V850/850E ICE SERVER
Reference Manual

Document #10952 Rev.9.5

DISCLAIMER

GREEN HILLS SOFTWARE, INC. AND ADVANCED DATA CONTROLS CORP. MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.

Further, Green Hills Software, Inc. and Advanced Data Controls Corp. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Green Hills Software, Inc. and Advanced Data Controls Corp. to notify any person of such revision or changes.

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

Chapter 1
Introduction

This manual describes how to use MULTI with the V850/V850E ICE made by Renesas Electronics. The manual provides information on files included in the package and how to customize certain files. This manual is intended for users who already know about basic operations of MULTI. For detailed use of MULTI, please refer to MULTI related manuals.

850eserv2 has succeeded the function of 850eserv almost and 850eserv2 supports TimeMachine which is the Advanced Capability of MULTI. You can use Trace produce in Trace List window and TimeMachine, with the combination of MULTI V4.0.7 or later,850eserv2 V2.000 or later, and IECUBE or IECUBE2 made by Renesas Electronics.
You can use IECUBE2(V850E2 Core) with combination of MULTI V5.1.6 or later and 850eserv2 V2.005.
You can use RH850 with E1 Emulator with combination of MULTI V6.1.4 or later and 850eserv2 V2.023.
You can use IECUBE2(RH850) with combination of MULTI V6.1.4(V2013.1.5) or later and 850eserv2 V2.028.

Concerning the functional difference of 850eserv2 and 850eserv, please refer to "Chapter 15 The changed part from 850eserv" on page 211.

It is used in order to show the part where this kind of sign especially is important

Chapter 2
Notational Conventions

Unless otherwise specified, the following conventions apply throughout the manual (They are subsets of "MULTI User's Guide"):

* Highlighted text (**bold characters**) contained in ordinary text indicates that it must be typed or output as specified.
* *Italicized characters* contained in ordinary or highlighted text indicates that they must be replaced with appropriate values.
* Commands or options enclosed in **`[ ]`** indicates that they are options.
* **`A`**|**`B`**|**`C`** indicates that one of **`A`**, **`B`**, and **`C`** should be typed as necessary.
* **`<cr>`** indicates the point where to press the Enter key.

This list provides information for making target connections using the emulators and interfaces listed in the table below.

In-circuit emulator	Interface	Referred to in this book as
IE-V850E1-CD-NW	PCMCIA	OCD Emulator(N-Wire Emulator)
MINICUBE	USB	OCD Emulator(MINICUBE)
QB-MINI2	USB	MINICUBE2
E1 Emulator	USB	E1 Emulator
E20 Emulator	USB	E20 Emulator
IECUBE	USB	IECUBE
IECUBE2	USB	IECUBE2

Chapter 3
Files

The V850/V850E ICE server (**850eserv2**) package includes the following files:

file	contents
850eserv2.exe	V850/V850E ICE server by Renesas Electronics
850win.exe	850eserv2 Configuration window application
tipserv.exe	TIP supporting application

Chapter 4
Setup

This chapter describes the following items.

- Setup with Windows

Setup with Windows

Insert the supplied disk into the appropriate drive of the PC then, from the Windows Program Manager, execute setup.exe on the diskette. This completes basic installation of the server.

Connect to ICEs

<<In case of N-Wire Emulator>>

Before using 850eserv2, you must properly configure the PC, interface card, ICE, card-type ICE, and target system. Make sure that these have been properly set up. 850eserv2 uses the interface card to communicate with each ICE. Before operating 850eserv2, install the Renesas Electronics interface card device driver to enable the card. Refer to the interface card manual for details of the device driver.

<<In case of MINICUBE,MINICUBE2,E1/E20 emulator,IECUBE,IECUBE2>>

Before using 850eserv2, you must properly configure the PC, USB driver and each ICE. Make sure that these have been properly set up. 850eserv2 uses the USB to communicate with the each ICE. Before operating 850eserv2, install the Renesas Electronics USB driver to enable the USB.
You can download USB driver for each ICE from following web page:

```
http://www.renesas.com/ghs_debug_if
```

For details of the each ICE and USB driver, please refer to the each ICE Setup Manual included with the each ICE.

The File Necessary at the Execution

The EXEC library, Device file and USB driver by Renesas Electronics are needed to debug in 850eserv2. You can download these tools from following web page:

```
http://www.renesas.com/ghs_debug_if
```

Uses EXEC library corresponding to each ICE.

ICE	EXEC library name
N-Wire Emulator, MINICUBE, E1/E20 Emulator(JTAG), IECUBE	EX850G32.DLL
MINICUBE2, E1/E20 Emulator(Serial)	EX850O32.DLL
MINICUBE+V850E2Core, E1/E20 Emulator(JTAG)+V850E2Core, IECUBE2(V850E2Core)	EX850G32E2R.DLL
MINICUBE2+V850E2Core, E1/E20 Emulator(Serial)+V850E2Core	EX850O32E2R.DLL
E1/E20 Emulator(LPD)+V850E2Core	EX850L32E2R.DLL

ICE	EXEC library name
E1/E20 Emulator(LPD)+RH850, IECUBE2(RH850)	EXRH850G3.DLL

Please prepare the EXEC library in MULTI or 850eserv2 installed directory.

Setting of environment variable

There are two Windows environment variables that must be set: IEPATH and DEVICE_FILE. The following sections describe each environment variable. Windows environment variables can be set from "Advanced" tag in System Properties window(righet-click on "My Computer" and choose "Properties").

The IEPATH Environment Variable

The IEPATH environment variable should be set to specify the directory where the Device file is located. For example, the Device file is located in ***C:\green\v850e***:

```
Variable name : IEPATH
Variable Value: C:\green\v850e
```

The DEVICE_FILE Environment Variable

The DEVICE_FILE environment variable should be set to specify the Device file name that contains the appropriate target-specific information. The specified device file must exist in the directory specified by IEPATH. For example, the Device file name is ***DF3283.800*** for V850ES/SG2:

```
Variable name : DEVICE_FILE
Variable Value: DF3283.800
```

Chapter 5
Starting 850eserv2

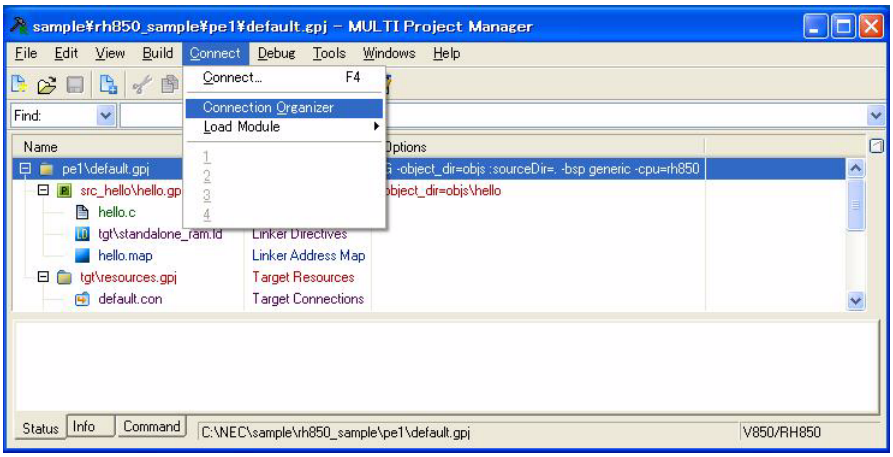
This chapter describes the following items.

- Connects via Connection Oganizer
- Connects via MULTI command pane
- Downloads user program

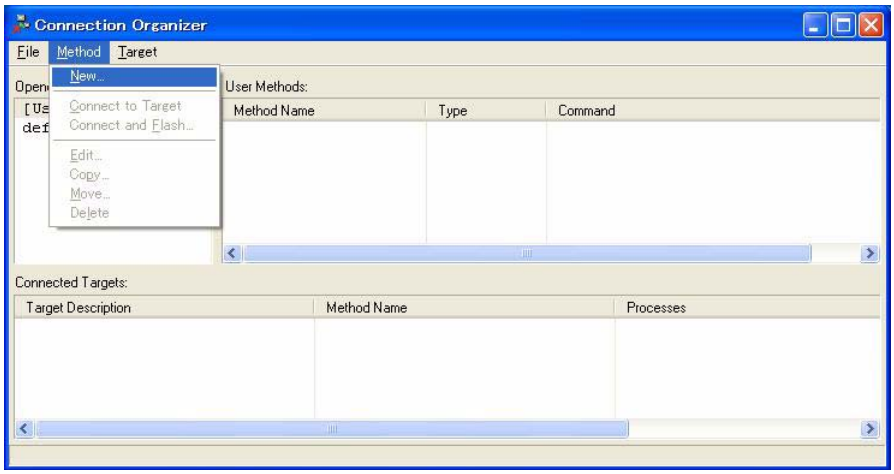
You can start 850eserv2 via **connect** command in MULTI command pane or Connection Editor wondow in Connection Oganizer. It describes two method in this chapter.

Connects via Connection Oganizer

You can connect to 850eserv2 via Connection Oganizer. Connection Oganizer can be opened via "Connect > Connection Oganizer" in MULTI Project Manager or "Target > Show Connection Oganizer" in MULTI debugger menu.



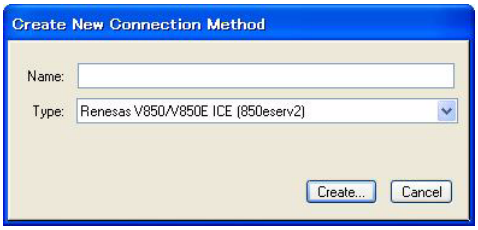
Connection Oganizer overview



This window is opened when starts Connection Oganizer. The new connection method can be created via "Connection Editor" window which is opened via "Method > New" in menu. The created connection

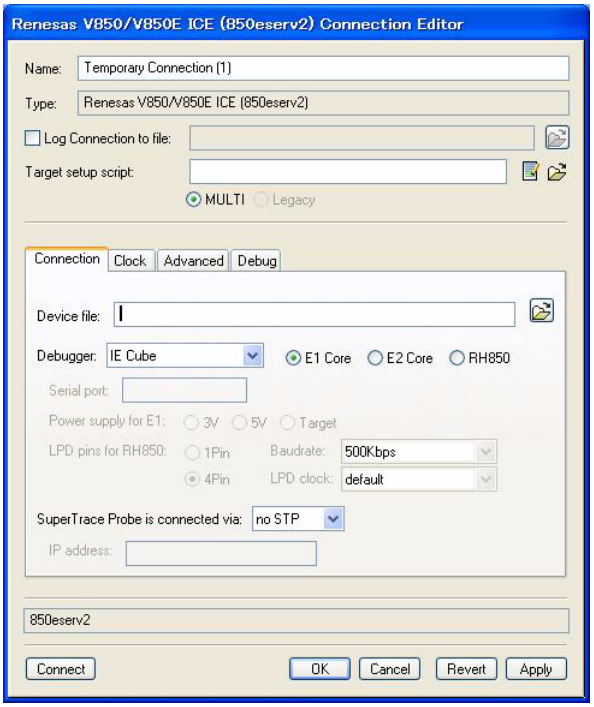
method is dispayed in "User Method". You can connect to 850eserv2 when you click this method and select "Method > Connect to Target" in menu.

Selects Connection method type



This window is opened when you select "Method > New" in Connection Oganizer's menu. Please input arbitrary name to "Name:" and select "Renesas V850/850E ICE(850eserv2)" in "Type:". When you push "Create" button, Connection Editor window is opened.

Creates connection method



You can create a new connection method which is used to connect to 850eserv2 in this window. 850eserv2 is connected when push "Connect" button after set connection method. When it connects to 850eserv2 via MULTI Project Manager, MULTI debugger is opened automatically. In this case, please load executable image via "File > Debug Program" in MULTI debugger's menu.

Name:	Inputs Connection method name.
Type:	Shows Connection method type.
Log Connection to file:	Output connection log between MULTI and 850eserv2 to specified file.
Target Setup script:	Specifies setup script(`.mbs` file). This file is loaded automatically when MULTI downloads target program.
Connect:	Connects to 850eserv2.

Connection Tag

Device File:	Specifies target Device file. If you do not specify an appropriate Device file, you may be not able to debug correctly.
Debugger:	Selects use ICE. When you connect V850E1 core or V850E2/ME3, you should select "E1 core". When you connect V850E2 core, you should select "E2 core". When you connect RH850, you should select "RH850". If you select MINICUBE2 or E1/E20 Emulator(Serial), you should input a port name which is used to connect to MINICUBE2 into "**Mini Cube 2 port:**". If you select E1 Emulator, you should select supplied power from ICE to target(**3V** or **5V**),or supplied from target(**Target**). If you select E1/E20 Emulator(LPD) and "RH850", you should select number of pins used LPD connection to "**LPD pins:**". When you select **1Pin,** you should specify "**Baudrate:**" to 2000, 1000, 500(Kbps). When you select **4Pin**, you should specify "**LPD clock:**" to 11000, 5500(KHz) or **default**. When you select **default**, it connects to ICE by default LPD clock of device.
SuperTrace Probe is connected via:	Selects connection method with SuperTrace Probe. Default is no use with SuperTrace Probe(no STP). If you select "ethernet", you should input the IP address which is registered in SuperTrace Probe into "**IP address:**".

Clock Tag

Device Clock Rate:	Specifies target operating frequencies(Device Clock). When not set target's operating frequencies, or when the wrong value is set, Emulator cannot access to target device correctly. For detail of each target operating frequencies, please refer to each device manuals.
Main Clcok:	Specifies Main clock per kHz.
Sub Clcok:	Specifies Sub clock per Hz.
Use sub clock when break is hit?	If you select "Yes", emulator use Sub clock when program is breaking.

When you select supplied power from ICE to target(**3V** or **5V**), the target may not work correctly if selected supplied power unmatches the device specifications or the power voltage target requires. Please check it when select **3V** or **5V**.

Advanced Tag

Protect I/O memory:	Disables I/O memory protect. By default, you cannot view or change the I/O memory in MULTI and **SFR** command is needed for this purpose.
Write 0's into BSS section:	Zero clears when dawnloads.
Debug with RD850 or AZ850:	Uses RD850 or AZ850 by Renesas Electronics.
Fast download to Flash memory:	Downloads all sections to Code flash memory at a time. If this option is specified, 850eserv2 downloads all sections to Code flash memory at a time to reduce writing count to flash memory and become writing speed faster after progress bar is 100%, you cannot cancel downloading to Code flash memory.
Hot Plugin Debugging:	Launches Hot-plugin mode. For detail of Hot-plugin debugging, please refer "Chapter 10 Hot-plugin debugging" on page 43
Set Jtag I/F when end debugging:	Sets connection interface to JTAG when debugging is finished. When debugger is finished unusually, connection interface can not be changed to JTAG. It can be used with RH850.
Use software break-point to flash:	Use software breakpoint by writing instruction to Code flash area. It can be used with RH850.

Disable option byte settng	Disable option byte setting via **OPBYTE** command. It can be used with RH850.
Disable to change clock when write to flash	Do not change device clock when it writes to flash memory. It can be used with RH850.
ICE name:	When it connects multiple ICE at same time, specify ICE name. Specified ICE name is can be get from after the last "/" of Device instance path(In WindowsXP, Device instance ID) in USB driver.
Registry ID Code:	Specifies attestation ID code for unlock the RSU(ROM Security Unit). If your Device has RSU, you need to specify 20(V850E1 core), 24(V850E2 core) or 32(RH850) digits attestation ID code and unlock the RSU. For detail of RSU, please refer to "Chapter 12 The attestation code" on page 53.
Environment variable:	Specify the environment variables.

In RH850, usually it uses hardware breakpoint at breakpoint. When turn on to **Use software breakpoint to flash:**, it uses software breakpoint by writing instruction to Code flash area. In this case, it uses hardware breakpoint to out of Code flash area.

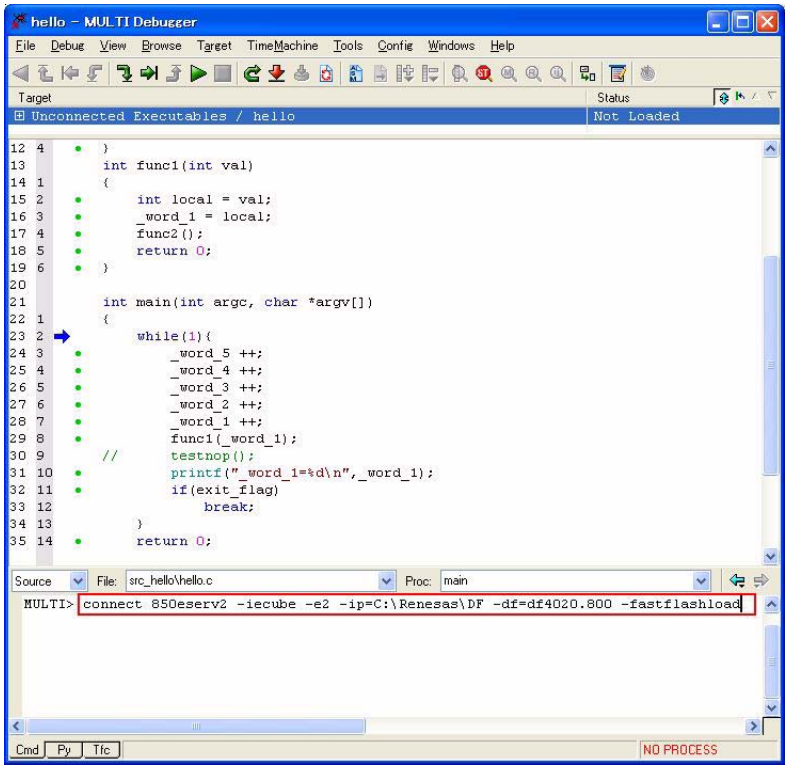
Debug Tag

The items in this tag are using to debug into 850eserv2. Therefore, you do not need to specify these usually.

Connects via MULTI command pane

connect command

You can use **connect** command to connect to 850eserv2. **connect** command is input in MULTI debugger command pane(Red pane in following image) and can be attached each starting options. For the detail of these starting options, please refer to "Server Starting Options" on page 18.



Following is an exapmle of **connect** command:

```
connect 850eserv2 -iecube -df=DF3283.800
```

If the connection to 850eserv2 is successful, the following will appear on MULTI command pane:

```
Connected to target '850eserv2...'
```

If following message is displayed in MULTI command pane:

```
Please specify DEVICE FILE using DEVICE_FILE environment variable
```

Please specify Device file name by environment variable DEVICE_FILE or starting option -df. Or if following message is displayed in MULTI command pane:

```
Module 'ex850g32.dll' not found
Unable to connect to emulator
```

Please confirm to prepare the EXEC library in MULTI or 850eserv2 installed directory. For the detail of other error message, please refer to "Appendix A ERROR Message From ICE" on page 225.

Server Starting Options

There are those below in starting option. Starting option distinguishes capital letter and the small letter. It attaches these options with **connect** command.

<<Only OCD emulator and E1/E20 emulator(JTAG) specifiable options>>

-cdnw	Connects through PCMCIA to N-Wire emulator（IE-V850E1-CD-NW）.
-2m	Sets the DCK to 10 MHz because of compatibility with 2-meter cables. Default DCK of N-Wire emulator is 20MHz.(Only N-Wire emulator)
-minicube	Connects through USB to MINICUBE.
-e1jtag	Connects through USB to E1 Emulator JTAG.
-e20jtag	Connects through USB to E20 Emulator JTAG.
-e1lpd	Connects through USB to E1 Emulator LPD in V850E2 Core.
-e20lpd	Connects through USB to E20 Emulator LPD in V850E2 Core.
-e1lpd1=<*baudrate*>	Connects through USB to E1 Emulator LPD by 1pins mode in RH850. You should specify "*baudrate*" to 2000, 1000, 500(Kbps).
-e1lpd4=<*LPD clock*>	Connects through USB to E1 Emulator LPD by 4pins mode in RH850. You should specify "*LPD clock*" to 16500, 11000, 5500(KHz).
-e20lpd1=<*baudrate*>	Connects through USB to E20 Emulator LPD by 1pins mode in RH850. You should specify "*baudrate*" to 2000, 1000, 500(Kbps).
-e20lpd4=<*LPD clock*>	Connects through USB to E20 Emulator LPD by 4pins mode in RH850. You should specify "*LPD clock*" to 16500, 11000, 5500(KHz).
-dck20	Sets the DCK to 20 MHz. Default DCK of MINICUBE is 10MHz. (Only MINICUBE or E1/E20 emulator(JTAG))

When -e1lpd or -e20lpd are specified, it needs the EXEC library for only LPD (Low Pin debug) (EX850L32E2R.DLL). When its connects to RH850, it needs the EXEC library for only RH850(EXRH850G3.DLL).

<Only MINICUBE2 and E1/E20 emulator(Serial) specifiable options>>

-minicube2	Connects through USB to MINICUBE2.
-e1serial	Connects through USB to E1 Emulator Serial.
-e20serial	Connects through USB to E20 Emulator Serial.
-p=<*port name*>	Specify use port name to connect target. If device does not support specified port name, it can not connect to MINICUBE2. In that case, error message and the port name list that can be specified is displayed.
-writemon	Does not display error message when write to Monitor reserved area.

Example:

```
connect 850eserv2 -minicube2 -p=uarta0
```

<<Only OCD emulator specifiable options>>

-hotplugin	Launches Hot-plugin mode. For detail of Hot-plugin debugging, please refer "Chapter 10 Hot-plugin debugging" on page 43

<<Only E1 emulator specifiable options>>

-t3v	E1 emulator supplies 3V power to target.
-t5v	E1 emulator supplies 5V power to target.

When you select supplied power from ICE to target(use -t3v or -t5v), the target may not work correctly if selected supplied power unmatches the device specifications or the power voltage target requires. Please check it when use -t3v or -t5v.

<<Only IECUBE and IECUBE2 specifiable options>>

-iecube	Connects through USB to IECUBE or IECUBE2(attached "-e2" or "-rh850").
-tc	Specifies that the target board be connected to IECUBE or IECUBE2. If you specify this option, it detects unusual power status. Be sure to power on the target board.

<<Only IECUBE specifiable options>>

-stp *<IP address or hostname>*	Specifies Green Hills SuperTrace Probe for LAN connection. IP address or hosname is required.
-usb	Specifies Green Hills SuperTrace Probe for USB connections.

<<Only OCD emulator, E1/E20 emulator(JTAG) and IECUBE2 specifiable options>>

-nocache	This option is available for V850E2 Core. Downloaded data into Code flash area is not cached. The default behavior(without -nocache) is that downloaded data is cached.
-opbyte_jtag	This option is available for RH850. Sets connection interface to JTAG when debugging is finished. When debugger is finished unusually, connection interface can not be changed to JTAG.
-opbyte_disable	This option is available for RH850. Disable option byte setting.
-noflashclock	This option is available for RH850. Do not change device clock when it writes to flash memory.
-useswbp	This option is available for RH850. Use software breakpoint by writing instruction to Code flash area.

In RH850, usually it uses hardware breakpoint at breakpoint. When specifies -useswbp, it uses software breakpoint by writing instruction to Code flash area. In this case, it uses hardware breakpoint to out of Code flash area.

<< Each ICE common options >>

-e1	Connects V850E1 core or V850E2/ME3
-e2	Connects V850E2 core. If "-e1", "-e2" or "-rh850" are omitted, 850eserv2 connects "-e1" core.
-rh850	Connects RH850. If "-e1", "-e2" or "-rh850" are omitted, 850eserv2 connects "-e1" core.

-id <*ID code*>	Specifies attestation ID code for unlock the RSU(ROM Security Unit). If your Device has RSU, you need to specify 20(V850E1 core), 24(V850E2 core) or 32(RH850) digits attestation ID code and unlock the RSU. For detail of RSU, please refer to "Chapter 12 The attestation code" on page 53. If your Device does not have RSU, ID code becomes invalid.
-fastflashload	Downloads all sections to Code flash memory at a time. If this option is specified, 850eserv2 downloads all sections to Code flash memory at a time to reduce writing count to flash memory and become writing speed faster after progress bar is 100%, you cannot cancel downloading to Code flash memory.
-noiop	Disables I/O memory protect. By default, you cannot view or change the I/O memory in MULTI and **SFR** command is needed for this purpose. If this option is specified, you can view or change the I/O memory in MULTI.
-tip	Uses RD850 or AZ850 by Renesas Electronics.
-df =<*DF name*>	Specify the device filename after "=". It is a usage similar to environment variable DEVICE_FILE. You can specify directory specification. In this case, you can omit "-ip".
-ip=<*path*>	Specify the directory where the device file is located after "=". It is a usage similar to environment variable IEPATH.
-dclock= <*Main_clock,* *Sub_clock,* *swon* | *swoff*>	Specifies target operating frequencies(Device Clock). The parameters which should be specified are same as **DCLOCK** command. Please refer to "DCLOCK" on page 85. In RH850, this option cannot be omitted.
-ice <*ice name*>	When it connects multiple ICE at same time, specify ICE name. Specified ICE name is can be get from after the last "/" of Device instance path(In WindowsXP, Device instance ID) in USB driver.
-env <*variable*>	Specify the environment variables.

Example:

```
connect 850eserv2 -iecube -df=DF3288y.800 -ip=\green\v850e
-env V850_A=5,V850_B=args
```

When -df, -ip and environment variable DEVICE_FILE, IEPATH are specified simultaneously, the device file and the directory specified by -df and -ip are given to priority.

<< Each Server common options >>

As for options common to server, it can use even with the server of other companies make ICE correspondence other than 850eserv2.

-nobss	Variable section without an initial value section not to be downloaded.
-nodata	Variable section with an initial value section not to be downloaded.
-notext	Program code section not to be downloaded.
-noload	All section not to be downloaded.
-bss	Variable section without an initial value section to be downloaded.
-data	Variable section with an initial value section to be downloaded.
-text	Program code section to be downloaded.
-loadall	All section to be downloaded.
-nosyscalls	It dosen’t set a breakpoint in .syscall section when downloading.
-setup<*filename*>	Script file which is specified is executed when downloading.

<<Starting options available list>>

Option	N-Wire Emulator	MINICUBE	MINICUBE 2	E1/E20 Emulator (JTAG)	E1/E20 Emulator (Serial)	IECUBE	IECUBE2
-cdnw	Enable	-	-	-	-	-	-
-minicube	-	Enable	-	-	-	-	-
-minicube2	-	-	Enable	-	-	-	-
-e1jtag	-	-	-	Enable	-	-	-
-e20jtag	-	-	-	Enable	-	-	-
-e1lpd	-	-	-	Enable	-	-	-
-e20lpd	-	-	-	Enable	-	-	-
-e1lpd1	-	-	-	Enable	-	-	-
-e1lpd4	-	-	-	Enable	-	-	-
-e20lpd1	-	-	-	Enable	-	-	-
-e20lpd4	-	-	-	Enable	-	-	-
-e1serial	-	-	-	-	Enable	-	-
-e20serial	-	-	-	-	Enable	-	-
-iecube	-	-	-	-	-	Enable	Enable
-e1	Enable	Enable	Enable	Enable	Enable	Enable	-
-e2	-	Enable	Enable	Enable	Enable	-	Enable
-rh850				Enable			
-2m	Enable	-	-	-	-	-	-
-dck20	-	Enable	-	Enable	-	-	-
-nocache	-	Enable	-	Enable	-	-	Enable
-opbyte_jtag	-	-	-	Enable	-	-	Enable
-opbyte_disable				Enable			Enable
-noflashclock				Enable			Enable
-useswbp	-	-	-	Enable	-	-	Enable

Option	N-Wire Emulator	MINICUBE	MINICUBE 2	E1/E20 Emulator (JTAG)	E1/E20 Emulator (Serial)	IECUBE	IECUBE2
-ice	-	Enable	Enable	Enable	Enable	-	Enable
-p	-	-	Enable	-	Enable	-	-
-writemon	-	-	Enable	-	Enable	-	-
-hotplugin	-	Enable	-	-	-	-	-
-t3v				Enable (E1 only)	Enable (E1 only)		
-t5v				Enable (E1 only)	Enable (E1 only)		
-stp	-	-	-	-	-	Enable	-
-usb	-	-	-	-	-	Enable	-
-tc	-	-	-	-	-	Enable	Enable
-id	Enable	Enable	Enable	Enable	Enable	-	Enable
-fastflashload	Enable	Enable	Enable	Enable	Enable	-	Enable
-noiop	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-df	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-ip	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-env	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-tip	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-nobss	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-nodata	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-notext	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-noload	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-bss	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-data	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-text	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-loadall	Enable	Enable	Enable	Enable	Enable	Enable	Enable

Option	N-Wire Emulator	MINICUBE	MINICUBE 2	E1/E20 Emulator (JTAG)	E1/E20 Emulator (Serial)	IECUBE	IECUBE2
-nosyscalls	Enable	Enable	Enable	Enable	Enable	Enable	Enable
-setup	Enable	Enable	Enable	Enable	Enable	Enable	Enable

Downloads user program

Please specify the target operating frequencies(Device Clock) by **DCLOCK** command after MULTI connects to 850eserv2 correctly. When not set target's operating frequencies, or when the wrong value is set, Emulator cannot access to target device correctly. In RH850, specify the target operating frequencies(Device Clock) via **Device Clock Rate:** in the Connection Editor window or -dclock option. If specified clock is different to the clock of oscillator on the target, The next error is displayed,

```
RSU info write error 0xca5: fatal err (could not connect to target)
Please check target connection and main clock specified with -dclock.
```

and 850eserv2 launching is failed.

When you push "Go" or "Step" button after initializing the target, the user program is downloaded and executed. Or when you input **load** into MULTI command pane, the user program is downloaded.
If following message is displayed in MULTI command pane,

```
Download failed, error during write 0xnnnnnnnn - 0xmmmmmmmm
debug server: download of "a.out" failed
Load: Download failed.
```

downloading is failed for the specified address cannot be used on the emulator or emulation memory is not mapping as RAM. Please check its link address.

In case of MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core, or IECUBE2(V850E2Core), if you have not set the Flash Option area via **OPBYTE** command, following warning message is displayed.

```
WARNING : Flash Option area settings are default value.
Use 'opbyte' command to change Flash Option area settings.
```

CPU is reset when you set the Flash Option area and attestation ID code for RSU.

Chapter 6
Editing MULTI Resource Files

This chapter describes the following items.

- Overview
- Example of resource files
- MULTI setup script

Overview

A file used to set server-specific commands, etc. for MULTI debuggers is called a resource file. Typically, `multi.rc` file is used. This file is stored in the following directory.

```
C:\Documents and Settings\(user name)\Application Data\GHS
```

After MULTI start, a resource file located this directory will be loaded automatically, and included commands will be executed sequentially. In this file, include font settings and configure commands such as button and mouse. The following is an example of a simple

resource file:

```
/* sample initial configuration commands */
button toggle $_DISPMODE := !$_DISPMODE
mouse mouse3=disconnect
configure tabsize=8
```

Note that server Target commands are available only for a server on which MULTI is installed. If you specify a Target command for the server in the resource file `multi.rc`, it will be ignored as an error. Do not include any server Target command in these resource files.
MULTI has the ability to automatically load resource files related to a file to be debugged. For example, suppose that debugged module file name is `test.out` and start the debugger by specifying `test.out`.
If the `multi.rc` file exsits, MULTI will load it, and if the current directory contains a resource file called `test.out.rc`, MULTI will also load and execute it automatically. That is, if you attache the extension of a debugged file name with `.rc`, it will become the name of its resource file.

In this way, you can use different resource files for each object file to be debugged. Resource files are loaded only when debugging of their object file starts. Therefore, you may want to make initial settings of the board in prior to debugging by including Target commands in the resource files. Note that if you include the **connect 850eserv2** command in `test.out.rc`, you must place it before the other target commands.

You can also use a resource file whose name has nothing to do with the object file name. In this case, however, start MULTI in debug mode and handle that file as a play back file. That is, the file cannot be loaded automatically, but must play back according to a MULTI command as follows:

```
<iecube.rc
```

Example of resource files

The following is an example of resource file:

<<In the case of N-Wire Emulator>>

```
SERVERTIMEOUT=500
connect 850eserv2 -cdnw -id ffffffffffffffffffff
target dclock 5000 32768 swoff /* Sets target clocks */
```

<<In the case of MINICUBE>>

```
SERVERTIMEOUT=500
connect 850eserv2 -minicube -id ffffffffffffffffffff
target dclock 8000 32768 swoff /* Sets target clocks */
target sfr wdtm2=0x0 /* Stops Watchdog timer reset*/
```

<<In the case of MINICUBE2>>

```
SERVERTIMEOUT=500
connect 850eserv2 -minicube2 -p=csib0 -id ffffffffffffffffffff
target dclock 5000 32768 swoff /* Sets target clocks */
```

<<In the case of E1 Emulator(JTAG)>>

```
SERVERTIMEOUT=500
connect 850eserv2 -e1jtag -id ffffffffffffffffffff
target dclock 8000 32768 swoff /* Sets target clocks */
target sfr wdtm2=0x0 /* Stops Watchdog timer reset*/
```

<<In the case of IECUBE>>

```
SERVERTIMEOUT=500
connect 850eserv2 -iecube
target dclock 5000 32768 swoff /* Sets target clocks */
target sfr wdtm2=0x0 /* Stops Watchdog timer reset*/
target pinmask reset
target sfr vswc=0x1
/* If you set following, download speed becomes faster */
target sfr prcmd=0x0
target sfr pcc=0x0
target sfr pllctl=0x3
```

<<In the case of IECUBE2>>

```
SERVERTIMEOUT=500
connect 850eserv2 -iecube -e2
target dclock 5000 32768 swoff /* Sets target clocks */
target opbyte 1 0x12345678     /* Sets Flash Option area */
load
target sfr vswc=0x1           /* Sets SFR setting after download */
```

Each target commands are described in "Chapter 13 Target Commands" on page 55. Since these commands are executed prior to program downloading, it is generally pointless to see and modify what is in memory or register (it could lead to a guard area access error).

Please be sure to set up target's operating frequencies by DCLOCK command or the Connection Editor window at the time of initialization.

SERVERTIMEOUT

SERVERTIMEOUT is a MULTI system variable and used to set the timeout for server connection. Since MULTI's server connection time is short by default, you may not connect to the server without setting this system variable. You do not need to change the above setting when using.

Starting 850eserv2

850eserv2 is started by **connect** command. As for details please refer to "connect command" on page 17.

Target command

If you want to execute certain ICE commands each time MULTI debugger starts, you can write them in the resource file as shown in the example above. This allows these commands to be executed automatically when MULTI debugger starts. Thus, you do not need to enter settings of emulation execution mode, memory map, and so on from the TARGET pane each time MULTI starts.

MULTI setup script

An Other Method of initialization target or emulator is to use MULTI setup script. MULTI setup script is `.mbs` extentsion file, you write initialization target or emulator in this file. This file is loaded automatically when MULTI downloads target program.

Chapter 7
I/O Pane

The I/O pane allows you to perform the standard input-output process for the user program. It appears in the area below the source pane in the main Debugger window when the I/O tab is selected.

However, to use the I/O pane, it is necessary to declare a special section (`.syscall`) to the linker's section map file. Since the `.syscall` section includes program code, specify the address of this section to be treated as same as the .text section.

```
.text 0x10000 :
.syscall :
```

Declare `.syscall` after .text as shown above. The `.syscall` section is required to use the I/O pane. Green Hills Software's internal library uses this section to perform the standard input-output process for the I/O pane. Therefore, you need to link Green Hills Software's library as shown below, as well as adding the `.syscall` section.

```
libansi.a libind.a libstartup.lib libsys.lib libarch.a libmalit.a
```

Chapter 8
Target Pane

The TARGET pane allows you to exchange commands directly with ICE. It appears in the area below the source pane in the main Debugger Window when the Trg tab is selected. Commands you can use on this pane are called TARGET commands.
As for explanation of target command, please refer to "Chapter 13 Target Commands" on page 55.

When you use MULTI as line mode, you can use following command with “target” prefix. For example, On Target pane:

```
m -d1 0x100000
```

The following command is as a debugger command of MULTI in line mode (or screen mode):

```
target m -d1 0x100000
```

Both are the same meaning of command. A capital letter and a small letter are not distinguished in a target command part.

Chapter 9
Trace Analysing

This chapter describes the following items.

- Use SuperTrace Probe
- Use QB-V850E2-SP
- Enable or Disable trace
- Launching TimeMachine
- Trace Trigger setting
- Trace Configurations

850eserv2 supports Trace analysing. In V850E1Core, Trace analysing can be used with IECUBE. In V850E2Core, Trace analysing can be used with IECUBE2. In RH850, Trace analysing can be used with E1/E20 emulator or IECUBE2.
850eserv2 supports Timemachine debugging by collected trace datas.

Use SuperTrace Probe

You can connect to SuperTrace Probe by GreenHills with the STP bourd mounted IECUBE. SuperTrace Probe can capture up to max 1 GB of trace data and analyze its.

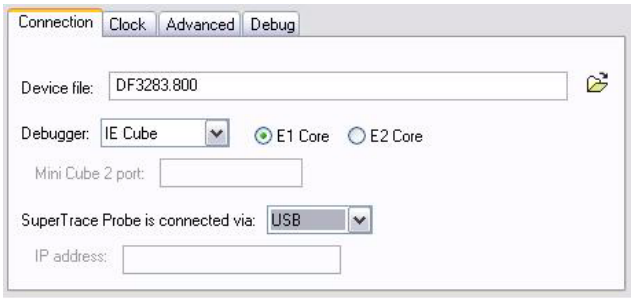
Setting of SuperTrace Probe

You should set the initialization of SuperTrace Probe. If you connect to SuperTrace Probe via LAN, you should set the IP address in SuperTrace Probe and connect to network. If you connect to SuperTrace Probe via USB, you should install the USB driver into your computer and connect SuperTrace Probe and your computer by USB cable. You might need to update the latest firmware of SuperTrace Probe.
Please refer to SuperTrace Probe manual for detail of SuperTrace Probe initialization.

Connects to SuperTrace Probe

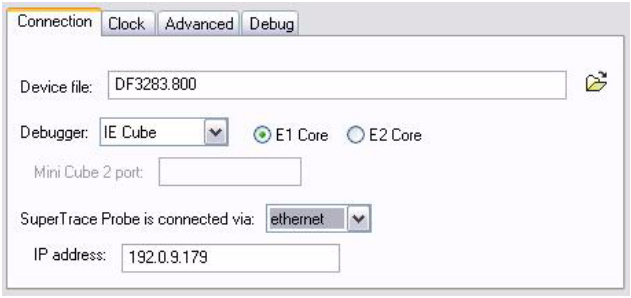
850eserv2 connects to SuperTrace Probe via USB or LAN. Selects connection method to SuperTrace Probe via Connection Oganizer. For the detail of Connection Oganizer, please refer to "Connects via Connection Oganizer" on page 12.

Connects via USB



Selects "USB" in **SuperTrace Probe is connected via:**.

Connects via LAN

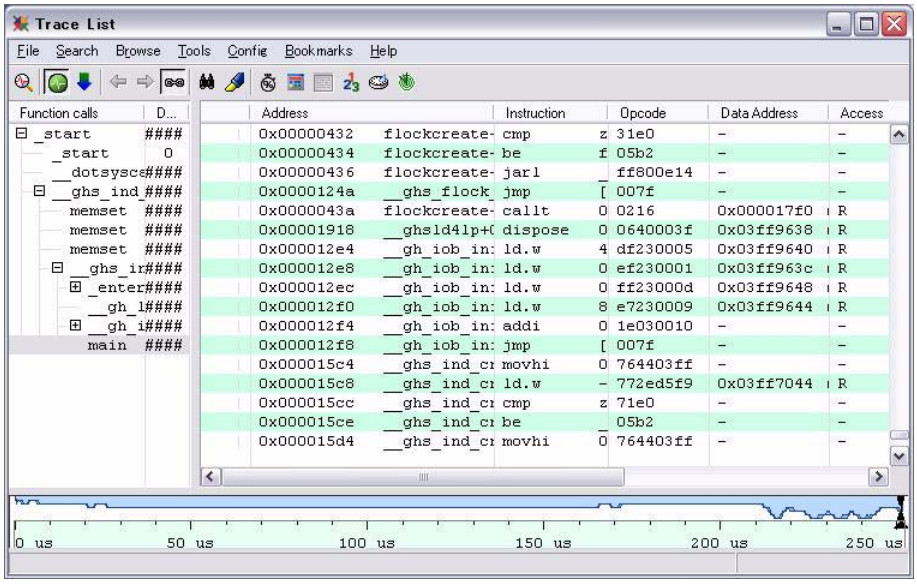


Selects "ethernet" in **SuperTrace Probe is connected via:** and inputs SuperTrace Probe's IP address in **IP address:**.

Use QB-V850E2-SP

You can connect to QB-V850E2-SP by Renesas Electronics with IECUBE2. QB-V850E2-SP can capture up to max 2.25 GB of trace data and analyze its.
There is no special setting for using QB-V850E2-SP, you can use QB-V850E2-SP immediately when your IECUBE2 connects to QB-V850E2-SP.
Please refer to QB-V850E2-SP manual for detail of QB-V850E2-SP.

Enable or Disable trace



Opens Trace List window when you select "TimeMachine > Trace List" in MULTI debugger menu.

 is Enable or Disable trace button, 850eserv2 collects trace data after pushing this button and executing program. Trace is disabled if this button is pushed again.

Trace data is displayed in Trace List window when trace is disabled or  button is pushed.

 This is the description in MULTI V5. In case of MULTI V4, Trace List window is opened "Tools > Trace > Trace List" in MULTI debugger menu.

Launching TimeMachine

TimeMachine is launched when  is pushed after 850eserv2 collects trace data. TimeMachine can reverse executing or stepping in range of trace data.

Please refer to "MULTI: Debugging" for detail of TraceList window or TimeMachine.

Trace Trigger setting

You can set the trace collecting conditions by Trace Triggers. Trace Triggers can be set via "Set Triggers" dialog. "Set Triggers" dialog can be opend via "Config > Set Triggers" in Trace List menu.

It describes for settable triggers in 850eserv2. Please refer to "MULTI: Debugging" for detail of Set Triggers dialog.

`Start Trace` Sets trace start condition.
It can not be set when DMA trace is enable.

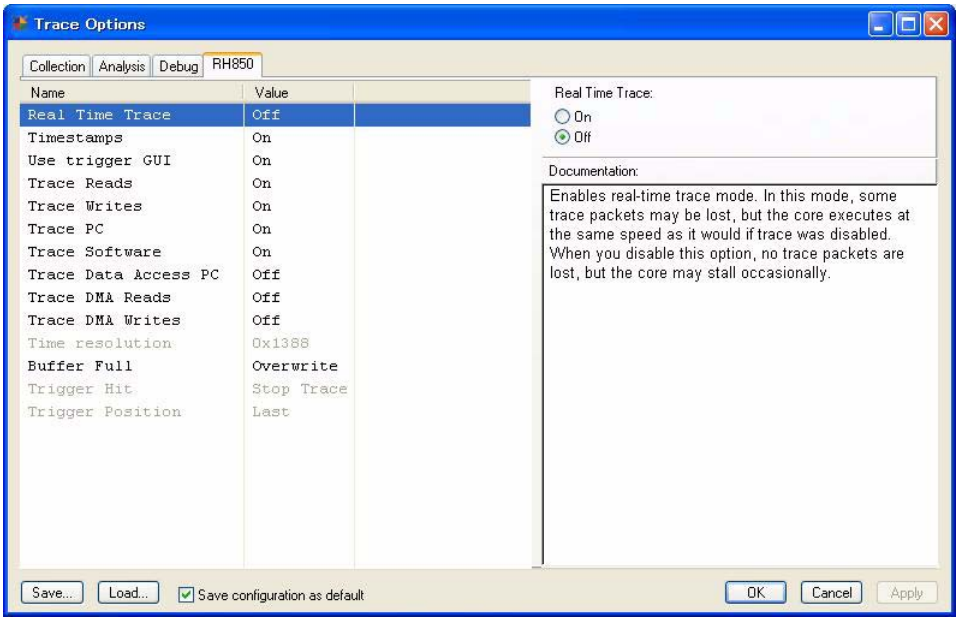
`Stop Trace` Sets trace stop condition.
It can not be set when DMA trace is enable.

`Qualify Trace` Sets trace collecting condition.
It can not be set when DMA trace is enable.

`Delay Trace` Sets Delay Trigger/
Trace is stopped when it gets a certain number of trace frame after hits delay trigger.
You sets the gettable number of trace frame and the behavior when hits delay trigger via the Target specific configurations.
It can not be set with QB-V850-SP.

Trace Configurations

You can set the trace configurations by the "IE Cube", "IE Cube2" or "RH850" tabs in "Trace Options" dialog. "Trace Options" dialog can be opend via "Config > Options" in Trace List menu.



It describes for settable configurations in 850eserv2. There are items that are not displayed by the environment. Please refer to "MULTI: Debugging" for detail of Trace Options dialog and other configurations.

Real Time Trace	Enables real-time trace mode. In this mode, some trace packets may be lost, but the core executes at the same speed as it would if trace was disabled. When you disable this option, no trace packets are lost, but the core may stall occasionally.
Timestamps	Enables timestamps. When timestamps are enabled, the trace collection device records a timestamp with each packet.
Use trigger GUI	Enables the trigger GUI.
Trace Read	Enables data read trace packets for memory accesses.
Trace Write	Enables data write trace packets for memory accesses.
Trace PC	Enables branch program counter (PC) trace packets.

Trace Software	Enables software trace packets. It can be enabled with RH850.
Trace Data Access PC	Enables PC trace packets for memory accesses. It can be enabled with RH850.
Trace DMA Read	Enables DMA read trace packets.
Trace DMA Write	Enables DMA write trace packets.
Buffer Full	Specifies the behavior when the trace buffer fills. You can choose to overwrite old trace data with new data, stop trace collection, or halt the process. **Stop Execution** can not be chosen with QB-V850E2-SP. In RH850, **Stop Trace** can not be chosen with **Real Time Trace** is **Off**.
Trigger Hit	Specifies the behavior when hits delay trigger. You can choose to stop trace collection or halt the process when it gets a certain number of trace frame after hits delay trigger. This option is only available if a Delay Trace trigger has been set up. In RH850, **Stop Trace** can not be chosen with **Real Time Trace** is **Off**. In RH850+E1/E20 emulator, **Stop Execution** can not be chosen.
Trigger Position	Specifies the desired position of the Delay Trace trigger in the trace buffer, after trace has stopped. **Last** means trace will stop very soon after hitting the trigger (so the trigger will end up being near to the last thing in the trace buffer). **First** means trace will continue to be collected until the trigger is near the beginning of the buffer. **Middle** means trace continues to be collected until half of the buffer has been filled. This option is only available if a Delay Trace trigger has been set up.
Time resolution	Specifies the time resolution of trace packet timestamps. This field defaults to 5000 picoseconds when timestamps are enabled. The value that multiplying trace clocks of trace frame(Cycles field on TraceList) and specified time resolution value is the execution time of trace frame(Time field on TraceList). It can be enabled with **Timestamps** is **On**. In multicore-device, you can specified each core's time resolution.

You obtainable the specified time resolution value for next calculating.

<<In case of V850E2Core>>

```
Time resolution = (1/(device frequency)*1000000)*dividing rate*trace edge
```

<<In case of RH850>>

```
Time resolution = (1/(device frequency)*1000000)
```

Chapter 10
Hot-plugin debugging

This chapter describes the following items.

- Launch 850eserv2 in Hot-plugin mode
- Attach to target program

Hot-plugin debugging is the function that attaches running target program and debugs it. You can do Hot-plugin debugging only OCD emurator+V850E2Core and it needs Hot-plugin adapter by Renesas Electronics. This chapter describes the procedure of Hot-plugin debugging.

Launch 850eserv2 in Hot-plugin mode

Launch 850eserv2

It needs to launch Hot-plugin mode in 850eserv2 when you do Hot-plugin debugging. Adds "-hotplugin" option to starting **connect** command, or enables "Hot plugin Debugging:" in Connection Oganizer.

```
<< In case of launch by connect command>>
   connect 850eserv2 -minicube -e2 -df=DF3512.800 -hotplugin
```

Separates ICE and target connecting when 850eserv2 launches in Hot-plugin mode. If 850eserv2 detects target power ON, the following error message is displayed in MULTI debugger command pane and 850eserv2 aborts launching.

```
   "-hotplugin" Error: Target has already been connected with ICE
```

Connect to Hot-plugin adapter

When this dialog box is opened, connects ICE and target by Hot-plugin adapter and pushes "OK" button in this dialog box. The following message is displayed in MULTI debugger command pane when the connection succeeds.

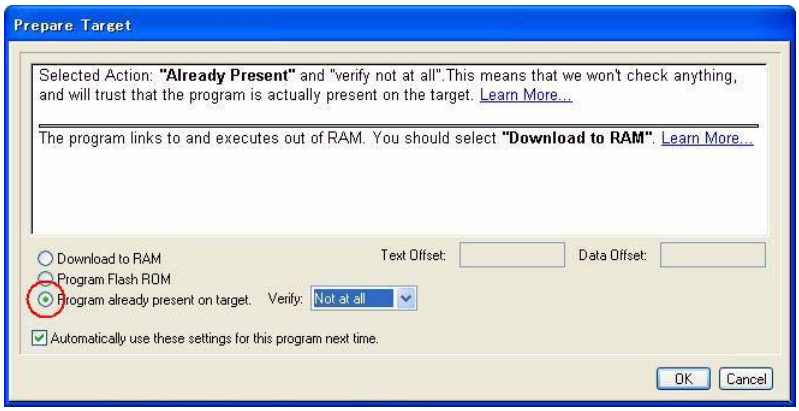
```
   850eserv2 launches in Hot-Plugin mode.
   Please select 'Program already present on target'.
```

If 850eserv2 could not detect target power ON after pushing "OK" button, the following error message is displayed in MULTI debugger command pane and 850eserv2 aborts launching.

```
   "-hotplugin" Error: Target power was not detected
```

Attach to target program

After 850eserv2 launched in Hot-plugin mode, "Prepare Target" window is opened via "Debug > Prepare Target" in MULTI debugger menu. Please refer to MULTI debugger's manual for detail of "Prepare Target" window.



Selects "Program already present on target" to attach to target program. MULTI compares the ELF file and contents on target memory when you select "Sparsely" or "Completely" in "Verify:"

Sparsely	Compares a few areas in each sections. MULTI halts target program temporarily when comparing.
Completely	Compares all areas in each sections. MULTI halts target program temporarily when comparing.
Not at all	Not Compare. MULTI does not halt target program.

When you select "Sparsely" or "Completely", MULTI does the process that halts target program > reads memory and compares > runs target program internally. If you do not want to halt target program temporarily when attach to target program, selects "Not at all".

When you push "OK" button in this window, 850eserv2 attaches to target program and the status of MULTI debugger is Running.

You can not download the program to target in Hot-plugin debugging. If you download via **load** command, **c** command or "Download RAM" in "Prepare Target" window, the following error message is displayed in MULTI debugger command pane and downloading is not done.

```
Could not download while Hot-Plugin mode.
Please select 'Program already present on target'.
```

Chapter 11
Multi-core debugging

This chapter describes the following items.

- Multi-core debugging

Multi-core debugging

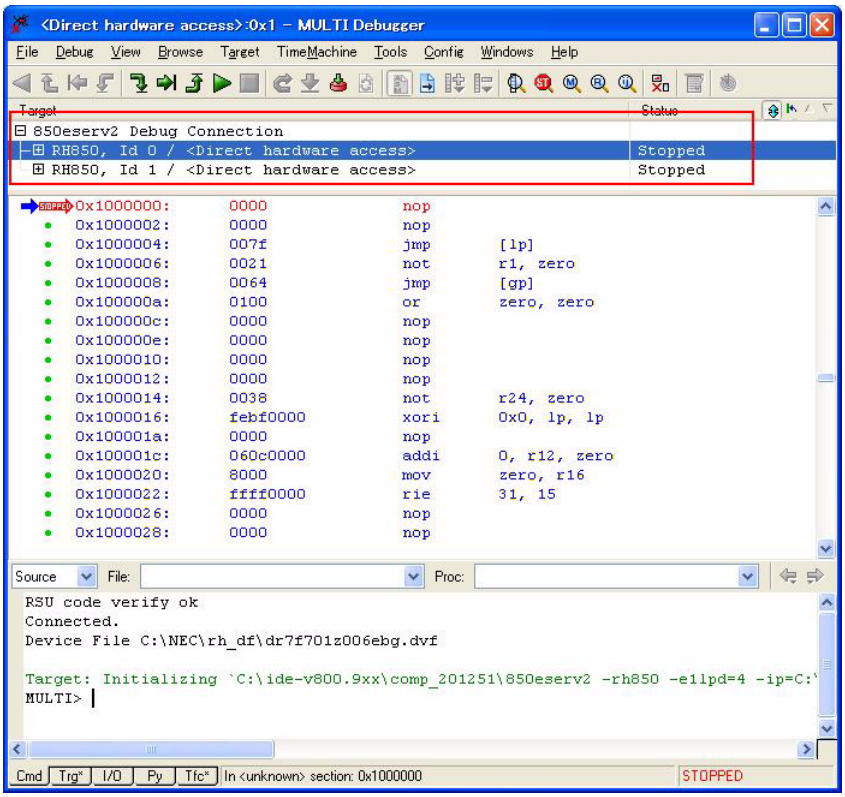
850eserv2 supports Multi-core debugging with combination of E1/20 Emulator and Multi-core device. This chapter describes the Multi-core debugging.

Launch 850eserv2

For the launching 850eserv2, please refer to "Chapter 5 Starting 850eserv2" on page 11.
This chapter describes assuming that it connects to 850eserv2 via method of "Connects via Connection Oganizer" on page 12.

In RH850, Selects "E1 Emulator LPD" in **`Debugger:`** and "RH850" in Connection Editor window. And Specifies 1 or 4 by number of pins used LPD connecting in **`LPD pin:`**.

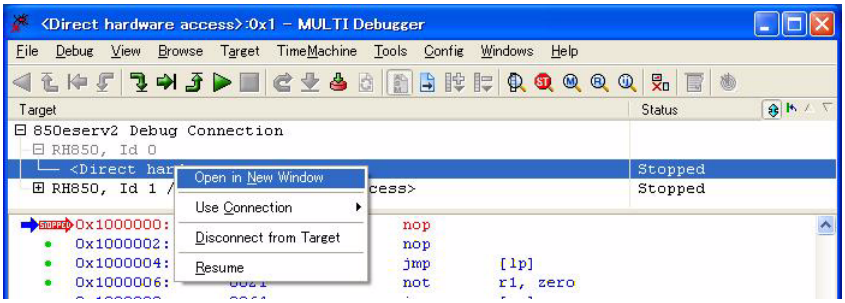
When it connects to 850eserv2 via MULTI Project Manager, MULTI debugger is opened automatically.



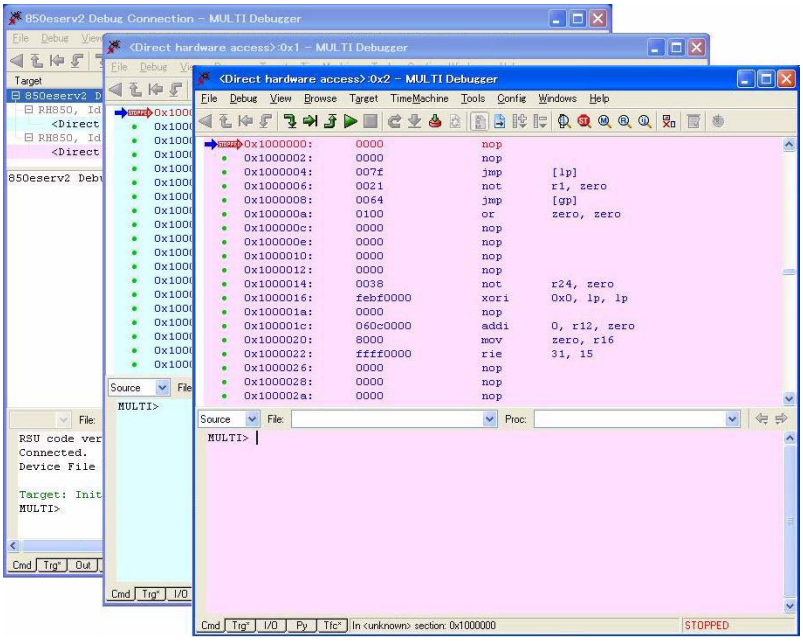
Red pane in this image in top of MULTI debugger is "Target List". Target List displays any connected cores. For the detail of Target List, please refer to "Target List" on page 49.

Target List

Target List displays now connected cores. MULTI debugger debugs to selected core in Target List. When you change other core in Target List, Debug window in MULTI debugger changes to display selected core.

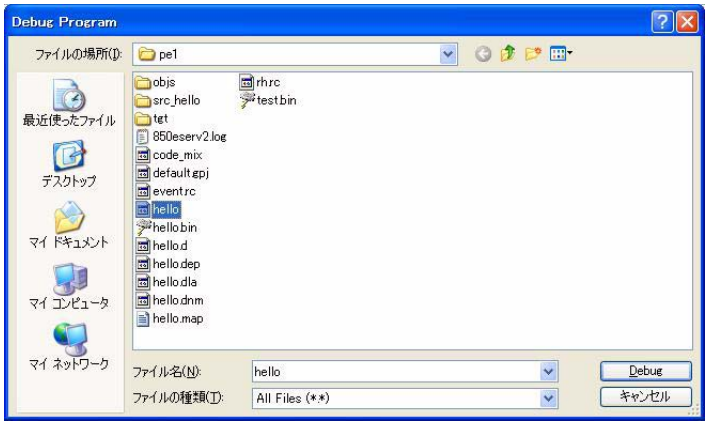


When you select one core in Target List and select "Open in New Window" via right-click, New window assigned selected core is opened.

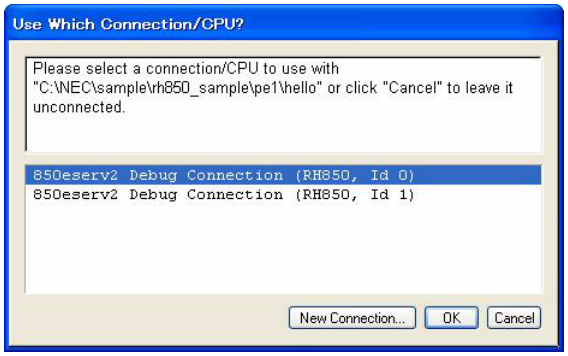


Load executable image to core

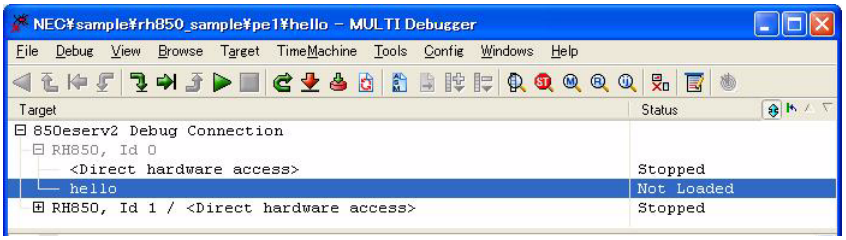
It is necessary to assign an execution image to core. When you select "File > Debug Program" in Debugger menu, the following window is opened.



Selects an executable image file and push "Debug" button.
"Use Which Connection/CPU" window is opened and displayed each cores now connected.



An executable image is assigned to selected core via this window.
An executable image name is displayed in Target List.

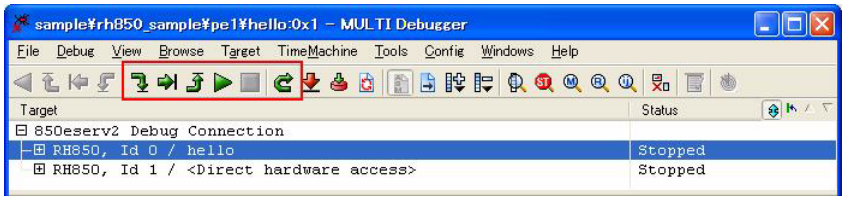


When push  button on Debugger's toolber or input **`load`** command via command pane in bottom of Debugger after executable image is assigned to core, executable image is downloaded to core.

It is necessary to assign and download execution images to all cores.

Start debugging

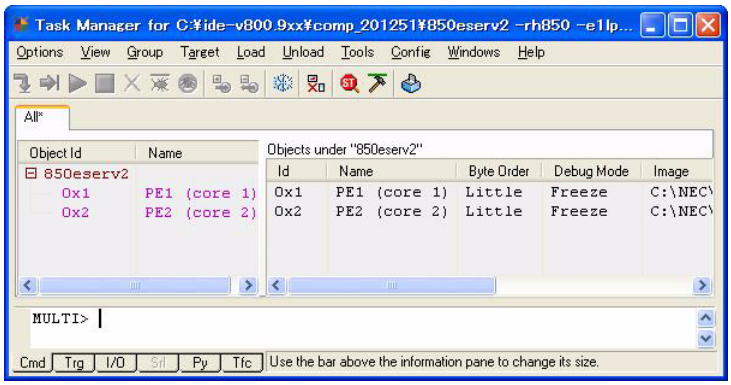
Each buttons(Red pane in following image) can handle each core's run control.



Each cores are synchronized. When one core is run, other core's statuses become to "Running(Frozen on core x)" and the target is not running yet. When all cores are run, all core's statuses become to "Running" and the target is actually running.
When one core is halted or hit on breakpoint, this core's status becomes to "Stopped" and other core's statuses become to "Running(Frozen on core x)".
When one core is reset, other each cores are reset.

Task Manager

When you select "View > Task Manager" in Debugger menu, "Task Manager" window is opened.



Each connected cores are displayed in left pane in Task Manager. The number of "Object Id" is ID that MULTI assignes to each cores internally. "Name" displays each core's name. The right pane displays the detail of each core's status.
MULTI debugger command or 850eserv2 target command can not be issued in Task Manager.
When you click one core in list, each buttons on toolber become to enable and can handle selected core's run control.
If you select "Group → Continue Tasks in Current Group" in menu, all cores are run.

Event setting

You can set events in resource that each cores have. An event you set in one core can be used in only this core. It can not be used in other core. An event set in other core can not be referred.

When you set an event via target command, you should input target command via target pane after select core which sets an event in Target List.
When you set an event via 850win, you can change core via "Select Core:" in Main window(850eserv2 Configuration) or pulldown menu in top of Event List window.

Chapter 12
The attestation code

This chapter describes the following items.

- About the attestation ID code

About the attestation ID code

When connecting with the target device which mounts RSU(ROM Security Unit) from 850eserv2, it is necessary to specify 20(V850E1 core), 24(V850E2 core) or 32(RH850) digits attestation ID code as "-id" option for security unlock.
Please refer to "Server Starting Options" on page 18 about the details of "-id" option.
ID code is specified by 20 or 24 digits hexadecimal without "0x" prefix. When you connect to 850eserv2 via MULTI command pane, you need to attach "-id" option and specify ID code after "-id". When you connect to 850eserv2 via Connection Editor window, you need to specify ID code into "Registry ID Code:" in Connection Editor window.

850eserv2 compare the specified ID code and the attestation code stored in the target device. In V850E1 core, the attestation code is stored in 0x70-0x79 address of Code Flash area. In V850E2 core and RH850, the attestation code is stored in the area of attestation ID in Flash Option.
If the specified ID code matches to the attestation code stored in the target device, 850eserv2 can be started.
When it succeeds to start 850eserv2, 850eserv2 saves specified ID code into registry. When 850eserv2 is started for the next time, you can omit ID code specifying. If ID code is omitted, 850eserv2 loads ID code from registry and attests by this ID code.
If the specified ID code and the attestation code stored in the target device are different, following error message is dispayed in MULTI command pane and connection fails:

```
RSU code verify error
```

In case of MINICUBE2 or E1/E20 emulator(Serial), When the monitor cannot be correctly operated, The EXEC library compulsorily rewrites all codes to "f" and releases RSU.

It is operation of the download, memory access, etc., and when writing in to a internal flash memory, cautions are required in not overwriting this attestation code domain (0x70 - 0x79) accidentally.
850eserv2 display the following warning, when there is writing to this domain.

```
WARNING : RSU id-code (0x70-0x79) has changed
New id-code is xxxxxxxxxxxxxxxxxxxx
```

The highest bit in ID code(0x79 bit 7) is defined as a use permission flag of a OCD emulator or E1/E20 emulator.
Even if specified ID code is in agreement, 850eserv2 cannot be started when the this bit is "0".
It cannot set this bit to "0" in 850eserv2.
For the detail of RSU, please refer to each emulator or each target device manuals.

Chapter 13
Target Commands

This chapter describes the following items.

- Command List Common to Server
- TARGET Commands Common to Servers
- Command List 850eserv2 Peculiar
- Target Commands 850eserv2 Peculiar
- 850eserv2 Scripts
- Expressions
- Assignment
- Keyword
- Conditions
- Loop
- Extension of Variables
- Script Examples

Since most of 850eserv2's **TARGET** commands are categorized by item, they need several parameters. If the number of parameters is larger or smaller than needed, an error will result. The item names themselves are available as help commands. A target command has commands peculiar to 850eserv2, and commands common to server. As for command common to server, it can use even with the server of other companies make ICE correspondence other than 850eserv2.

Command List Common to Server

These commands can be executed from a Target pane.

Command	Description
ADDRESSOF	Get address of symbols
AMASK	Change of the location to download
BREAK	Set Software breakpoints
BLOCKFILL	Memory filling
CLOSE	Close of file descriptor
DELBREAK	Delete Software breakpoints
DELRANGE	Delete access prohibition domain registration
DELRANGEALL	Delete all access prohibition domain registration
ECHO	Display command in script-file
FPRINT	Write character sequence and numerical value
FPRINTB	Write character sequence and numerical value (binary)
FREAD	Read character sequence and numerical value
FREADB	Read character sequence and numerical value (binary)
GETENV	Get Environment variable
HALT	halting
LISTRANGE	Display access prohibition domain registration
LISTVARS	Display script variable

Command	Description
LOAD	Specification of section to download
M	Memory access
MEMTEST	Memory check
NOFAIL	Normal end of commands
NOLOAD	Specification of section which is not downloaded
OPEN	Open of file descriptor
PRINT	Display character sequence and numerical value
RANDOM	Get random value
REG	Register access
REGNUM	Register access (register number specification)
RST	Reset target CPU
SCRIPT	Execution of a script file
SETRANGE	Set access prohibition domain registration
SETUP	Script execution before download
SLEEP	Sleep of the appointed time
STATUS	Display status
STEP	Single step
SYSCALLS	Control of Break to .syscall Section
UNDEF	Delete variable

TARGET Commands Common to Servers

ADDRESSOF

Syntax:

```
addressOf <symbol>
```

Description:

Obtains an address for a specified symbol. The obtained address will be returned as a return value.

<symbol>	Symbol name

Example:

```
address = addressOf foo
print $address
```

AMASK

Syntax:

```
amask <mask> <value>
```

Description:

Changes the location for downloading a program. The address of an actual download location is as follows:

```
<load address> = (<original address> & <mask>) | <value>
```

<mask>	Mask pattern
<value>	Logical sum pattern

BREAK

Syntax:

```
break <address>
```

Description:

Sets software breakpoint. Breakpoints set by this command are not recognized by MULTI debugger. To delete breakpoints, use the **delbreak** Target command.

<address>	Address

BLOCKFILL

Syntax:

```
blockfill [-d1|-d2|-d4] <address> <count> <value>
```

Description:

Fills a specified memory area with a specified value.

-d1 | -d2 | -d4	Access size (1, 2 or 4 bytes); 4 bytes by default
<address>	Start address
<count>	Number of counts
<value>	Value to be filled

CLOSE

Syntax:

```
close <fd>
```

Description:

Closes a specified file descriptor. Usually, assign a return value of the **open** command to a variable; specify this variable as **<fd>**.

<fd>	File descriptor

Example:

```
fp = open test.txt
close fp
```

DELBREAK

Syntax:

```
delbreak <address>
```

Description:

Deletes software breakpoints. Breakpoints deleted by this command are not be recognized by MULTI debugger. This command is used to delete breakpoints that were set by the **break** Target command.

<address>	Address

DELRANGE

Syntax:

```
delbreak <address>
```

Description:

Delete access prohibition domain registrations. The top address of the address range set up by the **setrange** command is specified to be **<address>**.

<address>	Address

DELRANGEALL

Syntax:

```
delbreak
```

Description:

Delete all access prohibition domain registrations.

ECHO

Syntax:

```
echo [on|off]
```

Description:

Selects whether or not to display commands executed by a script. If you do not specify a parameter, the current settings will be selected.

on	Displays executed commands
off	Does not display executed commands

FPRINT

Syntax:

```
fprint <fd> <string>
```

Description:

Writes string or variable values to an ASCII format file.

<fd>	File descriptor (Normally specify a return value of the open command)
<string>	String or variable

FPRINTB

Syntax:

```
fprintb <fd> <integer>
```

Description:

Writes string or variable values to a binary format file.

<fd>	File descriptor (Normally specify a return value of the open command)
<integer>	String or variable

FREAD

Syntax:

```
fread <fd> <identifier>
```

Description:

Reads an ASCII string from a specified file and assigns it to a variable. As a return value, the readout byte count is returned. If an error occurs, **-1** will be returned.

<fd>	File descriptor (Normally specify a return value of the open command)
<identifier>	Variable for assigning a value

FREADB

Syntax:

```
freadb <fd> <identifier>
```

Description:

Reads a value in binary format from a specified file and assigns it to a variable. As a return value, the readout byte count is returned. If an error occurs, **-1** will be returned.

<fd>	File descriptor (Normally specify a return value of the open command)
<identifier>	Variable for assigning a value

GETENV

Syntax:

```
getenv <envName> <identifier>
```

Description:

Assigns the setting of a specified environment variable to a variable specified by **<identifier>**.

<envName>	Environment variable name
<identifier>	Variable for assigning a value

HALT

Syntax:

```
halt
```

Description:

Forcefully stops the target CPU.

LISTRANGE

Syntax:

```
listrange
```

Description:

Display access prohibition domain registrations.

LISTVARS

Syntax:

```
listvars
```

Description:

Lists all variables used in scripts.

Example:

```
str1=”foo”
i=100
listvars
i
str1
```

LOAD

Syntax:

```
load [text|data|bss|all]
```

Description:

Specifies sections to be downloaded to the target. In combination with the **noload** command, you can specify only certain sections to be downloaded.

text	Program code section (All sections downloaded to ROM area)
data	Variable section with an initial value (.data, .sdata, .zdata, etc.)
bss	Variable section without an initial value (.bss, .sbss, etc.)
all	All sections

M

Syntax:

```
m [-d1|-d2|-d4] <address>[=<val>]
```

Description:

Reads or writes data from or to memory at a specified address.

-d1|**-d2**|**-d4**	Access size (1, 2 or 4 bytes); 4 bytes by default
<address>	Address (in hexadecimal; **0x** not added to the beginning)
<val>	Data to be written

Example:

```
m 1000
    7ca62b78
m 1000=12345678
m -d2 1000
    1234
```

MEMTEST

Syntax:

```
memtest <start> <end>
```

Description:

Checks, on a byte-by-byte basis, if a specified memory area is available for **read**/**write**. If you specify a large memory space, it may take long time to finish checking.

<start>	Start address
<end>	End address

NOFAIL

Syntax:

```
nofail <command>
```

Description:

Executes a specified command and always returns Normal Termination as the execution results.

<command>	Command and command parameter

Example:

```
ret=nofail m -d4 $address
```

NOLOAD

Syntax:

```
noload [text|data|bss|all]
```

Description:

Specifies sections not to be downloaded. It acts in a contrary manner to the **load** command.

text	Program code section (All sections downloaded to ROM area)
data	Variable section with an initial value (.data, .sdata, .zdata, etc.)
bss	Variable section without an initial value (.bss, .sbss, etc.)
all	All sections

This command is especially effective when an object exists on the flash memory, because it suppresses downloading to the flash memory.
The disassembly of program code in the Debugger window in MULTI is done by reading data from the executable file, not from the program being debugged. Therefore, the program code section is displayed to be downloaded when downloading it after **text** is specified.The noload command is reflected in the debugger window by specifying MULTI system variable **_ASMCACHE** is 0. Please refer to "MULTI: Debugging" for MULTI system variable.

OPEN

Syntax:

```
open <filename>
```

Description:

Opens a specified file in write mode and return its file descriptor. To **close** the file, use the close command.

<filename>	File name

Example:

```
fp = open test.txt
fprint fp aaa
close fp
```

PRINT

Syntax:

```
print <string>
```

Description:

Displays a specified string or variable value.

<string>	String or variable

RANDOM

Syntax:

```
random [<max>]
```

Description:

Generate random value of from 0 to **<max>**-1.

<max>	Max value

REG

Syntax:

```
reg [<regname> [<val>]]
```

Description:

Reads or writes data from or to a specified register. If you do not specify a parameter, all register names and their values will appear. If you specify a register name, only that register will appear. If you specify a register value, the register will be set to that value.

Example:

```
reg r0
reg r0 1
```

REGNUM

Syntax:

```
regnum <regname>[=<val>]
```

Description:

Reads or writes data from or to the register with a specified register number. If you do not know the register number, use the **reg** command.

RST

Syntax:

```
rst [stop|run]
```

Description:

Resets the target.

stop	After the reset, stops the CPU.
run	After the reset, runs the CPU freely (If you use this during the debug, MULTI's status could become inconsistent with the actual CPU status).

SCRIPT

Syntax:

```
script <filename>
```

Description:

Executes a group of commands included in a specified script file.

<filename>	Script file name

Example:

```
script crc32.scr
```

SETRANGE

Syntax:

```
setrange <address> <length>
```

Description:

The access prohibition domain by debugger is registered. The domain from the address specified by **<address>** to **<length>** is made prohibition of access. It is operation of refer to the memory from debugger that access is set as the object of prohibition, and it cannot make this domain prohibition of access from the user program currently performed.

<address>	Address
<length>	Length of access prohibition domain registrations.

SETUP

Syntax:

```
setup <filename>
```

Description:

Executes a specified script file before downloading a user program .

<filename>	Script file name

SLEEP

Syntax:

```
sleep <seconds>
```

Description:

Places the server into sleep mode for an internally specified period of time. You can use this command when, for example, you need to suspend script file processing for a certain period.

<seconds>	Number of seconds

STATUS

Syntax:

```
status
```

Description:

Displays the status and return its value. The return value corresponds to the following status types

In process

Break (software breakpoint, hardware breakpoint)

Single step

Target halted

Target running

STEP

Syntax:

```
step
```

Description:

Executes a single machine-language step from a PC-pointed address.

SYSCALLS

Syntax:

```
syscalls [on|off]
```

Description:

When .syscall section exists in the object to download, MULTI sets up an internal software break point at the head of .syscall section at the time of download. This command controls a setup of the break point. The function of printf() etc. cannot be used when not setting a break point to a .syscall section. Please refer to the "MULTI: Building Applications for Embedded V800" for the details of the .syscall section.

When you measure the execution time of the section which contained the function of printf() etc. by the **TIMER** command, please set **SYSCALLS** command as **OFF**. Since an internal break point setup for an input and output function is deterred by this setup, the clear of the timer by the break which a user does not mean is prevented, and exact execution time measurement is attained by it.

UNDEF

Syntax:

```
undef <viriable>
```

Description:

Deletes a specified variable and frees up memory space allocated to this variable.

<viriable>	Variable name

Example:

```
x=5
undef x
print $x
Error: Variable undefined:
```

Command List 850eserv2 Peculiar

You can enter all of these commands directly into the 850eserv2 Target pane. You can also enter these commands into the MULTI Debugger command pane using the target command.
The code in ( ) is Alias.

Data and Register Commands

Command	Description
ASSEMBLE(A)	Assembles code one line at a time
PIO	Displays or changes the Programmable P/O values
PIOBASE	Appoints the based address of the Programmable I/O
SFR	Displays or changes the SFR, Extended External I/O or Programmable I/O values
UNASSEMBLE(U)	Disassembles object code

Event, Break Setting Commands

Command	Description
BRA	Sets bus event detectors
BRS	Sets execution event detectors
FBREAK	Sets Software break setting mode in Flash memory.
FLSF	Sets and Displays Fail-safe-break
HWBRK(B)	Causes a break
LINK	Sets sequential events
PB	Sets and Displays Peripheral break
SHOWALL(SA)	Displays current trace analyzer settings

Timer Command

Command	Description
TIMER	Shows executed clocks

Command	Description
TMEVENT	Sets and Displays Timer events

Trace Commands

Command	Description
PROFILE	Gets or Sends to MULTI Profile Data
TCLEAR	Clears trace buffer
TDISPLAY(TD)	Displays trace buffer
TFILTER(TF)	Specifies trace buffer filtering
TIMEBASE	Sets or displays the frequency of the CPU clock and the scalar value of the trace time-tag counter
TMODE	Specifies trace mode
TRACE	Specifies conditions for tracing
TRUN	Restarts the trace analyzer
TSEARCH	Searches trace buffer
TSIZE	Specifies trace buffer size
TSTOP(TS)	Halts the trace analyzer
TTIMER	Displays trace clocks of TraceList

Emulator Configuration Commands

Command	Description
CACHE	Sets and displays CPU cache configuration
CPU	Displays or changes the internal ROM/RAM size
DCLOCK	Sets and displays the target's minimum operating frequencies
HSPLOAD	Downloads at high speed
ILLOPBK	Sets operational mode when the illegal opcode exception occurs

Command	Description
LOADOPT	Sets download option
MAP	Sets the emulator memory configuration
MODE	Sets the CPU mode
PINMASK(PIN)	Sets the masked processor pins

Flash Command

Command	Description
BLANKCHECK	Checks blank datas in Data Flash memory
DFDUMP	Dumps to Data Flash memory
DFLASH	Sets the emulation of Data Flash memory
DFLASHERR	Sets the error emulation of Data Flash memory
DFMAP	Maps the Data Flash area
DFSAVE	Saves to ID tag format from Data Flash memory
EFCONFIG	Registers external flash memory information.
FERASE	Erases flash memory blocks
FLASH	Shows CPU Flash memory information
FLASHRESET	Sets reset vector address
FLASHSELF	Sets Flash Self Emulation function
FLOAD	Downloads to Flash memory
FMACROERR	Sets the error emulation of Flash Macro Service
FMACROERRSC	Sets Status Check error emulation area by Flash Macro Service
FSECFLAG	Sets the security flag that can set by Flash-Writer
FSHIELDWINDOW	Sets flash memory block that can be written
IDCODE	Sets attestation ID code for unlock the RSU(ROM Security Unit)

Command	Description
IDTAG	Writes ID-tag to Data Flash memory
LOCKBIT	Sets flash memory block that can not be written
OPBYTE	Sets Flash Option area.

Realtime RAM Command

Command	Description
RMEM	Displays the contents of the real-time RAM periodically
RRAMBASE	Sets or displays the base address of the real-time RAM area
SW	Switchs trace mode or realtime RAM mode

Other Commands

Command	Description
HELP	Displays command summary
VERIFY	Turns memory verify on or off
VERSION	Shows version of 850eserv2

Target Commands 850eserv2 Peculiar

Target command 850eserv2 peculiar is explained from the following page.

ASSEMBLE

ASSEMBLE(A) <***address***> <***code***>

address	Specify target address for assebmles.
code	Specify assemble code.

This command is not available for V850E2Core and RH850.
The specified target address assembles. It becomes an error when there is no specification of a target address and the code which assembles.
Symbolic constants can be used when specifying either the address or operands in the source line.
The operators **sdaoff**, **zdaoff**, **hi**, and **lo** are supported. The name **ZERO** is treated as register **RO**.

Example:

```
850eserv2>assemble 0x13a st.w r18,-0x8000[zero]
0x00013A: 97608001 st.w r18, -32768[zero]
850eserv2>assemble 0x140 add -4,sp
0x000140: 1a5c add -4, sp
850eserv2>
```

BLANKCHECK

BLANKCHECK [on|off]

on	Enables blank checking
off	Disables blank checking(Default)
auto	Switches automatic blank checking
<none>	displays now setting

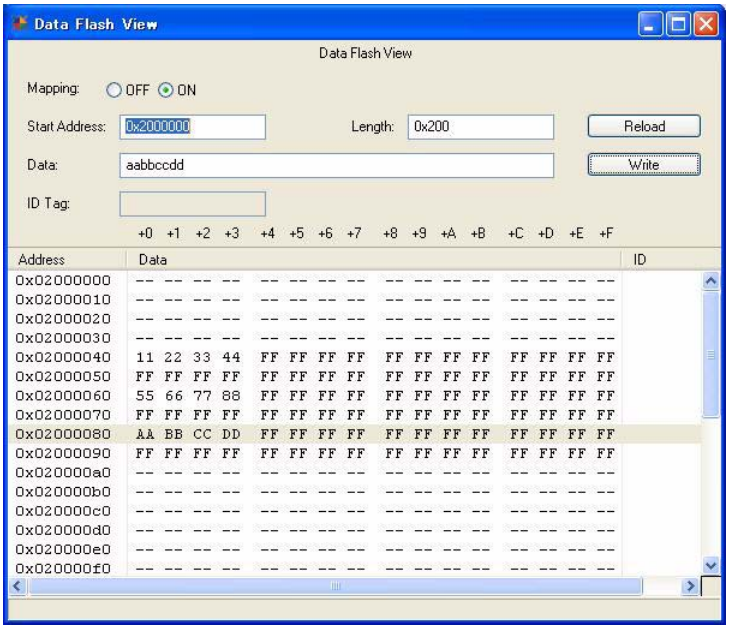
This command is available for only the device equipped with the Data Flash memory. It is necessary to select the Data Flash area by **DFMAP** command before this command is used.
In enables blank checking, when it reads memory value in Data flash memory, 850eserv2 does not display the indefinite values. When it displays Data flash memory via memory view in MULTI debugger(memview), "Data Flash" window in 850eserv2 configuration window or **DFDUMP** command**,** the indefinite values are displayed as "--". When it read the indefinite values via other commands, 850eserv2 returns error.
If **auto** is specified, 850eserv2 do blank checking automatically when memory view is opened or updated. In memory view(memview) of MULTI debugger is opend, memory view is updated automatically when target is running to stoped. If **auto** is specified, 850eserv2 do blank checking in this timing. In this case, it may take some time depending on the state of blank memory.
If **auto** is specified again, automatic blank checking is disabled.

When it displays via memory view in MULTI debugger(memview), unchecks "memory > Use Block Reads" in memview's menu.

Example:

```
850eserv2>dfmap on
850eserv2>blankcheck on
Data Flash memory blank checking...
850eserv2>
850eserv2> dfdump 0x2000000
0x02000000: -------- -------- -------- --------
0x02000010: -------- -------- -------- --------
0x02000020: -------- -------- -------- --------
0x02000030: -------- -------- -------- --------
0x02000040: 22114433 FFFFFFFF FFFFFFFF FFFFFFFF
850eserv2> m -d2 0x02000000
Memory read at 0x02000000 failed!
Error: General Error
Command 'm -d2 0x02000000' failed.
850eserv2>
```

<< In display via Data Flash window>>



BRA

BRA [# [A=*address_or_range*] [x=*data/mask*] [*status*] [S=*type*] [P=*mask8*] [M=*address_mask*]] | [D #]

Sets or displays the hardware bus event detectors. If no arguments are specified, the **BRA** command will display the current settings.

Each of the optional arguments is described in detail below.

* The **#** argument specifies the identification number of the event detector.

  Because some devices have up to two bus event detectors, the maximum identification number varies depending on device. If you specify an ID that is already being used, the previous setting will be overwritten.

* **A=***address_or_range* specifies an address or range for the event as follows:
  A=*address* specifies a single address expression,
  A=*start,end* specifies an address range beginning at start and ending at end,
  A=*start,* **L** *length* specifies an address range beginning at *start* and extending for *length* bytes.

  You cannot specify the address range for an event if the device does not support it.
  You cannot enter both a single address and a range. If you specify a range for the address field, it uses two bus event detectors. If you do not enter an address value, any address will be used.

* The **x=***data/mask* option specifies the access size and data/mask values.
  The data comparator detects reads and writes of specific values at the addresses. The size of the access (x) can be specified as **B**(byte), **H**(halfword), **W**(word), **L**(long), **L12**(12bytes), **L16**(16bytes) or **!** (exclusive of the specified data size). **L**, **L12** and **L16** can be specified only with RH850. If **L12** or **L16** are specified, *data/mask* can not be specified.

  You can use a symbolic constant to specify the data, but you cannot include a mask if you do. If you do not enter a data size/value, a halfword access and a mask of all ones will be assumed.

* The *status* option specifies the type of bus transaction to detect.
  Valid arguments for *status* are: **RW** (read or write; this is the default), **RO** (read only), or **WO** (write only).

* **M=***address_mask* specifies an address mask value of 32-bit width.
  The *address_mask* value is entered as a 32-bit hexadecimal or decimal value.

* **S**=*type* specifies the type of bus event. It can be specified only with RH850.
  Valid arguments for type are **f** (break before the execution), or **e** (break after the execution).

* **D #** specifies the event number to be deleted.

* **P=***mask8* specifies an external probe qualifier.

The event number and event condition that can be set by the **BRA** command are different each ICE and the device. Please refer to the user's manual of ICE and the device for details of the event condition. In case of IECUBE2(V850E2Core), BRA event cannot be specified for start or stop trace condition.

In Multi-core debugging, you can set BRA events in resource that each cores have. BRA event you set in one core can be used in only this core. It can not be used in other core. BRA event set in other core can

not be referred. For the detail of event in MULTI-core debugging, please refer to "Event setting" on page 51.

There are bus event detectors, each consisting of an address/range, a data/mask, a status, and an external probe comparator (some devices have up to two bus event detectors only). The output of each of these comparators are ANDed together to produce an event. If a starting and ending address range is specified, then two of the bus event detectors will create the event (one for the start and one for the end).

Hex values are indicated by a `0x` prefix, while binary input is indicated by an `0b` prefix. To specify a value with a mask, use `X` characters for the nibbles or bits you want to be ignored. For example, `H=0x245C` will only detect a data access of `0x245C`, while `H=0x245X` will detect any data value in the range `0x2450-0x245F`.

Example:

<< In the case of OCD emulator and Nx85ET(RCUO+TEU+TRCU)>>

```
850eserv2>bra 1 a=0x12345678 w=0x87654321 ro p=0x0 m=0xffffffff
850eserv2>bra
BRA1: A=0x2345678 W=0x87654321 RO P=0x1 M=0xffffffff
```

<< In the case of IECUBE or IECUBE2 >>

```
850eserv2>bra 1 a=0x12345678 w=0x87654321 wo m=0xffffffff
850eserv2>bra
BRA1: A=0x2345678 W=0x87654321 WO M=0xffffffff
850eserv2>
```

BRS

BRS [# [A=*address_or_range*] [P=*mask8*] [S=*type*] [M=*address_mask*]] | [D #]

Sets or displays the hardware execution event detectors. If no arguments are specified, the **BRS** command will display the current settings for all detectors. To display the current settings for one detector, specify the identification number (**#**) without any other arguments.

Each of the optional arguments is described in detail below.

* The **#** argument specifies the identification number of the execution event detector.

  Because some devices have up to two execution event detectors, the maximum identification number varies depending on device. If you specify an ID that is already being used, the previous settings will be overwritten.

* **A=***address_or_range* specifies an address or range for the event as follows:
  A=*address* specifies a single address expression,
  A=*start,end* specifies an address range beginning at *start* and ending at *end*,
  A=*start*, **L** *length* specifies an address range beginning at start and extending for *length* bytes.

  You cannot specify the address range for an event if the device does not support it.
  You cannot enter both a single address and a range. If you specify a range for the address field, it uses two execution event detectors. If you do not enter an address value, any address will be used.

* **P=***mask8* specifies an external probe qualifier.

* **S**=*type* specifies the type of execution event.
  Valid arguments for type are **f** (break before the execution), or **e** (break after the execution).
  In case of MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core or IECUBE2(V850E2Core), you can specify **t** for timer event. The event specified **S=t** cannot be used with Link event, hardware breakpoint or trace condition. If you do not specify **TIMEBASE c=** with IECUBE2, you can omit **S**=**t** for timer event.
  In case of OCD emulator and (Nx85E901(RCU0),RCU1),default is **f**. In case of ICE other than that, default is **e**.

* **M=***address_mask* specifies an address mask value of 32-bit width.
  The *address_mask* value is entered as a 32-bit hexadecimal or decimal value.

* **D #** specifies the event number to be deleted.

The event number and event condition that can be set by the **BRS** command are different each ICE and the device. Please refer to the user's manual of ICE and the device for details of the event condition.

In Multi-core debugging, you can set BRS events in resource that each cores have. BRS event you set in one core can be used in only this core. It can not be used in other core. BRS event set in other core can not be referred. For the detail of event in MULTI-core debugging, please refer to "Event setting" on page 51.

There are execution event detectors in the emulator, each consisting of an address and an external probe comparator (some devices have up to two execution event detectors only). The output of the two comparators are ANDed together to generate an event. They will cause an event when the specified address

or an address in a range is executed. If you specify a range of addresses, two of the available execution event detectors will be used, one for the start and one for the end.

Example:

<<In the case of OCD emulator and Nx85ET(RCU0+TEU+TRCU)>>

```
850eserv2> BRS 1 A=0x3C200CA, 0x3C200DE P=0xFFFF s=f M=0xFFFFFFFF
850eserv2> BRS
BRS1: A=0x03C200CA,0x03C200DE P=0x0 s=f M=0x0
```

<<In the case of IECUBE or IECUBE2>>

```
850eserv2> BRS 1 A=0x3C200CA, 0x3C200DE s=e M=0xFFFFFFFF
850eserv2> BRS 2 A=0x3C200CA s=f M=0x1234
850eserv2> BRS
BRS1: A=0x03C200CA,0x03C200DE s=e M=0x0
BRS2: A=0x03C200CA s=f M=0x1234
850eserv2>
```

CACHE

CACHE [(I|D)=(off|4|8|16)]

I=	Selects I-Cache from the following	
	off	Disables the selected cache
	4	Sets cache to 4kB 2way associative
	8	Sets cache to either 8kB 2way associative or 8kB 4way associative with the device
	16	Sets cache to 16kB 2way associative
D=	Selects D-Cache from the following	
	off	Disables the selected cache
	4	Sets cache to 4kB direct map
	8	Sets cache to either 8kB 2way associative or 8kB 4way associative with the device
	16	Sets cache to 16kB 4way associative

In case of IECUBE or IECUBE2, you cannot use this command.
Selecting a mode flushes the selected cache when OCD emulator, MINICUBE2 or E1/E20 emulator is used.
When type of order cash of device is set, unless correct type is set, there is a possibility cash clearing operation not being just done. Please be sure to verify the type of order cash with the manual of the device.

Example:

```
850eserv2>cache i=4
I-Cache=4KB 2way associative
850eserv2>
```

<<In the case of OCD emulator, E1/E20 emulator>>

```
850eserv2>CACHE i=off
850eserv2>CACHE d=off
850eserv2>CACHE
850eserv2>
```

CPU

CPU [R=*rom*] [A=*ram*]

In case of OCD emulator, MINICUBE2, E1/E20 emulator, QB-V850E-VM or IECUBE2, it is not possible to change though it is possible to display.
Changes or displays the CPU internal ROM and RAM sizes (in KByte) for emulation.
The **R=***rom* argument sets the internal ROM size and the **A=***ram* argument sets the internal RAM size (see below for appropriate values). If no arguments are specified, the **CPU** command displays the current internal ROM and RAM sizes.

Appropriate values for **R=***rom* for the IECUBE ICE are 1-1024 depending on the contents of the device file corresponding to the target.
Appropriate values for **A=***ram* for the IECUBE ICE are 1-60 depending on the contents of the device file corresponding to the target.

Example:

```
brk>cpu r=128 a=12
brk>cpu
IROM 0x00000000-0x0001ffff = 128K
IRAM 0x0fffc000-0x0fffefff = 12K
```

DCLOCK

DCLOCK [*main_clock sub_clock* (swon|swoff)]

main_clock	Specifies Main clock per KHz
sub_clock	Specifies Sub clock per Hz
swon	Uses Sub clock
swoff	Uses Main clock (default)
<none>	displays now setting

Please be sure to set up target's operating frequencies by this command at the time of initialization.
When not setting up target's operating frequencies, or when the right value is not set up,connection with hardware cannot be made correctly but an unexpected error may be caused.

It sets the transmitter of the clock which is mounted on the target, or the value of the oscillator.
It can set the main clock and the sub clock. As for presence of the main clock and the sub clock and frequency etc., please refer to each device manuals.

Example:

The oscillation machine of main clock 5MHz and sub clock 32.768KHz being mounted by the target, when the sub clock is used.

```
850eserv2>dclock 5000 32768 swon
850eserv2>dclock
auto clock switch = on
target main clock =   5000kHz
target sub clock  =  32768Hz
850eserv2>
```

DFDUMP

DFDUMP [*address*] [l=*length*]

address	Specify display address. If omit it, displays from the address specified last time. Default is the starting address of the Data Flash area selected by **DFMAP** command.
l=*length*	Specify display length(byte unit). If omit it, displayed by the length specified last time. Default is 0x50 byte.

This command is available for only the device equipped with the Data Flash memory. It is necessary to select the Data Flash area by **DFMAP** command before this command is used.
Displays the Data and ID tag in Data Flash memory from specified *address* to specified *length*. If specified *address* is not the Data Flash area selected by **DFMAP** command or specified *length* exceeds the size of Data Flash area, this command returns error.

It is necessary to specify *address* and *length* by four byte boundary. If specified *length* becomes 0, displayed by the length specified last time.

Example:

```
850eserv2>dfdump 0x1f8002 l=0x30
0x001F8000: 00000000 FFFFFFFF FFFFFFFF FFFFFFFF IDtag 0 1 1 1
0x001F8010: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF IDtag 1 1 1 1
0x001F8020: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF IDtag 1 1 1 1
850eserv2>
```

DFLASH

DFLASH [on [w|e = min|typ|typw|max]] [off]

on	Enable Data Flash memory emulation	
off	Disable Data Flash memory emulation(default)	
w =	Select following emulation mode writing time to Data Flash memory	
	min	No Retry
	typ	Assumption frequency on Flash macro spec(default)
	typw	Maximum Assumption frequency on Flash macro spec
	max	Maximum retry frequency
e =	Select following emulation mode erasing time to Data Flash memory	
	min	No Retry
	typ	Assumption frequency on Flash macro spec(default)
	typw	Maximum Assumption frequency on Flash macro spec
	max	Maximum retry frequency

This command is available for IECUBE or IECUBE2 and only the device equipped with the Data Flash memory.
It is necessary to select the Data Flash area by **DFMAP** command before this command is used.
This command sets the emulation of Data Flash memory. If you specify **ON**, you can use the emulation writing and erasing time to Data Flash memory, and you can use **DFLASHERR** command. If you specify **OFF**, all the emulation to Data Flash memory becomes invalid. **OFF** and other options cannot be specified simultaneously.
If you use the emulation writing time to Data Flash memory, you specify **W** option and the mode. If you use the emulation erasing time to Data Flash memory, you specify **E** option and the mode.

Example:

```
850eserv2>dfmap cs0
0x100000 - 0x1fffff is mapped 'TARGET ROM (POWER CHECK OFF)'
850eserv2>dflash on w=min
850eserv2>dflash w=typ e=max
850eserv2>dflash
Data Flash emulation enable
Write : Typ
Erase : Max
850eserv2>dflash off
850eserv2>dflash
Data Flash emulation disable
850eserv2>
```

DFLASHERR

DFLASHERR [wwrite|berase|bverify|bblank|ecc [*address*]] [off]

wwrite	Return error/non-error(default) value from Word writie function
berase	Return error/non-error(default) value from Block erase function
bverify	Return error/non-error(default) value from Block verify function
bblank	Return error/non-error(default) value from Block blank check function
ecc	Return error/non-error(default) value from ECC error function
address	Specify the address where the function returns error value. It can specify only the address in Data Flash area specified by **DFMAP** command
off	Return error/non-error(default) value from all function
<NONE>	Display now setting

This command is available for IECUBE or IECUBE2 and only the device equipped with the Data Flash memory.
It is necessary to select the Data Flash area by **DFMAP** command and setting **DFLAH ON** before this command is used.
This command sets the error emulation of Data Flash memory. If you specify each options and *address*, Specified function returns error value when specified function accesses specified address. When each options and *address* are specified simultaneously, specified function always returns error value. When 850eserv2 is finished, the address set last time becomes invalid. When *address* is omitted and only each options are set, it becomes the toggle of error/non-error.
If you specify **OFF**, all functions become to return non-error value.


It is necessary to set DFLCTL register on your program before this command is used


The default address of each options are 0x0. Therefore *address* cannot be omitted when it specifies first time.


It can not set **ecc** error emulation while Data Flash Liblary is running.


If you want to set **ecc** error emulation in BlockErase or BlankCheck function, It is necessary to disable **berase** and **bblank** error emulation. These options can not set **ecc** option at the same time.

Example:

* Setting before **DFLASHERR** command is used.

```
850eserv2>dfmap cs0
0x100000 - 0x1fffff is mapped 'TARGET ROM (POWER CHECK OFF)'
850eserv2>dflash on
```

* In case of Word writing function in Data Flash library returns the error value when it accesses 0x1f8000.

```
850eserv2>dflasherr wwrite 0x1f8000
WordWrite error 0x1f8000 was enabled
850eserv2>dflasherr
Generate WordWrite error       : 0x1f8000
850eserv2>
```

* In case of Word writing function in Data Flash library returns the non-error value.

```
850eserv2>dflasherr wwrite
WordWrite error was disabled
850eserv2>dflasherr
850eserv2>
```

* In case of all function in Data Flash library returns the non-error value.

```
850eserv2>dflasherr off
All error was disabled
850eserv2>dflasherr
850eserv2>
```

DFMAP

DFMAP [cs0-cs3] [off]

cs0-cs3	Selects mapping Data Flash area with CS area selectable device .
on	Mapping to Data Flash area with CS area fixed device.
off	Disable mapping to Data Flash area(default)
<NONE>	Display now setting.

This command is available for only the device equipped with the Data Flash memory.
This command maps to the Data Flash area. If it dose not map to the Data Flash area by this command, it dose not access to the Data Flash area. In case of CS area selectable device, when **cs0-cs3** is selected, the Data Fash area allocated in the CS area is selected. The Data Flash area that can be accessed is only an area selected by this command. Default is non mapping. In case of CS area dixed device, when **on** is specified, the Data Fash area allocated in the device fixation area.

In case of IECUBE, If you have not mapped the Data Flash area by **MAP** command, 850eserv2 automatically maps 1MB including the Data Flash area as target memory ROM area after inputting **DFMAP** command.
For example, when you specify **DFMAP CS0**, 850eserv2 automatically maps 0x100000-0x1fffff including CS0(0x1f8000-0x1fffff) at target memory ROM area.

When it read memory from Data flash area when mapping is disabled, it maps to Data Flash area automatically.

Example:

```
850eserv2>dfmap cs1
850eserv2>dfmap
Data Flash type is Selectable
Data Flash area is CS1 : 0x3f8000 - 0x3fffff
850eserv2>dfmap off
850eserv2>dfmap
Data flash area is disable
850eserv2>
```

DFSAVE

DFSAVE filename start length [old]

filename	File name to save. A file is possible for directory specification
start	Start address of the memory to save
length	Number of bytes to save
old	Saves old format from S-record file

This command is available for only the device equipped with the Data Flash memory.
It is necessary to select the Data Flash area by **DFMAP** command before this command is used.
DFSAVE command saves a file with a special file format, in which the data records (4 bytes) and ID tag (4 bytes) information for each address is altered from Data Flash memory. The address in S-record file is converted to the offset from top of Data Flash area. If you specify **old** option, the address in S-record file is not converted to the offset.
If the specified *start* is not the Data Flash area selected by **DFMAP** command or specified *length* exceeds the size of Data Flash area, this command returns error.
The file saved by this command can download by **FLOAD** command with **idtag** option.

It is necessary to specify *start* and *length* by four bytes boundary. If specified *length* becomes 0, saves four bytes.

<<MULTI and 850eserv2 save function list>>

Command	Format	Code Flash area	Data Flash area (Data)	Data Flash area (ID tag)	Internal ROM area	Internel RAM area
MULTI memdump	S-record	Enable	Enable	Disable	Enable	Enable
	Raw binary	Enable	Enable	Disable	Enable	Enable
850eserv2 dfsave	S-record	Disable	Enable	Enable	Disable	Disable
	Raw binary	Disable	Disable	Disable	Disable	Disable

Example:

```
850eserv2>dfsave IDtagform.hex 0x1f8000 0x8000
Data Flash Memory saved
850eserv2>fload srec IDtagform.hex idtag
Flash Memory loaded
850eserv2>dfsave c:\green\v850\IDtagform.hex 0x1f8000 0x8000
Data Flash Memory saved
850eserv2>
```

EFCONFIG

EFCONFIG [filename address size parallel serial] | [id] | [k]

filename	File name of external flash memory information (specify full path)
address	Top address of flash memory
size	Data access size of flash memory(specify 1, 2 or 4)
parallel	Parallel number of flash memory(specify 1 or more)
serial	Serial number of flash memory(specify 1 or more)
id	Displays ID informations(maker and vendor codes) of flash memory
k	Disables to download to external flash memory
<NONE>	Displays informations of flash memory

Registers external flash memory information. This command is available for only the device equipped with the external flash memory. You can download to external flash memory after to registers external flash memory information via this command.
The information file(.fdb file) that corresponds to your flash memory is necessary for registering external flash memory information. You can download this file from following Renesas Electronics web page:

```
http://www.renesas.com/ghs_debug_if
```

You specify full path of .fdb file, top address, access size, parallel number and serial number to this command for registering external flash memory information.

You can write to external flash memory via target command or memory window after to registers external flash memory information.

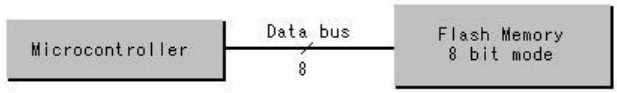
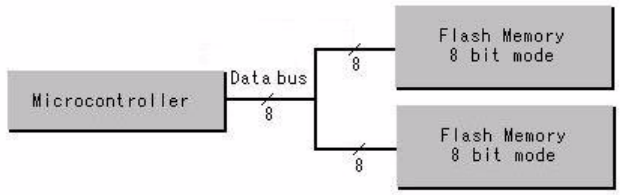
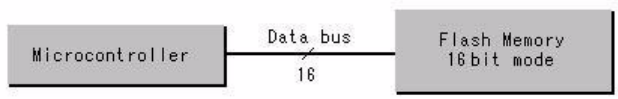
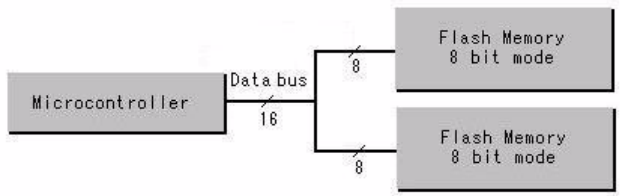
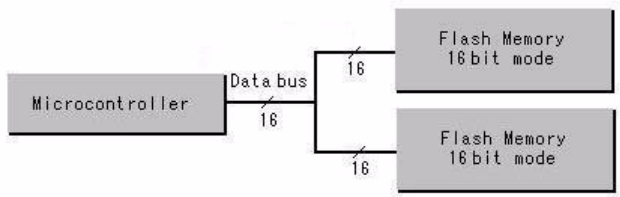
If you specify no argument in this command, it displays registered flash memory information and total size of flash memory(Kbyte). If you specify only *id* option, it displays ID information(maker and vendor codes) of flash memory.
If you specify only *k* option, it removes registered flash memory information and disables to download to external flash memory.

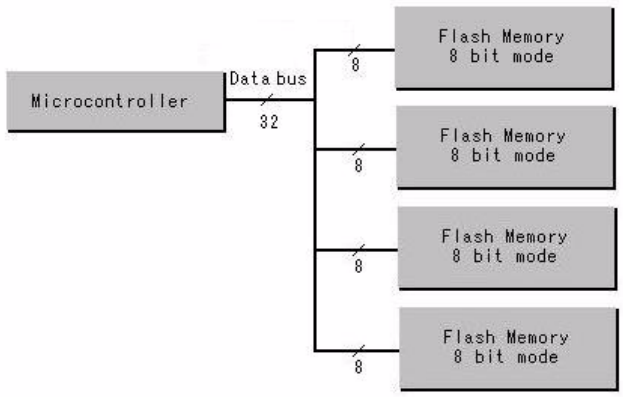
When you set a software breakpoint in external flash memory area via MULTI, 850eserv2 sets a before execution event hardware breakpoint instead software breakpoint. The maximum identification number of before execution events varies depending on device. If all before execution events are already used, it cannot set a software breakpoint to external flash memory area.
When you download the object included .syscall section to external flash memory, 850eserv2 use one before execution event because it sets a breakpoint to .syscall section automatically.
When you do C source level step or return from the function, 850eserv2 sets a breakpoints temporarily. If all before execution events are already used in this time, these work is failed.

Cautions

* Sectors that are protected cannot be erased or written.
* If the device has a function to prohibit erasing and writing when power is turned on/off, it cannot be erased or written, because a dedicated unlock command is not supported.
* Special sectors such as Secured Silicon Sector area that can permanently hold Electric Serial Numbers which can be randomly assigned cannot be accessed (read, erased, or written), because no dedicated command is supported.

Parameter setting value list

Connection of flash memory	size	parallel	serial
	1	1	1
	1	1	2
	2	1	1
	2	2	1
	2	1	2

Connection of flash memory	size	parallel	serial
	4	4	1

Example:

* Download procedure to external flash memory.

1. Registers external flash memory information.

```
850eserv2>efconfig C:\fdpfiles\MBM29LV800B.fdb 0x100000 2 1 1
```

2. Downloads program via MULTI command pane.

```
MULTI>load
```

* Displays informations of flash memory.

```
850eserv2>efconfig
Top address    : 0x100000
Data bus size  : 2
Parallel num   : 1
Serial num     : 1
Sector Info num: 4
      Sector 0 size : 16 Kbyte
      Sector 0 num  : 1
```

* Displays ID informations(maker and vendor codes) of flash memory.

```
850eserv2>efconfig id
Sector 1 :
      Vender code : 0x0004
      Device code : 0x225B
```

* Disables to download to external flash memory.

```
850eserv2>efconfig k
```

FBREAK

FBREAK [on|off]

on	Use ROM correction function in ICE and write break instruction to Flash memory.
off	Use ROM correction function in ICE only.
<NONE>	Display now setting.

This command is available for OCD emulator, MINICUBE2, or E1/E20 emulator and only the Flash device supported with OCD function. And this command is not available for the Flash device that has 4K or more block size.

FBREAK command sets software breakpoint setting mode in Flash memory
In default setting, it sets software breakpoint to Flash memory to use ROM correction function in ICE. Therefore, it cannot set software breakpoints to exceed the number of maximum of ROM correction function in Device.
When "**on**" is specified, it sets software breakpoint to Flash memory to write break instruction directly when it tried to set software breakpoints to exceed the number of maximum of ROM correction function. Therefore, it can set more software breakpoints, but it is necessary more longer time to write or remove software breakpoint to Flash memory.
When "**off**" is specified, it sets software breakpoint to Flash memory to use ROM correction function in ICE as well as the default setting. All software breakpoints are removed when it sets "**off**".

FERASE

FERASE <*address* | ALL>

address	Erase Flash block including specified address
ALL	Erase all Flash block

Erases Flash memory. Specify either the address or the "**all**" option.
When an address in Code Flash area is specified, the Flash block including specified address is erased. But cannot specify the Code Flash address in IECUBE. In this case, pleasee use **BLOCKFILL** command.

When an address in Data Flash area is specified, All Data Flash area is erased. However, only the device equipped with Data Flash area can be specified. It is necessary to select the Data Flash area by **DFMAP** command before this command is used.

When an address in External Flash area is specified, the Flash block including specified address is erased. It is necessary to register external flash memory information by **EFCONFIG** command before this command is used.

When "**all**" is specified, all Code Flash blocks are erased. And if the device equipped with Data Flash area and select Data Flash area by **DFMAP** command, all Data Flash area is erased.

Example:

```
850eserv2>ferase 0x100
850eserv2>ferase all
850eserv2>
```

FLASH

Shows Flash memory information.
The status of FLMD0 switch is shown as one of the following three states:
'OCD' means OCD emulator can configure FLMD0 pin level. 'High' means FLMD0 pin level is set high. 'Low' means FLMD0 pin level is set low.
Data Flash information is displayed only the device equipped with the Data Flash area and select Data Flash area by **DFMAP** command.
External flash memory informatin is not displayed in this command.

Example:

```
850eserv2>FLASH
Code Flash block count = 128
Data Flash block count = 16
boot swap on
FLMD0 switch : OCD
Code Flash block 0 end = 0xfff
Code Flash block 1 end = 0x1fff
Code Flash block 2 end = 0x2fff
Code Flash block 3 end = 0x3fff
...
Code Flash block 127 end = 0x7ffff
Data Flash block 128 end = 0x1f87ff
Data Flash block 129 end = 0x1f8fff
Data Flash block 130 end = 0x1f97ff
...
850eserv2>
```

FLASHRESET

FLASHRESET(FR) [*address*]

address	Specify reset vector address.
<NONE>	Display now setting.

This command is available for IECUBE.
This command cannot be used when the flash self emulation function is disabled. In this case, if **FLASHSELF ON** is specified, the flash self emulation function is enabled.Only when the flash memory process is Type1 and it specifies from flash memory self programming Ver 3.00(**FLASHSELF newspec** is specified) or when the device supports this function, this command can be used.
It specify reset vector address. When CPU is reset, the program is executed from the address specified by this command. The address that can be specified for reset vector is in the installed flash area. When it specifies outside the flash area, it becomes an error.

Example:

```
850eserv2>flashreset 0x100
850eserv2>flashreset
Reset vector address : 0x100
850eserv2>
```

FLASHSELF

FLASHSELF(FS) [on | off | newspec | oldspec] (IECUBE)
FLASHSELF(FS) [on | off | init | boot <value>] (IECUBE2)

on	Enables flash self emulation function.
off	Disables flash self emulation function(default).
newspec	This command is available for IECUBE. Since flash memory self programming Ver 3.00 When the flash self process is only Type1, it is possible to set it.
oldspec	This command is available for IECUBE Before flash memory self programming Ver 2.00 (default) When the flash self process is only Type1, it is possible to set it.
init	This command is available for IECUBE2. Initializes Code flash area and each settings.
boot <value>	This command is available for IECUBE2. Sets Boot Swap Cluster area. It can set 0x0 to 0xff.
<NONE>	Display now setting.

This command is available for IECUBE or IECUBE2.
Flash self emulation function is made enable or disable.
In the case of IECUBE, when the flash memory process is Type1, **newspec** and **oldspec** (default) are specified. When the flash memory process is except Type1, **newspec** and **oldspec** (default) can be specify. When **newspec** is specified, **read** option in **FSECFLAG** command and **FLASHRESET** command can be used.
In the case of IECUBE2, when it specifies **init** option, all datas of Code flash area are cleard and the settings of **FSECFLAG** and **FSHIELDWINDOW** command are initialized.
It sets Boot Swap Cluster area after **boot** option. Boot Block Cluster area is set automatically for this setting. It can set 0x0 to 0xff in **value**, Boot Swap Cluster area and Boot Block Cluster area are set as next list for specified **value**.

<<The setting of Boot Swap Cluster area and Boot Block Cluster area>>

value	Boot Swap Cluster area	Boot Block Cluster area
0x0	4KB	4KB
0x1	8KB	8KB
0x2	16KB	12KB
0x3	16KB	16KB

value	Boot Swap Cluster area	Boot Block Cluster area
0x4 ～ 0x7	32KB	20 ～ 32KB
0x8 ～ 0xf	64KB	36 ～ 64KB
0x10 ～ 0x1f	128KB	68 ～ 128KB
0x20 ～ 0xff	256KB	132 ～ 1024KB

When the flash self emulation is made enable, the following limitations are generated.

* You can set break before the execution only one. Debugger uses one break before the execution for flash self emulation.
* It cannot modify the internal ROM size and the internal RAM size after flash self emulation function is made enable.
* If the internal ROM size is set to other than the default size, flash self emulation function cannot be made enable.
* If the SP register is 0, user's programs break unjust.

Status Check error emulation in Type4.
When you set **FLASHSELF on** in Type4**,** 850eserv2 sets work area of Self library to use Status Check error emulation automatically.
850eserv2 sets the address of symbol "SelfLib_StatusCheck" in work area. If 850eserv2 could not find this symbol, it displays error message. In case, please set work area by **FMACROERRSC** command.
When you use Status Check error emulation function, please do not convert Flash self library to ROM.

Example:

In case of the flash self process isType1and using the flash memory self programming since Ver 3.00.

```
850eserv2>flashself on
850eserv2>flashself newspec
850eserv2>flashself
FlashSelf function enable
Type1 newspec
850eserv2>
```

FLOAD

FLOAD [srec|raw] <filename> [start[length]] [idtag [old]]

srec	Motorola S-record form(default)
raw	Raw binary data file
filename	file name to load. A file is possible for directory specification
start	start address of the memory to load(Only **raw** specify)
length	number of bytes to load(Only **raw** specify)
idtag	Downloads special format file including ID-tag to Data Flash area (Only **srec** specify)
old	Downloads old special format file(Only **idtag** specify)

The specified file is downloaded to Flash memory. It can be downloaded to Data Flash memory or Code Flash memory. If specified file is downloaded to Data Flash memory, It is necessary to select the Data Flash area by **DFMAP** command before this command is used. If specified file is downloaded to External Flash memory, It is necessary to register external flash memory information by **EFCONFIG** command before this command is used.

When **srec** and **raw** specification are omitted, a file is read in S-record form. If the download range is outside Flash memory, it downloads to normal memory. A file is possible for directory specification.

Specification of start and length cannot be specified when **srec** is specified.
If **idtag** is specified, the 850eserv2 downloads a file with a special file format, in which the data records (4 bytes) and ID tag (4 bytes) information for each address is altered. Please see figure below

```
S3 25 003F8200 12345678 FFFFFFFF ddaaddaa FFFFFFFF
ddaaddaa FFFFFFFF ddaaddaa FFFFFFFF 90
S3 25 003F8220 ddaaddaa 00000000 ddaaddaa FFFFFFFF
ddaaddaa 00000000 FFFFFFFF FFFFFFFF 11
```

Although the ID tag is 4 bytes of size, only bit 0 of the first byte is used to hold the value.
850eserv2 converts the address that described in special format file to the offset from top of Data flash area and downloads data and ID tag to this offset. If you specify **old**, 850eserv2 downloads data and ID tag to the address that described in special format file directly.
The data value and ID tag in Data Flash memory can be saved by **DFSAVE** command.

Specification of start is required when **raw** is specified. When length is omitted,the size of file is specified.

<<MULTI and 850eserv2 load function list>>

Command	Format	Code Flash area	Data Flash area (Data)	Data Flash area (ID tag)	External Flash area	Internal ROM area	Internel RAM area
MULTI memload	S-record	Enable	Enable	Disable	Enable	Enable	Enable
	Raw binary	Enable	Enable	Disable	Enable	Enable	Enable
850eserv2 fload	S-record	Enable(*)	Enable(*)	Enable	Enable	Enable	Enable
	Raw binary	Enable(*)	Enable(*)	Disable	Enable	Enable	Enable

(*) Can load at high speed.

Example:

```
850eserv2>fload srec SrecFile
Flash Memory loaded
850eserv2>
850eserv2>fload raw RawFile 0x1000 0x2000
Flash Memory loaded
850eserv2>
850eserv2>fload srec IDtagFile idtag
Flash Memory loaded
850eserv2>
```

FLSF

FLSF [ON|OFF|memgrd|memwp|sfrgrd|sfrwp|sfrrp|romgrd|romwp|ramgrd|ramgrdv]

OFF	Invalid all Fail-safe break.
ON	Effective default setting (memgrd,memwp,sfrgrd,sfrwp,sfrrp,romgrd,romwp,ramgrd,ramgrdv)
<NONE>	Display now Fail-safe breaks.
memgrd	Enables or disables external-bus-related guard mapping area access breaks
memwp	Enables or disables external-bus-related write detection mapping area access breaks.
sfrgrd	Enables or disables SFR guard area access breaks.
sfrwp	Enables or disables SFR write access detection breaks.
sfrrp	Enables or disables SFR read access detection breaks.
romgrd	Enables or disables guard area access breaks in IROM.
romwp	Enables or disables write access breaks in IROM.
ramgrd	Enables or disables guard area access breaks in IRAM.
ramgrdv	Enables or disables the verify check following a break of a guard area in IRAM.

Sets up Fail-safe breaks. This command is available for IECUBE.
It will enable memgrd, memwp, sfrgrd, sfrwp, sfrrp, romgrd, romwp, ramgrd, and ramgrdv (default) if you set ON. If you do not enable guard any area access breaks in IRAM, the IRAM guard area will not be verify-checked. Therefore, enabling ramgrdv causes ramgrd to be also enabled, and disabling ramgrd causes ramgrdv to be also disabled.
In case of QB-V850E-VM, when the write access was generated in the area of 0x100000-0x1fffff, "break by write protect (external bus)" Fail-safe break is generated.

Example:

```
850eserv2>flsf off
850eserv2>flsf memgrd
850eserv2>flsf
guard mapping area access break (external bus)
850eserv2>flsf on
850eserv2>flsf
guard mapping area access break (external bus)
guard area access break (irom)
guard area access break (iram)
guard area break & verify (iram)
mapping area write access break (external bus)
write protect break (irom)
guard area access break (sfr)
write access break (sfr)
read access break (sfr)
850eserv2>
```

FMACROERR

FMACROERR(FME) [berase|bverify|wwrite|bblank|setinfo|flmdchk|on|off]

berase	Return error/non-error value from FlashBlockErase function
bverify	Return error/non-error value from FlashBlockVerify function
wwrite	Return error/non-error value from FlashWordWrite function
bblank	Return error/non-error value from FlashBlockBlankCheck function
setinfo	Return error/non-error value from FlashSetInfo function
flmdchk	Return error/non-error value from FlashFLMDCheck function
on	Return error value from all function
off	Return non-error value from all function
<NONE>	Display now setting

This command is available for IECUBE.
This command cannot be used when the flash self emulation function is disabled. In this case, if **FLASHSELF ON** is specified, the flash self emulation function is enabled.
It emulates error of flash macro service. The error value that flash memory returns when suffering can be compulsorily returned. The error value that flash memory returns when suffering never returns in usual emulation.
Each option is set to return error/non-error by toggle. When you specify **on**, the error is returned by all functions of FlashBlockErase, FlashBlockVerify, FlashWordWrite, FlashBlockBlankCheck, FlashSet-Info, and Flash-FLMDCheck. When you specify **off**, the non-error is returned by all functions.

850eserv2 cannot read the level of FLMD0 terminal. It is assumed that the high level is input to the FLMD0 terminal while processing the flash self programming, and FlashFLMDCheck function always returns the non-error usually. Therefore, please set **flmdchk** option when you want to return the error when the row level is input to FLMD0 terminal by the Flash-FLMDCheck function.

Example:

* When you want to return the error value by the FlashBlockErase function.

```
850eserv2>fmacroerr berase
850eserv2>fmacroerr
Generate FlashBlockErase error
```

* When you want to return the error value by the all function.

```
850eserv2>fmacroerr on
850eserv2>fmacroerr
Generate FlashBlockErase error
Generate FlashBlockVerify error
Generate FlashWordWrite error
Generate FlashBlockBlankCheck error
Generate FlashSetinfo error
Generate FlashFLMDCheck error
```

* When you want to return the non-error value by the all function.

```
850eserv2>fmacroerr off
850eserv2>fmacroerr
```

FMACROERRSC

FMACROERRSC(FSC) [work=*address*]

work	Sets work area address of Flash Self library
<NONE>	Sets default address

This command is available for IECUBE.
This command cannot be used when the flash self emulation function is disabled. In this case, if **FLASHSELF ON** is specified, the flash self emulation function is enabled.
It sets work area address of Flash Self library to use Status Check error emulation.
When you specify **work** option, specified address is set to work area of Flash Self library. When you specify no option, default address is set to work area of Flash Self library.

Example:

```
850eserv2>fmacroerrsc work=0x3ff7000
Work area is 0x3ff7000
850eserv2>fmacroerrsc
Work area is 0x3ffefd0
850eserv2>
```

FSECFLAG

FSECFLAG(FCF) [cerase|berase|program|read|bootcluster|on|off]

cerase	Enable(default) or Disable Chip Erase
berase	Enable(default) or Disable Block Erase
program	Enable(default) or Disable Write
read	Enable(default) or Disable Read When the flash memory self programming since Ver 3.00 and the flash self process is Type1, or the device supports this function, it is possible to set it.
bootcluster	Enable(default) or Disable Boot Block Cluster Reprogramming When the device supports this function, it is possible to set it.
on	Disable all setting
off	Enable all setting
<NONE>	Display now setting.

This command is available for IECUBE and IECUBE2.
This command cannot be used when the flash self emulation function is disabled. In this case, if **FLASHSELF ON** is specified, the flash self emulation function is enabled.
It emulates the default value of security flag when it set security to the flash memory using Flash Writer PG-FP4 etc.
Each option is set to enable or disable by toggle. When you specify **on**, the all functions of Chip Erase, Block Erase, Write, Read, and Boot Block Cluster Reprogramming are set disable. When you specify **off**, all functions are set enable.
In the case of IECUBE, **read** can be specified only the flash memory self programming since Ver 3.00 and the flash self process is Type1, or the device supports this function. In case of Type1, please specify **FLASHSELF newspec**. If it sets **read** with unsupported device, it becomes an error.
bootcluster can be specified only the device supports this function. If it sets **bootcluster** with unsupported device, it becomes an error.
In the case of IECUBE2, each security flags can not be set to disable after they are set to enable. However, when it specifies **FLASHSELF init**, all security flags are initialized.

Example:

In case of the flash self process isType1and using the flash memory self programming since Ver 3.00.

* When you want to disable Chip Erase

```
850eserv2>fsecflag cerase
850eserv2>fsecflag
Disable Chip Erase
```

* When you want to disable all setting

```
850eserv2>fsecflag on
850eserv2>fsecflag
Disable Chip Erase
Disable Block Erase
Disable Program
Disable Read
```

* When you want to enable all setting

```
850eserv2>fsecflag off
850eserv2>fsecflag
850eserv2>
```

FSHIELDWINDOW

FSHIELDWINDOW [s=*start_block*] [e=*end_block*]

s=*start_block*	Specifies start block.
e=*end_block*	Specifies end block.

This command is available for MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core or IECUBE2(V850E2Core).
Flash memory areas from ***start_block*** to ***end_block*** can be written via user program with Flash self emulation function. Other areas are protected to be written. You can refer flash memory blocks via **FLASH** command.
This setting is not effect writing via debugger.

Example:

```
850eserv2>fshieldwindow s=5 e=100
850eserv2>fshieldwindow
Flash Shield Window start block : 5
Flash Shield Window end block   : 100
850eserv2>
```

HELP

HELP [command]

command	Display help of specified command
<NONE>	Display help of all commands.

HSPLOAD

HSPLOAD [ON | OFF]

ON	Allows high-speed download at **HSPLOAD**.
OFF	Does not allow high-speed download at **HSPLOAD**.(default)

This command is available for IECUBE.
This command does not perform the download. High-speed download is achieved by executing a load command after use of **HSPLOAD**.

That if you use this command for download, a CPU reset is inserted, thereby initializing SFR and the registers.

HWBRK

HWBRK(B) [K | [event-expression] [...]]

event-expression	OR(|) list of **BRA**,**BRS**,**LINK**.
K	Clear break status
<NONE>	Display now setting.

Specifies or displays which event detectors cause breaks.
Once you have used the **BRA**, **BRS** and **LINK** commands to specify how the event detectors work, you can use the **HWBRK** command to assign some or all of them to cause a break. The *event_expression* argument specifies the event(s) and/or link(s). Multiple events and/or links can be specified if they are separated by a pipe (|) symbol. The **K** option clears all breakpoints.

In Multi-core debugging, you can set hardware breakpoints in each cores. You can specify the **BRA**, **BRS** and **LINK** events set in this core. The events set in other core can not be specified.

If you enter the command without any parameters, the list of currently assigned events will be displayed. Any new **HWBRK** command overrides the previous break setting.

From the menu "View" > "Breakpoints..." > "Hardware" of MULTI, or by the fact that "Set Hardware Breakpoint" is selected from the right click, it is possible directly to install the breakpoint. The breakpoint(**BRS** and **BRA** event) installed by MULTI GUI is displayed as (MULTI HW-Break) at the time of reference, and cannot perform change and deletion from a target command.

When you set a software breakpoint in external flash memory area via MULTI, 850eserv2 sets a before execution event hardware breakpoint instead software breakpoint. This breakpoint is displayed as (MULTI HW-Break) at the time of reference, and cannot perform change and deletion from a target command.

Example:

```
850eserv2>BRS 1 a=main
850eserv2>BRA 2 A=data2, L 0x10 W=0x00xx
850eserv2>HWBRK BRS1|BRA2
850eserv2>HWBRK
break on:
BRS1: A=main <0x001000ba> P=0xXX
BRA2: A=data2 <0x001002f0>, 0x100300 W=0x000000XX RW
P=0xXX
850eserv2>LINK a 1=brs1
850eserv2>HWBRK BRA2|LINKA
850eserv2>HWBRK
Break on:
BRA2: A=data2 <0x001002f0>, 0x100300 W=0x000000XX RW
P=0xXX
LINKA: 1=BRS1
850eserv2>
```

IDCODE

IDCODE <code>

code	Sets attestation ID code.

This command is available for MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core or IECUBE2(V850E2Core).
IDCODE command sets 24 digits attestation ID code for unlock the RSU(ROM Security Unit).
Attestation ID code is specified by 24 digits hexadecimal without "`0x`" prefix.
When connecting with the target device which mounts RSU(ROM Security Unit) from 850eserv2, it is necessary to 24 digits of attestation ID code set by this command as "-id" option for security unlock.

Example:

```
850eserv2>idcode 123456789012345678900abcd
RSU IDcode set complete.
```

IDTAG

IDTAG <address> <=IDtag>

address	Specify address for write ID-tag to Data Flash memory
IDtag	Specify ID-tag. It can specify only "0" or "1".

This command is available for only the device equipped with the Data Flash memory. It is necessary to select the Data Flash area by **DFMAP** command before this command is used.
It can write the ID-tag attached to the data in Data Flash area. The ID-tag can specify only 0 or 1. The ID-tag is attached to four byte boundary data, Therefore, specified address is adjusted to four byte boundary.

Example:

```
850eserv2>idtag 0x1f8000=0x0
850eserv2>dfdump 0x1f8000 l=0x30
0x001F8000: 00000000 FFFFFFFF FFFFFFFF FFFFFFFF IDtag 0 1 0 1
0x001F8010: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF IDtag 1 1 1 1
0x001F8020: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF IDtag 1 1 1 1
850eserv2>
```

ILLOPBK

ILLOPBK [ON | OFF]

ON	When the illegal opcode exception occurs, it does not stop.
OFF	When the illegal opcode exception occurs, it stops.(default)

This command is not available for MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core, IECUBE2 or RH850.
ILLOPBK command sets when detecting illegal opcode, whether it does or dosen't do the emulation of device operation. Default is **OFF**. When illegal opcode is detected while program execution, it breaks with the address which detects illegal opcode.
If you set **ILLOPBK ON**, when illegal opcode is detected while program execution, the program automatically jumps to exception trap (address `0x60`) same as device operation, and continues execution. In this time, the address which detects illegal opcode is displayed.When the program jumped to exception trap, NP, EP and ID bit in PSW register are set internally.
However DBPC and DBPSW register is used in debugging, These register cannot be set internally. Since DBPC and DBPSW register cannot be set when jumping to exception trap, so when the program returns through DBRET from exception trap, the program crashes.
Therefore, when using **ILLOPBK** by **ON**, disposal such that the program made to move to RESET from exception trap is necessary.

Example:

<<MULTI command pane>>

```
MULTI> target illopbk on
MULTI> target illopbk
illegal opcode execution break ON
MULTI> C
        : illegal opcode execution break
illegal opcode address : 0x144
MULTI>

MULTI> target illopbk off
MULTI> target illopbk
illegal opcode execution break OFF
MULTI> C
        : illegal opcode execution break
Stopped by unknown trap/breakpoint.
MULTI>
```

LINK

LINK [A] [K | *n=event_expression...*] [C=*count*]

Links events with a sequencer, where **A** specifies the sequencer(you can specify A only), *n* specifies the level of enables or disables (see below), and **C** specifies a pass count for the output of the sequencer. The **K** argument clears (kills) all settings.

In Multi-core debugging, you can set sequentially event in each cores. You can specify the **BRA**, **BRS** events set in this core. The events set in other core can not be specified.

If you enter the **LINK** command with only the link number or without any options, the current settings of the link event(s) will be displayed.

You can link events together with one of three event sequencer. Each sequencer has 4 levels of enables and one level of disable, specified by *n=event_expression* as follows:

* *1=event_expression* specifies a list of Enable4 events.
* *2=event_expression* specifies a list of Enable3 events.
* *3=event_expression* specifies a list of Enable2 events.
* *4=event_expression* specifies a list of Enable1 events.
* *D=event_expression* specifies a list of disable events.

The condition of sequentially event that can be set by the **LINK** command are different each ICE and the device. Please refer to the user's manual of ICE and the device for details of the sequentially event condition.

You can link any number of events to each enable level or to disable. To enter a list of events for an input, put a pipe character (|) between the events.

When the events occur in the proper sequence, a link event is generated that may be tied to hardware break point, trace trigger or timer event function. its will be used as the trigger condition for the PASS counter.

You do not have to specify all levels. Levels that are not specified are assumed to be enabled. For example, if you specify Enable4 and Enable3 only, a link will be generated when the two conditions are satisfied. Always begin the events setting from Enable4, Enable3, Enable2, and Enable1.

You can clear the current settings for the link by using the **K** argument.

The **C** option specifies a pass count for the output of the sequencer and should be specified with one or more *n=event_expression* lists.

Example:

```
850eserv2>BRS 1 a=main
850eserv2>BRA 2 a=data1
850eserv2>BRS 3 a=data2
```

```
850eserv2>BRS 4 a=data3
850eserv2>LINK a 1=brs3|brs4 2=bra2 3=brs1
850eserv2>LINK
LINKA: 1=BRS3, BRS4 2=BRA2 3=BRS1
PASS COUNT=1
850eserv2>LINK k
850eserv2>LINK
Link:
<None>
850eserv2>
```

LOADOPT

LOADOPT [ERASE [ON|OFF]] [RESET [ON|OFF]]]

ERASE	Selects whether flash memory is erased before downloading	
	ON	Flash memory is erased before downloading
	OFF	Flash memory is not erased before downloading
RESET	Selects whether the target is reset after downloading	
	ON	The target is reset after downloading
	OFF	The target is not reset after downloading

This command does not downloading. When you input **LOAD** command after setting download options via this command, each options are executed before/after downloading.

When you set **ERASE** option, it is neccesary to the long time to start download because all flash memory areas are erased before downloading,

Example:

```
850eserv2> loadopt erase on
850eserv2> loadopt reset on
850eserv2> loadopt
Flash memory is erased before downloading
Target is reset after downloading
850eserv2>
```

LOCKBIT

LOCKBIT [on|off] [s=*start_block*] [e=*end_block*]

on	Protects the blocks in Code flash memory area.
off	Released the blocks protection.
s=*start_block*	Specifies start block.
e=*end_block*	Specifies end block.

This command is available for RH850.
When **on** is specified, the Code flash memory areas from ***start_block*** to ***end_block*** can not be written(protected) via user program. When **off** is specified, the specified blocks protection are released and can be written. The default of each block depends on the device. You can refer flash memory blocks via **FLASH** command.
This setting is not effect writing via debugger.

Example:

```
850eserv2> lockbit on s=0 e=5
850eserv2> lockbit
Code Flash block 0 is Locked
Code Flash block 1 is Locked
Code Flash block 2 is Locked
Code Flash block 3 is Locked
Code Flash block 4 is Locked
Code Flash block 5 is Locked
Code Flash block 6 is Released
Code Flash block 7 is Released
```

MAP

MAP [K | (W=|R=|TR=|TR2=|U=|U2=|G=*start,end*|*start*, L *length*)]
[CS0|CS1|CS2|CS3|CS4|CS5|CS6|CS7]

This command is available for IECUBE.
Sets the emulator memory configuration, as described below. If no arguments are specified, the **MAP** command displays the current memory maps.
You can use the following options with **MAP** to specify address ranges where the emulator will read or write memory:

W	Specifies emulator read/write memory.
R	Specifies emulator read only memory.
TR	Specifies target read only memory.
TR2	Specifies target read only memory. Power source check of the target is excluded.
U	Specifies target read/write memory.
U2	Specifies target read/write memory. Power source check of the target is excluded.
G	Specifies a guard area.
CS0-CS7	Specifies the chip selection domain mapped at the time of **W** or **R** specification.
K	Cancels (kills) all memory maps.

The specified memory can be designated by *start*, *end* (an address range beginning at *start* and ending at *end*) or *start*, **L** *length* (an address range beginning at *start* and extending for *length* bytes).
Target memory is provided on your development board. The emulator will drive the external bus lines to access it through the pod. For all other types of memory, the emulator will not drive the external lines.

For all other memory mapping, **R** (read only) and **W** (read/write) specify where in the address space the emulator will provide memory blocks. For blocks specified with **R, TR** or **TR2**, the emulator will provide an illegal access break if an attempt is made to write to it.

With IECUBE as for emulation memory is the hardware's option. On default, all the regions are non-mapped (Guard).
In IECUBE which carried the emulation memory board, mapping of an emulation memory is possible as the **W** or **R** option. An emulation memory consists of 16MB (1MB memory of 16 bank). It can assign arbitrary chip selection domains from **CS0** to **CS7**. Two or more banks can be assigned to one chip selection domain. However, when the address which were specified and chip selection domain are not in agreement, access is impossible for an emulation memory. Refer to each device manuals for the address of chip selection domain. Since V850ES core device distributes chip selection domain automatically, it can omit specification of **CS0-CS7** option.
When the emulation memory board is not carried, **W** or **R** option cannot be specified.

Example:

```
850eserv2>map w=0x000000,0x0fffff cs0
850eserv2>map r=0x200000,0x2fffff cs1
850eserv2>map tr2=0x100000,0x1fffff
```

MODE

MODE [romless16 | romless8 | single16 | single | target]

Specifies the CPU mode using one of the following arguments:

romless16	specifies ROM-less mode with 16-bit external bus.
romless8	specifies ROM-less mode with 8-bit external bus.
single16	specifies single-chip mode with 16-bit external bus.
single	specifies single-chip mode with 8-bit external bus.
target	selects a mode by pins `MD2` to `MD0` on the target

If no argument is specified, the **MODE** command displays the current CPU mode. The arguments **romless8**, **single16**. If the target does not have mode changed this command fails.

Example:

```
850eserv2>mode single16
850eserv2>mode
Mode=SINGLE16
850eserv2>

<<In case of the device which does not have mode changed>>
850eserv2>mode
Mode change not supported
850eserv2>
```

OPBYTE

OPBYTE [<bit-position> <data>]

bit-position	Specifies Flash Option bit position in following: 0 : 0-31bit 1 : 32-63bit 2 : 64-95bit 3 : 96-127bit 4 : 128-159bit 5 : 160-191bit 6 : 192-223bit 7 : 224-255bit 8 : 256-287bit(Only V850E2 Core)
data	Specifies 32bit data in range of specified **bit-position**.
<NONE>	Displays now settings.

This command is available for V850E2Core or RH850.
OPBYTE command sets Flash Option area with V850E2 core or RH850.
For detail of Flash Option area by each device, please refer to each device manuals.

Example:

```
850eserv2>opbyte 0 0x11223344
850eserv2>opbyte 1 0x55667788
850eserv2>opbyte 2 0x99aabbcc
```

PB

PB [ON|OFF]

This command is not available with MINICUBE2 or E1/E20 emulator(Serial).
It sets up whether the circumference emulation function of ICE is stopped at the time of a break.
(Peripheral break) When there is no this function in Device, 850eserv2 displays "Peripheral break is not supported" or "Peripheral break set error 0x400: param err (illegal data)" error.

OFF	Peripheral break is not done.(default)
ON	It makes Peripheral break effective.

Example:

```
850eserv2>pb
Peripheral break disabled
850eserv2>pb on
850eserv2>pb
Peripheral break enabled
850eserv2>pb off
850eserv2>pb
Peripheral break disabled
850eserv2>
```

PINMASK

PINMASK(PIN) [K | *symbol*]

Sets masked processor pins using the following arguments:

K	unmasks (kills) all masks
symbol	choose a symbol from the list below
<NONE>	Display current masked pin

More than one *symbol* cannot be specified. Any new **PINMASK** overrides previous settings.

List of pin/mode symbols for Single Pin Mask in OCD Emulator, E1/E20 emulator(JTAG)

Symbol	Description
reset	masks reset pin and internal reset.
hwstop	masks stop pin
wait	masks wait pin
hold	masks hold pin
nmi0	masks NMI0 pin while user program runs
nmi1	masks NMI1 pin while user program runs
nmi2	masks NMI2 pin while user program runs
dbint	masks DBINT(ext break input) pin
nmi0b	masks NMI0 pin while at a break
nmi1b	masks NMI1 pin while at a break
nmi2b	masks NMI2 pin while at a break
vbresz	masks VBRESZ pin while user program runs(V850E2/ME3 only)
vbreszb	masks VBRESZ pin while at a break(V850E2/ME3 only)
cpinit	masks CPINIT pin while user program runs(V850E2/ME3 only)

cpinitb	masks CPINIT pin while at a break(V850E2/ME3 only)

List of pin/mode symbols for Single Pin Mask in MINICUBE2, E1/E20 emulator(Serial)

Symbol	Description
reset	masks reset pin and internal reset.

List of pin/mode symbols for Single Pin Mask in IECUBE

Symbol	Description
reset	masks reset pin and internal reset.
hwstop	masks stop pin
wait	masks wait pin
treset	masks external reset pin
hold	masks hold pin
nmi0	masks NMI0 pin while user program runs
nmi1	masks NMI1 pin while user program runs
nmi2	masks NMI2 pin while user program runs
nmi0b	masks NMI0 pin while at a break
nmi1b	masks NMI1 pin while at a break
nmi2b	masks NMI2 pin while at a break

List of pin/mode symbols for Single Pin Mask in IECUBE2

Symbol	Description
reset	masks reset pin and internal reset.
hwstop	masks stop pin
wait	masks wait pin

treset	masks external reset pin
hold	masks hold pin

As for nmi0,nmi1,nmi2,nmi0b,nmi1b and nmi2b basically in running and while breaking please make the same status. However, when you use with combo break, it is not necessary to use the same status.

Example:

```
850eserv2>pinmask
mask=

850eserv2>pinmask wait
850eserv2>pinmask hwstop
850eserv2>pinmask nmi1
850eserv2>pinmask nmi1b
850eserv2>pinmask
mask=
WAIT
HWSTOP
NMI1
NMI1_CB

850eserv2>pinmask k
All pinmask settings are cleared
850eserv2>pinmask
mask=

850eserv2>
```

PIO

PIO [*name*[=*newvalue*]]

Displays or changes the Programmable I/O values.

If *name* is specified: Displays the Programmable I/O specified by *name*.
If *name* and =*newvalue* is specified: Changes the value of the Programmable I/O *name* to *newvalue*.

If no arguments are specified: Displays all Programmable I/Os.

Before using this command, by all means please set the based address with **PIOBASE**. It accesses **PIO** command the address of this based address + offset. When there is no this function in Devicefile, you connot this command.

If you access Programmable I/Os without the based address in **PIOBASE**, 850eserv2 sets the based address from BPC register value automatically.

Example:

```
850eserv2>pio C0GMCS=0x1
850eserv2>pio C0GMCS
c0gmcs=0x01
850eserv2>
```

PIOBASE

PIOBASE [*address*]

Appoint the based address of the programmable I/O. When there is no this function in Devicefile, you connot this command. When there is no argument, present setting is indicated.

When you will access the programmable I/O, you have to specify setting for SFR's BPC register and the base address. 850eserv2 will access the programmable I/O, based on the base address specified by the **PIOBASE** command. By default, the base address is set at `0x0`. Please refer to each device manuals for about the address of programmable I/O and SFR's BPC register.

If you access Programmable I/Os by **PIO** or **SFR** command without the based address setting, 850eserv2 sets the based address from BPC register value automatically. If the base address by this command and BPC value are different when you access Programmable I/Os, 850eserv2 displays the warning message and use the base address to access Programmable I/Os.

Example:

```
850eserv2>sfr BPC=0x8ffb
850eserv2>piobase 0x3fec000
850eserv2>
```

PROFILE

PROFILE [data | done | set]

This command is available for IECUBE.
In case of IECUBE2 or RH850, this command is not available. Profile function is available via TraceList window. For detail of TraceList, please refer to "MULTI: Debugging".
In case of IECUBE, trace function including this command cannot be used when realtime RAM function is using. Turn off realtime RAM function to use **SW TRC** to be able to use trace function.

PROFILE set sets up the trace mode in which a profile is possible. All the trace events set up by the **TRACE** command are deleted. Moreover, trace buffer is cleared.

PROFILE data gathers and accumulates information from the trace buffer to the **850eserv**2 internal profile data buffers.

PROFILE done writes the data to an output file readable by MULTI and clears the **850eserv2** internal profile data buffers. Therefore, after each **PROFILE done** command, you must reload the target program and repeat the process (that, you must. reissue the **PROFILE data** and **PROFILE done** commands) if you want to gather more profiling data.

To use the **PROFILE** command, you must open the MULTI Profiler window before you download your target program. For further information about the MULTI Profiler, please refer to "MULTI: Debugging".

Example:

```
850eserv2>profile set
Set All Trace
Change Trace Mode(TMODE):
  Time Tag(TT=) = CPU clock / Frame time tag
  Trace Compensates(CP=) = OFF
  Trace Mode(TM=) = A(BranchPC+DataAccessPC+InstructionPC)

850eserv2>profile data
profile data: adding trace buffer to profile data...
profile data: trace buffer added.
850eserv2>profile done
profile done: profile data written.
850eserv2>
```

RMEM

RMEM [#] [K] [L=*length*] [T=*interval*]

#	ID number 0-7 is appointed. When it omitted, 0 is appointed.
L=*length*	Specifies the length of memory in bytes to be displayed. The value of length must be greater than or equal to 1, lesser than or equal to 256. The default is 128 bytes.
T=*interval*	Specifies an update interval in seconds. The default is 2 seconds.
K	Removes existing settings.
<NONE>	Display current settings.

In case of IECUBE, Realtime RAM function including this command cannot be used when trace function is using. Turn off trace function to use **SW RRAM** to be able to use realtime RAM function.

Displays the contents of realtime RAM periodically while a program is running, as specified by the following arguments. If no arguments are specified, the **RMEM** command displays the current settings.

The base address of realtime RAM must be specified with the **RRAMBASE** command. before you can use the **RMEM** command.

When a program starts to execute after the **RMEM** command has been issued, the contents of the realtime RAM region are displayed in the target pane and are updated at the specified interval. When program execution breaks, updating stops. If you resume program execution, updating restarts unless **RMEM K** is entered.

You can view up to 8 points at the same time. Unless you set an ID base address specified by **RRAMBASE**, you cannot use the ID.

In Multi-core debugging, sets Realtime RAM to the core ID specified by **RRAMBASE** command.

If you specify a short period with the **T=** option, the display of realtime RAM contents may be updated too frequently. You can stop updating with the **K** option.

When uses Realtime RAM, it sets **SYSCALLS OFF** before downloading.

When you are using 850win, realtime RAM is displayed to Realtime RAM window in 850win,and it is not displayed to MULTI pane.

Example:

```
850eserv2>rrambase 0xffff7500
850eserv2>rrambase 1 0xffff7508
850eserv2>rmem l=0x10
850eserv2>rmem 1 l=0x20 t=0x6
850eserv2>rmem
0 : length=16 interval=3
1 : length=32 interval=6
2 : Disable
3 : Disable
4 : Disable
5 : Disable
6 : Disable
7 : Disable
850eserv2>
run>
rmem 0 :
0xFFFF7500: 10 00 00 00 F8 78 0B 00 BA 78 00 00 E8 03 00 00
rmem 1 :
0xFFFF7508: D0 5E 00 00 E8 03 00 00 00 00 00 00 00 00 00 00
0xFFFF7518: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
850eserv2>
```

RRAMBASE

RRAMBASE [#] [*address*]

#	ID number 0-7 is appointed. When it omitted, 0 is appointed.
<**NONE**>	Display current settings.

In case of IECUBE, Realtime RAM function including this command cannot be used when trace function is using. Turn off trace function to use **SW RRAM** to be able to use realtime RAM function.

Sets *address* as the base address of the realtime RAM. If *address* is not specified, displays the current setting. An area of 256 bytes from the specified address will be specified.
You can set up to 8 points at the same time. Unless you set an ID base address, you cannot use **RMEM** with the ID.

In Multi-core debugging, sets Realtime RAM base address to the selecting core.

Example:

```
850eserv2>rrambase 0xffff7500
850eserv2>rrambase 1 0xffff7508
850eserv2>rrambase
Rambase is at:
0 : 0xffff7500
1 : 0xffff7508
2 : non mapping
3 : non mapping
4 : non mapping
5 : non mapping
6 : non mapping
7 : non mapping
850eserv2>
```

SFR

SFR [*name*] [l=*length*] Displays Register.

name	Displays specify name's or address's register. If ***name*** is omitted, it displays the next register from last displayed register.
l=*length*	Specifies display registers number. It displays from ***name*** to ***length*** numbers. If **l=*length*** is omitted, it displays only ***name*** register.

SFR [[s|e]=*num*] [l=*length*] Displays Register.

s=*num*	Displays ***num***th register from top of all registers.
e=*num*	Displays ***num***th register from bottom of all registers.
	If **[s|e]=*num*** is omitted, it displays the next register from last displayed register.
l=*length*	Specifies display registers number. It displays from ***name*** to ***length*** numbers. If **l=*length*** is omitted, it displays last ***length***.

SFR <*name*>=<new*value*> Changes Register.

name	Changes specify name's or address's register value.
=new*value*	New value to change.

Displays or changes SFR, Extended External I/O register or Programmable I/O register values.
The register names come from the Device specification.
If you specify ***name*** without **=** or **[s|e]=*num***, it displays **l=*length*** registers from specified register.
If you specify ***name*** with **=new*value*,** it changes ***name*** register value to **new*value***.

SFR command can not refer bit size access. The register view of MULTI debugger can refer bit size of SFR which is registered address and bit-field.

The value of SFR which value is destroyed when it reads is not displayed when **l=*length*** was specified. However, when only SFR name is specified, it can access.

If you access Programmable I/Os without the based address in **PIOBASE**, 850eserv2 sets the based address from BPC register value automatically.

Example:

```
850eserv2>sfr bpc l=3
bpc (0x03fff064)                    RW          0x0000
bsc (0x03fff066)                    RW          0x5555
vswc (0x03fff06e)                   RW          0x77
850eserv2>sfr
dsa0l (0x03fff080)                  RW          0xAF9F
dsa0h (0x03fff082)                  RW          0x0861
dda0l (0x03fff084)                  RW          0x0928
850eserv2>sfr s=0 l=2
pdl (0x03fff004)                    RW          0x0000
pdll (0x03fff004)                   RW          0x00
850eserv2>sfr l=3
pdlh (0x03fff005)                   RW          0x00
pdh (0x03fff006)                    RW          0x00
pct (0x03fff00a)                    RW          0x00
850eserv2>sfr bpc=0x8ffb
850eserv2>sfr bpc
bpc (0x03fff064)                    RW          0x8FFB
850eserv2>sfr 0x03fff064=0x8ff0
850eserv2>sfr 0x03fff064
bpc (0x03fff064)                    RW          0x8FF0
```

SHOWALL

SHOWALL(SA)

Displays the current settings of programmed events, break settings, link settings, trace conditions, CPU settings, and memory verify setting.

Example:

```
850eserv2>showall
BRS1: A=0x00100082 P=0xXX
BRS2: A=0x001000B0 P=0xXX
BRA:
<None>
Break on:
Link event:
<None>
Trace start:
BRS1: A=0x00100082 P=0xXX
Trace end (delay = <none>)
BRS2: A=0x001000B0 P=0xXX
ROM size=32 RAM size=2
NoHalt:
Disabled
Verify:
On
850eserv2>
```

SW

SW [TRC|RRAM]

TRC	Switch trace mode
RRAM	Switch realtime RAM mode
<NONE>	Display now setting.

This command is available for IECUBE.
In IECUBE, trace function and realtime RAM function have become exclusive, and cannot be used simultaneously. Trace mode and realtime RAM mode are switched by this command.

TCLEAR

TCLEAR

This command is available for IECUBE or IECUBE2.
In case of IECUBE, trace function including this command cannot be used when realtime RAM function is using. Turn off realtime RAM function to use **SW TRC** to be able to use trace function.
Clears IECUBE tarce buffer by this command.

TDISPLAY

TDISPLAY(TD) [I | F] [($|S|E|T) *rel frame*] [L=#*frames*] [> *filename* <*a*|*d*>]

I	Specifies Instruction mode.(default)
F	Specifies Data Access (or Frame) mode.
$	Sets the starting frame of the display relative to the current frame.(default)
S	Sets the starting frame of the display relative to the start of the buffer.
E	Sets the starting frame of the display relative to the end of the buffer.
T	Sets the starting frame of the display relative to the trigger point.
rel frame	Specifies the number of frames (+/-) relative to the starting frame
L=#*frames*	Sets the number of frames (max. 32768) to display. Default is 20 frames. As for maximum frames are 1048575(0xfffff).
>*filename*	Indicatory contents are written to the file. A file is possible for directory specification. The following option appointment is necessary. **a** append the data to the existing file. **d** the existing file will be deleted and then the trace data will be written to the specfied file.
<NONE>	Display some frames from current frame.

This command is available for IECUBE.
In case of IECUBE2 or RH850, this command is not available. Trace displaying is available via TraceList window. For detail of TraceList, please refer to "MULTI: Debugging".
In case of IECUBE, trace function including this command cannot be used when realtime RAM function is using. Turn off realtime RAM function to use **SW TRC** to be able to use trace function.

Displays the trace buffer. If no arguments are specified, **TD** displays frames from the current frame.
There are two trace buffer display modes: Instruction mode (**I**) and Data Access mode (**F**). In Instruction mode, a disassembly of executed instructions is displayed and no stall cycles are displayed. In Data Access mode, program data reads/writes are displayed. The default mode is Instruction mode. If no display mode is specified, the last mode specified is used.

Trace Display Status Codes

The following codes are used in the **Status** field of the **TD** trace display with Data Access mode (**F**):

`dly`	Delay Trigger frame
`sfm_S`	Starting frame of Section Trace or enable the tracer.(Command Qualify Trace)
`sfm_E`	Ending frame of Section Trace or enable the tracer.(Command Qualify Trace)
`over`	Time Stamp overflow
`dma`	DMA trace frame
`reti`	Branch by reti instruction
`jmp`	Branch by jmp instruction

With the frame where "`sfm_S`", "`sfm_E`" and "`dly`" occur only frame number and PC value are indicated and the opcode and mnemonic is not indicated.
With Data Qualify Trace, "`sfm_S`" and "`sfm_E`" are not indicated.

Default mode is Instruction mode (**I**). If display mode is not specified, the mode specified at the end is used.

When a new trace is made, the current frame is reset to the trigger point or to the end of the trace buffer if no trigger was specified. Each time a portion of the buffer is displayed, the current frame is updated to the end of the displayed data. If no starting frame number is specified, the current frame is used.

The default number of displayed frames is 20. You can change the default number with the **L** option. If you do not specify a number of frames, the last setting will be used. When displaying in Instruction mode, there will be fewer lines of disassembled code displayed than the number of frames, since instruction execution may span more than one frame for data access. Trace data can be displayed while a program is running, but you must first stop the tracer.

With the IECUBE, two instructions can be displayed in one frame because the IECUBE can execute two instructions in one clock.

By default, the **Time** field of the trace display shows the number of CPU clocks that have elapsed from the previous frame to the current frame. If you specify **TMODE TS=T**, actual time is displayed instead of the number of clocks. To get the proper time value, set the clock frequency and scalar value with the **TIMEBASE** command.

Display format	Time unit
XX:XX':XX" h	hours: minutes: second
XX".XXX s	second.MS
XXX.XXX ms	MS.US
XXX.XXX us	US.NS
XXX.XXX ns	NS.PS

When indicating time tag, by all means please input the operational frequency and dividing ratio of CPU beforehand in **TIMEBASE** command. It calculates the time when it is indicated in operational frequency and the dividing ratio which are input with **TIMEBASE**.

Error in measured time

The run time can be measured for a longer time by setting the frequency dividing rate; however, error will arise correspondingly.

For example, the run time for a certain frame has finished in one clock. In this case, if you set 33 MHz for the operating frequency and X1 for the frequency dividing rate, ICE will return 1 as the time tag counter value and 30.303 ns will be displayed as the run time.

If you set X4 for the frequency dividing rate in the above case, ICE will return 1 as the time tag counter value, like the case of X1, and the run time will be 121.212 ns. This difference will cause the error to arise in changing the frequency dividing rate.
Similarly, also if you change the frequency dividing rate to X8, X16, X32..., X1M, the counter value will always be 1 and the error will arise correspondingly.
In addition, because the frequency dividing rate you can specify is 2n, this will also cause error.
Error in trace lead time tag will arise a small amount.
If the time tag is overflowed in access mode, “over” will appear.

Trace data on IECUBE

* The trace time tag counter restarts counting from zero when an overflow occurs. Therefore, the time tag after “over” does not indicate the correct time since the trace started. Only a rough value is displayed rather than the accurate value for the reason of the hardware specification. If you set a hardware or software break during execution, error will arise in time tag because the CPU will stop momentarily.

* Timer display is not available for any trace frame complemented in completion mode.If you specify TC=R option with the TMODE command, dropping of trace data may occur. Frames where dropping has occurred will be indicated as <Lost Data>. Frames at fetch addresses canceled by an interruption or at fetch addresses related to the debug monitor will be indicated as <Invalid Data>.

* In IECUBE, two kinds can be chosen as trace time tag mode, external clock used for measuring time and CPU clock. Although external clock used for measuring time is fixation in 50MHz, CPU clock can set up frequency and clock source division ratio by the TIMEBASE command. Since one count is four clocks when it measures with a CPU clock, the value of 4 count or less of trace data cannot be measured. Therefore, a count value may be displayed as 0x0.

* The trace time tag represents the time required to read into the IECUBE trace memory the divided frame information sent from the evaluation chip. Because the frame information from this chip is stored in the queue in the chip, it does not always represent the accurate time; you should use it as a guide. The clock count displayed is the value obtained by dividing this time of day by 20 ns (50 MHz), and is unrelated with the actual CPU clock. In addition, note that if you set a hardware or software break during execution, error will arise in time tag because the CPU will stop momentarily.

If you specify "> filename", the tracing results will be written to filename. A file is possible for directory specification.
Be sure to insert one or more spaces before and after the ">". Please specify "a" or "d" after "filename". If you want to append the data to the existing file, specify "a". If you specify "d", the existing file will be deleted and then the trace data will be written to the specified file. However, when the specified file does not exist, "a" or "d" can be omitted.

Example:

```
850eserv2>td s0 l=3

|FRAME|   TIME   |    PC    | OPCODE |   ADDRESS   |   DATA   | MNEMONIC
-------------------------------------------------------------------------------
|  0  |    18    |0x0010009E|ff8000f8|             |          |jarl 0x100196,lp
|  1  |     1    |0x00100196|  1a5c  |             |          |add -4, sp
|  2  |     1    |0x00100198|ff630001|WD 0x00114460|0x001000A2|st.w lp, 0[sp]

850eserv2>tmode ts=t
850eserv2>td s0 l=3

|FRAME|   TIME   |    PC    | OPCODE |   ADDRESS   |   DATA   | MNEMONIC
-------------------------------------------------------------------------------
|  0  |719.999 ns|0x0010009E|ff8000f8|             |          |jarl 0x100196,lp
|  1  | 39.000 ns|0x00100196|  1a5c  |             |          |add -4, sp
|  2  | 39.000 ns|0x00100198|ff630001|WD 0x00114460|0x001000A2|st.w lp, 0[sp]

850eserv2> td s0 1=100 > trc.txt
850eserv2>
```

TFILTER

TFILTER(TF) [I= [T][O] | . | *] [F= [T][E][C][O][A][D][S][M] | . | *]

This command is available for IECUBE.
In case of IECUBE2 or RH850, this command is not available. Trace filtering is available via TraceList window. For detail of TraceList, please refer to "MULTI: Debugging".
In case of IECUBE, trace function including this command cannot be used when realtime RAM function is using. Turn off realtime RAM function to use **SW TRC** to be able to use trace function.

Specifies or displays what is filtered out before display by **TDISPLAY** in the trace buffer. This command affects only the trace display and does not affect the trace buffer. If no options are specified, **TFILTER** displays the current filter settings.

Use the **I** or **F** options to specify which fields and bus status types are displayed, where the letters following the equal sign (=) indicate the fields and status types, as described below. Some fields are permanent and cannot be removed. Any previous setting of the **I**= or **F**= argument will be overwritten.

For Instruction mode, there are two possible fields that can be controlled with the **I**= argument. The **T** option specifies time-tag and the **O** option specifies opcode. The frame number, PC address, and mnemonic fields are permanent. To enable all of the Instruction mode fields, use **I**=*. To disable all but the permanent fields, use **I**=..

For Data Access (or Frame) mode, there are eight possible fields that can be controlled with the **F**= command. The **T** option specifies time-tag, the **E** command specifies external probe data, the **C** option specifies program counter information, the **O** option specifies opcode, the **A** option specifies address, the **D** option specifies data, the **S** command specifies status, and the **M** command specifies mnemonic. The frame number field is permanent. To enable all of the Data Access mode fields, use **F**=*. To disable all but the permanent fields, use **F**=..

Example:

```
850eserv2>tfilter i=o
850eserv2>tfilter

Trace Filter:

Instruction mode:
     Frame:  Permanent
      Time:  Disabled
        PC:  Permanent
    Opcode:  Enabled
  Mnemonic:  Permanent

Frame mode:
     Frame:  Permanent
      Time:  Enabled
 Ext Probe:  Enabled
        PC:  Enabled
    Opcode:  Enabled
   Address:  Enabled
      Data:  Enabled
  Mnemonic:  Enabled
    Status:  Enabled

850eserv2>td s0
|FRAME|     PC     |  OPCODE  |     ADDRESS    | DATA | MNEMONIC
-------------------------------------------------------------------------------
|  0  | 0x001001E8 | 87110000 | RD 0x03FFE000 | 0x91 | ld.b 0[r17], r16
|  1  | 0x001001E8 | 87110000 | RD 0x03FFE000 | 0x92 | ld.b 0[r17], r16
|  2  | 0x001001E8 | 87110000 | RD 0x03FFE000 | 0x93 | ld.b 0[r17], r16
|  3  | 0x001001E8 | 87110000 | RD 0x03FFE000 | 0x94 | ld.b 0[r17], r16
|  4  | 0x001001E8 | 87110000 | RD 0x03FFE000 | 0x95 | ld.b 0[r17], r16
|  5  | 0x001001E8 | 87110000 | RD 0x03FFE000 | 0x96 | ld.b 0[r17], r16
850eserv2>
```

TIMEBASE

TIMEBASE [C=*clock*] [R=*rate*] [T=*rate*]

This command is available for MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core, IECUBE or IECUBE2.
Sets the frequency of the CPU clock and the scalar value of the trace time-tag counter or timer counter. If no arguments are specified, the **TIMEBASE** command displays the current settings.

The **C=** command specifies the frequency of trace time tag clock or timer clock in MHz. For example, if the clock frequency of your CPU is 25 MHz, you can specify this with **C=25**. In case of IECUBE, default value is 50. In case of MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core or IECUBE2, default value is 200.

In case of IECUBE, when using CPU clock by trace time-tag counter(specified **TMODE TT=CC**,**AC)**, it is necessary to set CPU clock. When using External clock for time measurement by trace time-tag counter(specified **TMODE TT=AE)**, it is not necessary to change a setup into 50MHz for fixation.
In case of IECUBE2 or RH850, if **C=** is specified, RUN-BREAK is measured by the RCU JTAG clcok and timer events are measured by specified CPU clock. For detail of the RCU JTAG Clcok, please refer to ICE's manual. If **C=** is not specified, it measures by time measurement clock(200MHz).

 Please set up the same value as the frequency of Target CPU of operation.

The **R=** option specifies the scalar value of the trace time-tag counter, which counts the number of clocks for each trace frame. The **R=** can not be specified in MINICUBE+V850E2Core, MINICUBE2+ V850E2Core, E1/E20 emulator+V850E2Core or IECUBE2.
If **R=X1** is specified, 1 CPU clock increases the counter by 1. If **R=X2** is specified, 2 CPU clocks increase the counter by 1. If **R=X1K** is specified, 1024 CPU clocks increase the counter by 1. In short, the larger the value of *rate*, the longer timer value the counter can hold. The counter holds at least 1 any time trace data is displayed, which can cause a loss of precision. **XQ** is same as **X1**, however if you specify **XQ** and clock mode is CPU clock mode(**TMODE TT=AC,CC**), the time-tag's accuracy becomes more precisely.

The following mnemonics can usually be specified as arguments to **R=**:
X1, X2, X4, X8, X16, X32, X64, X128, X256, X512, X1K, X2K, X4K, XQ.

The **T=** option specifies the scalar value of the timer counter.
If **T=X1** is specified, 1 CPU clock increases the counter by 1. If **T=X2** is specified, 2 CPU clocks increase the counter by 1. If **T=X1K** is specified, 1024 CPU clocks increase the counter by 1. In short, the larger the value of *rate*, the longer timer value the counter can hold.
In case of IECUBE2, **T=** option can be secified without **C=**.

The following mnemonics can usually be specified as arguments to **T=**:
X1, X2, X4, X8, X16, X32, X64, X128, X256, X512, X1K, X2K, X4K.

Example:

When you set the frequency of CPU clock to 33.33, the scalar value of the trace time-tag counter to X4, and the scalar value of the timer counter to X2.

```
850eserv2>timebase c=33.33 r=x4 t=x2
CPU Clock = 33.33MHz
Trace Timetag Rate = X4
Timer Rate = X2
850eserv2>
```

TIMER

Displays the current timer value as a number of clocks. The number of execution clocks until RUN-BREAK: takes a break from execution is displayed.
TIMER1-TIMER7 are as a result of timer event set up by the TMEVENT command.
The event and clock source division ratio used as measurement conditions are displayed.
A measurement result is displayed with the applied total number of clocks, and is cleared and measured for every program execution.
When the timer event is not being used, it is displayed as ***** no use *****.

To get the proper time value, set the clock frequency and scalar value with the **TIMEBASE** command.
The format at the time display is the following:

Display format	Time unit
XX:XX':XX" h	hours: minutes: second
XX".XXX s	second.MS
XXX.XXX ms	MS.US
XXX.XXX us	US.NS
XXX.XXX ns	NS.PS

<<In case of OCD emulator+V850E1Core,
E1/E20 emulator(JTAG)+V850E1Core>>

Measures by 20MHz(When use **-2m** starting option, measures by 10MHz).
Only the time from RUN to BREAK displays. It can display also during execution.

<<In case of MINICUBE2+V850E1Core,
E1/E20 emulator(Serial)+V850E1Core>>

Measures by 10MHz(When use **-dck20** starting option, measures by 20MHz).
Only the time from RUN to BREAK displays. It cannot display during execution.
Clock count becomes 0 after Step.

<<In case of IECUBE>>

Measures by 50MHz.
TIMER1-TIMER7 become effective.When not specifying the link event, an link event name is not displayed. Total(addition time count), Max(the maximum section time count), Min(the minimum section time count), Average(average time count), and Pass(number of times of section passage) are displayed. When it overflows, it is displayed as ***over***. When Max overflows, after overflow because it does again to count from 0, one time passed designated section accurate count value of coming becomes uncertainty. Therefore, it is displayed on Total as 0x?????????????.
When Total, Max and Pass overflow, it is displayed because 0x?????????? Average becomes an unfixed value. When a timeout break occurs by the number of section passage (**CP** of **TMEVENT** command) set up in the timer event, it is displayed on Pass as ***break***. When a timeout break occurs by the

section time count (**CT** of **TMEVENT** command) set up in the timer event, it is displayed on Max as ***break***. It can display also during execution.

<<In case of MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core, E1/E20 emulator+RH850, IECUBE2>>

If CPU clock is specified by **TIMEBASE** command, RUN-BREAK is measured by the RCU JTAG Clcok(In serial connecting, RUN-BREAK is measured by 10MHz) and timer events are measured by specified CPU clock. For detail of the RCU JTAG clcok, please refer to ICE's manual. If CPU clock is not specified, it measures by time measurement clock(200MHz). TIMER1 become effective.

Notes of timer measurement

When there is a break point which the MULTI side sets up internally all over the measurement section of timer, in order to break at once internally, a timer is cleared and it becomes impossible to measure a timer. MULTI usually sets a breakpoint as the .syscall section. Please refer to "MULTI: Building Applications for Embedded V800" for the details of the .syscall section.

By the breakpoint set up internally, it breaks at once and RUN again after that. Before RUN then, the timer is cleared as initialization processing. Since a timer will be cleared here, timer measurement becomes impossible. RUN-BREAK and a timer event are cleared.

For this reason, in RUN-BREAK, the execution time from the breakpoint which MULTI set up internally is displayed. Since in the case of a timer event it breaks before coming to a timer end, execution time cannot be measured.

If you set **SYSCALLS OFF** command before download, it will not set up breakpoint internally at the top of .syscall section.

Example:

<<In case of OCD emulator+V850E1/ES Core, E1/E20 emulator(JTAG)+V850E1/ES Core>>

```
RUN-BREAK : rate( x2) = 10".099 s(0x06051c35 Clocks)
DCK 20MHz
```

<<In case of MINICUBE2+V850E1Core, E1/E20 emulator(Serial)+V850E1Core>>

```
RUN-BREAK : rate = 9".988 s(0x0000000000018628 Clocks)
```

<<In case of IECUBE>>

* When Total overflows (Pass is surely overflowing.)

```
RUN-BREAK : rate( x1) = 00:02':51" h(0x00000001ffffffff Clocks)
TIMER1    :START=BRS1 END=BRS2  rate( x1)
Total  =      5".101  s(0x000000f341349 Clocks)  ***over***
Max    =      20.020 us(0x0000003e9 Clocks)
Min    =      19.979 us(0x0000003e7 Clocks)
Average=     0x????????? Clocks
Pass   =     0x00000ffff Counts  ***over***
...
TIMER7    : ***** no use *****
Time Measurement Clock 50MHz
```

* When Max overflows

```
RUN-BREAK : rate( x1) = 00:02':51" h(0x00000001ffffffff Clocks)
TIMER1    :START=BRS1 END=  rate( x1)
Total  =     0x???????????? Clocks  ***over***
Max    =     00:02':51" h(0x1fffffff Clocks)  ***over***
Min    =     26".768  s(0x04fc6aab9 Clocks)
Average=     0x????????? Clocks
Pass   =     0x000000001 Counts
...
TIMER7    : ***** no use *****
Time Measurement Clock 50MHz
```

* When Pass overflows

```
RUN-BREAK : rate( x1) = 10".080  s(0x000000001e0aadf8 Clocks)
TIMER1    :START=BRS1 END=BRS2  rate( x1)
Total  =      1".997  s(0x0000005f3a0ea Clocks)
Max    =       4.219 us(0x0000000d3 Clocks)
Min    =       4.179 us(0x0000000d1 Clocks)
Average=     0x????????? Clocks
Pass   =     0x00000ffff Counts  ***over***
...
TIMER7    : ***** no use *****
Time Measurement Clock 50MHz
```

<<In case of MINICUBE2+V850E2Core, E1/E20 emulator(Serial)+V850E2Core>>

* When uses RUN-BREAK

```
RUN-BREAK :  1".681  s(0x00000000000041aa Clocks)

TIMER1    : ***** no use *****
CPU Clock 200.00MHz
```

* When uses timer event

```
RUN-BREAK : 994.100 ms(0x00000000000026d5 Clocks)

TIMER1    :START=BRS1 END=BRS2  rate( x1)
Total  =      40.550 ms(0x00000007bc065 Clocks)
Max    =      10.444 us(0x000000829 Clocks)
Min    =       9.864 us(0x0000007b5 Clocks)
Average=      10.014 us(0x0000007d3 Clocks)
Pass   =     0x000000fd0 Counts

CPU Clock 200.00MHz
```

<<In case of MINICUBE+V850E2Core, E1/E20 emulator(JTAG)+V850E2Core, IECUBE2>>

* When uses RUN-BREAK

```
850eserv2> timer
RUN-BREAK :  4".799  s(0x00000000016e315b Clocks)
RCU JTAG Clock

TIMER1    : ***** no use *****
CPU Clock 200.00MHz
```

* When uses timer event

```
RUN-BREAK :  2".252  s(0x0000000000abe199 Clocks)
RCU JTAG Clock

TIMER1    :START=BRS1 END=BRS2  rate( x1)
Total  =      96.567 ms(0x000000126b32b Clocks)
Max    =      19.004 us(0x000000ed9 Clocks)
Min    =      18.969 us(0x000000ed2 Clocks)
Average=      18.975 us(0x000000ed3 Clocks)
Pass   =     0x0000013e1 Counts

CPU Clock 200.00MHz
```

TMEVENT

TMEVENT[#[T=linkevent][S=startevent][E=endevent][R=tidiv]
[CP=cplimit][CT=ctlimit][CTMIN=ctmin_limit]][D #]

#	Specifies a sequential number of a time event. In IECUBE, you can appoint 1-7. In V850E2Core, you can appoint 1 only.
S=*startevent*	Specifies a start event name.(Only a single event can be specified)
E=*endevent*	Specifies a start event name.(Only a single event can be specified)
T=*linkevent*	Specifies a sequential event name (linka) for timer start/stop. This option is available for IECUBE.
R=*tidiv*	Specifies the scalar value of the timer counter. This option is available for IECUBE. Each timer events can not set individual scalar value. You can use the following values: **X1**, **X2**, **X4**, **X8**, **X16**, **X32**, **X64**, **X128**, **X256**, **X512**, **X1K**, **X2K**, and **X4K**.
CP=*cplimit*	A break is taken by the specified number of section passage. In the case of 0, it does not set up. In IECUBE, it cannot specify simultaneously with **CT**. It can be specified 0x1-0xfffe. In V850E2Core, it can be specified 0x1-0xffffffff.
CT=*ctlimit*	A break is taken by the specified section time count. In the case of 0, it does not set up. In IECUBE, it cannot specify simultaneously with **CP**. It can be specified 0x1-0xfffffffe. In V850E2Core, it can be specified 0x1-0xffffffff.
CTMIN= *ctmin_limit*	A break is taken by the specified section time minimum count. (Only V850E2Core) In the case of 0, it does not set up. It can be specified 0x1-0xffffffff.
D #	Specifies deletion of an event.

Cannot specify before execution event and address range event to ***startevent***, ***endevent***, or ***linkevent***.

This command is available for MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core, IECUBE or IECUBE2(V850E2Core).
In Multi-core debugging, you can set timer events in each cores. You can specify the BRA, BRS events set in this core. The events set in other core can not be specified.

Sets and displays timer events. You can set up timer events apart from the run time measurement timers, because the timer events are not shared with the run time measurement timer. **S=** and **E=** specify a single event. **T=** specifies a sequential event. **T=** and **S=**,**E=** cannot be specified simultaneously.

In case of MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core or IECUBE2, you can specify only **S=t** specified event via **BRS** command.
However, clock source division ratio cannot be set up individually. Timer measurement is performed by clock source division ratio set up at the last.
CP= takes a break by the specified number of section passage. **CT=** takes a break at the specified section time count. In IECUBE, it becomes an error when **CP=** and **CT=** are specified simultaneously.
CTMIN= can be specified in MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core or IECUBE2, it takes a break at the specified section time minimum count.

The timeout break of a section time count takes a break later than set-up ct value. Since a break is taken synchronizing with a CPU clock, and the error of a CPU clock is included, it happens. If a CPU clock is set up at high speed, since the error of a CPU clock will become small, a gap of the count value when taking a break with the set-up time count value becomes comparatively small.

Example:

<<In case of IECUBE>>

```
850eserv2>brs 1 a=0x79a
850eserv2>brs 2 a=0x7aa
850eserv2>link a 1=brs1 2=brs2
850eserv2>tmevent 1 t=linka r=x1 cp=0x100 ct=0x200
850eserv2>tmevent 2 s=brs1 e=brs2 r=x1
850eserv2>
```

<<In case of MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core, IECUBE2>>

```
850eserv2>brs 1 a=0x79a s=t
850eserv2>brs 2 a=0x7aa s=t
850eserv2>tmevent 1 s=brs1 e=brs2 ct=0x200
850eserv2>
```

TMODE

TMODE [M=F|N|B][TS=C|T][DMA=ON|OFF][TC=R|F][TT=CC|AC|AE][CP=ON|OFF]
[TM=B|A|D|BD|DP|BDP|AD](IECUBE)
TMODE [M=F|N|B][DMA=ON|OFF][TC=R|F][TM=B|BR|BW|BRW|R|RW|W]
[DMASTAT=ON|OFF][DMAMODE=DMA1|DMA2|DMA3|DMA4|DMA5](IECUBE2)

M= Specify the trace memory control mode

- ***N*** Non-stop mode (default)
- ***F*** Full stop mode
- ***B*** Full break mode
- ***D*** Delay stop mode
- ***DB*** Delay break mode(IECUBE2 only)

TS= Specify the format of TIME displayed in TDISPLAY(IECUBE only)

- ***C*** Display in time tag counter format (default)
- ***T*** Display in execution time format

DMA= Specify which acquires DMA trace frame

- ***ON*** DMA trace is acquired
- ***OFF*** DMA trace is not acquiredt

DMASTAT= Specify to display DMA trace status message(IECUBE2 only)

- ***ON*** Displays DMA trace status message
- ***OFF*** Does not displays DMA trace status message (default)

DMAMODE= Selects DMA trace mode(IECUBE2 only)

DMA1	Read/Write+Transfer Address+Transfer size+Transfer data(Default)
DMA2	Read/Write+Transfer Address+Transfer size
DMA3	Read/Write+Channel number
DMA4	Read/Write+Channel number+Transfer count
DMA5	Read/Write+Transfer data+Channel number+Transfer count

TC= Specify the trace control mode

R	Real-time trace mode (default in IECUBE)
F	Complete trace mode (default in IECUBE2)

TT= Specify the time tag mode(IECUBE only)

CC	Clear the time tag for every frame based on the CPU clock
AC	Accumulate time tags across frames based on the CPU clock.
AE	Clear the time tag for every frame based on an external clock used for measuringtime (default)

CP= Complements the interval between branch traces(IECUBE only)

ON	Complements the interval between branch traces (default).
OFF	Does not complement the interval between branch traces.

TM= **(IECUBE)**	Selects collectable trace packets.	
	B	BranchPC
	A	BranchPC+DataAccessPC+InstructionPC
	D	DataAddress+Data
	BD	BranchPC+DataAddress+Data (default)
	DP	DataAddress+Data+DataAccessPC
	BDP	BranchPC+DataAddress+Data+DataAccessPC
	AD	BranchPC+DataAccessPC+InstructionPC+DataAddress+Data
TM= **(IECUBE2,** **RH850)**	Selects collectable trace packets.	
	B	BranchPC and instruction between branch
	BR	BranchPC and instruction between branch + Data read access
	BW	BranchPC and instruction between branch + Data write access
	BRW	BranchPC and instruction between branch + Data read access + Data write access (default)
	R	Data read access
	RW	Data read access + Data write access
	W	Data write access
	RD (RH850)	Data read access + Data access PC
	RWD (RH850)	Data read access + Data write access + Data access PC
	WD (RH850)	Data write access + Data access PC
<NONE>	Displays current setting	

This command is available for IECUBE or IECUBE2.
In case of IECUBE, trace function including this command cannot be used when realtime RAM function is using. Turn off realtime RAM function to use **SW TRC** to be able to use trace function.

Please set trace configurations via Trace Options window in MULTI. You can open Trace Options window via "Config > Target Specific Options" menu in TraceList window. Trace configurations that is not able to set via Trace Options window can be set via this command. When you set trace configurations via Trace Options window, the setting of this command is set automatically. For detail of Trace Options window, please refer to "MULTI: Debugging".

Trace configurations of Trace Options window is set automatically when opens TraceList window, turns trace on and downloads program, and setting of this command is updated in this timing.

Specifying **M=N** sets trace memory to nonstop mode, which is the default mode. In nonstop mode, the tracer overwrites old trace data if the trace buffer becomes full and continues to trace until a **TSTOP** command is entered. Specifying **M=F**, sets trace memory to full stop mode. In full stop mode, the tracer stops tracing when the trace buffer becomes full. Specifying **M=B**, the target stops when the trace buffer becomes full. In case of IECUBE2 with QB-V850E2-SP, you can not specify **M=B**. Specifying **M=D**, the tracer stops tracing after collecting some trace frames when delay trigger is hit. You can specify collecting trace frames number when delay trigger is hit via **TRACE** command. In IECUBE, When you specify **T=** with the **TRACE** command, **M=D** will be specified automatically. In IECUBE2 or RH850, When you set "Delay Trace" via Set Trigger dialog in TraceList window, **M=D** will be specified automatically. Specifying **M=DB**, the target stops after collecting some trace frames when delay trigger is hit. **M=DB** can be specified in IECUBE2 and **M=D** or **M=DB** can be specified while delay trigger is set via **TRACE** command or Set Triggers dialog in TraceList window. **M=N**, **M=F** or **M=B** cannot be specified while delay trigger is set. In RH850, could not select **M=F** and **M=D** with **TC=F**. In RH850 + E1/E20 emulator, could not select **M=DB**.

TS= can be specified with IECUBE.
If **TS=C** is specified, counts of the time-tag counter are displayed in the TIME field of **TDISPLAY**. If **TS=T** is specified, the time-tag is displayed in real-time units of microseconds, milliseconds, seconds or minutes.
For IECUBE, only a rough value is displayed rather than the accurate value for the reason of the hardware specification. If you set a hardware or software break during execution, error will arise in time tag because the CPU will stop momentarily.

Specifying **DMA=ON** is specified, it acquired the DMA trace. If **DMA=OFF** is specified, it not acquired the DMA trace. In IECUBE, default is **DMA=ON**. In IECUBE2, default is **DMA=OFF**, and **DMA=ON** is set automatically when you turn on "DMA Read" or "DMA Write" via Trace Options window in TraceList.
In IECUBE2, trace Start, Stop or Qualify condition cannot set via **TRACE** command or Set Triggers dialog in TraceList window while **DMA=ON**.

DMASTAT= and **DMAMODE=** can be specified with IECUBE2 and **DMA=ON**.
DMASTAT=ON displays status of DMA transfer is running and finished. Default is **DMASTAT=OFF**.
DMAMODE= sets DMA trace mode. It can specify **DMA1-DMA5**, each setting can acquire and display following DMA trace message.

Setting	Read/ Write	Address	Data size	Data	Channel	Count
DMA1	Enable	Enable	Enable	Enable	Disable	Disable
DMA2	Enable	Enable	Enable	Disable	Disable	Disable
DMA3	Enable	Disable	Disable	Disable	Enable	Disable
DMA4	Enable	Disable	Disable	Disable	Enable	Enable
DMA5	Enable	Disable	Disable	Enable	Enable	Enable

When **TC=R** is appointed, the taking dropping of trace occurs. When **TC=F** is appointed, when the taking dropping of trace becomes may occur, stopping the execution pipeline of CPU inside, temporarily in order to process, the real time characteristic for the user program is lost.

TT= can be specified with IECUBE.
Use **TT=** to specify the time tag mode. If you specify **CC** or **AC**, the time tag will be calculated with the CPU clock you set with **TIMEBASE** and the value will be displayed as the time tag in trace data. Since one count is four clocks when it measures with a CPU clock(**CC**,**AC**), the value of 4 count or less of trace data cannot be measured. Therefore, a count value may be displayed as 0x0. If you specify **AE**, the time tag will be measured with the external clock for time measurement (50MHz) and data will be accumulated from frame to frame. The trace time tag represents the time required to read into the IECUBE trace memory the divided frame information sent from the evaluation chip. Because the frame information from this chip is stored in the queue in the chip, it does not always represent the accurate time; you should use it as a guide. The clock count displayed is the value obtained by dividing this time of day by 20 ns (50MHz), and is unrelated with the actual CPU clock. In addition, note that if you set a hardware or software break during execution, error will arise in time tag because the CPU will stop momentarily.

CP= can be specified with IECUBE.
IF you specify **CP=ON**, frames between branch addresses will be complemented. The complemented frames dose not indicate the Time Tag. This setting is valid for the mode which acquires branch instruction (**B**, **BD**, and **BDP**) with **TM** options only.

Selects collectable trace packets via **TM=**. Trace data to be collected varies depending on the trace mode setting. In case of IECUBE, while **S**, **E**, **Q**, or **T** is specified with the **TRACE** command, you cannot specify **A** or **AD**. If you specify **S**, **E** or **Q** with the **TRACE** command in **A** or **AD** action mode, **A** and **AD** will be change to **B** and **BD** respectively.

Example:

<<In case of IECUBE>>

* Only branch address paying attention with all trace, when trace is acquired

```
850eserv2>trace a
850eserv2>tmode tm=b
```

* Only branch address paying attention with section trace, when trace is acquired

```
850eserv2>brs 1 a=foo1
850eserv2>brs 2 a=foo2 (foo1,foo2 is function name)
850eserv2>trace s=brs1 e=brs2
850eserv2>tmode tm=b
```

* Only branch address paying attention with trace enable (qualify trace), when trace is acquired

```
850eserv2>brs 1 a=foo (foo is function name)
850eserv2>trace q=brs1
850eserv2>tmode m=b cp=on tm=bdp
850eserv2>tmode
Trace Mode:
Trace Type = All Trace
Trace Memory(M=) = Full Break
DMA Trace(DMA) = On
Time Stamp(TS=) = Count
Trace Cycle(TC=) = Real time
Trace Compensates(CP=) = ON
Trace Mode(TM=) = BDP(BranchPC+DataAddress+Data+DataAccessPC)
850eserv2>
```

<<In case of IECUBE2>>

* Collects Status, Read/Write, Transfer Address, Transfer size and Transfer data in DMA trace mode.

```
850eserv2>tmode dmastat=on dmamode=dma1
```

* Collects Read/Write, Transfer data, Channel number and Transfer count in DMA trace mode.

```
850eserv2>tmode dmastat=off dmamode=dma5
```

TRACE

TRACE(T) [A | [S|E|Q|T]=*event_expression*][D=*delay*][K]

A	Tracing all frames
S=*event_expression*	Specifies the event(s) and/or link(s) that start of Section Trace.
E=*event_expression*	Specifies the event(s) and/or link(s) that stop (end) of Section Trace.
Q=*event_expression*	Specifies the event(s) that enable or disable the tracer. (Qualify Trace)
T=*event_expression*	Specifies the trigger event(s) and/or link(s) that halt the tracer after a delay point.
D=delay	Selects getting frames after a delay point. **F** Maximum trace buffer **M** half of trace buffer **L** About some frames (default)
K	Clear all trace setting
<NONE>	Display current setting

 Cannot specify before execution event to ***event_expression***.

This command is available for IECUBE.
In case of IECUBE, trace function including this command cannot be used when realtime RAM function is using. Turn off realtime RAM function to use **SW TRC** to be able to use trace function.
In case of IECUBE2 or RH850, this command is not available. Trace conditions can be set via Set Triggers dialog in TraceList window. For detail of TraceList, please refer to "MULTI: Debugging".

Specifies under what conditions tracing occurs. If no arguments are specified, the **TRACE** command displays the current trace settings. Setting the **K** option clears (kills) all trace settings.
By default, the tracer will trace all frame transactions. You may reset the tracer to this default mode by entering **TRACE A**. You can also set the tracer to trace only at specified places in your code. You can start and stop the tracer with the **S** (start) and **E** (stop) arguments, enable or disable the tracer with the **Q** (qualify) argument, and/or halt the tracer after the specified delay time with the **T** (trigger) argument.
In IECUBE2, **S** (start), **E** (stop) or **Q** (qualify) cannot be specified while DMA trace is enable.

The value of ***event_expression*** can be one or more event and/or link mnemonic(s) separated by a pipe character (|). For example, `BRA1|LINKA|BRS2`.

You can assign any or all of the events and/or the link specifications to the **T**, **S**, and **E** inputs. The start and stop inputs are edge-triggered, which means that once the event(s) occur, tracing will start or stop.

Q inputs may only be tied to events. The qualifying input is level-triggered, which means that tracing only occurs if the event(s) are true; once the event(s) are no longer true, tracing will cease. The start/stop and qualify conditions are ORed together.

The **T** (trigger) option programs the tracer to stop after an event or link occurs. You can delay the halting of the tracer by specifying **D=*delay***. This specifies how long after a trigger event the tracer will halt, where ***delay*** is the number of frames you want to record after the trigger. For the ICE specification, you can only set a rough delay value, which is the amount to be traced in relation to the maximum trace buffer. You can only specify **D=** as **L** (about some frames), **M** (half of trace buffer), or **F** (maximum trace buffer). If you set a trigger event, **M** option of the **TMODE** will be automatically specified as **D** (delay stop).
In case of IECUBE2 with QB-V850E2-SP, you can not specify **T**.

Example:

```
850eserv2>TRACE A
850eserv2>TRACE
Trace ALL
850eserv2>TRACE K
Trace events are cleared .
850eserv2>LINK
Link:
<None>
850eserv2>TRACE S=BRS2 E=BRS1
850eserv2>trace
Trace start:
BRS1: A=main <0x00100070> P=0xXX
Trace end (delay = <none>)
BRS2: A=demoC <0x00100084> P=0xXX
850eserv2>trace k
850eserv2>LINK a 1=brs1 2=brs2
850eserv2>trace r=linka
850eserv2>
```

TRUN

This command is available for IECUBE or IECUBE2.
In case of IECUBE, trace function including this command cannot be used when realtime RAM function is using. Turn off realtime RAM function to use **SW TRC** to be able to use trace function.
Restarts the trace analyzer if your program is running but the trace analyzer is stopped.

TSIZE

tsize [size]

size	The size of trace buffer is specified. In case of IECUBE, following sizes can be specified: 8K,32K,64K,128K,256K(default is 8K) In case of IECUBE2, following sizes can be specified: 8K,32K,64K,128K,256K,512K(default is 512K) In case of IECUBE2 with QB-V850E2-SP, following sizes can be specified: 8K,32K,64K,128K,256K,512K,1M,2M,4M,8M,16M,32M,64M,128M (default is 512K)
<none>	Display current setting.

This command is available for IECUBE or IECUBE2.
In case of IECUBE, trace function including this command cannot be used when realtime RAM function is using. Turn off realtime RAM function to use **SW TRC** to be able to use trace function.

The number of frames which the trace memory in ICE actually uses is set up.
If the number of frames to be used is large, reading of trace data will take time,such as **TDISPLAY**, However, since 1-4 frames is stored in one memory, the trace data of 256K-256 K*4 frame range is displayed.

Example:

```
850eserv2>tsize 128k
850eserv2>tsize
TraceMemorySize : 128K
850eserv2>
```

TSTOP

TSTOP(TS)

This command is available for IECUBE or IECUBE2.
In case of IECUBE, trace function including this command cannot be used when realtime RAM function is using. Turn off realtime RAM function to use **SW TRC** to be able to use trace function.

Halts the trace analyzer but keeps the CPU Evachip running. The emulator mode will switch from TRACE to RUN mode. Once in RUN mode, you can change the events, links, break, and trace settings, or display the contents of the trace buffer. Use **TRUN** to restart the analyzer.

TSEARCH

TSEARCH[frame][A=address_or_range][x=data/mask][C=status][L= frame]

This command is available for IECUBE.
In case of IECUBE2 or RH850, this command is not available. Trace searching is available via TraceList window. For detail of TraceList, please refer to "MULTI: Debugging".
In case of IECUBE, trace function including this command cannot be used when realtime RAM function is using. Turn off realtime RAM function to use **SW TRC** to be able to use trace function.

Searches the trace buffer for a particular type of frame information and displays, in Frame mode, the frames that match the specified conditions. The search conditions consist of an address range, a data/mask value, a bus status type, and an external probe value, as described below. You can enter a subset of these conditions, in which case the other conditions will be ignored. If you do not specify any conditions, the last specified set of conditions will be used. The arguments for setting the conditions are described below.

* The *frame* argument specifies the starting frame for the search. It is the offset from the top (the oldest frame) of the trace buffer, not the frame number that is displayed with a **TDISPLAY** command. That is, frame number 0 specifies the oldest frame. If a frame number is not specified, the search begins from the frame next to the current trace pointer, not from the last position of the previous search. If a frame that matches the specified condition(s) is found, the current trace pointer is moved to it. The trace contents are displayed in both Instruction and Data Access mode and the search is stopped. If no frame matching the specified condition(s) is found, the trace pointer stays at the starting frame of the search.

* **A=***address_or_range* specifies an address or range to search for, as follows:
A=*address* specifies a single address expression, **A=***start,end* specifies an address range beginning at *start* and ending at *end*, **A=***start*, **L** *length* specifies an address range beginning at *start* and extending for *length* bytes.

* The *x=data/mask* option specifies the access size and data/mask values. The size of the access (*x*) can be can be **B** (byte), **H** (halfword), **W** (word), or **D** (any size, the default).

* The **C=***status* option specifies the type of bus transaction to search for.
Valid arguments for status are: **RW** (read or write; this is the default), **RO** (read only), **WO** (write only),**F**(command fetch),or **ALL**(all status).

* **L**= *frame* option specifies the number of frames which it indicates.(defaults 20 frames)

Example:

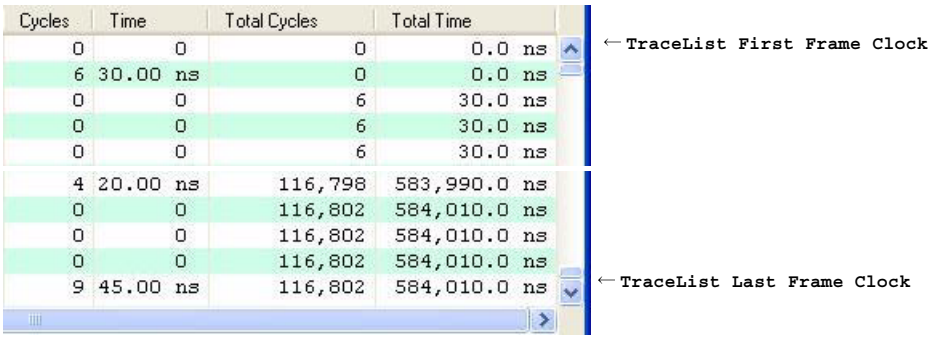
```
850eserv2>tsearch a=0x100100 c=rw
|FRAME|TIME|    PC     | OPCODE |   ADDRESS    | DATA | MNEMONIC
-------------------------------------------------------------------------
| 14  | 26 |0x00100028|  0000  |RD 0x00100100|0x676D| nop
| 16  | 16 |0x0010002E|076a0000|WD 0x00100100|0x0000| st.h zero ,0[r10]
| 18  | 35 |0x00100034|  0000  |RD 0x00100100|0x676D| nop
850eserv2>
```

TTIMER

This command is available for IECUBE2(V850E2Core).
When trace buffer is overwrited, "Total Cycles" in Trace List window is displayed from 0, this command can display actuality "Total Cycles". In this case, 850eserv2 displays the message "IECUBE2 Trace buffer is wrapped" into MULTI command pane and **TTIMER** command displays overwrited first frame's clock(**`Overwrited First Frame Clock`**).

Example:

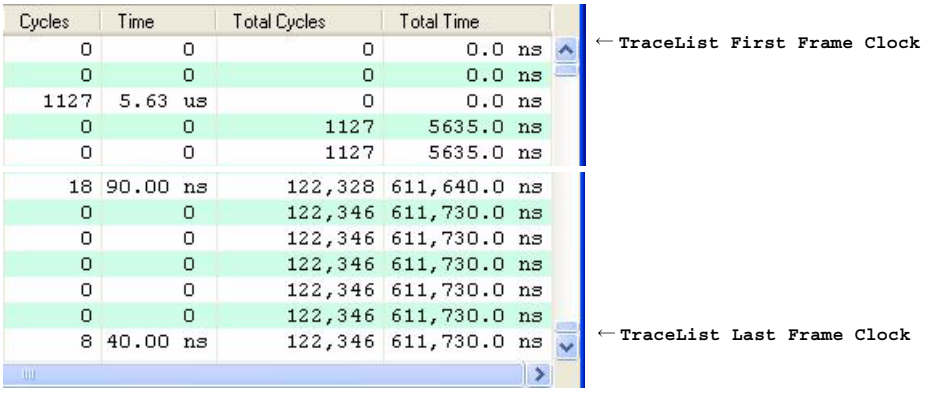
* In case of trace buffer is not overwrited.



Cycles	Time	Total Cycles	Total Time
0	0	0	0.0 ns
6	30.00 ns	0	0.0 ns
0	0	6	30.0 ns
0	0	6	30.0 ns
0	0	6	30.0 ns
4	20.00 ns	116,798	583,990.0 ns
0	0	116,802	584,010.0 ns
0	0	116,802	584,010.0 ns
0	0	116,802	584,010.0 ns
9	45.00 ns	116,802	584,010.0 ns


```
850eserv2>ttimer
TraceList First Frame Clock  : 0 clock
TraceList Last Frame Clock   : 116822 clock
```

* In case of trace buffer is overwrited.



Cycles	Time	Total Cycles	Total Time
0	0	0	0.0 ns
0	0	0	0.0 ns
1127	5.63 us	0	0.0 ns
0	0	1127	5635.0 ns
0	0	1127	5635.0 ns
18	90.00 ns	122,328	611,640.0 ns
0	0	122,346	611,730.0 ns
0	0	122,346	611,730.0 ns
0	0	122,346	611,730.0 ns
0	0	122,346	611,730.0 ns
0	0	122,346	611,730.0 ns
8	40.00 ns	122,346	611,730.0 ns


```
850eserv2>ttimer
TraceList First Frame Clock  : 0 clock
Overwrited First Frame Clock : 14959717 clock
TraceList Last Frame Clock   : 15082083 clock
```

UNASSEMBLE

UNASSEMBLE(U) [*start* | *start, end* | *start,* L *lines*]

This command is not available for V850E2Core and RH850.
Disassembles object code beginning at the specified address.

If *start*, *end* are specified: Disassembles object code in the range beginning at *start* and ending at *end*.
If *start*, **L** *lines*: Disassembles object code in the range beginning at *start* and extending for *lines* lines.

If no address is specified: Disassembles code beginning where the last **U** command left off.
If no range is specified: Disassembly continues for 11 lines.

Example:

```
850eserv2>U main
main:
0x1000ba: 5cla add -4, sp
0x1000bc: 63ff0100 st.w 1p, 0[sp]
0x1000c0: 2096e803 movea 0x3e8, zero, r18
0x1000c4: 64974180 st.w r18, -0x7fc0[gp]
0x1000c8: 40360200 movhi 0x2, zero, r6
0x1000cc: 2636a086 movea -0x7960, r6, r6
0x1000d0: bfffbcff jarl 0x10008c, 1p
0x1000d4: 23ff0100 ld.w 0[sp], 1p
0x1000d8: 441a add 4, sp
0x1000da: 7f00 jmp [1p]
0x1000dc: 0969 or r9, r13
850eserv2>U main, L 5
main:
0x1000ba: 5cla add -4, sp
0x1000bc: 63ff0100 st.w 1p, 0[sp]
0x1000c0: 2096e803 movea 0x3e8, zero, r18
0x1000c4: 64974180 st.w r18, -0x7fc0[gp]
0x1000c8: 40360200 movhi 0x2, zero, r6
850eserv2>
```

VERIFY

VERIFY [on | off]

Sets or clears the memory write verify flag, which turns on or off memory verify.

If **on** is specified, all subsequent memory writes are verified.
If **off** is specified, all subsequent writes are not verified.

If no parameters are set, the **VERIFY** command displays the current setting.

Example:

```
850eserv2> VERIFY off
850eserv2> VERIFY
VERIFY IS OFF
850eserv2>
```

VERSION

Display **850eserv2** version and hardware requirement and the like.

Example:

<<In case of N-Wire emulator >>

```
850eserv2>version
850eserv2 Version: 1.010
IE type=IE-V850E1-CD-NW (RCU1)
Executor Version=V850 G2 Executor V1.78 Copyright 2006
Device File Name=DF3259YG2.800
Device File Format Version=V2.18
Device File File Version=V1.20
```

<<In case of MINICUBE>>

```
850eserv2>version
850eserv2 Version: 1.010
IE type=MINICUBE (V850E2)
Executor Version=V850 G2 Executor V1.78 Copyright 2006
Device File Name=df3380.800
Device File Format Version=V2.19
Device File File Version=V1.00
MINICUBE Control Code=A
MINICUBE Firmware Version=V1.11
Control Board Version(0001)=V1.00
MINICUBE Board Version(0002)=V1.00 (FPGA Version=1.00)
```

<<In case of MINICUBE+V850E2Core>>

```
850eserv2>version
850eserv2 Version: V2.008
IE type=MINICUBE (V850E2Core)
Executor Version=V850E2 Executor V1.00
Device File Name=df3536.800
Device File Format Version=V1.00
Device File File Version=V1.00
MINICUBE Control Code=D
MINICUBE Firmware Version=V1.11
Control Board Version(0003)=V1.00 (FPGA Version=2.00)
```

<<In case of E1/E20 emulator(JTAG)>>

```
850eserv2>version
850eserv2 Version: V2.019
IE type=E1 Emulator JTAG (V850E2Core)
Executor Version=V850E2 Executor V1.07  [22 Mar 2011]
Device File Name=df4020.800
Device File Format Version=V1.00
Device File File Version=E1.00d
E1Emulator Control Code=E
E1Emulator Firmware Version=V1.16
Control Board Version(0003)=V1.00 (FPGA Version=3.00)
```

<<In case of MINICUBE2>>

```
850eserv2>version
850eserv2 Version: 1.010
IE type=MINICUBE2
Executor Version=V850 All Flash Executor V1.73 Copyright 2006
Device File Name=DF3707.800
Device File Format Version=V2.18
Device File File Version=V1.00
MINICUBE2 Control Code=A
MINICUBE2 Firmware Version=V3.00
Selectable port : UARTA0 / CSIB0
Select port : CSIB0
```

<<In case of E1/E20 emulator(Serial)>>

```
850eserv2>version
850eserv2 Version: 1.010
IE type=E1 Emulator Emulator Serial
Executor Version=V850 All Flash Executor V1.73  Copyright 2006
Device File Name=DF3707.800
Device File Format Version=V2.18
Device File File Version=V1.00
E1Emulator Control Code=A
E1Emulator Firmware Version=V3.00
Selectable port : UARTA0 / CSIB0
Select port : CSIB0
```

<<In case of IECUBE>>

```
850eserv2>version
850eserv2 Version: V1.010
IE type=NU85E Full ICE Generation 2 (IECUBE)
Executor Version=V850 G2 Executor V1.78 Copyright 2006
Device File Name=df3288Y.800
Device File Format Version=V2.18
Device File File Version=V2.01
IECUBE Control Code=B
IECUBE Firmware Version=V1.10
Control Board Version(0001)=V3.00 (FPGA Version=1.01)
CPU Board Version(0003)=V3.01
I/O Board Version(0109)=V3.00 (FPGA Version=2.00)
STP Interface Board Version(0012)=V1.00 (FPGA Version=1.11)
```

<<In case of IECUBE2>>

```
850eserv2>version
850eserv2 Version: V2.008
IE type=IECUBE2
Executor Version=V850E2 Executor V1.00
Device File Name=df3536.800
Device File Format Version=V1.00
Device File File Version=V1.00
IECUBE2 Control Code=@
CT Board Firmware Version=V1.00
CT Board Version(0002)=V1.00 (FPGA Version=0.13)
DB Board Firmware Version=V1.00
DB Board Version(0104)=V1.00 (FPGA Version=13.01)
```

850eserv2 Scripts

In addition to the Target commands mentioned above, **850eserv2** allows you to use script languages from the command line. You have two options to execute a script:

- To enter a script line by line from the prompt on the Target pane. If you use while or if statements in this case, the script won't be executed until the top statement is enclosed with **{....}**.
- To prepare a file in which a **script** is written and use the script command to specify and execute the file. If you specify the script file with the **-setup** option, this file will be executed each time you download a user program.

Notes on Using Scripts

Notes the following when using **850eserv2** script languages:

- Each statement must end within a line.
- If a line starts with "**#**" and no alphanumeric character precedes "**#**," that line will be commented out.
- It is not necessary to declare variables in advance.
- Variable type is detected automatically. For, example, you can change an integer-type variable to string-type data later.
- Variables and function names are case-sensitive.
- Commands are not case-sensitive.

Expressions

Expressions used in script languages are similar to those used in C language. The following table lists operators available for expressions in the order of precedence. Note that when included in a string, an integer-type notation is treated like integer-type data.

Operator	Integer-type variable	String-type variable
(	A group operator indicating that parenthesized parts must be evaluated first	A group operator indicating that parenthesized parts must be evaluated first
*	Multiply	Invalid
/	Divide	Invalid
%	Remainder	Invalid
+	Add	Concatenation of strings
-	Subtract	Invalid
<<	Shift left	Invalid
>>	Shift right	Invalid
<	Smaller than	Smaller than (for alphabetic characters)
<=	Smaller than or equal to	Smaller than or equal to (for alphabetic characters)
>	Greater than	Greater than (for alphabetic characters)
>=	Greater than or equal to	Greater than or equal to (for alphabetic characters)
=	Equal to	Equal to
!=	Unequal to	Unequal to
&	Bitwise AND	Invalid
^	Bitwise exclusive OR	Invalid
|	Bitwise OR	Invalid
&&	Logical AND	Invalid

Operator	Integer-type variable	String-type variable
||	Logical OR	Invalid

Assignment

Identifier = Expression

The expression is evaluated, and the result is stored as a variable in the identifier. Both string and integer types are supported. You can use alphanumeric characters and underscores "_" in identifiers. An identifier beginning with a number results in an error.

Keyword

Keywords can be used only in a small letter.

Word	Meaning
if	Conditions
else	Other
endif	Conditions end
elif	Another conditions
while	Loop
endwhile	Loop end

Conditions

```
if expression
    statements
endif
```

If the evaluation result of **`expression`** is `0`, nothing will take place. Otherwise, block statements between ***if*** and ***endif*** will be executed.

```
if expression
    statements
else
    statements
endif
```

If the evaluation result of **expression** is **0**, block statements between ***else*** and ***endif*** will be executed. Otherwise, block statements between ***if*** and ***else*** will be executed.

```
if expression
   statements
elif expression
   statements
endif
```

If the evaluation result of **expression** in the ***if*** statement is not **0**, block statements between ***if*** and ***elif*** will be executed. If the evaluation result of **expression** in the ***elif*** statement is not **0**, block statements between ***elif*** and ***endif*** will be executed. If both of the above **expression**s are evaluated as **0**, nothing will take place.

Loop

```
while expression
     statements
endwhile
```

Statements between ***while*** and ***endwhile*** are repeatedly executed until **expression** is evaluated as **0**. At the first iteration, **expression** is evaluated before the loop executes the whole statements. Be careful not to make it an endless loop. If the loop becomes endless, you have to kill **850eserv2** and start it over again.

Extension of Variables

If you use a script variable as an argument for an **850eserv2**-offered command, add "**$**" to the beginning of the variable.

- If you specify a variable to which "**$**"is added: the variable is an integer, a decimal number will be passed as an argument; if the variable is a string, the string itself will be passed as an argument.
- If you specify a variable to which "**$$**" is added: this variable will be passed in hexadecimal as an argument. This is used for commands where a hexadecimal number is specified as an argument (e.g., **m** command).

The script parser uses a longest possible identifier name begging with "**$**." In the following example, the user tries to show a variable "**foo**" together with a string "**bar**." However, the parser interprets them as a variable named "**foobar**" and, as a result, an error message "variable undefined!" will appear.

```
foo="foo"
PRINT $foobar
Error: variable undefined!
```

Script Examples

This section shows a few **850eserv2** script examples.

<Example 1>

This example uses a file named **test.ascii**. This file includes the following text:

```
This is a test of script file access in ascii mode.
This should print 3 lines of which this is the second.
And this is the third.
```

The following script is an example of access to **test.ascii**:

```
file = OPEN test.ascii
filecontents=""
totalchars=0
while(numlinechars=fread file line)
     filecontents=filecontents+line
     totalchars=totalchars+numlinechars
endwhile
CLOSE file
endl="\n"
PRINT Read: $filecontents$endl
PRINT Total of $totalchars characters read.$endl
```

The output results of this script are as follows:

```
Read: This is a test of script file access in ascii mode.
This should print 3 lines of which this is the second.
And this is the third.
Total of 130 characters read.
```

<Example 2>

The following script is another example of file access:

```
i=100
file = OPEN temp.bin
while(i>0)
     FPRINTB file $i
     i=i-1
endwhile
CLOSE file
file = OPEN temp.bin
sum=0
while(freadb file i)
     sum=sum+i
endwhile
```

```
CLOSE file
endl="\n"
PRINT The numbers between 1 and 100 sum to $sum!$endl
```

The output results of this script are as follows:

```
The numbers between 1 and 100 sum to 5050!
```

<Example 3>

In the next example, the system calculates the CRC32 value for a memory area ranging from **0x010000** to **0x010100** and displays the results on the Target pane. This script uses loop, conditional branches, expressions and variables, and shows how they are actually used.

```
# Change the following values to specify the memory
# range you want to calculate a CRC32 for.
# Note: locations from memstart to memend-1 are used
# to cmpute the CRC32 value.

memstart=0x010000
memend=0x010100

# This is the CRC32 polynomial. This is the same as
# is used in ehternet packets.

p=2+4+16+32+128+256+1024+2048+4096+65536+4194304+8388608+67108864

r=0

ptr=memstart
while(ptr<memend)
      currbyte=M -d1 $$ptr
      currbit=128
      while(currbit)
            test=r&(1<<31)
            r=r<<1
            r=r|(currbyte&currbit)
            if(test)
                  r=r^p
            endif
            currbit=currbit>>1
      endwhile
      ptr=ptr+1
endwhile

# This loop is for the 32 zeros appended to the
# original memory contents
i=0
while(i<32)
      test=r&(1<<31)
```

```
      r=r<<1
      if(test)
           r=r^p
      endif
      i=i+1
endwhile

# Now the resulting 32 bit CRC is in r
# Many of the same ASCII control codes that are used
# in C are supported in debug server scripts.

endl="\n"
PRINT CRC32 = $$r$endl
```

Chapter 14
Configuration Window

This chapter describes the following items.

- Starting Configuration Window
- Main window(850eserv2 Configuration)
- Trace Options window
- Trace Condition window
- Realtime RAM window
- Event List window
- Execute(BRS) event editor
- Access(BRA) event editor
- LINK event editor
- Timer Event List window
- Timer Event Editor
- Setting Hardware Breakpoints window
- Data Flash View
- Setting Device Clock window
- External Memory Mapping List
- External Memory Mapping Editor
- Setting Internal ROM/RAM size window

Starting Configuration Window

When 850win.exe that 850eserv2 configuration window starts from 850eserv2, you input a following command to MULTI command pine.

```
target 850win
```

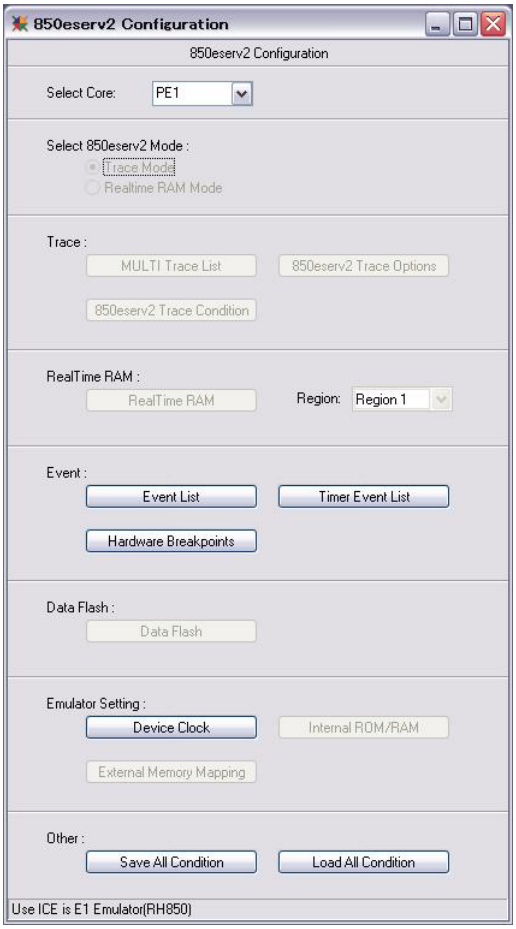
If 850win.exe doesn't exist in a directory including MULTI and 850eserv2, 850eserv2 displays the following error message.

```
850win open error
```

Trace condition setting from MULTI "Set Triggers" window is ignored while using 850win. Please set trace condition setting from "Trace Condition" window in 850win.

When 850eserv2 is started for the first time, 850win is started and "Setting Device Clock" window in 850win is opened automatically. If specified Device file is the same as last time's it when 850eserv2 is started for the next time, 850eserv2 sets the last operating frequencies automatically and 850win is not started.

Main window(850eserv2 Configuration)



When 850win is started from 850eserv2, this Main window(850eserv2 Configuration) is opened.
Each buttons on Main window opens the each function setting window. The buttons that it can not use in your environment are become to gray out.

Select core:

Selects core in Multi-core debugging. In other case, this pulldown is not available.

Select 850eserv2 Mode:

`Trace Mode`	Changes 850eserv2 mode to Trace mode in IECUBE.
`Realtime RAM Mode`	Changes 850eserv2 mode to Realtime RAM mode in IECUBE.

Trace:

Trace List	Opens "Trace List" window in MULTI. When 850eserv2 mode is Realtime RAM mode in IECUBE, this button is not available.
850eserv2 Trace Options	Opens Trace Options window. When 850eserv2 mode is Realtime RAM mode in IECUBE, this button is not available.
850eserv2 Trace Condition	Opens Trace Start Condition window. When 850eserv2 mode is Realtime RAM mode in IECUBE, this button is not available.

Realtime RAM:

Realtime RAM	Opens Realtime RAM window. Selects using Realtime RAM region from "Region" pulldown. In case of IECUBE, when 850eserv2 mode is Trace mode, this button is not available.

Event:

Event List	Opens Event List window. BRS, BRA and LINK events can set from "Event List" window.
Timer Event List	Opens Timer Event List window. Timer events can set from "Timer Event List" window.
Hardware Breakpoints	Open Setting Hardware Breakpoints window.

Data Flash:

Data Flash	Opens Data Flash memory view. In case of the device without Data Flash Memory, it is not available this button.

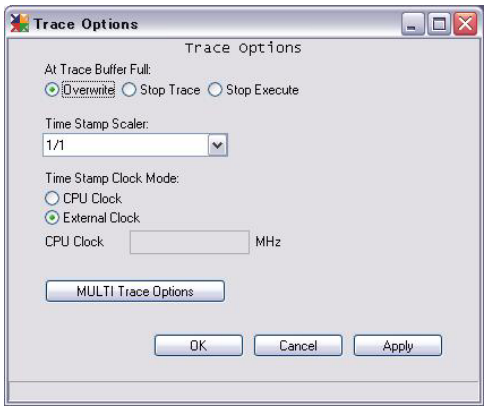
Emulator Setting:

Device Clock	Opens Device Clock setting window.
Internal ROM/RAM	Opens Internal ROM/RAM size setting window.
External Memory Mapping	Opens External memory mapping setting window.

Other:

Save All Condition	Saves All 850eserv2 conditions to .mbs file. The following conditions are saved: The unavailable functions in your environment are not saved. In Multi-core debugging, now selecting core's conditions are saved. * IECUBE2: Realtime RAM settings, Each events and Data flash mapping. * IECUBE and Trace mode : Trace Condition, Trace Options, Each events, External memory mappings, Internal ROM/RAM size and Data flash mapping. * IECUBE and Realtime RAM mode : Realtime RAM settings, Each events, External memory mappings, Internal ROM/RAM size and Data flash mapping. * OCD emulator, MINICUBE2 or E1/E20 emulator: Realtime RAM settings, Each events and Data flash mapping.
Load All Condition	Loads All 850eserv2 conditions from .mbs file. In Multi-core debugging, conditions are loaded to the core saved via **Save All Condition**.

Trace Options window



Each trace options is set from this window.

At Trace Buffer Full:

`Overwrite`	Trace is overwrited when trace buffer is full.(default)
`Stop Trace`	Trace is stoped when trace buffer is full.
`Stop Execute`	Break execute when trace buffer is full.

Time Stamp Scaler:

Selects scaler value of the trace time stamp counter. (default is 1/1)

Time Stamp Clock Mode:

`CPU Clock`	Uses specified CPU clock for trace time tag. CPU clock is specified in "CPU clock" field.
`External Clock`	Uses IECUBE's external clock(50MHz) for trace time tag(default)

MULTI Trace Options:

Opens "Trace Options" window in MULTI.

Trace Condition window

Trace start or end condition is set from this window.
When it selects "Unconditional Trace(All)", other condition's field(Start Trace, Stop Trace, Qualify Trace and Delay Stop) are not available.

Trace condition setting from MULTI "Set Triggers" window is ignored while using 850win. Please set trace condition setting from this window.

Start Trace:

Sets Start trace trigger by the events on text field.

Add	Adds the events to text field from pulldown.
Clear Start Trace	Clear Start Trace text field.

Stop Trace:

Sets Stop trace trigger by the events on text field.

Add	Adds the events to text field from pulldown.
Clear Stop Trace	Clear StopTrace text field.

Qualify Trace:

Sets Qualify trace trigger by the events on text field. Cannot set a LINK event.

Add	Adds the events to text field from pulldown.
Clear Qualify Trace	Clear Qualify Trace text field.

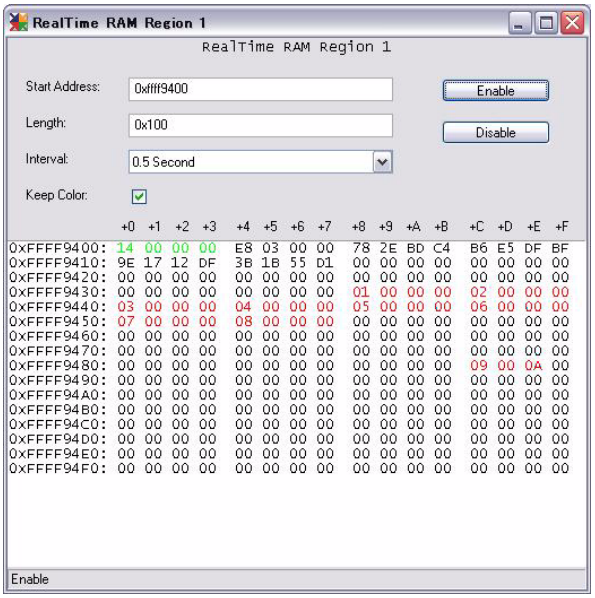
If it sets BRA events to Qualify trace trigger, it is necessary to set "Trace Packets" to "Data Access PC" and "Data Address and Data" from IECUBE Target Options window("config > Target Specific Options" in Trace List window).

Delay Stop:

Sets Delay stop trigger by the events on text field and selects the number of frames which are taken in after Delay stop trigger from "First", "Middle" or "Last".

Add	Adds the events to text field from pulldown.
Clear Delay Stop	Clear Delay Stop text field.
Fiest	About maximum of trace buffer.
Middle	About half of trace buffer.
Last	From trigger frame About about some frames.

Realtime RAM window



Realtime RAM window dislpays contents of RAM in specified region cyclically during execution by user program, and Realtime RAM condition is set from this window.
In case of IECUBE, the value's color in list is following meaning. In case of MINICUBE, MINICUBE2, E1/E20 emulator or IECUBE2, all values are displayed in the black color.
In Multi-core debugging, It can specify core via the pulldown in top of this window. In other case, this pulldown is not available.

Red	Write access.
Green	Read access.
Blue	Read and Write access.

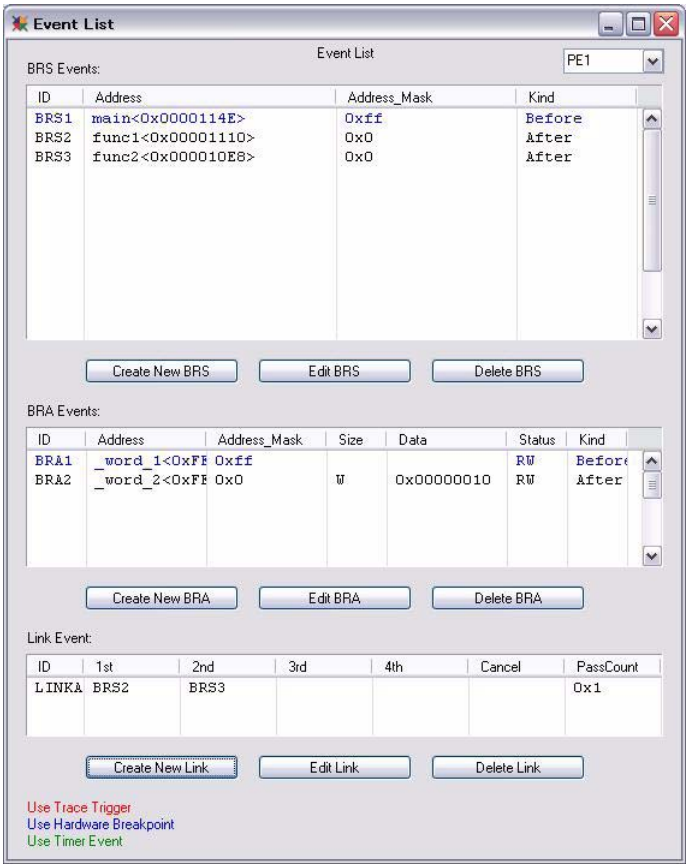
Now Realtime RAM status(Enable/Disable) in this region is dysplayed the bottom of window.

Start Address:	Sets start address of Realtime RAM in this region. It can specify a symbol.
Length:	Sets displayed length of Realtime RAM in this region. You can specify 1 - 256 bytes.
Interval:	Selects updating interval from 0.5, 1, 2 or 5 seconds.

Keep Color: Usually value's color is cleared by every update. If you set this check, value's color is not cleared until Realtime RAM setting is changed. This check is common in all the regions.

"Interval" value is a rough time. When measuring with more regions, the timing of each region update might shift.

Event List window



Event List window displays now setting BRS(max 12), BRA(max 6) and LINK(max 1) events.
And create new events, edits now setting events and delete now setting events from this window.
The event set by Trace condition displays red, the event set by Hardware breakpoint displays blue, the event set by Timer event displays green.
In Multi-core debugging, It can change the core via the pulldown in top of this window. In other case, this pulldown is not available.

BRS Events:

`Create New BRS`	Creates new BRS event from "Execute(BRS) event editor".
`Edit BRS`	Edits BRS event that has been selected with cursor in "BRS events" list.
`Delete BRS`	Deletes BRS event that has been selected with cursor in "BRS events" list.

BRA Events:

Create New BRA	Creates new BRA event from "Access(BRA) event editor".
Edit BRA	Edits BRA event that has been selected with cursor in "BRA events" list.
Delete BRA	Deletes BRA event that has been selected with cursor in "BRA events" list.

LINK Event:

Create New LINK	Creates new LINK event from "LINK event editor".
Edit LINK	Edits LINK event.
Delete LINK	Deletes LINK event.

Execute(BRS) event editor

BRS event condition is set from this window.
Setting event ID and execute type is selected from pulldown in top of window.
When it selects "After" in execute type, it is not available "Address Mask:".
In case of OCD emulator + (Nx85E901(RCU0),RCU1), MINICUBE2 or E1/E20 emulator(Serial), it is available only "Before" in execute type.
In case of MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core or IECUBE2(V850E2Core), it is available "Timer" in execute type.

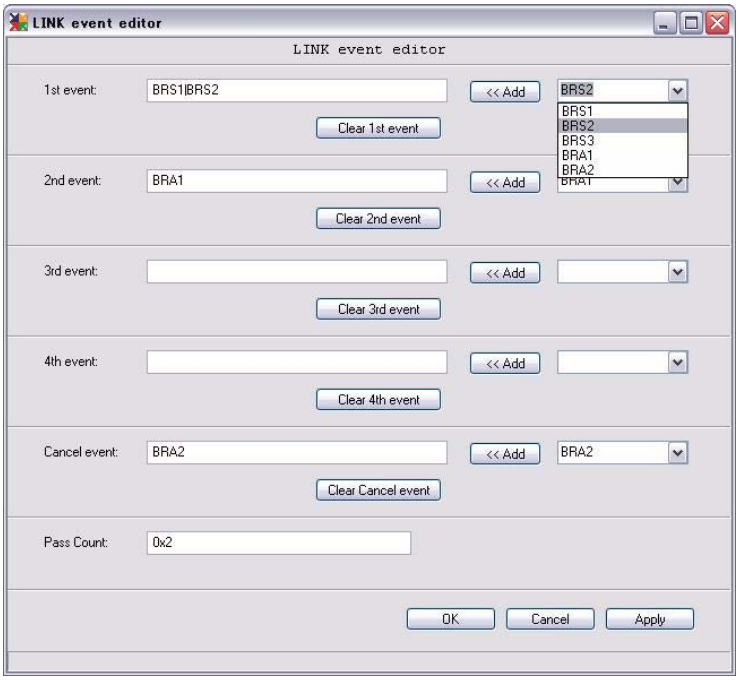
Address:	Sets event address. If it sets range, puts a comma between start address and end address. It can specify a symbol.
Address Mask:	Sets address mask value of 32-bit width.

Access(BRA) event editor

BRA event condition is set from this window.
Setting event ID and bus event type is selected from pulldown in top of window. The bus event type can be splecified only with RH850.

Address:	Sets event address. If it sets range, puts a comma between start address and end address. It can specify a symbol.
Address Mask:	Sets address mask value of 32-bit width.
Data size:	Selects data access size. "Long", "12Bytes" and "16Bytes" can be specified only with RH850. If "12Bytes" or "16Bytes" are selecetd, **Data:** can not be specified.
Data:	Sets data value. Hex values are indicated by a `0x` prefix, while binary input is indicated by an `0b` prefix. To specify a value with a mask, use `X` characters for the nibbles or bits you want to be ignored.
Status:	Selects access type.

LINK event editor



LINK event condition is set from this window.
In Multi-core debugging, you can set sequentially event in each cores. You can specify the **BRA**, **BRS** events set in this core. The events set in other core can not be specified.
In case of OCD emulator+(Nx85E901(RCU0),RCU1), MINICUBE2 or E1/E20 emulator(Serial), it is not available to "Pass Count:". In case of other ICE, it cannot specify before execution event in 1st - 4th and Cancel level.

1st - 4th event:

Sets 1st - 4th level in LINK event by BRS or BRA events on text field.

Add	Adds the events to text field from pulldown.
Clear 1st - 4th event	Clear 1st - 4th level text field.

Cancel event:

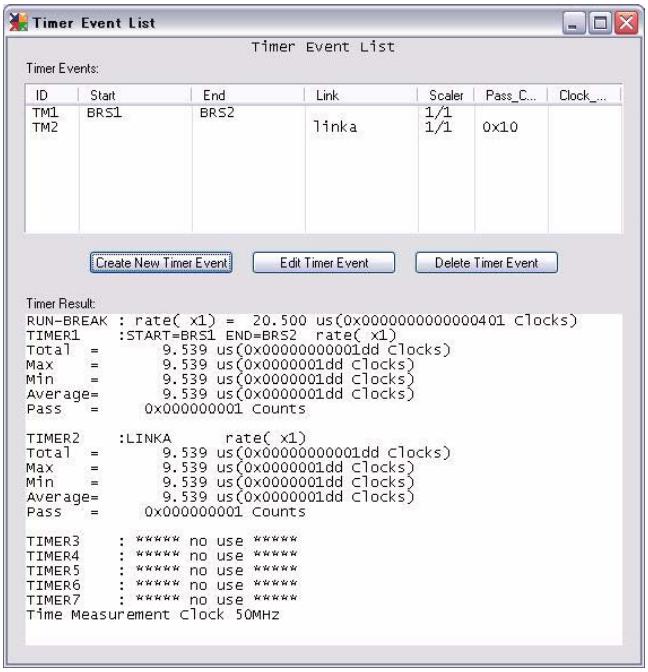
Sets Cancel conditon in LINK event by BRS or BRA events on text field.
When specofied Cancel conditon is met, LINK event is canceled.

`Add`	Adds the events to text field from pulldown.
`Clear Cancel event`	Clear Cancel event text field.

Pass Count:

Sets pass count value.

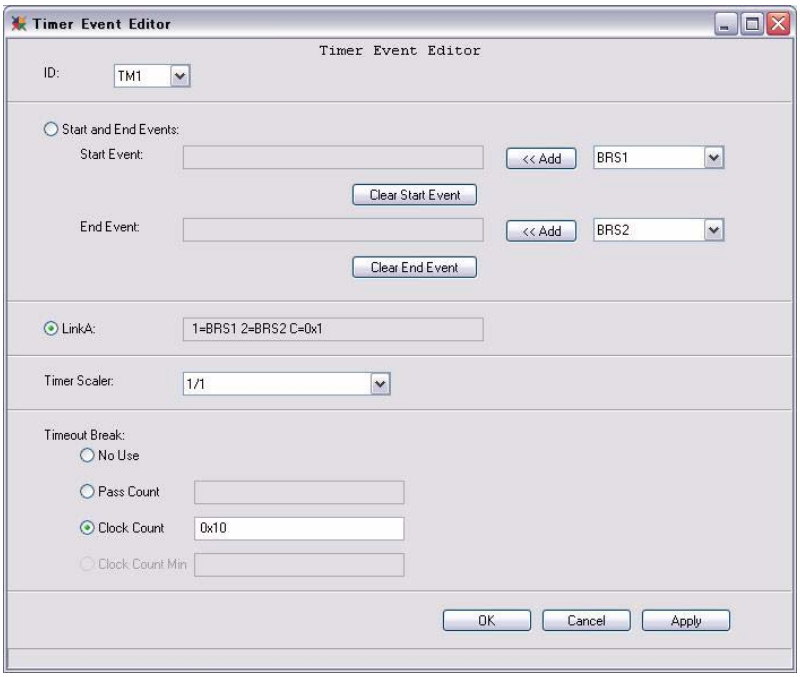
Timer Event List window



Timer Event List window displays now setting Timer events(Max 7) and timer result of executing user program.
And create new Timer events, edits now setting Timer events and delete now setting Timer events from this window. In case of OCD emulator+V850E1Core, MINICUBE2+V850E1Core or E1/E20 emulator+V850E1Core, it is available to these buttons.
In Multi-core debugging, It can change the core via the pulldown in top of this window. In other case, this pulldown is not available.

Create New Timer Event	Creates new Timer event from "Timer event editor".
Edit Timer Event	Edits Timer event that has been selected with cursor in "Timer events" list.
Delete Timer Event	Deletes Timer event that has been selected with cursor in "Timer events" list.

Timer Event Editor



Timer event condition is set from this window.
In Multi-core debugging, you can set timer events in each cores. You can specify the **BRA**, **BRS** events set in this core. The events set in other core can not be specified.
Setting event ID is selected from pulldown in top of window. You cannot specify before execution event and address range event.
In case of MINICUBE+V850E2Core, MINICUBE2+V850E2Core, E1/E20 emulator+V850E2Core or IECUBE2, it is not available to "LinkA". And you can specify only **S**=**t** specified event via **BRS** command.

Start and End events:

Sets start and end events by simgle BRS or BRA event.
It cannot specify simultaneously with "LinkA".

Add	Adds the events to text field from pulldown.
Clear Start(End) Event	Clear Start event (End event) text field.

LinkA:

Sets start and end events by LINK event.
Start trigger is level-1 and end trigger is level-2 in LINK event.
If it doesn't set LINK event, it cannot select "LinkA" button.
It cannot specify simultaneously with "Start and End events".

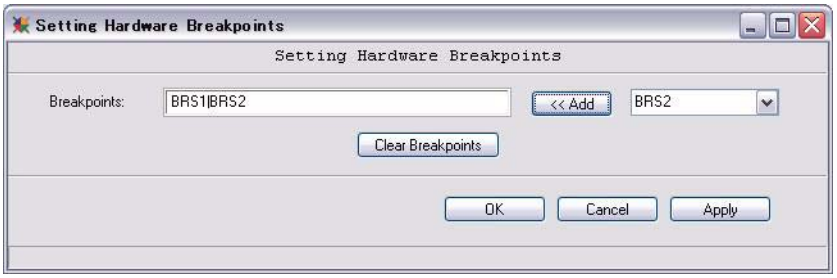
Timer Scaler:

Sets a clock source division ratio. Clock source division ratio cannot be set up individually.
Timer measurement is performed by clock source division ratio set up at the last.

Timeout Break:

No use	Not use the Timeout break function.
Pass Count	A break is taken by the specified number of section passage. In the case of 0, it does not set up. In IECUBE, it can be specified 0x1-0xfffe. In V850E2Core, it can be specified 0x1-0xffffffff.
Clock Count	A break is taken by the specified section time count. In the case of 0, it does not set up. In IECUBE, it can be specified 0x1-0xfffffffe. In V850E2Core, it can be specified 0x1-0xffffffff.
Clock Count Min	A break is taken by the specified section time minimum count. (Only V850E2Core) In the case of 0, it does not set up. It can be specified 0x1-0xffffffff.

Setting Hardware Breakpoints window



Hardware breakpoints are set from this window.
In Multi-core debugging, you can set hardware breakpoints in each cores. You can specify the **BRA**, **BRS** and **LINK** events set in this core. The events set in other core can not be specified.
If "Breakpoints:" field is empty, all hardware breakpoints set now are removed.

Breakpoints:

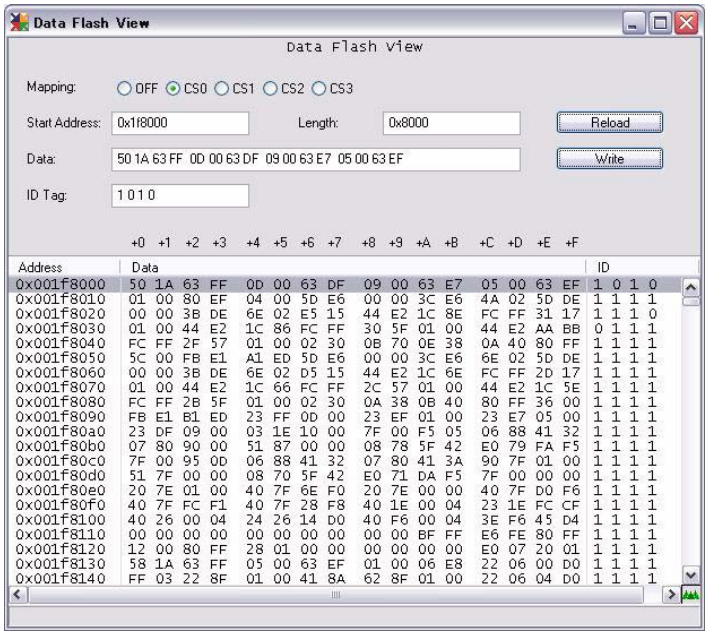
Sets hardware breakpoints by BRS or BRA events on text field.

`Add`	Adds the events to text field from pulldown.
`Clear Breakpoints`	Clear "Breakpoints:" text field.

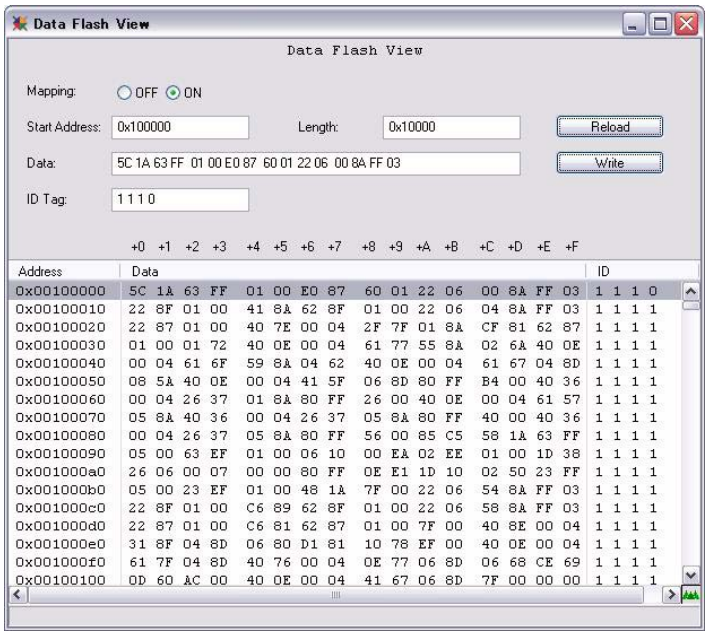
The hardware breakpoint set via this window is not marked to MULTI source window.

Data Flash View

<< In the case of CS area selectable device>>



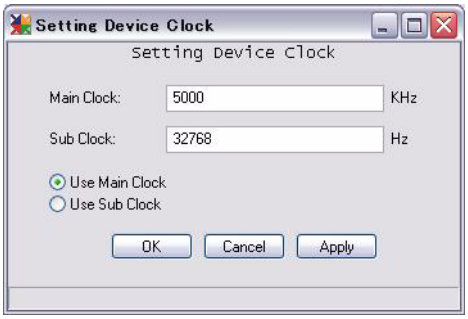
<< In the case of CS area fixed device>>



Data Flash view can display and edit data and the ID-tag in the Data Flash area.
This window can be displayed and be edited only in the Data Flash area selected "Mapping:". When it clicks on Data list with mouse's left button, the 16 bytes data and ID-tag are displayed in "Data:" and "ID Tag:" from selected address.

Mapping:	Maps Data Flash area. In case on CS area selectable device, it can specify "OFF" or "CS0-CS3". In case on CS area fixed device , it can specify "OFF" or "ON".
Start Address:	Displays starting address. It is necessary by four byte boundary. You cannot specify out of selecting Data Flash area.
Length:	Displays memory length. It is necessary by four byte boundary. You cannot specify to exceed from selecting Data Flash area
Data:	Editing Data. It can input by the hexadecimal number that excludes "0x".
ID Tag:	Editing ID tag. It can input only 0 or 1.

Setting Device Clock window

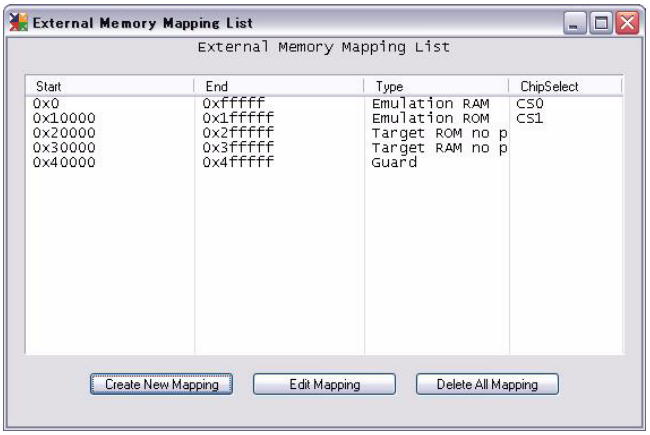


Target's operating frequencies(Device Clock) is set from this window.
When not set target's operating frequencies, or when the wrong value is set, Emulator cannot access target device correctly. Therefore, it is necessary to set the target's operating frequencies after 850eserv2 starts.

When 850eserv2 is started for the first time, 850win is started and this window is opened automatically. If specified Device file is the same as last time's it when 850eserv2 is started for the next time, 850eserv2 sets the last operating frequencies automatically and 850win is not started.

Main Clock:	Specifies Main clock per KHz
Sub Clock:	Specifies Sub clock per Hz
Use Main Clock:	Uses Main clock (default)
Use Sub Clock:	Uses Sub clock

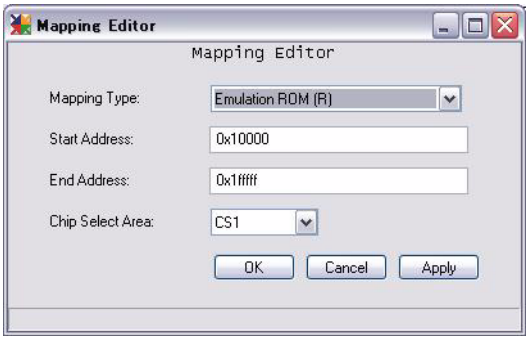
External Memory Mapping List



External Memory Mapping List window displays now setting external memory mapping. And create new mapping, edits now mapping and delete all mappings from this window.

Create New Mapping	Creates new mapping from "Mapping editor".
Edit BRS	Edits mapping that has been selected with cursor in list.
Delete BRS	Deletes all mappings

External Memory Mapping Editor



External memory mapping is set from this window. Please refer to "MAP" on page 125 for external memory mapping's details.

Mapping Type: Selects following mapping type.

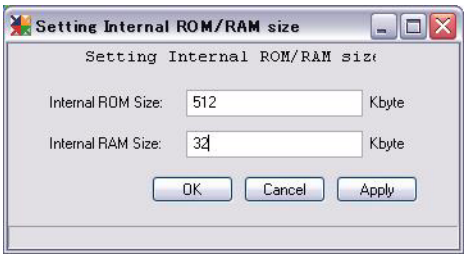
Emulation RAM (W)	Emulator read/write memory.
Emulation ROM (W)	Emulator read only memory.
Target ROM (TR)	Target read only memory.
Target ROM no power check (TR2)	Target read only memory. Power source check of the target is excluded.
Target RAM (U)	Target read/write memory.
Target RAM no power check (U2)	Target read/write memory. Power source check of the target is excluded.
Guard (G)	Guard area.

Start Address: Specifies mapping start address.

End Address: Specifies mapping end address.

Chip Select Area Selects Chip select area. This item is available when you specify "Emulation RAM (W)" or "Emulation ROM (R)".

Setting Internal ROM/RAM size window



Internal ROM and RAM sizes(in KByte) is set from this window. Default ROM and RAM sizes are included in Device File.

`Internal ROM size` Specifies Internal ROM size. You can specify 1-1024 Kbyte.

`Internal RAM size` Specifies Internal RAM size. You can specify 1-60 Kbyte.

Chapter 15
The changed part from 850eserv

This chapter describes the following items.

- Option List Which was Deleted from 850eserv
- Options Which was Deleted from 850eserv
- Command List Which was Deleted or Changed from 850eserv
- Commands Which was Deleted or Changed from 850eserv

850eserv2 has succeeded almost commands syntax and starting options of 850eserv,but commands and options of part were changed or deleted.
This chapter explains options and commands which there were changed or deleted from 850eserv.

Option List Which was Deleted from 850eserv

-network	Connects through Ethernet interface.	Deleted
-X0	Prevents 850eserv for zeroing the bss section.	Deleted
-X1	Enables 850eserv to flush the bss section.	Deleted
-noint	Doesn't become in interactive mode.	Deleted

Options Which was Deleted from 850eserv

-network

<<specification of 850eserv>>

Connects through Ethernet interface. This command can use only IE-70000-MC-SV3.

<<specification of 850eserv2>>

850eserv2 not support IE-70000-MC-SV3.

-X0

<<specification of 850eserv>>

Prevents 850eserv for zeroing the bss section.

<<specification of 850eserv2>>

If you specify **NOLOAD BSS**, prevents for zeroing the bss section when downloading. Please refer to "NOLOAD" on page 66.

-X1

<<specification of 850eserv>>

Enables 850eserv to flush the bss section.

<<specification of 850eserv2>>

If you specify **LOAD BSS**, enables to flush the bss section. when downloading. Please refer to "LOAD" on page 64.

-noint

<<specification of 850eserv>>

Doesn't become in interactive mode by **PIO**, **REGISTER**,and **SFR** command.

<<specification of 850eserv2>>

Interactive mode was abolished from all commands in 850eserv2.

Command List Which was Deleted or Changed from 850eserv

ASSEMBLE	Assembles code one line at a time	Changed
BATCH	Executes a batch file	Deleted
BREAK	Causes a break	Changed
CANCEL	Cancels batch file execution	Deleted
CLOCK	Specifies the clock source	Deleted
COMBO	Displays or changes the COMBO break status	Deleted
COMPARE	Compares two blocks of memory	Deleted
COPY	Copies one block of memory to another	Deleted
DFVIEW	Displays Data Flash Window	Changed
FILL	Fills a block of memory	Deleted
FIND	Searches memory	Deleted
HISTORY	Displays previous command invocation	Deleted
MEMORY	Displays or changes memory	Deleted
NOLOAD	Turns download suppression on or off	Changed
PAUSE	Pauses batch execution	Deleted
PIO	Displays or changes the Programmable P/O values	Changed
PROFILE	Get or Send to MULTI Profile Data	Changed
REGISTER	Displays or changes register values	Deleted
RESET	Resets the CPU and/or the emulation hardware	Deleted
RMEMVIEW	Displays the contents of the real-time RAM Window	Changed
SFR	Displays SFR values	Changed

SYMBOL	Turns symbol display on or off	Deleted
TDISPLAY	Displays the trace buffer	Changed
TFILTER	Specifies trace buffer filtering	Changed
TMODE	Specifies the trace mode	Changed
TRACEWIN	Display Trace Window	Changed
WAIT	Waits for an event	Deleted

Commands Which was Deleted or Changed from 850eserv

ASSEMBLE

<<specification of 850eserv>>

Assembles code one line at a time in interactive mode.

Syntax:

```
ASSEMBLE [address]
```

<<specification of 850eserv2>>

Interactive mode was abolished, only specified address compiles. Unless address and code are input, it becomes error.

Syntax:

```
ASSEMBLE <address> <code>
```

BATCH

<<specification of 850eserv>>

850eserv executes a specified batch file.

Syntax:

```
BATCH <filename> [param1][param2][...]
```

<<specification of 850eserv2>>

850eserv2 executes a specified script file from **SCRIPT** command of common to server. Please refer to "SCRIPT" on page 68 or "850eserv2 Scripts" on page 177.

Syntax:

```
SCRIPT <filename>
```

BREAK

<<specification of 850eserv>>

Display or Set event break.

Syntax:

```
BREAK [K|[event-expression][...]]
```

<<specification of 850eserv2>>

BREAK command sets software breakpoints. **HWBRK** command sets event breakpoints (hardware breakpoints).

Syntax:

```
BREAK <address>
HWBRK [K|[event-expression][...]]
```

CANCEL

<<specification of 850eserv>>

Cancels batch file execution when executing batch file.

<<specification of 850eserv2>>

This command was deleted.

CLOCK

<<specification of 850eserv>>

Specifies the clock source.

<<specification of 850eserv2>>

Because 850eserv2 doesn't support ICE that can use this command, this command was deleted.

COMBO

<<specification of 850eserv>>

Displays or changes the COMBO break status.

<<specification of 850eserv2>>

Because 850eserv2 doesn't support ICE that can use this command, this command was deleted.

COMPARE

<<specification of 850eserv>>

Compares two blocks of memory.

Syntax:

```
COMPARE <source1> <source2> <length> [size]
```

<<specification of 850eserv2>>

Please use MULTI's **COMPARE** command. Please refer to "MULTI: Debugging".

Syntax:

```
COMPARE <OP> <SRC1> <SRC2> <length> [size]
```

COPY

<<specification of 850eserv>>

Copies one block of memory to another.

Syntax:

```
COPY <source> <dest> <length> [size] [direction]
```

<<specification of 850eserv2>>

Please use MULTI's **COPY** command. Please refer to "MULTI: Debugging".

Syntax:

```
COPY <src> <dest> <length> [<size> [direction]]
```

DFVIEW

<<specification of 850eserv>>

Displays Data Flash Window.

<<specification of 850eserv2>>

This command was deleted, Data Flash memory is displayed by Data Flash view in 850win.
Please refer to "Chapter 14 Configuration Window" on page 185.

FILL

<<specification of 850eserv>>

Fills a block of memory in specify address.

Syntax:

```
FILL <dest> <length> [value] [size]
```

<<specification of 850eserv2>>

Please use **BLOCKFILL** command of common to server. Please refer to "BLOCKFILL" on page 59.

Syntax:

```
BLOCKFILL [-d1|-d2|-d4] <address> <count> <value>
```

FIND

<<specification of 850eserv>>

Searches memory.

Syntax:

```
FIND <source> <length> <value> [size]
```

<<specification of 850eserv2>>

Please use MULTI's **FIND** command. Please refer to "MULTI: Debugging".

Syntax:

```
FIND <src> <length> <value> [<size> [mask]]
```

HISTORY

<<specification of 850eserv>>

Displays previous command invocation.

<<specification of 850eserv2>>

This command was deleted.

MEMORY

<<specification of 850eserv>>

Displays or changes specify memory. It can write in continuously in interactive mode. It can access to flash memory.

Syntax:

```
MEMORY [FW|FE] [[B|H|W][C]<address>[=value][L=length] | all]
```

<<specification of 850eserv2>>

Please use **M** command of common to server. **M** command can access to Code Flash memory and Data Flash memory. Writing ID-tag to Data Flash area uses **IDTAG** command. Erasing Flash memory uses **FERASE** command. But **M** command not support interactive mode.
Please refer to "M" on page 64, "IDTAG" on page 119 and "FERASE" on page 98.

Syntax:

```
M [-d1|-d2|-d4] <address>[=<val>]
IDTAG <address> <=IDtag>
FERASE <address | all>
```

NOLOAD

<<specification of 850eserv>>

Toggles suppression of data downloading to the ICE.

Syntax:

```
NOLOAD
```

<<specification of 850eserv2>>

Please use **NOLOAD** command of common to server. Please refer to "NOLOAD" on page 66. Or please use **-noload** starting option of common to server. Please refer to "Server Starting Options" on page 18.

Syntax:

```
NOLOAD [text|data|bss|all]
```

PAUSE

<<specification of 850eserv>>

Pauses batch execution

<<specification of 850eserv2>>

This command was deleted.

PIO

<<specification of 850eserv>>

Displays or changes the Programmable I/O values. If you enter only the name of an Programmable I/O, its value is displayed and you are prompted to enter a new value.

Syntax:

```
PIO [name] [=newvalue]
```

<<specification of 850eserv2>>

Interactive mode was abolished, If you enter only the name of an Programmable I/O, its value is displayed. Please refer to "PIO" on page 133.

Syntax:

```
PIO [name] [=newvalue]
```

PROFILE

<<specification of 850eserv>>

It can not get profile data again as **data** option, after writing profile data in MULTI as **done** option.

<<specification of 850eserv2>>

It can get profile data again as **data** option, after writing profile data in MULTI as **done** option. Please refer to "PROFILE" on page 135.

REGISTER

<<specification of 850eserv>>

Displays or changes register values. And a disassembly of the next instruction to execute.

Syntax:

```
REGISTER [regname [=value]]
```

<<specification of 850eserv2>>

Please use **REG** command of common to server. Please refer to "REG" on page 67. But **REG** command does not disassemble.

Syntax:

```
REG [<regname> [<val>]]
```

RESET

<<specification of 850eserv>>

Resets the CPU and/or emulation hardware .If **T** is specifie, it clear trace buffer.

Syntax:

```
RESET [A|T]
```

<<specification of 850eserv2>>

Please use **RST** command of common to server. But **RST** command not support emulator all reset. If you will clear trace buffer, please use **TCLEAR** command. Please refer to "RST" on page 68 and "TCLEAR" on page 143.

Syntax:

```
RST [stop|run]
TCLEAR
```

RMEMVIEW

<<specification of 850eserv>>

Displays the contents of the realtime RAM Window.

<<specification of 850eserv2>>

This command was deleted, Realtime RAM is displayed by Realtime RAM window in 850win.
Please refer to "Chapter 14 Configuration Window" on page 185.

SFR

<<specification of 850eserv>>

Displays or changes the SFR values. If you enter only the name of a SFR, its value is displayed and you are prompted to enter a new value.

Syntax:

```
SFR [name] [=newvalue]
```

<<specification of 850eserv2>>

SFR command can access to SFR, Extended External I/O register or Programmable I/O register.
Interactive mode was abolished. Please refer to "SFR" on page 139.

Syntax:

```
SFR [name] [l=length]
SFR [[s|e]=num] [l=length]
SFR <name>=<newvalue>
```

SYMBOL

<<specification of 850eserv>>

Using **UNASSEMBLE** command and **BRS** command etc., when the symbol which corresponds to address exists, it indicates symbol name.

<<specification of 850eserv2>>

When the symbol which corresponds to address exists, 850eserv2 indicates symbol name in normally. This command was deleted.

TDISPLAY

<<specification of 850eserv>>

There are three trace buffer display modes: Instruction mode (**I**), Data Access (**F**) mode and Source Mixed mode(**X**). When you write the trace data to the file, you are prompted to enter "a" or "d" or ".".

Syntax:

```
TDISPLAY [I|F|x][($,S,E,T) rel frame][L=# of frames][> filename]
```

<<specification of 850eserv2>>

Source Mixed mode(**X**) was deleted. When you write the trace data to the file, you must specify entry mode (a|d) after the filename. When there is no appointment, it becomes error. However, when the specified file does not exist, "a" or "d" can be omitted. Please refer to "TDISPLAY" on page 144.

Syntax:

```
TDISPLAY [I|F][($,S,E,T) rel frame][L=# of frames][> filename <a|d>]
```

TFILTER

<<specification of 850eserv>>

Specifies or displays what is filtered out before display in the trace buffer in Instruction mode (**I**), Data Access (**F**) mode and Source Mixed mode(**X**).

Syntax:

```
TFILTER [I=(TO.*)][F=(TECOADSM.*)][X=(TD.*)]
```

<<specification of 850eserv2>>

Source Mixed mode(**X**) was deleted. Please refer to "TFILTER" on page 148.

Syntax:

```
TFILTER [I=(TO.*)][F=(TECOADSM.*)]
```

TMODE

<<specification of 850eserv>>

In the case of IECUBE, Trace mode and Real time RAM mode are switched **SW** option in **TMODE** command.

Syntax:

```
TMODE [M=[F|N|B|D]][TS=[C|T]][TC=[R|F]][CP=[ON|OFF]]
      [TT=[CC|AC|AE]][TM=[B|A|D|BD|DP|BDP|AD]][sw=[on|off]]
```

<<specification of 850eserv2>>

In the case of IECUBE, Trace mode and Real time RAM mode are switched **SW** command. Please refer to "TMODE" on page 158 and "SW" on page 142.

Syntax:

```
TMODE [M=[F|N|B|D]][TS=[C|T]][TC=[R|F]][CP=[ON|OFF]]
      [TT=[CC|AC|AE]][TM=[B|A|D|BD|DP|BDP|AD]]
SW [trc|rram]
```

TRACEWIN

<<specification of 850eserv>>

Display Trace window and sets each events, trace condition and timer condition.

<<specification of 850eserv2>>

850win has almost the same functions as Tracewin.
Please refer to "Chapter 14 Configuration Window" on page 185.

WAIT

<<specification of 850eserv>>

Waits for an event when executing batch file.

<<specification of 850eserv2>>

This command was deleted.

Appendix A
ERROR Message From ICE

This appendix describes the following items.

- Fatal Error
- User system abnormality
- Status Error
- Parameter Error
- Device Dependent Error
- IECUBE or IECUBE2 Starting Error
- IECUBE or IECUBE2 Starting Warning
- RSU verify Error
- Server Starting Error
- Emulation CPU status in running

Fatal Error

0x0100:fatal err (communication error)	Can not communicate with ICE. Please confirm the installation of the device driver for the PC interfaceboard. > The driver may not be correctly installed. Reinstall the driver.
0x0101:fatal err (hostname not found)	Can not find initialization file (expc.ini).
0x0102:fatal err (net-inf file not found)	Host name not found.
0x0103: fatal err (data send timeout)	Data transfer to ICE is timed out. > Please confirm the power of ICE, connection of the interface cable, or I/O address of the PC interface board.
0x0104:fatal err (exec timeout)	Data receive from ICE is timed out. > Please confirm the power of ICE, connection of the interface cable, or I/O address of the PC interface board.
0x0105:fatal err (missing device file read)	Failed in reading device file (dxxxx.800). > Necessary files may be damaged. Reinstall the device file. Check setting IEPATH environment variables.
0x0106:fatal err (illegal receive data)	Illegal data received. > Check the power of the in-circuit emulator, cable connections, and setting of the interface board and restart the debugger.
0x0107:fatal err (illegal pipe handle)	Error departure raised with communication with the incircuit emulator.
0x0108:fatal err (hostname error)	cannot read the network information file just.

0x0109:fatal err (USB communication error)	Abnormality occurred in the communication of USB > Please end the debugger, verify the power source of in-circuit-emulator, and the connection cable. Restart the debugger.
0x010a:fatal err (another EXEC has already operated)	The tool that uses EXEC cannot be started simultaneously. > Stop operated tool that uses EXEC.
0x01a0:fatal err (monitor timeout)	Monitor timeout occurred. > Please verify whether there is no abnormality in the signal and the clock pulse of ESET, WAIT and HLDRQ etc.
0x01a1:fatal err (exec file read err)	Failed in reading the exec file.
0x01a2:fatal err (BK board no connect)	Break board is not connected.
0x01a3:fatal err (EM board no connect)	Emulation board is not connected.
0x01a4:fatal err (illegal board set)	Board configuration of ICE is not consistent.
0x01a5:fatal err (POD/EM1 board no connect)	POD/EM1 board is not connected.
0x01a6:fatal err (exec running)	EXEC is running. > Stop operated tool that uses EXEC.
0x01a7:fatal err (micro program read err)	Failed in reading micro program file.
0x01a8:fatal err (ini file not found)	Failed in reading initialization file (expc.ini).
0x01a9:fatal err (packet send-buffer sizeover)	Packet transmission buffer size over.
0x01aa:fatal err (RPRM no connect)	Terminal emulator is not loaded.
0x01ab:fatal err (flash F/W file read err)	Failed in reading the flash F/W file.
0x01ac:fatal err (illegal device file)	Device file format type error.
0x01ad:fatal err (old device driver)	Device driver is older version. > Install newest driver.

0x01ae:fatal err (init file error)	Failed in reading init file.
0x01b2:fatal err (ICE firmware is older version)	MINICUBE2 firmware is older version. > Install newest MINICUBE firmware.
0x0600:fatal err (buffer allocate err)	Not enough memory for buffer. > There is not enough system memory. Close the applications being executed and the open files.
0x0601:fatal err (win32 resource insufficiency)	Resource of the operating system becoming insufficient.
0x0c00:fatal err (Monitor file read error)	Monitor file read error. > Necessary files may be damaged. Reinstall that files.
0x0c01:fatal err (Register function monitor timeout)	During access of register, CPU did time out. > Check the clock signal, etc. The register value may not be correct.
0x0c02:fatal err (Memory function monitor timeout)	During access of memory, CPU did time out. > Check the HOLD signal, WAIT signal, clock signal, etc. The memory value may not be correct.
0x0c03:fatal err (Sfr function monitor timeout)	During access of sfr register, CPU did time out. > Check the HOLD signal, WAIT signal, clock signal, etc. The IOR value may not be correct.
0x0c04:fatal err (external flash memory database file is not exist)	Specified external flash memory database file is not exist. > Check to exist external flash memory database file in specified path.
0x0c05:fatal err (Monitor program update failed)	Failed to update monitor program in MINICUBE2.

0x0ca0: fatal err (Exec I/F abort error)	Can not communicate with ICE. > Please confirm the power of ICE, connection of the interface cable, or I/O address of the PC interface board.
0x0ca1: fatal err (monitor file missing)	Monitor file not found. > Necessary files may be damaged. Reinstall that files. > Please check the device that corresponds to using emulator.
0x0ca2: fatal err (specified device file doesn't correspond to OCD)	Device files other than OCD emulator was used. > A device file should check in the thing corresponding to OCD emulator. Or install newest device file.
0x0ca3: fatal err (EXEC is too old)	Unknown flag exists in OCD information part. Using EXEC is too old. > Use newest EXEC.
0x0ca4: fatal err (specified device file doesn't correspond to IECUBE)	Device files other than IECUBE was used. > A device file should check in the thing corresponding to IECUBE.
0xca5: fatal err (could not connect to target)	Could not connect to target. > Please check target connection or specified Main clock.
0xca6: fatal err (failed to read option byte)	Failed to read option byte > Please check setting of flash security.
0xca7: fatal err (failed to write option byte)	Failed to write option byte.
0xca8: fatal err (LPD connection was failed)	LPD connection was failed. > Please check setting of LPD connection.
0xca9: fatal err (emulator firmware is not correct)	Emulator firmware is not correct.

0xcaa: fatal err (emulator firmware version error)	Emulator firmware version is not matched.
0xcab: fatal err (emulator fpga is not correct)	Emulator fpga is not correct.
0xcac: fatal err (emulator fpga version error)	Emulator fpga version is not matched.
0xcad: fatal err (long trace option fpga version error)	Long trace option fpga version is not matched.
0xcae: fatal err (unmatch target MCU and POD)	The combination of target MCU and POD are not correct.
0xcb3: fatal err (target POD fpga version error)	Target POD fpga version is not matched.
0xcb4: fatal err (long trace option is not connected)	Long trace option is not connected.

User system abnormality

0x0200:user system err (verify err)	Verification error occurred. Failed in writing memory. > In case of OCD emulator, MINICUBE2 or IECUBE, Please check setting of **dclock** command.
0x02a0:user system err (bus hold)	Bus hold error. > CPU is in the bus-hold status. Reset the debugger.
0x02a1:user system err (stand by mode)	Stand by mode.
0x02a2:user system err (cannot break)	Can not compulsory break.
0x02a3:user system err (reset continuation)	Reset under continuation. In case of MINICUBE2, When the RESET pin in MINICUBE2 doesn't become high-level when reset is released, it error occurs. following causes are thought. *The connection with the target is not correct. *The reset circuit of the target doesn't operate normally.

0x0c1e:user system err (Instruction code for Flash self emulation doesn't exist)	Necessary instruction code for Flash self emulation doesn't exist. > Please replace Flash library supported Type4 Status Check error emulation.
0x0c20:user system err (Guard area access)	Guarded area can not be accessed.
0x0c21:user system err (NOREADY)	Memory was unready status.
0x0c22:user system err (NOREADY cancel break)	Memory unready status was canceled.
0x0c23:user system err (bus hold)	Bus hold under continuation. > Check the setting of the target board, or mask the HOLD pin.
0x0c24:user system err (break reset error)	It cannot shift to debug mode. > Check the clock signal. This may be caused by a stopped clock or a slow clock. > In case of MINICUBE2, it is generated when there is no response from the monitor program after releasing reset.
0x0c25:user system err (mask rom area)	Flash macro service ROM was accessed or stepped in.
0x0c26:user system err (FLMD terminal write protect)	FLMD terminal is in a write-protected state. > FLMD is not in the write-enabled status. Check the status of the FLMD0 and FLMD1 pins.
0x0c27:user system err(security flag disabled)	Security flag is in a write-protected state. > The security flag of the flash memory has disabled writing, block erasure, or chip erasure. Nothing can be written to the flash memory.
0x0c28:user system err(flash write disabled)	Internal RAM is not enough, the writing to flash memory is not made. > The internal RAM size is less than 4 KB and flash self-programming cannot be executed.
0x0c29:user system err (blank check failed)	The blank check of flash memory failed

0x0c2a:user system err (erase failed)	The erasing of flash memory failed.
0x0c2b:user system err (write failed)	The writing of flash memory failed.
0x0c2c:user system err (verify failed)	The internal verification of flash memory failed.
0x0c2d:user system err (PLL lock failed)	The writing of flash memory failed because could not lock PLL.
0x0c2e:user system err (no response from flash macro service)	No response from flash macro service.
0x0c2f:user system err (return unjust value from flash macro service)	Return unjust value from flash macro service.
0x0c30:user system err (necessary to release flash SFR prohibition setting)	Necessary to release flash SFR prohibition setting.
0x0c31:user system err (break failed by stop mode)	Could not break by stop mode. > Please release STOP mode or reset CPU.
0x0c32:user system err (no support device mode)	This device mode is not supported to write. > Please use single-chip-mode 0 to write to Flash memory.
0x0c33:user system err (tried disable on chip debug)	Tried disable on chip debug. > Please do not write 0 to the most significant bit of ID code.
0x0c34:user system err (tried to write reserved area)	Tried to write reserved area by on chip debug.
0x0c35:user system err (temporary program write error)	Ttemporary program write error > Please check to specify a correct device file.
0x0c36:user system err (IROM size is illegal for flash self emulation)	Flash self emulation function cannot be made enable for internal ROM size is set to other than the default size.
0x0c3a:user system err (no support data flash)	This device doesn't support Data Flash. > Please confirm whether used device file supports Data Flash.

0x0c3b:user system err (flash environmental is other than data flash)	Because it is accessing Code Flash area, it cannot access Data Flash area.
0x0c3c:user system err (external flash memory is not possible to write)	External flash memory state is not possible to write.
0x0c3d:user system err (could not erase to external flash memory)	Failed to erase to external flash memory.
0x0c3e:user system err (could not write to external flash memory)	Failed to write to external flash memory.
0x0c3f:user system err (tried to write invalid value in on chip debug)	Tried to write invalid value in on chip debug.

Status Error

0x0300:status err (user program is running)	User program is running.
0x0301:status err (user program not running)	User program is being breaked.
0x0302:status err (trace is working)	User program is being traced.
0x0303:status err (no trace data)	Not traced.
0x0304:status err (trace memory invalid)	Trace memory is not set.
0x0306:status err (no trace block)	No trace block exists.
0x0307:status err (illegal event set num)	No event condition exists.
0x0308:status err (no timer measurement)	No timer measurement is done.
0x0309:status err (no trigger frame)	No trigger frame exists.
0x030a:status err (timer off)	Tracer is being stopped.
0x030d:status err (timer on)	Timer is running.
0x030e:status err (mem range err)	Memory copy area is overlapped.
0x030f:status err (already set)	Trace has been already set.
0x0310:status err (no condition event num)	Event condition is not set.

0x0311:status err (full timer num)	Too many valid timer event conditions.
0x0312:status err (no timer num)	Specified timer event is not set.
0x0313:status err (map range err)	Illegal map range. > Check the setting in "Memory Mapping (mapping setting area)" in the Configuration dialog box.
0x0314:status err (delay event-mode set err)	Only trace delay mode can set with delay trigger.
0x0315:status err (delay mode is full)	Delay trigger cannot set without trace delay mode.
0x0316:status err (over mapping num)	Too many valid mapping.
0x03a0:status err (target power off)	Target is not turned on. > Check the target power supply. Check the cable connecting the in-circuit emulator and target board.
0x03a1:status err (step executive)	Step execution is being done.
0x03a2:status err (realtime measure running)	Timer and Tracer are running.
0x03a3:status err (mixed events specified)	Existed together appointed the event.
0x0b04:status err (realtime measure running)	Target power is already supplied. > Please delete -t3v or -t5v starting option.
0x0c40:status err (invalid address condition)	Status of effective event conditions cannot be changed.
0x0c42:status err (escape break, cannot run)	Monitor has failed in shift in the debugging mode. Please reset the CPU.
0x0c43:status err (emulator access failed)	Can not communicate with ICE. (IE-V850E1-CD-NW) Can not communicate with ICE. Please confirm the power of ICE, connection of the interface cable.(IECUBE)
0x0c44:status err (trace packet data missing)	trace packet data missing.
0x0c45:status error (power off reset effective)	Inside of Power off reset emulation cannot carry out program execution.

0x0c46:user system error (flash self emulation effective)	It cannot modify internal ROM size and internal RAM size for the flash self emulation function is made enable.
0x0c47:status err (enable ROM correction emulation)	Could not use because ROM correction emulation is emabled.
0x0c48:status err (disable flash write mode)	Could not use because flash write mode is disabled.
0x0c4d:status err (security unlock failed)	Failed to unlock security.
0x0c4e:status err (could not use software break setting by self programming is enable)	Could not use software break setting because self programming is enabled
0x0c4f:status err (could not use self programming by software break is enable)	Could not use self programming because software break is enabled
0x0caf:status err (exceed trace block)	Trace block can not be stepped over.

Parameter Error

0x0400:param err (illegal data)	llegal condition. > Settings of the used in-circuit emulator and those of the Configuration dialog box may not match.
0x0401:param err (timer overflow)	Result of timer measurement overflowed.
0x0402:param err (pass counter overflow)	Too many event conditions with path count.
0x0403:param err (address range err)	Too many address range conditions.
0x0404:param err (event num overflow)	Too many simultaneously-usable-event conditions.
0x0407:param err (initial data area overflow)	Too many initialization data.
0x0408:param err (search data overflow)	Too large search data (> 16 byte).
0x0409:param err (search data area overflow)	Too large search data (> search range).

0x040a:param err (sequential overflow)	Too many Linking-event conditions.
0x04a0:param err (trigger event overflow)	Too many Trigger-event.
0x04a1:param err (emulation mem insufficiency)	Not enough memory for emulation.
0x04a2:param err (bus size overflow)	Too many partition of bus size.
0x04a3:param err (BRS event overflow)	Too many execution-event conditions.
0x04a4:param err (BRA event overflow)	Too many bus-event conditions.
0x04a5:param err (external data overflow)	Too many External data.
0x0c60:param err (invalid event condition)	Event before execution cannot be set up other than break conditions.
0x0c61:param err (Can ft use hardware breakpoint)	Can not register event numbers which can not be used for hardware break.
0x0c62:paramerr (eventisusedhardwarebreakpoint)	Event numbers reserved for hardware breaks can not be used.
0x0c63:param err (trace event set error)	Event link conditions cannot set.
0x0c64:param err (rom emulation area error)	Too many ROM-emulation-RAM areas.
0x0c65:param err (event busy)	The event which is appointed presently is in the midst of using.
0x0c66:param err (not emulation mameory area)	Emulation memory area was appointed.
0x0c67:param err (block size error)	Writing of flash memory during block is not made.
0x0c69:param err (data flash area out of range)	It tried to access out of Data Flash area. Or it tried to access Data Flash area from other areas. > Please specify address and length to become in Data Flash area. Or please specify address and length not accessed Data Flash area.

0x0c6a:param err (external flash area out of range)	It tried to access out of External Flash area. Or it tried to access External Flash area from other areas. > Please specify address and length to become in External Flash area. Or please specify address and length not accessed External Flash area.
0x0c6b:param err(pseudo interrupt to set event)	Tried to set pseudo interrupt to set event.
0x0c6c:param err(could not set selecting trace resource)	Could not set selecting trace resource.
0x0560:param err(necessary to exclusive setting value)	Necessary to exclusive setting value.
0x0566:param err(hardware break overflow)	Too many settable event num.
0x0567:param err(overlap to MCU resources)	Could not access to memory because overlap to MCU resources.

Device Dependent Error

0x0c70:device depend err (DCU access error)	There is a possibility of having made a mistake in the selection of the device file. Please select the device file which corresponds to the target chip.
0x0c71:device depend err (reset request failed)	Please verify the clock pulse. You can think clock stop and the low-speed clock. In case of MINICUBE2, When the RESET pin in MINICUBE2 doesn't become low-level when reset is released, it error occurs. Following causes are thought. *The connection with the target is not correct. *The reset circuit of the target doesn't operate normally.
0x0c72:device depend err (monitor area access failed)	Failure of monitor area access. Following causes are thought. *The connection with the target is not correct. *The selection by UART or CSI is wrong. *The operation frequency input by the **dclock** command and the operation frequency of the device on target are different.

0x0c73:device depend err (monitor execution failed)	Failure of monitor execution. In case of MINICUBE2, When the ID code is changed while executing the user program, it is generated.
0x0c74:device depend err (CPU resource access failed)	Failure of CPU core resource access > There is a possibility of having made a mistake in the selection of the device file.
0x0c75:device depend err (illegal debug mode)	Transited to illegitimate debugging mode. > Please CPU reset.
0x0c76:device depend err (DCU access init error)	Initial condition when starting the DCU access is abnormal.
0x0c77:device depend error (DCU access error(verify))	Please verify the device file. > Please check the connection of DCK,DMS,DDI,DDO,DRSTZ.
0x0c78:device depend error (trace memory read error)	Failed in reading the trace data.
0x0c79:device depend error (disable on chip debug)	On chip debugging is disabled.
0x0c7b:device depend error (mismatched with specified LPD pin mode)	Could not connect to target because specified LPD pin mode is mismatched.
0x0c7c:device depend error (RSU ID code verify error)	Could not connect to target because ID code unlock is failed.
0x0590:device depend error (trace start failed)	Failed ot trace starting.
0x0591:device depend error (trace stop failed)	Failed to trace stopping.
0x0592:device depend error (trace buffer stop failed)	Failed to get trace datas in trace buffer.
0x0593:device depend error (trace is not supported)	Trace is not supported in this device.
0x0594:device depend error (selecting trace is not supported)	Selected trace is not supported in this device.

0x0595:device depend error (disable trace output)	Trace output is disabled.

IECUBE or IECUBE2 Starting Error

Check the target power on. Or please delete "-tc" option.	Please verify the power source of the target. Or "-tc" please delete option.
Remove the target. Or please add "-tc" option and power on the target.	Please remove the target. Or "-tc" option adding, please turn on the power of the target.
Power off and remove the target. Or please add"-tc" option.	Please remove the target. Or "-tc" option please add.
EXEC X.XX does not support this ICE. Please update EXEC to V1.56 or later.	EXEC version X.XX is not supported the ICE of use. > Please check whether the version of EXEC is the newest. Please update EXEC, when EXEC before V1.55 is being used.

IECUBE or IECUBE2 Starting Warning

Check the exchange adapter is connected.	Please verify whether the conversion adapter is connected.

RSU verify Error

RSU id-code verify error	Failed to compare the attestation code specified as "-id" option. > Specified attestation code is wrong. Please specify the attestation code again. > There may be selected a different device file. Please check a device file that corresponds to the device.

Server Starting Error

Couldn't start '850eserv2 ' as a remote debug server. Verify that '850eserv2 ' exists and is executable by you. Connection: No remote connection established.	When only this error message is displayed, there may be no license of a server. > Please install the license of 850eserv2.
EXEC Library V1.XX is too old Please update EXEC to V1.XX or later	850eserv2 cannot be started because an old EXEC library is used. > Please update a EXEC library. You can download the latest EXEC library from following web page: ***http://www.renesas.com/ghs_debug_if***

Emulation CPU status in running

CPU status (0x1) WAIT	Emulation CPU status is Waiting.
CPU status (0x2) HOLD	Emulation CPU status is Holding.
CPU status (0x8) HARDWARE STOP	Emulation CPU status is Stopping or Idle mode.
CPU status (0x10) RESET	Emulation CPU status is Reset.
CPU status (0x20) HALT	Emulation HALT mode.
CPU status (0x1000) COMBO	User interrupt is being processed at combo break.