI IDE, respectively.

4. Interlocked Debugging of the G4MH Cores and DFP

4.1 Interlocked execution of the G4MH cores and DFP

This section describes interlocked execution and points for caution on debugging of the G4MH cores and DFP.

4.1.1 Procedure for interlocked execution of the G4MH cores and DFP

When the G4MH cores and DFP are in break (DB-Halt) states during interlocked debugging, the procedure for interlocked execution of the G4MH cores and DFP is as follows.

(1) Click on [Go] in the CS+ debugger.

(2) Click on [Resume] in the NSI IDE within the timeout time for waiting for the request for execution by the DFP.

4.1.2 Points for caution on interlocked execution of the G4MH cores and DFP

No. 1 Clicking on [Resume] in the NSI IDE without clicking on [Go] in the CS+ debugger

During interlocked debugging of the G4MH cores and DFP, clicking on [Resume] in the NSI IDE without clicking on [Go] in the CS+ debugger leaves the G4MH cores in the break state and only the DFP starts execution of a program.

While only the DFP is executing a program, clicking on [Suspend] in the NSI IDE or halting the DFP at a breakpoint transfers the DFP to the DB-Halt state. Following the procedure for interlocked execution of the G4MH cores and DFP in this state starts interlocked execution by the G4MH cores and DFP and both the G4MH cores and DFP start execution of the given programs.

No. 2 Clicking on [Go] in the CS+ debugger while the DFP is executing a program

During interlocked debugging of the G4MH cores and DFP, if [Go] is clicked in the CS+ debugger while the DFP executes a program, an error will occur and the G4MH cores will not start execution of the program.

No. 3 Clicking on [Go] in the CS+ debugger but not clicking on [Resume] in the NSI IDE within the timeout time

During interlocked debugging of the G4MH cores and DFP, if the timeout time after [Go] is clicked in the CS+ debugger has elapsed without a click on [Resume] in the NSI IDE, an error will occur and the G4MH cores will not start executing programs.

4.2 Interlocked breaks of the G4MH cores and DFP

This section describes interlocked breaks and a point for caution on breaks in debugging of the G4MH cores and DFP.

4.2.1 Operation of interlocked breaks for the G4MH cores and DFP

When the G4MH cores and DFP are in the interlocked execution state for interlocked debugging of the G4MH cores and DFP and a break (halt) of any of the kinds listed below occurs for a G4MH core or the DFP, interlocked breaks occur in the G4MH cores and DFP so that the break (halt) is generally applicable.

- Clicking on [Stop] in the CS+ debugger (forced break for a G4MH core)
- Clicking on [Suspend] in the NSI IDE (forced halt of the DFP)
- Occurrence of a break for a G4MH core at a breakpoint that has been set in the CS+ debugger (event break for the G4MH core)
- Occurrence of a break for the DFP at a breakpoint that has been set in the NSI IDE (event break for the DFP)

5. Inquiries

If trouble arises or you need more information on usage during the debugging of a DFP-equipped RH850/U2B-series device, contact the following companies. If you do not know which company to ask, contact Renesas Electronics.

- Trouble or information on usage during the debugging of a G4MH core by using the CS+ debugger: Renesas Electronics
- Trouble or information on usage during debugging of the DFP by using the NSI IDE: NSITEXE

Revision History	E2 Emulator Additional Document for User’s Manual (Setting up Interlocked Debugging of the DFP in an RH850/U2B-Series Device)

Rev.	**Date**	**Description**	
		Page	**Summary**
0.50	Apr.25.22	—	First Edition issued
1.00	Jul.20.22	10	Fixed 3.4.1 chapter title
		10	Add description of 9 harts debugging in 3.4.1
		13	Fixed 3.5.1 Number of instances of hart to be debugged from five to nine

E2 Emulator Additional Document for User's Manual
(Setting up Interlocked Debugging of the DFP in an
RH850/U2B-Series Device)

Publication Date: Rev.1.00 Jul.20.22

Published by: Renesas Electronics Corporation

E2 Emulator

Additional Document for User's Manual

(Setting up Interlocked Debugging of the DFP in an RH850/U2B-Series Device)

Renesas Electronics Corporation

R20UT5150EJ0100