User's Manual

R0E530650MCU00

User's Manual

Supported Devices:
M16C Family / M16C/60 Series
M16C/65 and 64A Groups

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corporation without notice. Please review the latest information published by Renesas Electronics Corporation through various means, including the Renesas Electronics Corporation website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev.6.02 Sep 2021

Downloaded from Arrow.com.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.

© 2021 Renesas Electronics Corporation. All rights reserved.

Downloaded from Arrow.com.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Downloaded from Arrow.com.

Preface

The R0E530650MCU00 is a full-spec emulator for MCUs of the M16C/60 Series M16C/65 and M16C/64A Groups. This user's manual mainly describes specifications of the R0E530650MCU00 and how to setup it.

All components of the R0E530650MCU00 are listed under "1.1 Package Components" (page 17). If you have any questions about the R0E530650MCU00, contact your local distributor.

The manuals relevant to usage of the R0E530650MCU00 are listed below. You can download the latest manuals from the Renesas Tools homepage (https://www.renesas.com/tools).

Related manuals

Item	Manual
Accessory	R0E0100TNPFJ00 User's Manual
	R0E0100TNPFK00 User's Manual
	R0E530650CFJ30 User's Manual
	R0E530650CFK10 User's Manual
Integrated development environment	High-performance Embedded Workshop User's Manual
C compiler	C/C++ Compiler Package for M16C Series, R8C Family C/C++ Compiler User's Manual
Assembler	C/ C++ Compiler Package for M16C Series, R8C Family Assembler User's Manual

Downloaded from Arrow.com.

Important

Before using this product, be sure to read this user's manual carefully.
Keep this user's manual, and refer to it when you have questions about this product.

Emulator:

"Emulator" in this document collectively refers to the following products manufactured by Renesas Electronics Corporation.

(1) E100 emulator main unit
(2) MCU unit
(3) Pitch converter board for connecting the user system

"Emulator" herein encompasses neither the customer's user system nor the host machine.

Purpose of use of the emulator:

This emulator is a device to support the development of systems that use the M16C Family M16C/60 Series M16C/65 and M16C/64A Groups of Renesas 16-bit single-chip MCUs. It provides support for system development in both software and hardware.

Be sure to use this emulator correctly according to said purpose of use. Please avoid using this emulator other than for its intended purpose of use.

For those who use this emulator:

This emulator can only be used by those who have carefully read the user's manual and know how to use it.

Use of this emulator requires basic knowledge of electric circuits, logical circuits, and MCUs.

When using the emulator:

(1) This product is a development-support unit for use in your program development and evaluation stages. When a program you have finished developing is to be incorporated in a mass-produced product, the judgment as to whether it can be put to practical use is entirely your own responsibility, and should be based on evaluation of the device on which it is installed and other experiments.
(2) In no event shall Renesas Electronics Corporation be liable for any consequence arising from the use of this product.
(3) Renesas Electronics Corporation strives to provide workarounds for and correct trouble with products malfunctions, with some free and some incurring charges. However, this does not necessarily mean that Renesas Electronics Corporation guarantees the provision of a workaround or correction under any circumstances.
(4) The product covered by this document has been developed on the assumption that it will be used for program development and evaluation in laboratories. Therefore, it does not fall within the scope of applicability of the Electrical Appliance and Material Safety Law and protection against electromagnetic interference when used in Japan.
(5) Renesas Electronics Corporation cannot predict all possible situations and possible cases of misuse that carry a potential for danger. Therefore, the warnings in this user's manual and the warning labels attached to the emulator do not necessarily cover all such possible situations and cases. The customer is responsible for correctly and safely using this emulator.
(6) The product covered by this document has not been through the process of checking conformance with UL or other safety standards and IEC or other industry standards. This fact must be taken into account when the product is taken from Japan to some other country.
(7) Renesas Electronics Corporation will not assume responsibility of direct or indirect damage caused by an accidental failure or malfunction in this product.

When disposing of the emulator:

Penalties may be applicable for incorrect disposal of this waste, in accordance with your national legislation.

Downloaded from Arrow.com.

Usage restrictions:

The emulator has been developed as a means of supporting system development by users. Therefore, do not use it as an embedded device in other equipment. Also, do not use it to develop systems or equipment for use in the following fields.

(1) Transportation and vehicular
(2) Medical (equipment that has an involvement in human life)
(3) Aerospace
(4) Nuclear power control
(5) Undersea repeaters

If you are considering the use of the emulator for one of the above purposes, please be sure to consult your local distributor.

About product changes:

We are constantly making efforts to improve the design and performance of this emulator. Therefore, the specification or design of this emulator, or this user's manual, may be changed without prior notice.

About rights:

(1) We assume no responsibility for any damage or infringement on patent rights or any other rights arising from the use of any information, products or circuits presented in this user's manual.
(2) The information or data in this user's manual does not implicitly or otherwise grant a license to patent rights or any other rights belonging to Renesas or to a third party.
(3) This user's manual and this emulator are copyrighted, with all rights reserved by Renesas. This user's manual may not be copied, duplicated or reproduced, in whole or part, without prior written consent from Renesas.

About diagrams:

Some diagrams in this user's manual may differ from the objects they represent.

Downloaded from Arrow.com.

Precautions for Safety

This chapter describes the precautions which should be taken in order to use this product safely and properly. Be sure to read and understand this chapter before using this product.
Contact us if you have any questions about the precautions described here.
This chapter describes the precautions which should be taken in order to use this product safely and properly. Be sure to read this chapter before using this product.

⚠ WARNING	WARNING indicates a potentially dangerous situation that will cause death or heavy wound unless it is avoided.
⚠ CAUTION	CAUTION indicates a potentially dangerous situation that will cause a slight injury or a medium-degree injury or property damage unless it is avoided.

In addition to the two above, the following are also used as appropriate.

△ means WARNING or CAUTION.
Example:



CAUTION AGAINST AN ELECTRIC SHOCK

⃠ means PROHIBITION.
Example:



DISASSEMBLY PROHIBITED

● means A FORCIBLE ACTION.
Example:



UNPLUG THE POWER CABLE FROM THE RECEPTACLE.

Downloaded from Arrow.com.

⚠ WARNING

Warnings for AC Power Supply:



If the attached AC power cable does not fit the receptacle, do not alter the AC power cable and do not plug it forcibly. Failure to comply may cause electric shock and/or fire.

Do not touch the plug of the AC power cable when your hands are wet. This may cause electric shock.

This product is connected signal ground with frame ground. If yours developing product is transform-less (not having isolation transformer of AC power), this may cause electric shock. Also, this may give an un-repairable damage to this product and yours developing product. While developing, connect AC power of the product to commercial power through isolation transformer in order to avoid these dangers.

If other equipment is connected to the same branch circuit, care should be taken not to overload the circuit.



When installing this equipment, insure that a reliable ground connection is maintained.



If you smell a strange odor, hear an unusual sound, or see smoke coming from this product , then disconnect power immediately by unplugging the AC power cable from the outlet.
Do not use this as it is because of the danger of electric shock and/or fire. In this case, contact your local distributor.

Before setting up this product and connecting it to other devices, turn off power or remove a power cable to prevent injury or product damage.

Warnings to Be Taken for Handling:



Do not modify this product. Personal injury due to electric shock may occur if this product is modified. Modifying the product will void your warranty.

Make sure nothing falls into the cooling fan on the top panel, especially liquids, metal objects, or anything combustible.

Warning for Installation:



Do not set this product in water or areas of high humidity. Make sure that the product does not get wet. Spilling water or some other liquid into the product may cause un-repairable damage.

Warning for Use temperature:

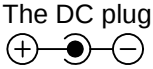


This equipment is to be used in an environment with a maximum ambient temperature of 35°C. Care should be taken that this temperature is not exceeded.

Caution for AC Adapter:



The DC plug of the AC adapter has the polarity shown below.



Use an AC adapter which complies with the safety standards of the country where it is to be used.

Downloaded from Arrow.com.

⚠CAUTION

Cautions to Be Taken for Turning On the Power:

Turn ON/OFF the power of the emulator and the user system as simultaneously as possible.
If you cannot turn on the powers simultaneously, turn on the emulator first and then the user system.

When turning on the power again after shutting off the power, wait about 10 seconds.

Cautions to Be Taken for Handling This Product:

Use caution when handling the product. Be careful not to apply a mechanical shock.

Do not touch the connector pins of the emulator and the target MCU connector pins directly. Static electricity may damage the internal circuits.

When attaching and removing the cable, hold the plug of the cable and do not touch the cable. Do not pull the emulator by the communications interface cable or the flexible cable. And, excessive flexing or force may break conductors.

Do not flex the flexible cable excessively. The cable may cause a break.

Do not use inch-size screws for this equipment. The screws used in this equipment are all ISO (meter-size) type screws. When replacing screws, use same type screws as equipped before.

Do not tape the flexible cable or apply adhesives to secure the cable. The shielding material on the surface of the cable may come off.

Note on Transporting the Product:

When sending your product for repair, use the packing box and cushioning material supplied with the product when it was delivered to you and specify caution in handling (handling as precision equipment). If packing of your product is not complete, it may be damaged during transportation. When you pack your product in a bag, make sure to use the conductive plastic bag supplied with the product (usually a blue bag). If you use a different bag, it may lead to further trouble with your product due to static electricity.

Caution to Be Taken for System Malfunctions:

If the emulator malfunctions because of interference like external noise, do the following to remedy the trouble.

(1) Exit the emulator debugger, and shut OFF the emulator and the user system.
(2) After a lapse of 10 seconds, turn ON the power of the emulator and the user system again, then launch the emulator debugger.

Caution to Be Taken for Disposal:

Penalties may be applicable for incorrect disposal of this waste, in accordance with your national legislation.

European Union regulatory notices:



The WEEE (Waste Electrical and Electronic Equipment) regulations put responsibilities on producers for the collection and recycling or disposal of electrical and electronic waste.
Return of WEEE under these regulations is applicable in the European Union only.
This equipment (including all accessories) is not intended for household use.
After use the equipment cannot be disposed of as household waste, and the WEEE must be treated, recycled and disposed of in an environmentally sound manner.
Renesas Electronics Europe GmbH can take back end of life equipment, register for this service at "http://www.renesas.eu/weee"

Downloaded from Arrow.com.

Contents

Page
Preface....4
Important....5
Precautions for Safety....7
Contents....10
User Registration....15
Terminology....16
1. Outline....17
1.1 Package Components....17
1.2 Other Tool Products Required for Development....17
1.3 System Configuration....18
1.3.1 System Configuration....18
1.3.2 Names and Functions of each part of the emulator....19
1.4 Specifications....21
1.4.1 Product Specifications....21
1.4.2 Regulatory Compliance Notices....22
1.4.3 Operating Environment....23
2. Setup....25
2.1 Flowchart of Starting Up the Emulator....25
2.2 Installing the Included Software....27
2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit....28
2.4 Connecting the Host Machine....29
2.5 Connecting the Emulator Power Supply....30
2.6 Turning ON the Power....31
2.6.1 Checking the Connections of the Emulator System....31
2.6.2 Turning the Power ON and OFF....31
2.7 Self-checking....32
2.8 Selecting the Clock Supply....33
2.8.1 Clock Source....33
2.8.2 Using an Internal Oscillator Circuit Board....34
2.8.3 Using the Oscillator Circuit on the User System....35
2.8.4 Using the Internal Generator Circuit....35
2.9 Connecting the User System....36
2.9.1 Connecting to a 100-pin 0.65mm Pitch Foot Pattern....37
2.9.2 Connecting to a 100-pin 0.5mm Pitch Foot Pattern....38
2.9.3 Connecting to an 80-pin 0.65mm Pitch Foot Pattern....39
2.9.4 Connecting to a 128-pin 0.5mm Pitch Foot Pattern....40
3. Tutorial....41
3.1 Introduction....41
3.2 Starting the High-performance Embedded Workshop....42
3.3 Connecting the Emulator....42
3.4 Downloading the Tutorial Program....43
3.4.1 Downloading the Tutorial Program....43
3.4.2 Displaying the Source Program....44
3.5 Setting Software Breakpoints....45
3.6 Executing the Program....46
3.6.1 Resetting the CPU....46
3.6.2 Executing the Program....46
3.7 Checking Breakpoints....47
3.7.1 Checking Breakpoints....47
3.8 Altering Register Contents....48
3.9 Referencing Symbols....49
3.10 Checking Memory Contents....50
3.11 Referencing Variables....51
3.12 Showing Local Variables....53
3.13 Single-Stepping a Program....53
3.13.1 Executing Step In Command....54
3.13.2 Executing the Step Out Command....55

Downloaded from Arrow.com.

3.13.3 Executing the Step Over Command....56
3.14 Forcibly Breaking a Program....57
3.15 Hardware Break Facility....58
3.15.1 Stopping a Program when It Executes a Specified Address....58
3.16 Stopping a Program when It Accesses Memory....59
3.17 Trace Facility....60
3.17.1 Showing the Trace Information Acquired by Fill Until Stop....61
3.17.2 Showing the Trace Information Acquired by Fill around TP....64
3.17.3 Showing a Function Execution History....66
3.17.4 Filter Facility....68
3.18 Stack Trace Facility....70
3.19 What Next?....71
4. Preparing to Debug....72
4.1 Starting the High-performance Embedded Workshop....72
4.2 Creating a New Workspace (Toolchain Unused)....73
4.3 Creating a New Workspace (Toolchain Used)....75
4.4 Opening an Existing Workspace....78
4.5 Connecting the Emulator....79
4.5.1 Connecting the Emulator....79
4.5.2 Reconnecting the Emulator....79
4.6 Disconnecting the Emulator....80
4.6.1 Disconnecting the Emulator....80
4.7 Quitting the High-performance Embedded Workshop....80
4.8 Setting Up the Debug....81
4.8.1 Specifying a Download Module....81
4.8.2 Setting Up Automatic Execution of Command Line Batch Files....82
5. Debugging Functions....83
5.1 Setting Up the Emulation Environment....84
5.1.1 Setting Up the Emulator at Startup....84
5.1.2 Setting Up the Target MCU....85
5.1.3 Setting Up the System....87
5.1.4 Creating a Memory Map....90
5.1.5 Setting Up Flash ROM Overwrite....91
5.1.6 Setting the Warning of Exceptional Events....92
5.1.7 Showing Progress in Boot-up Processing....93
5.2 Downloading a Program....95
5.2.1 Downloading a Program....95
5.2.2 Showing the Source Code....95
5.2.3 Turning columns in all source files off....97
5.2.4 Turning columns in one source file off....97
5.2.5 Showing Assembly Language Code....98
5.2.6 Correcting Assembly Language Codes....99
5.3 Displaying Memory Contents in Real Time....100
5.3.1 Displaying Memory Contents in Real Time....100
5.3.2 Setting RAM Monitor Update Intervals....101
5.3.3 Clearing RAM Monitor Access History....101
5.3.4 Clearing RAM Monitor Error Detection Data....101
5.4 Showing the Current Status....102
5.4.1 Showing the Emulator Status....102
5.4.2 Showing the Emulator Status in the Status Bar....103
5.5 Periodically Reading Out and Showing the Emulator Status....104
5.5.1 Periodically Reading Out and Showing the Emulator Information....104
5.5.2 Selecting the Items to Be Displayed....105
5.6 Using Software Breakpoints....106
5.6.1 Using Software Breakpoints....106
5.6.2 Adding/Removing Software Breakpoints....107
5.6.3 Enabling/Disabling Software Breakpoints....108
5.7 Using Events....110
5.7.1 Using Events....110
5.7.2 Adding Events....111
5.7.3 Removing Events....116
5.7.4 Registering Events....118

Downloaded from Arrow.com.

5.7.5 Entering Events Each Time or Reusing Events 120
5.7.6 Applying Events 121
5.8 Setting Hardware Break Conditions 122
5.8.1 Setting Hardware Break Conditions 122
5.8.2 Setting Hardware Breakpoints 122
5.8.3 Saving/Loading the Set Contents of Hardware Breaks 125
5.9 Looking at Trace Information 126
5.9.1 Looking at Trace Information 126
5.9.2 Acquiring Trace Information 126
5.9.3 Setting Trace Information Acquisition Conditions 128
5.9.4 Setting Trace Modes 130
5.9.5 Setting Trace Points 132
5.9.6 Setting Capture/Do not Capture Conditions 136
5.9.7 Selecting the Content of Trace Acquisition 138
5.9.8 Showing Trace Results 139
5.9.9 Filtering Trace Information 141
5.9.10 Searching for Trace Records 142
5.9.11 Saving Trace Information to Files 143
5.9.12 Loading Trace Information from Files 144
5.9.13 Temporarily Stopping Trace Information Acquisition 144
5.9.14 Restarting Trace Information Acquisition 144
5.9.15 Switching Timestamp Display 144
5.9.16 Showing the History of Function Execution 145
5.9.17 Showing the History of Task Execution 146
5.10 Measuring Performance 147
5.10.1 Measuring Performance 147
5.10.2 Showing the Result of Performance Measurement 147
5.10.3 Setting Performance Measurement Conditions 148
5.10.4 Starting Performance Measurement 150
5.10.5 Clearing Performance Measurement Conditions 151
5.10.6 Clearing the Performance Measurement Result 151
5.10.7 About the Maximum Measurement Time of Performance 151
5.11 Acquiring Code Coverage 152
5.11.1 Acquiring Code Coverage 152
5.11.2 Opening the Code Coverage Window 152
5.11.3 Allocating Code Coverage Memory (Hardware Resource) 153
5.11.4 Code Coverage in an Address Range 155
5.11.5 Adding Address Ranges 156
5.11.6 Changing Address Ranges 157
5.11.7 Removing Address Ranges 158
5.11.8 Code Coverage in a Source File 160
5.11.9 Adding Source Files 161
5.11.10 Removing Source Files 162
5.11.11 Showing Percentages and Graphs 164
5.11.12 Using the Sort Function 165
5.11.13 Searching for Unexecuted Lines 166
5.11.14 Clearing Code Coverage Information 167
5.11.15 Updating Coverage Information 167
5.11.16 Preventing Update of Coverage Information 167
5.11.17 Saving the Code Coverage Information to a File 168
5.11.18 Loading Code Coverage Information from a File 168
5.11.19 Coverage Information File Load Modes 169
5.11.20 Displaying Code Coverage Information in the Editor Window 171
5.12 Acquiring Data Coverage 172
5.12.1 Acquiring Data Coverage 172
5.12.2 Opening the Data Coverage Window 172
5.12.3 Allocating Data Coverage Memory (Hardware Resource) 173
5.12.4 Data Coverage in an Address Range 175
5.12.5 Adding Address Ranges 176
5.12.6 Changing Address Ranges 177
5.12.7 Removing Address Ranges 178
5.12.8 Data Coverage in a Section 179

Downloaded from Arrow.com.

5.12.9 Adding Sections 180
5.12.10 Removing Sections 181
5.12.11 Data Coverage in a Task Stack 182
5.12.12 Clearing Data Coverage Information 183
5.12.13 Updating Coverage Information 183
5.12.14 Preventing Update of Coverage Information 183
5.12.15 Saving the Data Coverage Information to a File 184
5.12.16 Loading Data Coverage Information from a File 184
5.13 Viewing Realtime Profile Information 186
5.13.1 Viewing Realtime Profile Information 186
5.13.2 Setting Realtime Profile Measurement Modes 188
5.13.3 Measuring Function Profiles 188
5.13.4 Setting Function Profile Measurement Ranges 189
5.13.5 Saving Function Profile Measurement Ranges 190
5.13.6 Loading Function Profile Measurement Ranges 190
5.13.7 Measuring Task Profiles 191
5.13.8 Setting Task Profile Measurement Ranges 192
5.13.9 Saving Task Profile Measurement Tasks 193
5.13.10 Loading Task Profile Measurement Tasks 193
5.13.11 Clearing Realtime Profile Measurement Results 194
5.13.12 Saving Realtime Profile Measurement Results 194
5.13.13 Setting the Measurement Interval 194
5.13.14 Maximum Measurement Time of the Realtime Profile 195
5.14 Detecting Exceptional Events 196
5.14.1 Detecting Exceptional Events 196
5.14.2 Detecting an Access Protect Violation 196
5.14.3 Setting an Access Protected Area 198
5.14.4 Detecting Initialization-Omitted 202
5.14.5 Detecting Stack Access Violation 203
5.14.6 Detecting a Performance Overflow 204
5.14.7 Detecting a Realtime Profile Overflow 204
5.14.8 Detecting a Trace Memory Overflow 205
5.14.9 Detecting a Task Stack Access Violation 205
5.14.10 Setting a Task Stack Area 206
5.14.11 Detecting an OS Dispatch 209
5.15 Using the Start/Stop Function 210
5.15.1 Opening the Start/Stop Function Setting Dialog Box 210
5.15.2 Specifying the Work Address 210
5.15.3 Specifying the Routine to be Executed 210
5.15.4 Limitations of the Start/Stop Function 211
5.15.5 Limitations to the Statements written in a Specified Routine 211
5.16 Using the Trigger Output Function 212
5.16.1 Using the External Trigger Cable for Output 212
5.16.2 Opening the Trigger Output Conditions Dialog Box 213
5.16.3 Manual Setting for Output through Trigger Pins 31 to 24 214
5.16.4 Setting for Output through Trigger Pins 20 to 16 215
5.16.5 Events 216
6. Troubleshooting (Action in Case of an Error) 217
6.1 Flowchart for Remediation of Trouble 217
6.2 Error in Self-checking 218
6.3 Errors Reported in Booting-up of the Emulator 219
6.4 How to Request Support 221
7. Hardware Specifications 222
7.1 Target MCU Specifications 222
7.2 Differences between the Actual MCU and Emulator 223
7.3 Connection Diagram 225
7.3.1 Connection Diagram for the R0E530650MCU00 225
7.4 External Dimensions 226
7.4.1 External Dimensions of the E100 Emulator 226
7.4.2 External Dimensions of the Converter Board R0E0100TNPFJ00 227
7.4.3 External Dimensions of the Converter Board R0E0100TNPFK00 228
7.4.4 External Dimensions of the Converter Board R0E530650CFJ30 229

Downloaded from Arrow.com.

7.4.5 External Dimensions of the Converter Board R0E530650CFK10 230
7.5 Notes on Using the MCU Unit 231
8. Maintenance and Warranty 235
8.1 User Registration 235
8.2 Maintenance 235
8.3 Warranty 235
8.4 Repair Provisions 236
8.5 How to Make Request for Repair 236
Revision History 237

Downloaded from Arrow.com.

User Registration

When you install debugger software, a text file for user registration is created on your PC. Fill it in and email it to your local distributor. If you have replaced an emulator main unit or emulation probe, rewrite an emulator name and serial number in the text file you filled in earlier to register your new hardware products.

Your registered information is used for only after-sale services, and not for any other purposes. Without user registration, you will not be able to receive maintenance services such as a notification of field changes or trouble information. So be sure to carry out the user registration.

For more information about user registration, please contact your local distributor.

Downloaded from Arrow.com.

Terminology

Some specific words used in this user's manual are defined below.

MCU unit (R0E530650MCU00)
This means the E100 emulator for the M16C/65 and M16C/64A Groups.

Emulator system
This means an emulator system built around the MCU unit (R0E530650MCU00). The emulator system is configured with an emulator main unit (R0E001000EMU00), MCU unit (R0E530650MCU00), emulator power supply, USB cable, emulator debugger and host machine.

Integrated development environment: High-performance Embedded Workshop
This tool provides powerful support for the development of embedded applications for Renesas microcomputers. It has an emulator debugger function allowing the emulator to be controlled from the host machine via an interface. Furthermore, it permits a range of operations from editing a project to building and debugging it to be performed within the same application. In addition, it supports version management.

Emulator debugger
This means a software tool that is started up from the High-performance Embedded Workshop, and controls the MCU unit and enables debugging.

Firmware
This means a control program stored in the emulator. This analyzes the contents of communications with the emulator debugger and controls the emulator hardware. To upgrade the firmware, download the program from the emulator debugger.

Host machine
This means a personal computer used to control the emulator.

Target MCU
This means the MCU to be debugged.

User system
This means a user's application system in which the MCU to be debugged is used.

User program
This means the program to be debugged.

Evaluation MCU
This means the MCU mounted on the emulator which is operated in a dedicated mode for use with tools.

#
This symbol indicates that a signal is active-low (e.g. RESET#).

Downloaded from Arrow.com.

1. Outline

This chapter describes the package components, the system configuration, and the specifications of the emulator functions and operating environment.

1.1 Package Components

The R0E530650MCU00 package consists of the following items. After you have unpacked the box, check if your R0E530650MCU00 contains all of these items.

Table 1.1 Package components

Item		Quantity
R0E530650MCU00 MCU board		1
Oscillator module (20MHz) mounted on the IC17 socket		1
R0E001000FLX10 flexible cable		2
R0E530650MCU00 Release Notes (English)		1
R0E530650MCU00 Release Notes (Japanese)		1
Repair Request Sheet (English)		1
Repair Request Sheet (Japanese)		1
CD-ROM	- M16C R8C E100 Emulator Software (M16C R8C E100 Emulator Debugger included) - User's Manual	1

* Please keep the R0E530650MCU00's packing box and cushioning materials at hand for later reuse in sending the product for repairs or for other purposes. Always use the original packing box and cushioning material when transporting the MCU unit.
* If you have any questions or are in doubt about any point regarding the packaged product, contact your local distributor.

1.2 Other Tool Products Required for Development

To proceed with the development of a program for M16C/60 Series M16C/65 and M16C/64A Groups MCUs, the products listed below are necessary in addition to those contained in the package and listed above. Procure them separately.

Table 1.2 Other tool products required for development

Product	Part No.
Emulator main unit E100	R0E001000EMU00
100-pin 0.65mm pitch QFP (PRQP0100JD-B Previous code: 100P6F-A)	R0E0100TNPFJ00
100-pin 0.5mm pitch LQFP (PLQP0100KB-A Previous code: 100P6Q-A)	R0E0100TNPFK00
80-pin 0.65mm pitch LQFP (PLQP0080JA-A Previous code: FP-80W)	R0E530650CFJ30
128-pin 0.5mm pitch LQFP (PLQP0128KB-A Previous code: 128P6Q–A)	R0E530650CFK10

* To purchase the product, contact your local distributor.

Downloaded from Arrow.com.

1.3 System Configuration

1.3.1 System Configuration

Figure 1.1 shows the configuration of the emulator system.

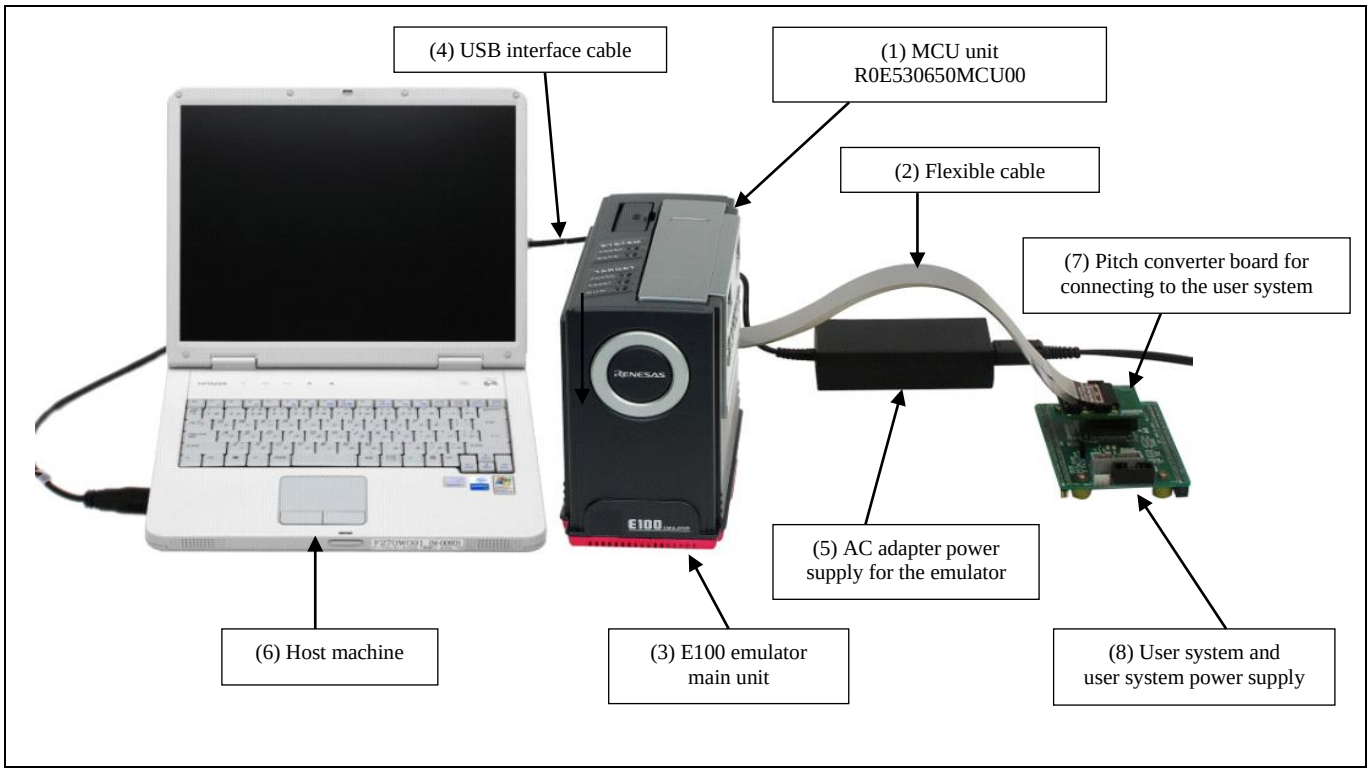


Figure 1.1 System configuration

(1) MCU Unit R0E530650MCU00 (this product)
This is an MCU board for the M16C/60 Series M16C/65 and M16C/64A Groups MCUs with 792 KB ROM and contains an evaluation MCU.
(2) Flexible cable R0E001000FLX10 (included)
(3) E100 Emulator main unit R0E001000EMU00
This is the E100 emulator main unit.
(4) USB interface cable
This is an interface cable for the host machine and emulator.
(5) AC adapter supply for the emulator
(6) Host machine
A personal computer to control the emulator.
(7) Pitch converter board for connecting the user system R0E0100TNPFJ00, etc.
(8) User system and user system power supply
User system is your application system. This emulator can be used without the user system.
The user system power supply is power supply for the user system. This emulator cannot supply power to the user system. Get a power supply separately.

Downloaded from Arrow.com.

1.3.2 Names and Functions of each part of the emulator

Figure 1.2 shows the names of each part of the emulator.



Figure 1.2 Names of each part of the emulator

(1) Power switch

This is a switch to turn the emulator ON and OFF.

(2) USB cable connector

This is a connector for connecting the USB cable of the emulator.

(3) Power connector

This is a connector for connecting the DC cable of the AC power adapter of the emulator.

(4) External trigger connector

This is a connector to connect the external trigger cable of the emulator.

Downloaded from Arrow.com.

(5) System Status LEDs
The system status LEDs indicate the emulator E100's power supply, operating state of firmware, etc. Table 1.3 lists the definitions of each system status LED.

Table 1.3 Definitions of the system status LEDs

Name	Status	Meaning
POWER	ON	Emulator system power is turned ON.
	OFF	Emulator system power is turned OFF.
SAFE	ON	Emulator system is operating normally.
	Flashing	Emulator system cannot communicate with the host machine.
	Flashing (every 2 seconds)	The self-checking is in progress.
	OFF	Emulator system is not operating normally (system status error).

(6) Target Status LEDs
The target status LEDs indicate operating state of the target MCU and power supply of the user system. Table 1.4 lists the definition of each target status LED.

Table 1.4 Definitions of the target status LEDs

Name	Status	Meaning
POWER	ON	Power is being supplied to the user system.
	OFF	Power is not being supplied to the user system.
RESET	ON	Target MCU is being reset, or reset signal of the user system is held low.
	OFF	Target MCU is not being reset.
RUN	ON	User program is being executed.
	OFF	User program has been halted.

Note on the Target Status POWER LED:
- If your MCU has two or more Vcc pins, the LED does not light up unless power is supplied to all the pins.

Downloaded from Arrow.com.

1.4 Specifications

1.4.1 Product Specifications

Table 1.5 lists the specifications of the R0E530650MCU00.

Table 1.5 Specifications of the R0E530650MCU00

Applicable MCU	M16C/60 Series M16C/65 and M16C/64A Groups MCUs with 792 KB ROM	
Applicable MCU mode	Single-chip mode, memory expansion mode, microprocessor mode	
Maximum ROM/RAM capacity	1. Internal flash ROM: 8KB+16KB+256KB+512KB 0E000h--0FFFFh, 10000h--13FFFh, 40000h--7FFFFh, 80000h--FFFFFh 2. Internal RAM: 47KB 00400h--0BFFFh	
Maximum operating frequency	Power supply voltage: 2.7 to 5.5V, 32MHz (with PLL) (Emulation memory 0wait: 12MHz, 1wait or more: 32MHz)	
Software break	4096 points (uses RAM for break point capability before execution)	
Hardware break	16 points (Execution address, bus detection, interrupt, external trigger signal)	
Combination, pass count	- Cumulative AND/OR/status transition - 255 pass counts	
Detection of exceptional events	Violation of access protection/Read from uninitialized memory/ Stack access violation/Performance overflow/Realtime profile overflow/ Trace memory overflow/Task stack access violation/OS dispatch	
Real-time tracing	192bits × 4M cycles (Address, data, status, CPU status, bus status, target status, task ID, timestamp, 32 external trigger inputs)	
Trace modes	Fill until stop/fill until full/fill around TP/repeat fill until stop/repeat fill until full	
Extraction/deletion of trace data	- Extracting or deleting data by specifying events or extracting the instruction that accesses the specified data - Extracting data before and after trace points	
Real-time RAM monitor	- 16,384 bytes (512 bytes × 32 blocks) - Data/last access	
Time measurement	- Execution time between program start and stop - Maximum/minimum/average execution time and number of passes through eight specified sections - Clock used to count times: 10ns to 1.6μs	
Coverage measurement	C0: 2 Mbytes (256 Kbytes × 8 blocks) C1: 1 Mbyte (128 Kbytes × 8 blocks)	
Profile	1 MB (128 KB × 8 blocks)	
Connection to user system	100-pin 0.65mm pitch QFP	R0E0100TNPFJ00
	100-pin 0.5mm pitch LQFP	R0E0100TNPFK00
	80-pin 0.65mm pitch LQFP	R0E530650CFJ30
	128-pin 0.5mm pitch LQFP	R0E530650CFK10
Emulator power supply	Supplied from an AC adapter (power supply voltage: 100 to 240 V, 50/60 Hz)	

Downloaded from Arrow.com.

1.4.2 Regulatory Compliance Notices

● European Union regulatory notices

This product complies with the following EU Directives. (These directives are only valid in the European Union.)

CE Certifications:

・Electromagnetic Compatibility (EMC) Directive 2014/30/EU

EN 55032 Class A

WARNING: This is a Class A product. This equipment can cause radio frequency noise when used in the residential area. In such cases, the ser/operator of the equipment may be required to take appropriate countermeasures under his responsibility.

EN 55024

・Information for traceability

・Authorised representative

Name: Renesas Electronics Corporation
Address: Toyosu Foresia, 3-2-24, Toyosu, Koto-ku, Tokyo 135-0061, Japan

・Person responsible for placing on the market

Name: Renesas Electronics Europe GmbH
Address: Arcadiastrasse 10, 40472 Dusseldorf, Germany

・Trademark and Type name

Trademark: Renesas
Product name: E100 Emulator MCU Unit
Type name: R0E530650MCU00

Environmental Compliance and Certifications:

・Waste Electrical and Electronic Equipment (WEEE) Directive 2012/19/EU

● United States Regulatory notices

This product complies with the following EMC regulation. (This is only valid in the United States.)

FCC Certifications:

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:
(1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

CAUTION: Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

Downloaded from Arrow.com.

1.4.3 Operating Environment

Make sure to use this emulator in the operating environments listed in Tables 1.6 to 1.8.

Table 1.6 Operating environmental conditions

Item	Description
Operating temperature	5 to 35°C (no condensation)
Storage temperature	-10 to 60°C (no condensation)

Table 1.7 Operating environment of the host machine (Windows® XP)

Item	Description
Host machine	IBM PC/AT compatible
OS	Windows® XP 32-bit edition [*1] [*3]
CPU	Pentium 4 running at 1.6 GHz or more recommended
Interface	USB 2.0 / USB 1.1 [*2]
Memory	1 Gbyte or larger (more than 10 times the file size of the load module) recommended
Pointing device such as mouse	Mouse or any other pointing device usable with the above OS that can be connected to the host machine
CD drive	Needed to install the emulator debugger or refer to the user's manual

Table 1.8 Operating environment of the host machine (Windows Vista® or Windows® 7)

Item	Description
Host machine	IBM PC/AT compatible
OS	Windows Vista® 32-bit edition [*1] [*4] Windows® 7 32-bit edition / 64-bit edition [*1]
CPU	Pentium 4 running at 3GHz or Core 2 Duo running at 1GHz or more recommended
Interface	USB 2.0 / USB 1.1 [*2]
Memory	2 Gbyte or larger (more than 10 times the file size of the load module) recommended (32-bit edition) 3 Gbyte or larger (more than 10 times the file size of the load module) recommended (64-bit edition)
Pointing device such as mouse	Mouse or any other pointing device usable with the above OS that can be connected to the host machine
CD drive	Needed to install the emulator debugger or refer to the user's manual

Notes:

*1: Windows and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other company or product names are the property of their respective owners.

*2: Operation with all combinations of host machine, USB device and USB hub is not guaranteed for the USB interface.

*3: The 64-bit edition of Windows® XP is not supported.

*4: The 64-bit edition of Windows Vista® is not supported.

Downloaded from Arrow.com.

1.4.4 Specifications of the AC Adapter

The user must prepare an AC adapter and a power cable. Use an AC adapter which complies with the safety standards of the country where it is to be used. Table 1.9 lists the recommended specifications of the AC adapter.

Table 1.9 Recommended Specifications of the AC Adapter

Item	Description
AC input voltage range	AC 100 to 240 V, 50/60Hz single phase
Output power	36 W
DC output voltage, current	12.0 V, 3.0 A
DC output polarity	EIAJ TYPE IV, inner-side positive/outer-side negative

Downloaded from Arrow.com.

2. Setup

This chapter describes the preparation for using the MCU unit, the procedure for starting up the emulator and how to change settings.

2.1 Flowchart of Starting Up the Emulator

The procedure for starting up the emulator is shown in Figures 2.1 and 2.2. For details, refer to each section hereafter. If the emulator does not start up normally, refer to "6. Troubleshooting (Action in Case of an Error)".

1	Check the package components.	Refer to "1.1 Package Components" (page 17).
▼		
2	User registration	Refer to "User Registration" (page 15).
▼		
3	Install the included software, etc.	Install them from the included CD-ROM.
▼		
4	Connect the MCU unit to the E100 main unit.	Refer to "2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit" (page 28).
▼		
5	Connect the host machine.	Connect the USB interface connector of the emulator and the USB port of the host machine.
▼		
6	Connect the emulator power supply.	Connect an emulator power supply to the power connector.
▼		
7	Turn on the power supply.	Turn ON the power to the emulator.
▼		
8	Check the LEDs on the emulator.	Check that the system status LEDs are lighting. When the user system is not connected, the POWER LED does not light up.
▼		
9	Try to use the emulator debugger according to the tutorial.	When using the emulator debugger for the first time, try to use the emulator functions, referring to "3. Tutorial" (page 41). If the emulator debugger does not start, follow steps 10 through 13 in this chart to perform a self-checking.
▼		
10	Create a project in the High-performance Embedded Workshop.	Start up the High-performance Embedded Workshop and create a project by following "4. Preparing to Debug" (page 72).
▼		
11	Start up the emulator debugger from the High-performance Embedded Workshop.	Start up the High-performance Embedded Workshop to launch the emulator debugger by following "4. Preparing to Debug" (page 72).
▼		
12	Set up the operating environment of the emulator debugger.	Make settings on the Device page in the Device Setting dialog box of the emulator debugger and select the "Start booting up on successful completion of self-checking" checkbox by following "4. Preparing to Debug" (page 72).
▼		
13	Self-checking	

Figure 2.1 Flowchart of starting up the emulator (for the first time)

Downloaded from Arrow.com.

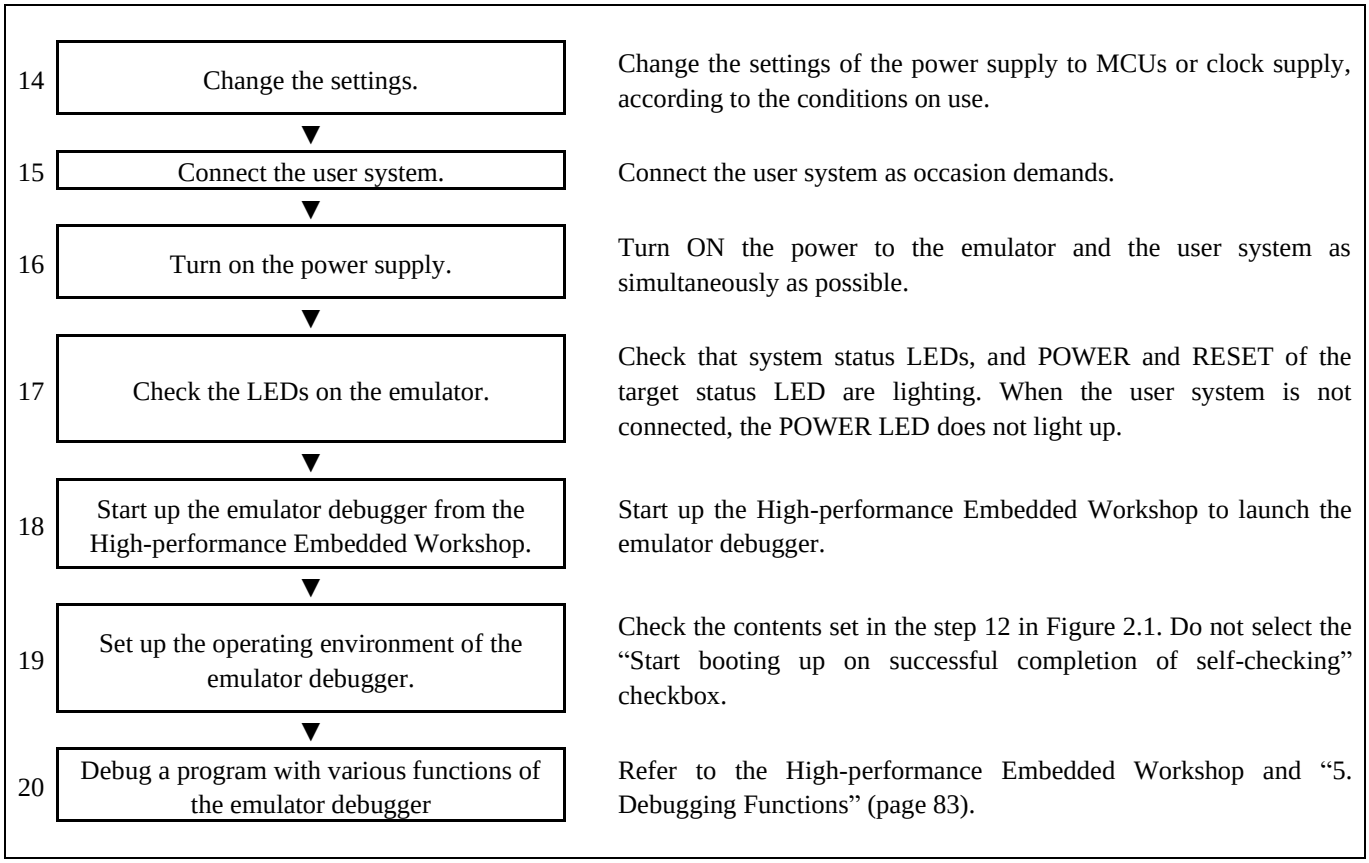


Figure 2.2 Flowchart of starting up the emulator (after the self-checking)

Downloaded from Arrow.com.

2.2 Installing the Included Software

If you have Windows® 7, Windows Vista® or Windows® XP on the host machine, this installation must be executed by a user with administrator rights. Note that users without administrator rights cannot complete the installation.

Place the CD-ROM in the CD-ROM drive and follow the instructions to install the software.

Downloaded from Arrow.com.

2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit

Figure 2.3 shows the procedure for connecting the MCU Unit to the E100 Emulator Main Unit.

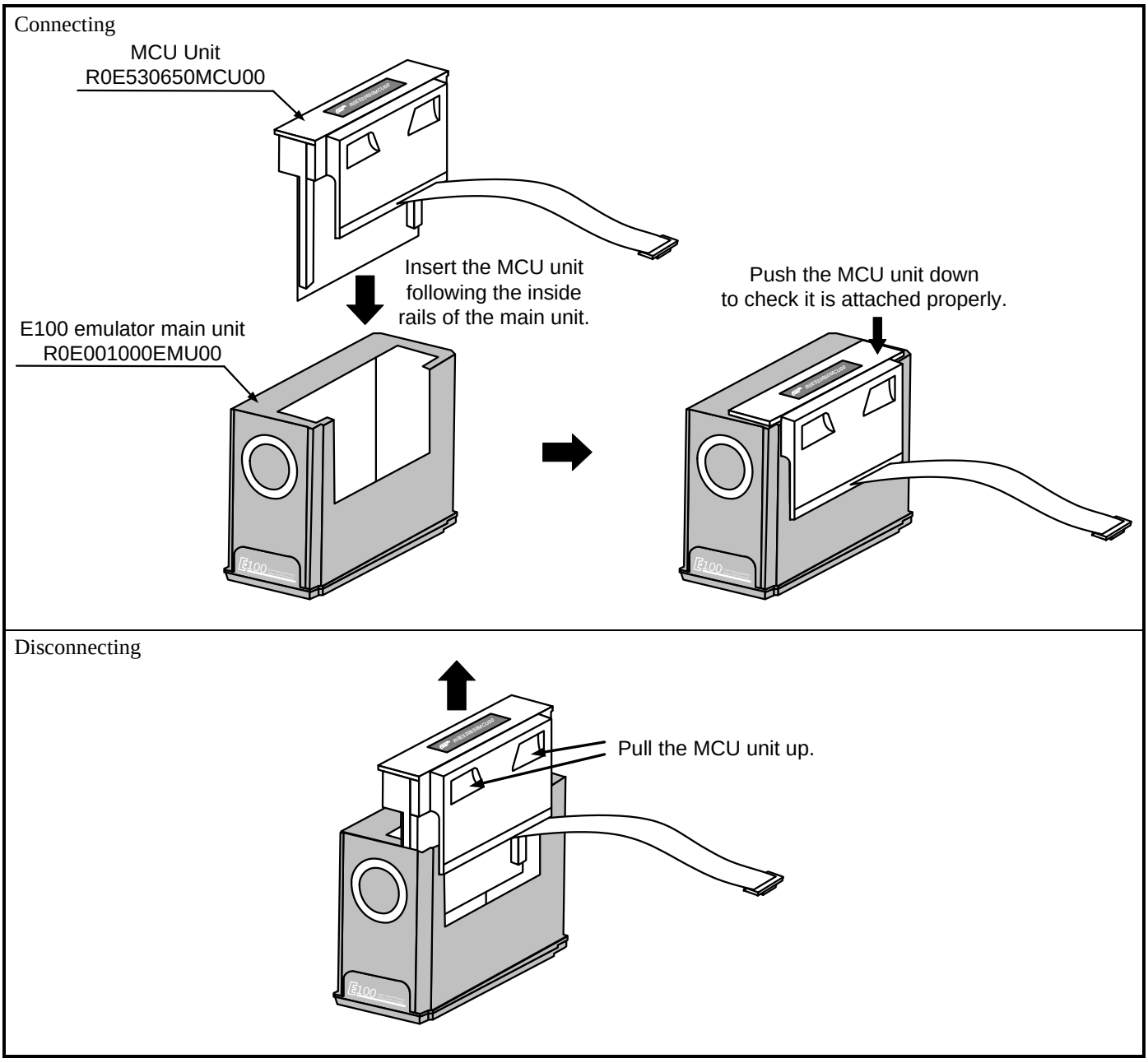


Figure 2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit

⚠ CAUTION

Note on Connecting the MCU Unit to the E100 Emulator Main Unit:

- Always shut OFF power when connecting the MCU unit to the E100 emulator main unit. Otherwise, internal circuits may be damaged.

Downloaded from Arrow.com.

2.4 Connecting the Host Machine

USB interface is used for connecting the emulator to the host machine. The USB cable is connected to the USB cable connector of the emulator and the USB port of the host machine.

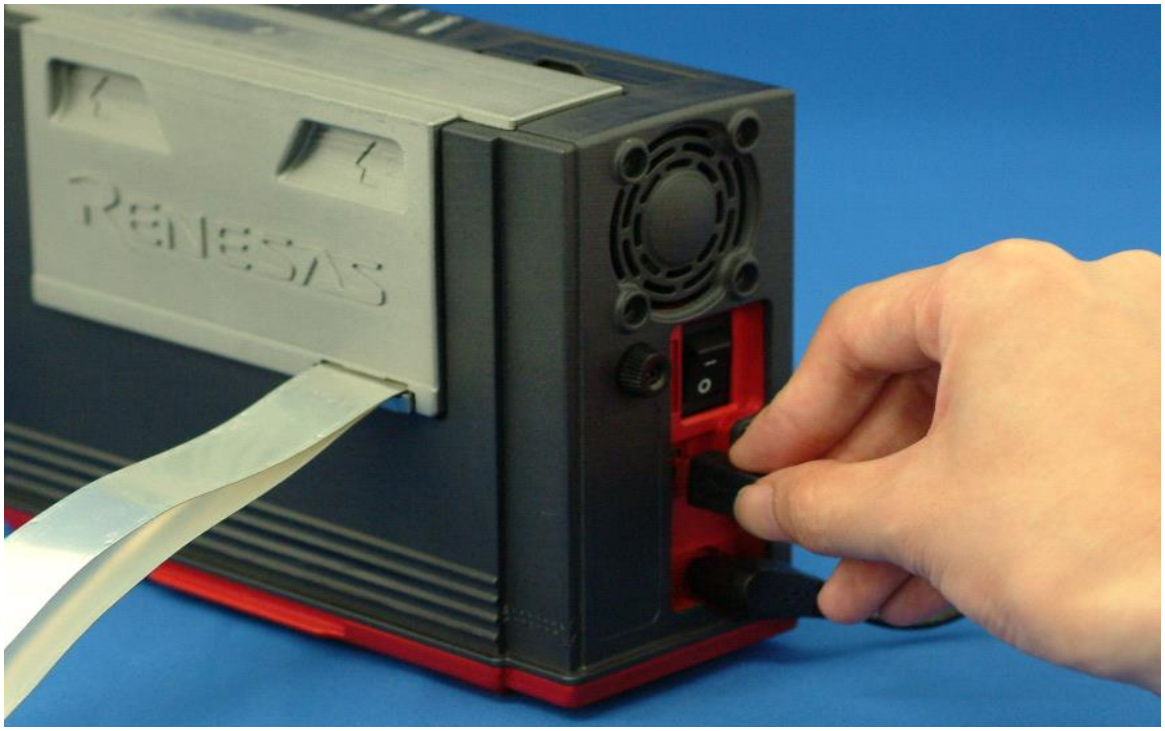


Figure 2.4 Connecting the host machine

Downloaded from Arrow.com.

2.5 Connecting the Emulator Power Supply

Power is supplied from an AC adapter to the emulator. The following shows how to connect the AC adapter.

(1) Turn OFF the power of the emulator.
(2) Connect the DC cable of the AC adapter to the emulator.
(3) Connect the AC power cable to the AC adapter.
(4) Connect the AC power cable to the outlet.



Figure 2.5 Connecting the emulator power supply

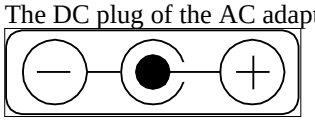
⚠ CAUTION

Cautions for AC Power Cable:

If the AC power cable does not fit the receptacle, do not alter the AC power cable and do not plug it forcibly. Failure to comply may cause electric shock and/or fire.
Do not touch the plug of the AC power cable when your hands are wet. This may cause electric shock.

Cautions for AC Adapter:

The DC plug of the AC adapter has the polarity shown below.



Use an AC adapter which complies with the safety standards of the country where it is to be used.

Downloaded from Arrow.com.

2.6 Turning ON the Power

2.6.1 Checking the Connections of the Emulator System

Before turning the power ON, check the connection of the interface cable with the host machine, emulator, and user system.

2.6.2 Turning the Power ON and OFF

- Turn ON/OFF the power of the emulator and user system as simultaneously as possible.
- When the SAFE LED of the system LEDs is flashing, check that the USB cable is connected to the host machine. When each of the target status LEDs is flashing, check that the MCU unit is connected.
- When turning ON the power again after shutting OFF the power, wait for about 10 seconds.

Notes on Power Supply:

- The emulator pin Vcc is connected to the user system in order to monitor user system voltage. For this reason, the emulator cannot supply power to the user system. Supply power to the user system separately.

  The voltage of the user system should be as follows.

  $2.7\ V \leq Vcc1 = Vcc2 \leq 5.5\ V$
- When you start the emulator without the user system, do not attach a converter board. When starting with a converter board, the MCU will be in a reset status.
- When you start the emulator without the user system, take care that metallic pieces are not touched to the connector at the head of the flexible cable.
- Do not leave either the emulator or user system powered on. The internal circuits may be damaged due to leakage current.

Downloaded from Arrow.com.

2.7 Self-checking

Self-checking is to check if the emulator functions operate properly. To run the self-check function of the emulator, follow the procedure below. While the self-checking is in progress, the states of the LEDs will change as shown in Figure 2.6. In case of ERROR, because the states of the target status LEDs will change depending on the types of errors, check the system status LEDs.

(1) If the user system is connected, disconnect the converter board and the user system.
(2) Turn on the emulator.
(3) Launch the emulator debugger, and select the "Start booting up on successful completion of self-checking" checkbox in the Device Setting dialog box.
(4) When you click OK, self-checking will start. If the normal result is displayed in about 60 seconds, self-checking has ended.

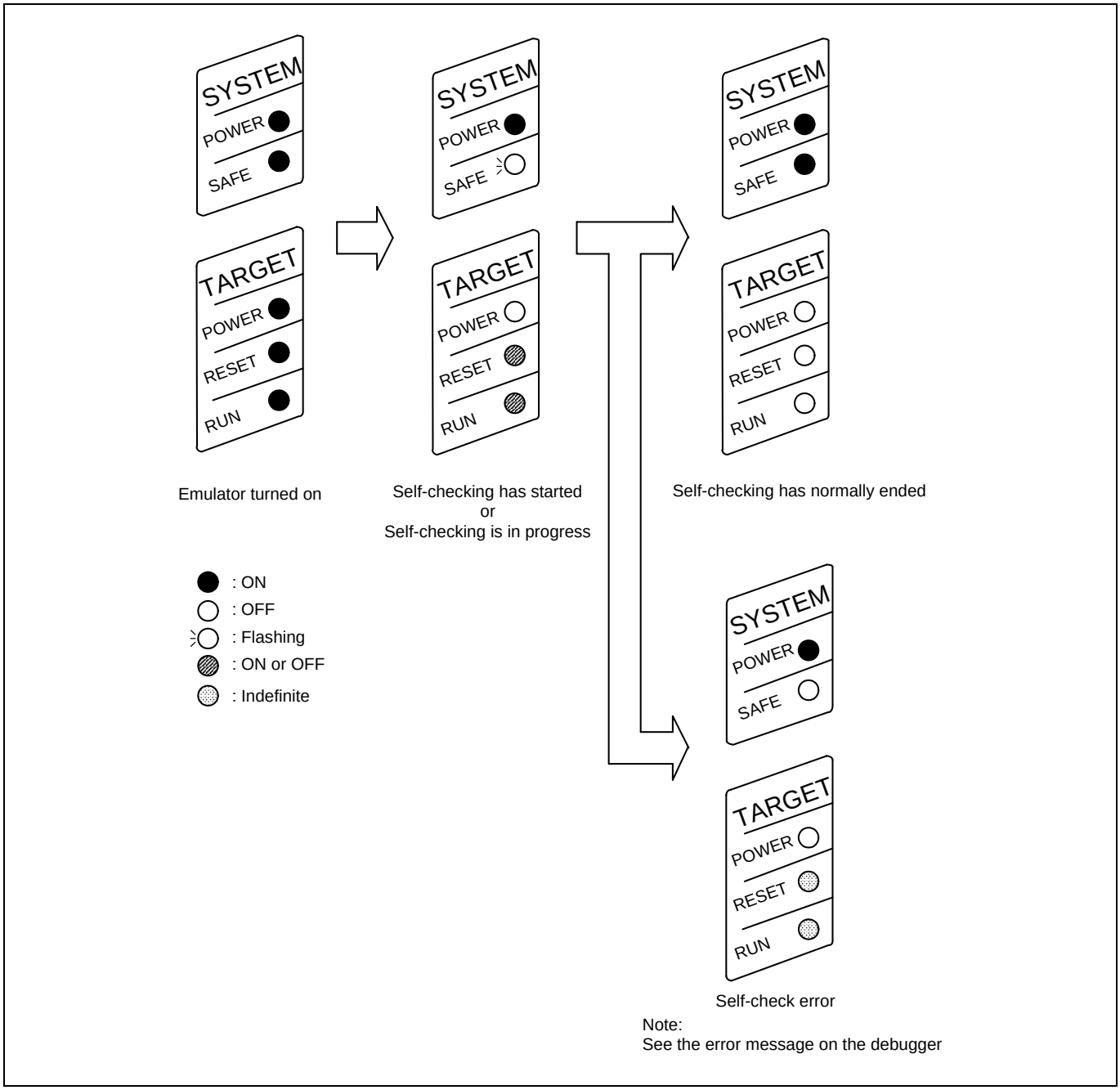


Figure 2.6 LED displays during the self-checking

Downloaded from Arrow.com.

2.8 Selecting the Clock Supply

2.8.1 Clock Source

You can choose the clock source supplied to the evaluation MCU in the Configuration properties dialog box of the emulator debugger. Table 2.1 shows the clock sources and their default settings.

Table 2.1 Clock supply to the MCU

Clock	Clock selection in the emulator debugger	Description	Default setting
Main (XIN-XOUT)	Emulator	IC17 mounting oscillator module	Yes
	User	Oscillator circuit on the user system	-
	Generate	Internal generator circuit (1.0 to 20.0 MHz)	-
Sub (XCIN-XCOUT)	Emulator	Internal oscillator circuit (32.768 kHz)	Yes
	User	Oscillator circuit on the user system	-

Note on Changing the Clock Supply:

- The clock supply can be set by the Configuration properties dialog box when starting up the emulator debugger or by an input of the emulator_clock command on the Command Line window.

Downloaded from Arrow.com.

2.8.2 Using an Internal Oscillator Circuit Board

Kinds of Oscillator Circuit Boards

An oscillator module (20MHz) is mounted on the IC17 at factory setting. If you change the frequency, replace the oscillator module.

(1) Replacing the Oscillator module

Remove the MCU unit from the E100 emulator main unit, and replace the oscillator module of the IC17 (see Figure 2.7).

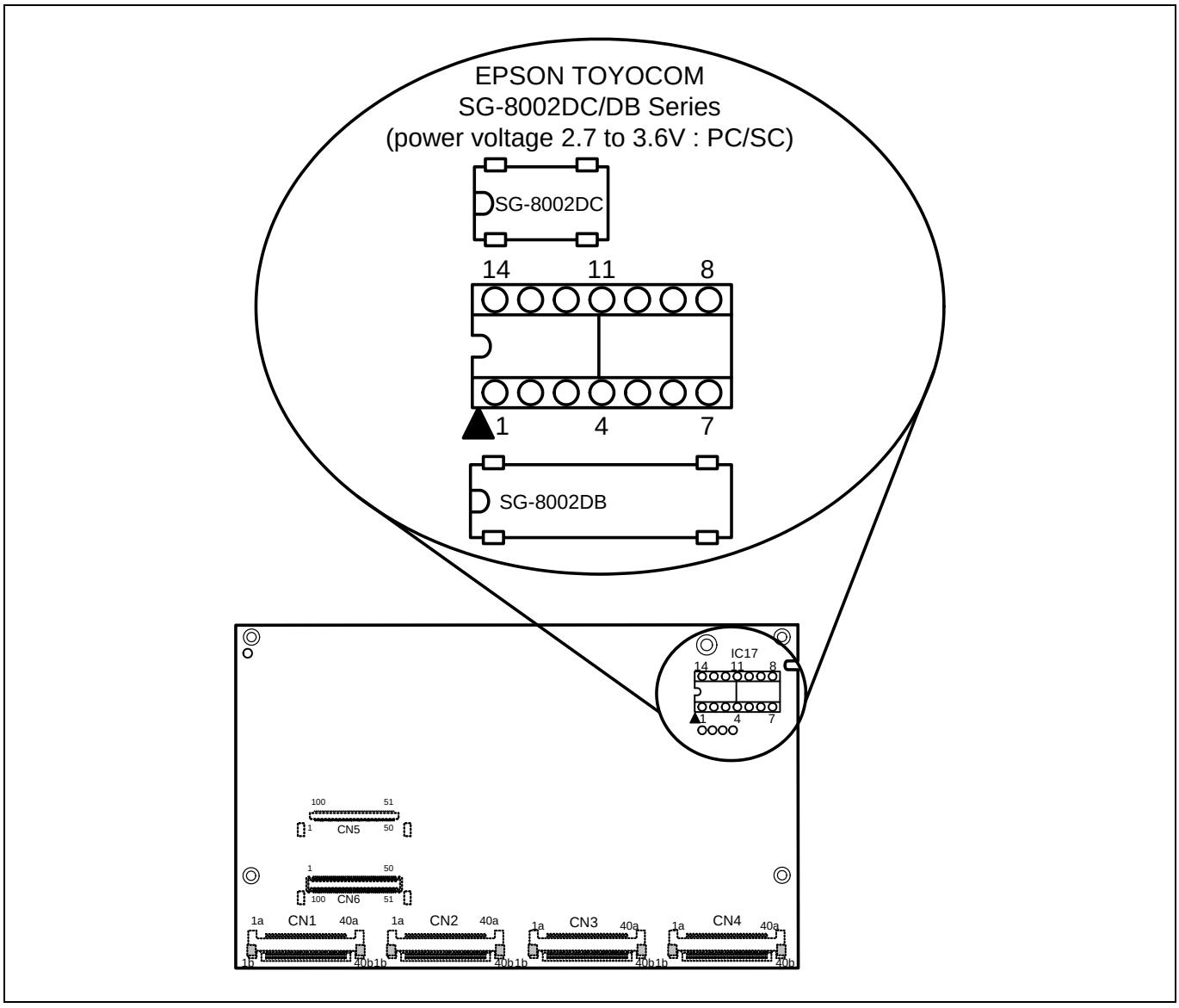


Figure 2.7 Replacing the oscillator module

⚠ CAUTION

Notes on Replacing the Oscillator Module and Oscillator Circuit Board:

- Always shut OFF power when replacing the oscillator module. Otherwise, internal circuits may be damaged.
- When replacing the oscillator module, remove it with a tool such as an IC extractor so as not to damage the board. If the board is damaged, the pattern on the board may be cut and the emulator may not be able to operate.

Downloaded from Arrow.com.

2.8.3 Using the Oscillator Circuit on the User System

To operate this product with an external clock, construct the oscillator circuit as shown in Figure 2.8 in the user system and input the oscillator output at 50% duty (within the operating range of the evaluation MCU) into pin X_{IN}. And pin X_{OUT} should be open. Choose "User" in the emulator debugger to use this clock.

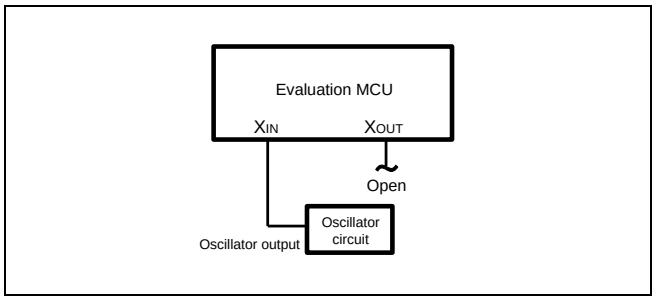


Figure 2.8 External oscillator circuit

Make note that in the oscillator circuit shown in Figure 2.9 where an oscillator is connected between pins X_{IN} and X_{OUT}, oscillation does not occur because a converter board and other devices are used between the evaluation MCU and the user system. It is the same for sub-clock oscillator circuits (X_{CIN} and X_{COUT}).

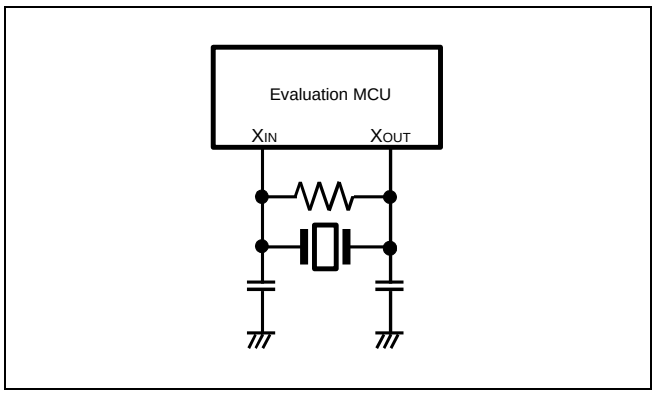


Figure 2.9 Circuit in which oscillation does not occur

2.8.4 Using the Internal Generator Circuit

The dedicated circuit in the E100 can generate any arbitrary frequency specified by the emulator debugger, and it can be supplied as a main clock. It does not depend on either the oscillator circuit board in the MCU unit or the oscillator circuit on the user system. If you want to debug programs without the user system or change a frequency temporarily, you can check its operation before purchasing an oscillator. If you want to use the internal generator circuit in the E100 as a main clock, choose "Generate" in the emulator debugger and specify a frequency you like to use this clock.

Although you can change a frequency between 1.0 and 99.9 MHz by 0.1 MHz for the E100, do not specify a value exceeding the maximum input frequency 20 MHz of the X_{IN} of the MCU.

Notes on Using the Internal Generator Circuit:

- The internal generator circuit is equipped for temporary debugging purposes. Temperature characteristics of frequencies are not guaranteed.
- Be sure to evaluate your system with an oscillator whose frequency is the same as that of the oscillator module or oscillator circuit (emulator) for final evaluation purposes.

Downloaded from Arrow.com.

2.9 Connecting the User System

Figure 2.10 shows how to connect this product to your user system.

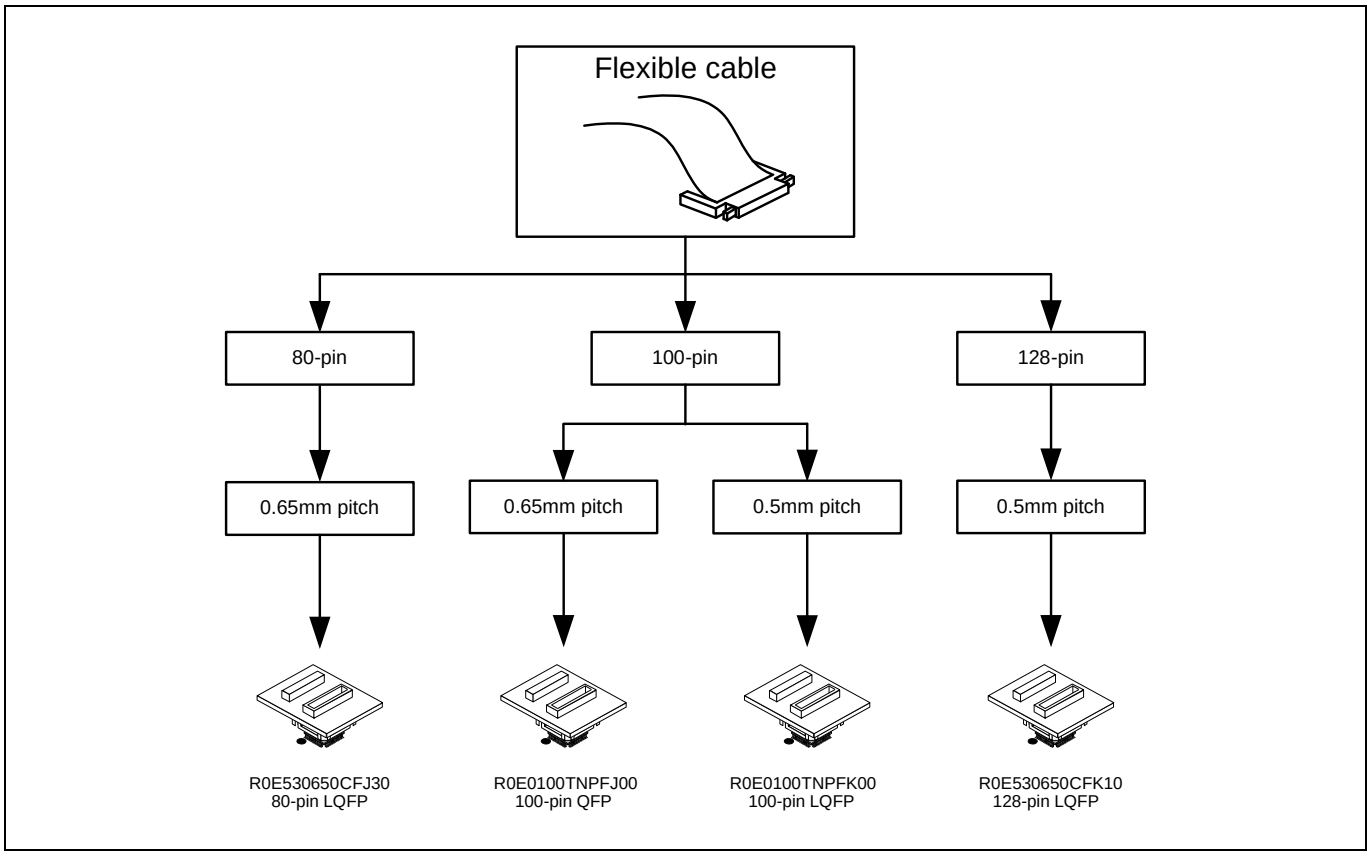


Figure 2.10 Connecting this product to the user system

⚠ CAUTION

Note on Connecting the User System:

- Take care not to attach a converter board in a wrong direction. It may cause a fatal damage to the emulator and user system.

Downloaded from Arrow.com.

2.9.1 Connecting to a 100-pin 0.65mm Pitch Foot Pattern

The following is a procedure of connecting to a 100-pin 0.65mm pitch foot pattern on the user system using the R0E0100TNPFJ00 (not included). For details on the R0E0100TNPFJ00 (not included), refer to its user's manual.

(1) Attach the NQPACK100RB included with the R0E0100TNPFJ00 to the user system.
(2) Attach the YQPACK100RB included with the R0E0100TNPFJ00 to the NQPACK100RB and secure it with the YQ-GUIDEs.
(3) Attach the R0E0100TNPFJ00 to the YQPACK100RB.
(4) Attach the CN2 side of the R0E0100TNPFJ00 to the CN2 side of the flexible cable.
(5) Attach the CN1 side of the R0E0100TNPFJ00 to the CN1 side of the flexible cable.

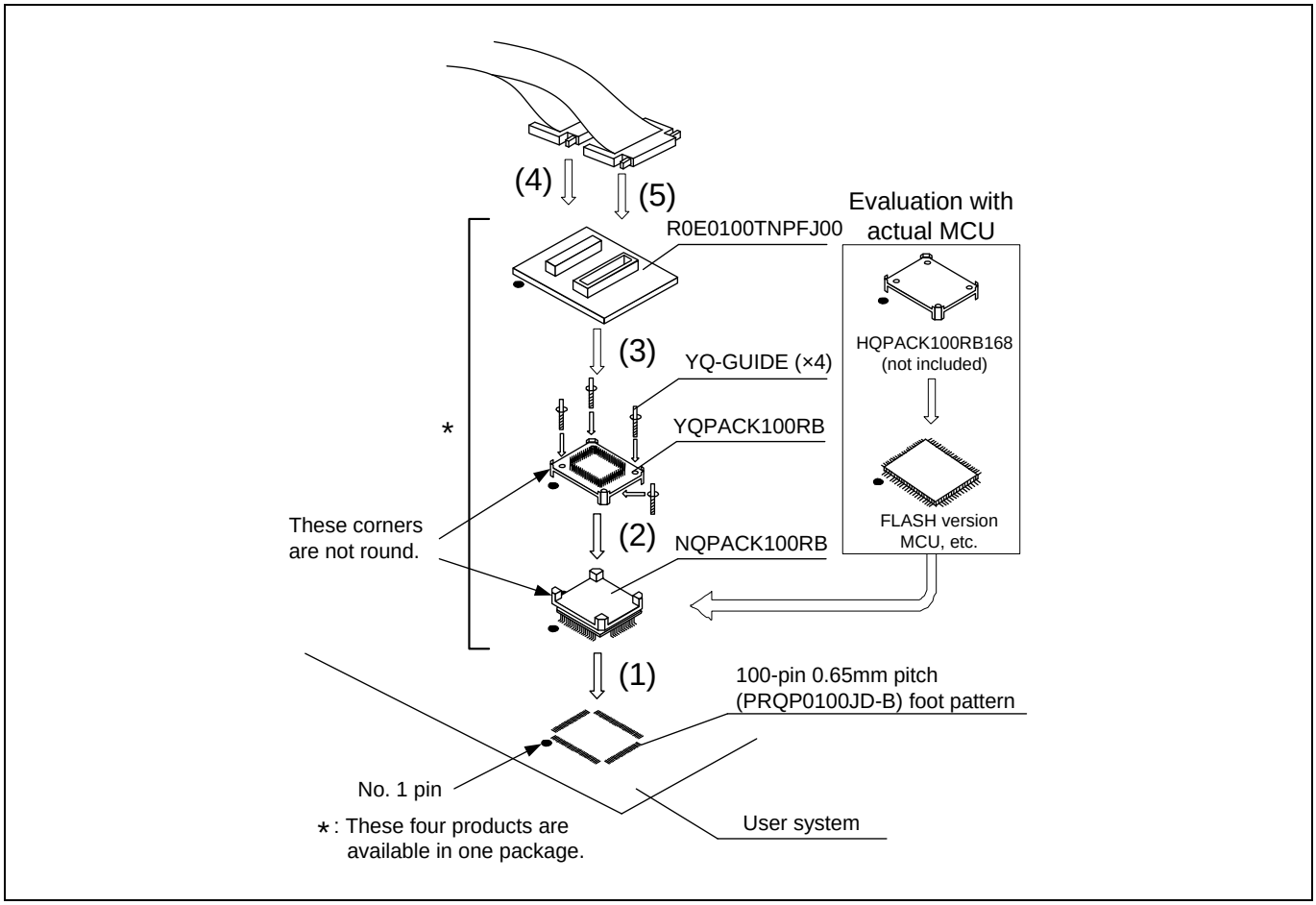


Figure 2.11 Connecting to a 100-pin 0.65mm pitch foot pattern

⚠ CAUTION

Notes on Connecting the User System:

- Take care not to attach a converter board in a wrong direction. It may cause a fatal damage to the emulator and user system.
- The connectors of the R0E0100TNPFJ00 are guaranteed for only 50 insertion/removal iterations.
- For purchasing the HQPACK100RB168, contact the following:
  Tokyo Eletech Corporation http://www.tetc.co.jp/e_index.htm

Downloaded from Arrow.com.

2.9.2 Connecting to a 100-pin 0.5mm Pitch Foot Pattern

The following is a procedure of connecting to a 100-pin 0.5mm pitch foot pattern on the user system using the R0E0100TNPFK00 (not included). For details on the R0E0100TNPFK00 (not included), refer to its user's manual.

(1) Attach the NQPACK100SD-ND included with the R0E0100TNPFK00 to the user system.
(2) Attach the YQPACK100SD included with the R0E0100TNPFK00 to the NQPACK100SD-ND and secure it with the YQ-GUIDEs.
(3) Attach the R0E0100TNPFK00 to the YQPACK100SD.
(4) Attach the CN2 side of the R0E0100TNPFK00 to the CN2 side of the flexible cable.
(5) Attach the CN1 side of the R0E0100TNPFK00 to the CN1 side of the flexible cable.

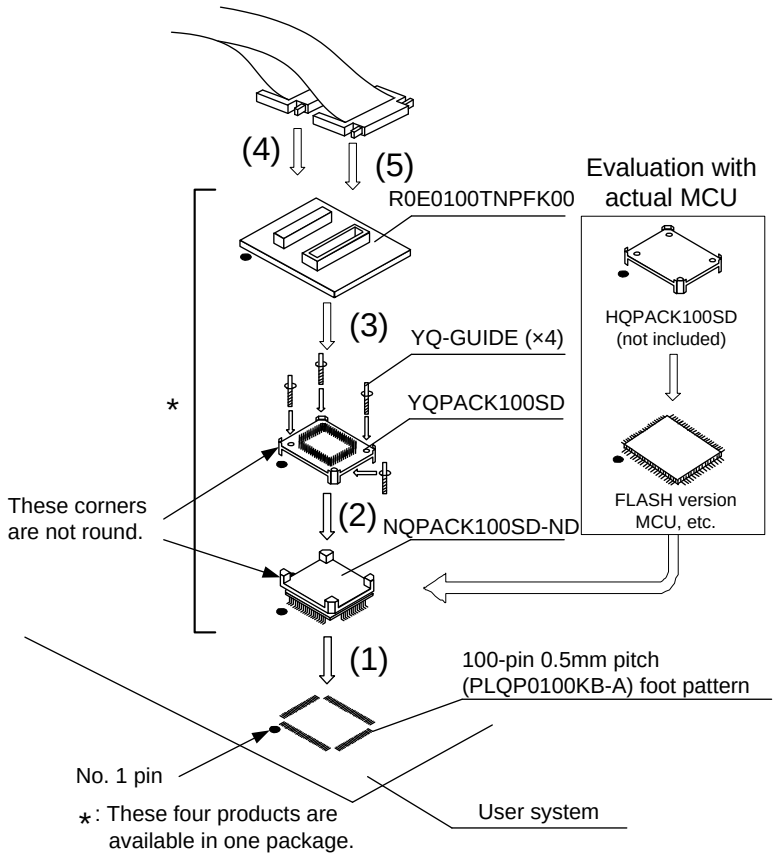


Figure 2.12 Connecting to a 100-pin 0.5mm pitch foot pattern

⚠ CAUTION

Notes on Connecting the User System:

- Take care not to attach a converter board in a wrong direction. It may cause a fatal damage to the emulator and user system.
- The connectors of the R0E0100TNPFK00 are guaranteed for only 50 insertion/removal iterations.
- For purchasing the HQPACK100SD, contact the following:
  Tokyo Eletech Corporation http://www.tetc.co.jp/e_index.htm

Downloaded from Arrow.com.

2.9.3 Connecting to an 80-pin 0.65mm Pitch Foot Pattern

The following is a procedure of connecting to an 80-pin 0.65mm pitch foot pattern on the user system using the R0E530650CFJ30 (not included). For details on the R0E530650CFJ30 (not included), refer to its user's manual.

(1) Attach the NQPACK080SB included with the R0E530650CFJ30 to the user system.
(2) Attach the YQPACK080SB included with the R0E530650CFJ30 to the NQPACK080SB and secure it with the YQ-GUIDEs.
(3) Attach the R0E530650CFJ30 to the YQPACK080SB.
(4) Attach the CN2 side of the R0E530650CFJ30 to the CN2 side of the flexible cable.
(5) Attach the CN1 side of the R0E530650CFJ30 to the CN1 side of the flexible cable.

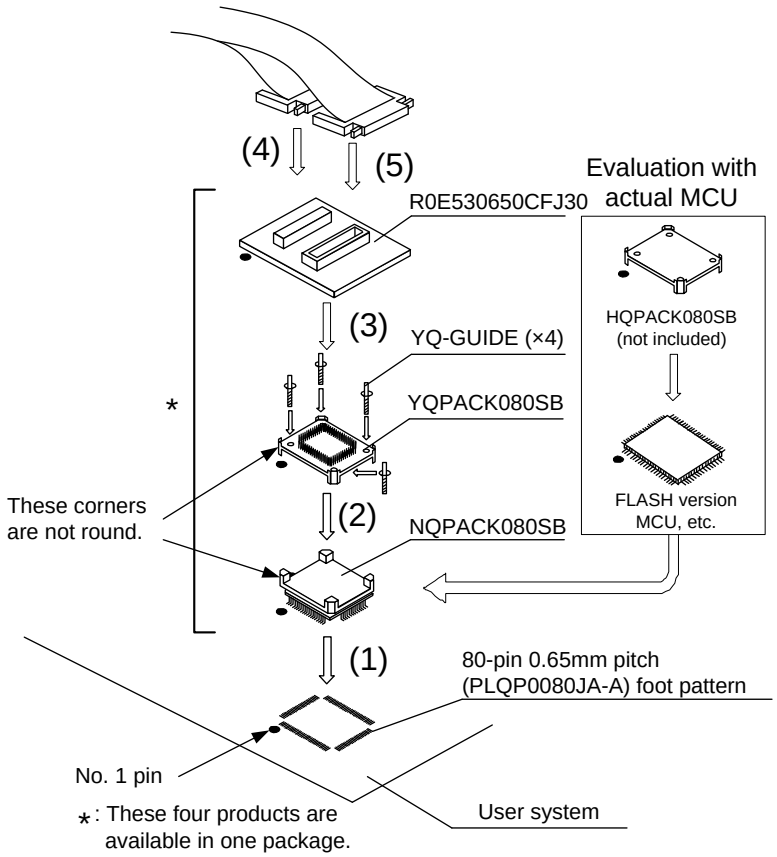


Figure 2.13 Connecting to an 80-pin 0.65mm pitch foot pattern

⚠ CAUTION

Notes on Connecting the User System:

- Take care not to attach a converter board in a wrong direction. It may cause a fatal damage to the emulator and user system.
- The connectors of the R0E530650CFJ30 are guaranteed for only 50 insertion/removal iterations.
- For purchasing the HQPACK080SB, contact the following:
  Tokyo Eletech Corporation http://www.tetc.co.jp/e_index.htm

Downloaded from Arrow.com.

2.9.4 Connecting to a 128-pin 0.5mm Pitch Foot Pattern

The following is a procedure of connecting to a 128-pin 0.5mm pitch foot pattern on the user system using the R0E530650CFK10 (not included). For details on the R0E530650CFK10 (not included), refer to its user's manual.

(1) Attach the NQPACK128RD included with the R0E530650CFK10 to the user system.
(2) Attach the YQPACK128RD included with the R0E530650CFK10 to the NQPACK128RD and secure it with the YQ-GUIDEs.
(3) Attach the R0E530650CFK10 to the YQPACK128RD.
(4) Attach the CN2 side of the R0E530650CFK10 to the CN2 side of the flexible cable.
(5) Attach the CN1 side of the R0E530650CFK10 to the CN1 side of the flexible cable.

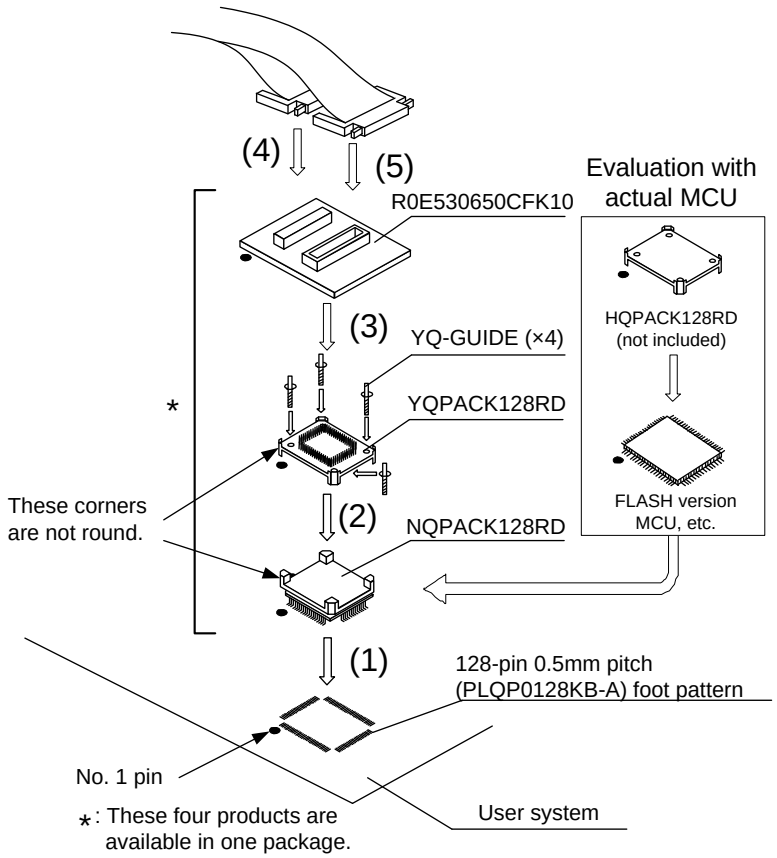


Figure 2.14 Connecting to a 128-pin 0.5mm pitch foot pattern

⚠ CAUTION

Notes on Connecting the User System:

- Take care not to attach a converter board in a wrong direction. It may cause a fatal damage to the emulator and user system.
- The connectors of the R0E530650CFK10 are guaranteed for only 50 insertion/removal iterations.
- For purchasing the HQPACK128RD, contact the following:
  Tokyo Eletech Corporation http://www.tetc.co.jp/e_index.htm

Downloaded from Arrow.com.

3. Tutorial

3.1 Introduction

The E100 emulator has a tutorial program available. This program is provided as a means for presenting to you the main features of the emulator, as will be explained in this document.

This tutorial program is written in C language, and is created to sort 10 pieces of random data in ascending/descending orders. The following outlines the processing performed by the tutorial program.

The main function calls the tutorial function repeatedly in order to execute a sort process repeatedly.
The tutorial function generates the random data to be sorted and calls the sort and the change functions in that order.
The sort function inputs an array that contains the random data generated by the tutorial function and sorts the input data in ascending order.
The change function inputs an array that was sorted in ascending order by the sort function and sorts the input data in descending order.

The tutorial program is a program designed to help users to understand how to use the functions of the emulator and the emulator debugger. When developing user systems and user programs, refer to the user's manuals of the target MCUs.

CAUTION
If the tutorial program is recompiled, the addresses in a recompiled program may not be the same as those described in this chapter.

Downloaded from Arrow.com.

3.2 Starting the High-performance Embedded Workshop

Open a workspace following the procedure described in Section 4.4, "Opening an Existing Workspace"

For the directory, specify the one that is given below.
OS installed drive\Workspace\Tutorial\E100\M16C

For the file, specify the one that is shown below.

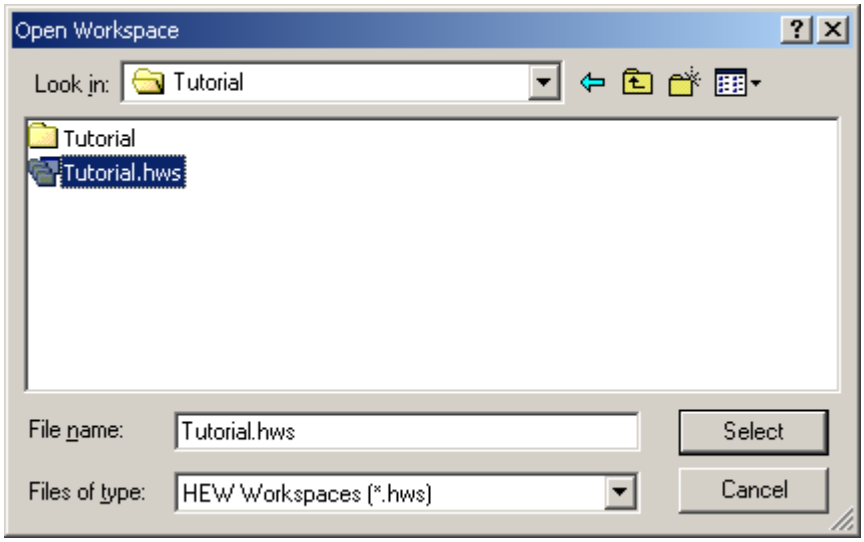


Figure 3.1 Open Workspace dialog box

3.3 Connecting the Emulator

When the debugger is connected to the emulator, a dialog box for setting up the debugger is displayed. In this dialog box, make initial settings of the debugger.
When you have finished setting up the debugger, you are ready to debug.

Downloaded from Arrow.com.

3.4 Downloading the Tutorial Program

3.4.1 Downloading the Tutorial Program

Download the object program you want to debug. Note, however, that the program to be downloaded and the address in the microcomputer to which downloaded differ with each microcomputer used. Read the strings, etc. on the screen as suitable for the microcomputer you are using.
Choose Download from Tutorial.x30 of Download modules.

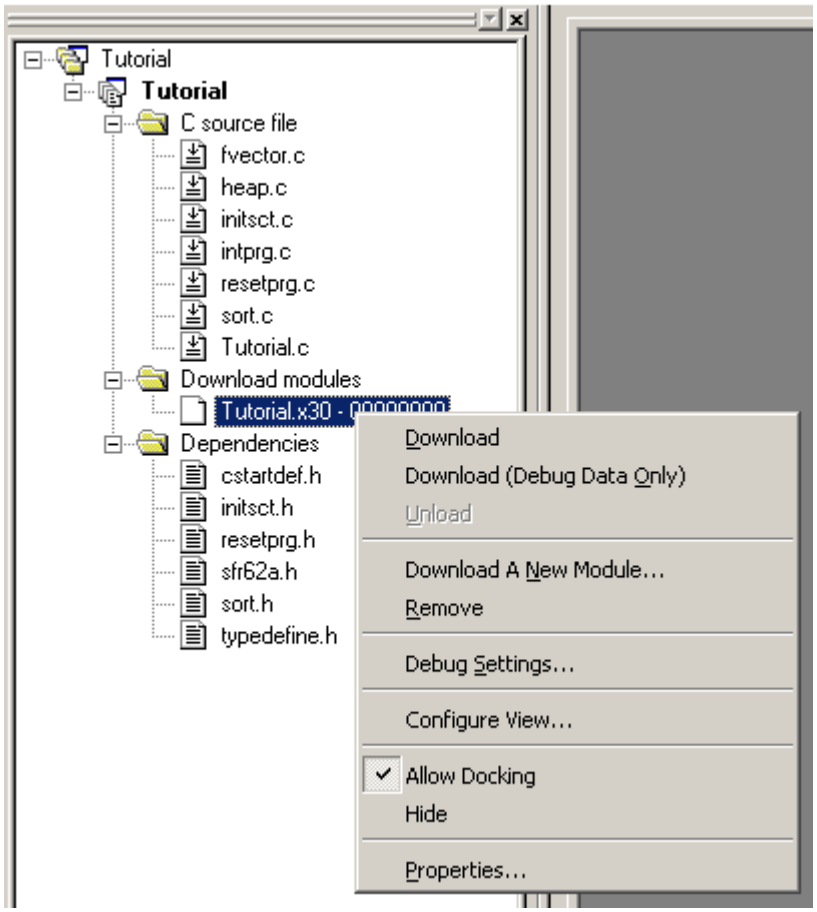


Figure 3.2 Download display of the tutorial program

Downloaded from Arrow.com.

3.4.2 Displaying the Source Program

In the High-performance Embedded Workshop you can debug a program at the source level.
Double-click Tutorial.c of C source file.

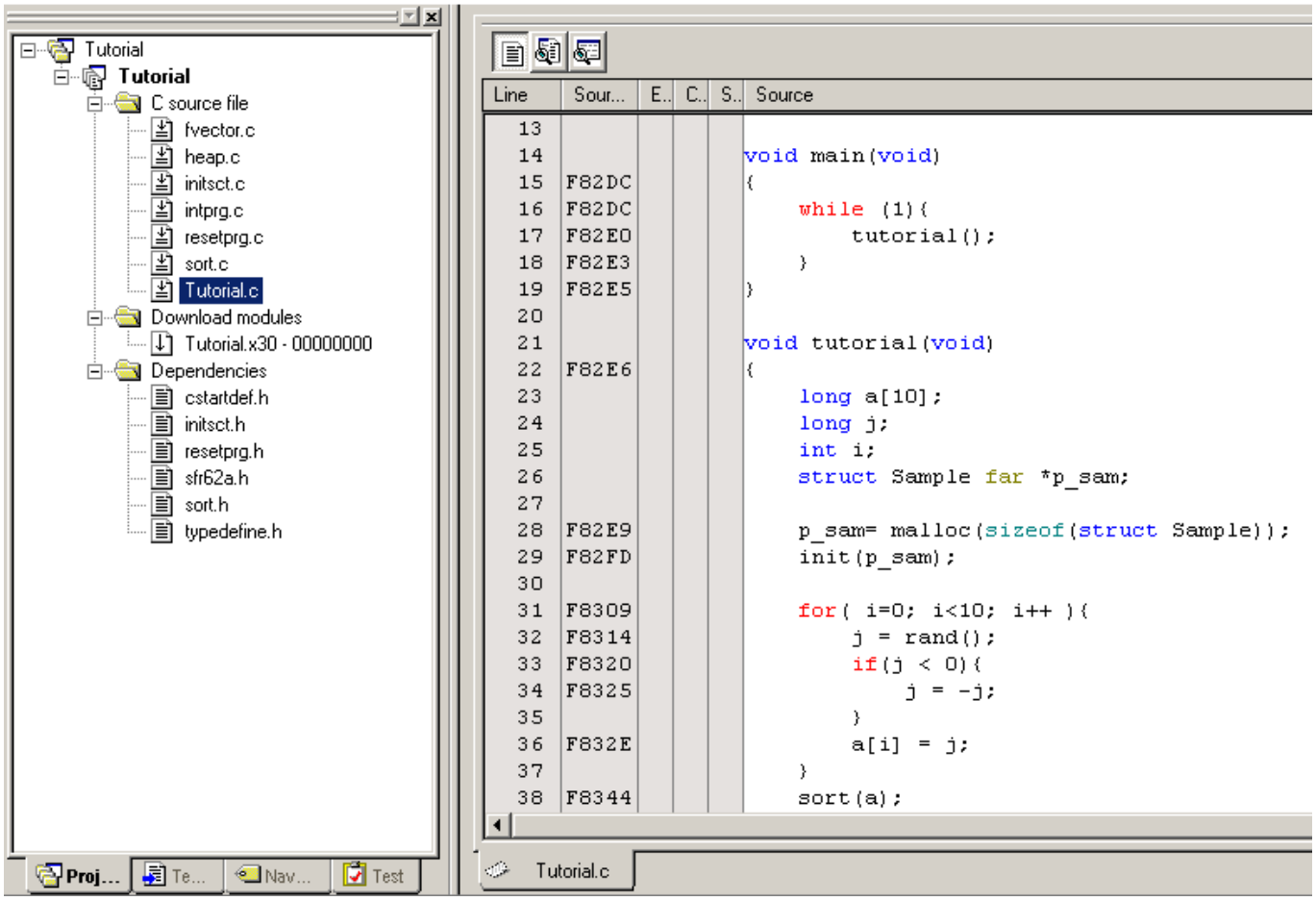


Figure 3.3 Editor window (displaying the source program)

If necessary, you can change the font and font size to make text more easily readable. For details on how to change, refer to the High-performance Embedded Workshop User's Manual.
The Editor window initially shows the beginning of a program. Using the scroll bar, you can look at another part of a program.

Downloaded from Arrow.com.

3.5 Setting Software Breakpoints

Software breakpoints are one of simple debug facilities.

The Editor window permits you to set software breakpoints easily. For example, you can set a software breakpoint at a place where the sort function is called.

Double-click a row in the S/W Breakpoints column corresponding to the source line that includes a sort function call.

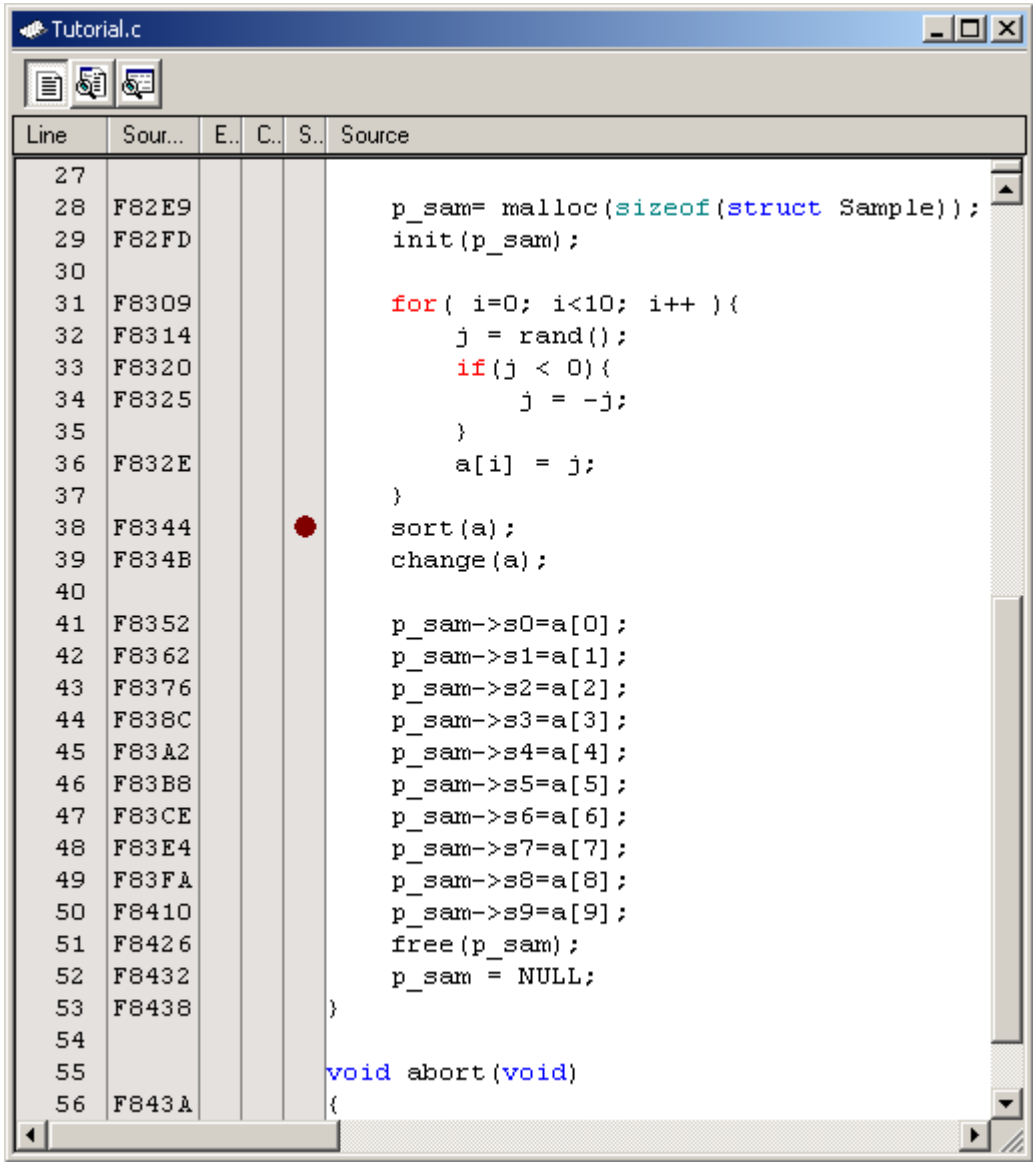


Figure 3.4 Editor window (setting a software breakpoint)

The source line that includes the sort function will be marked with a red circle, indicating that a software breakpoint has been set there.

Downloaded from Arrow.com.

3.6 Executing the Program

The following describes how to run the program.

3.6.1 Resetting the CPU

To reset the CPU, choose Reset CPU from the Debug menu or click the Reset CPU button  in the toolbar.

3.6.2 Executing the Program

To execute the program, choose Go from the Debug menu or click the Go button  in the toolbar.

The program will be executed continuously until a breakpoint is reached. An arrow will be displayed in the S/W Breakpoints column to indicate the position at which the program has stopped.

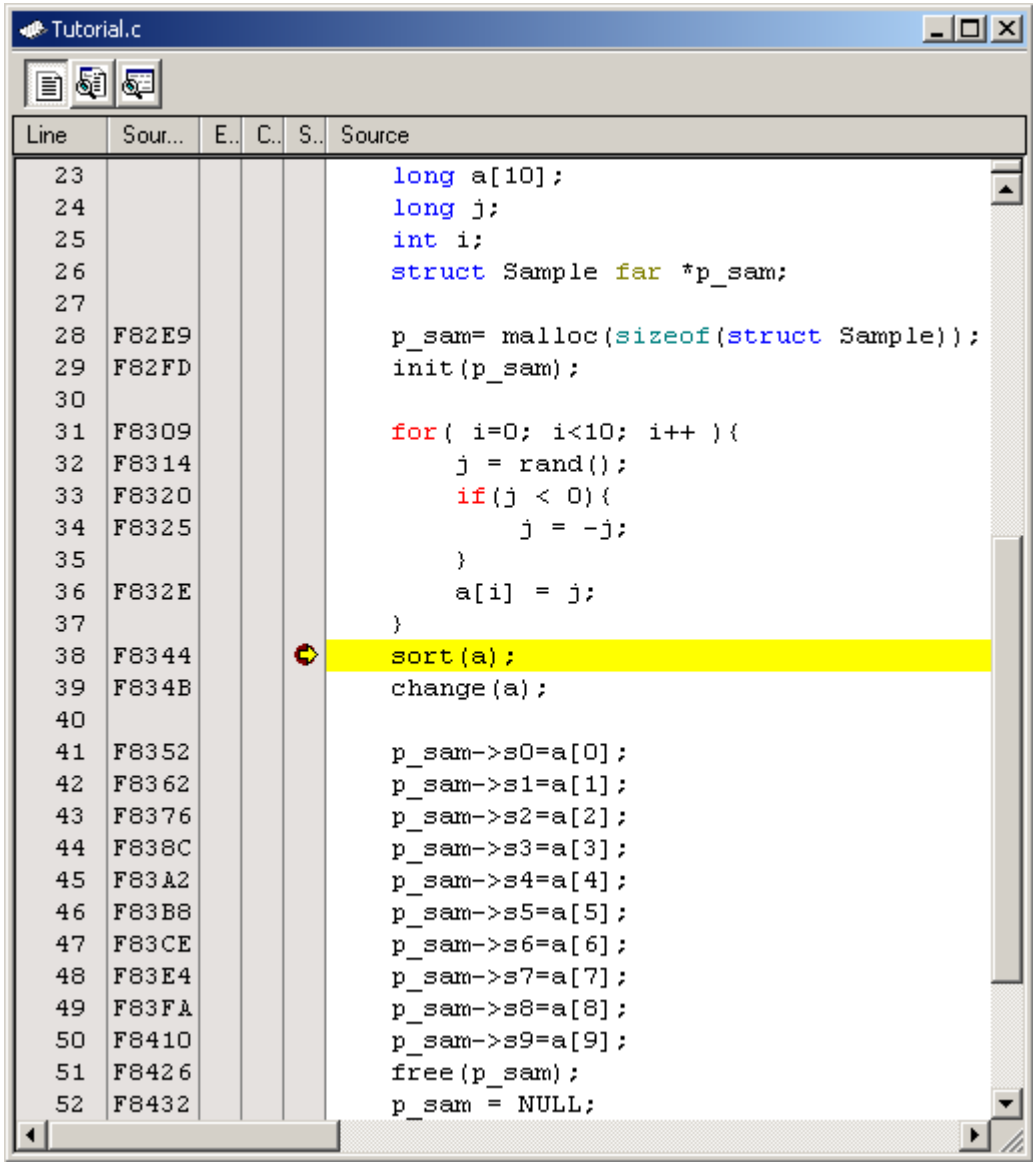


Figure 3.5 Editor window (program at a break)

Downloaded from Arrow.com.

The Status window permits you to check the cause of the break that last occurred.
Choose CPU –> Status from the View menu or click the View Status toolbar button . When the Status window is displayed, open the Target sheet in it and check.

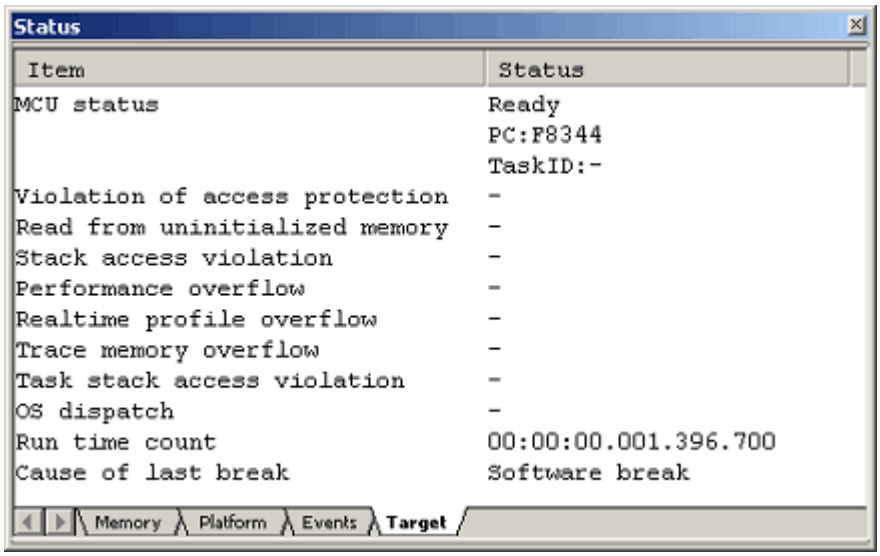


Figure 3.6 Status window

CAUTION
The contents displayed in this window differ with each product. For details about the displayed contents of each product, refer to Chapter 3, "Debugging," or online help.

3.7 Checking Breakpoints

Use the Breakpoints dialog box to check all software breakpoints set.

3.7.1 Checking Breakpoints

Press the keys Ctrl + B on the keyboard of your PC. The Breakpoints dialog box shown below will be displayed.

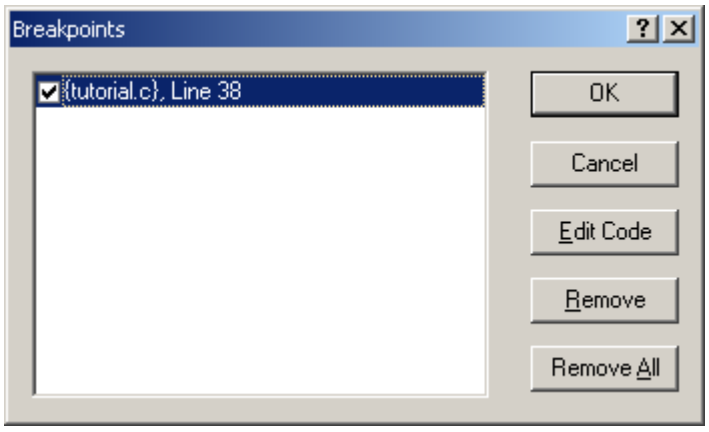


Figure 3.7 Breakpoints dialog box

Use this dialog box to remove a breakpoint or enable or disable a breakpoint.

Downloaded from Arrow.com.

3.8 Altering Register Contents

Choose CPU –> Registers from the View menu or click the Registers toolbar button . The Register window shown below will be displayed.

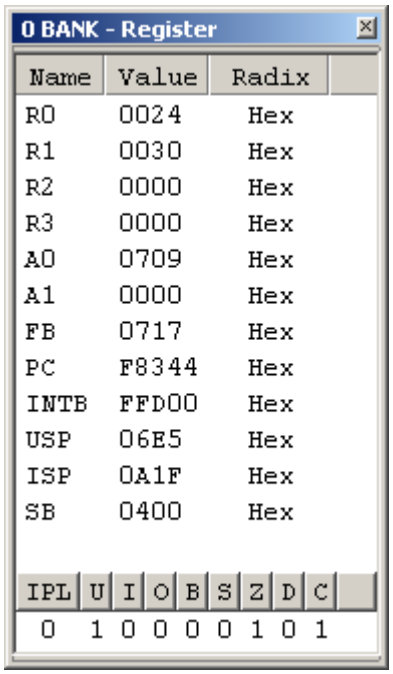


Figure 3.8 Register window

The content of any register can be altered.

Double-click the line for the register you want to alter. The dialog box shown below will be displayed, so enter a new value with which you want to alter the register.

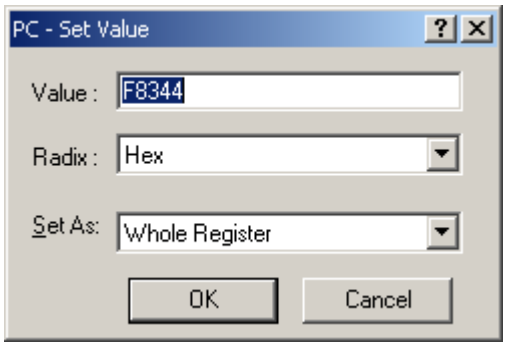


Figure 3.9 Set Value dialog box (PC)

Downloaded from Arrow.com.

3.9 Referencing Symbols

The Label window permits you to display the symbol information included in a module.

Choose Symbols –> Label from the View menu or click the Label toolbar button . The Label window shown below will be displayed. Use this window to look at the symbol information included in a module.

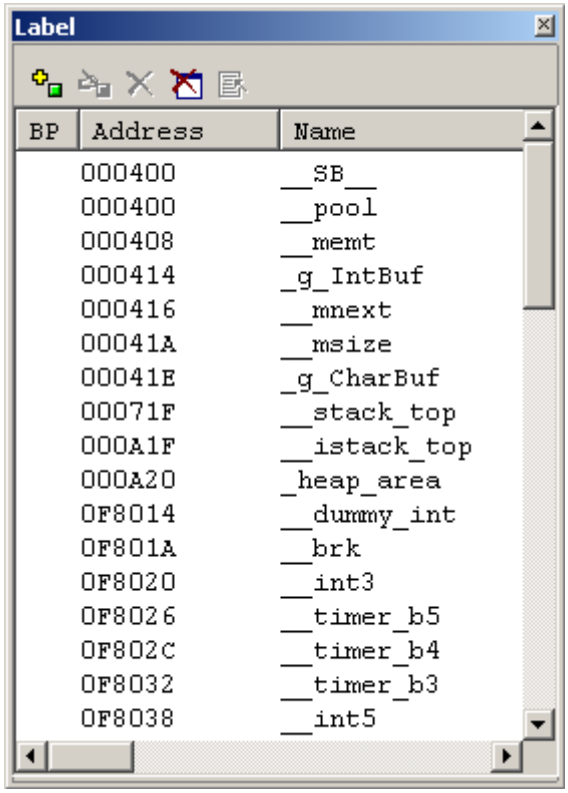


Figure 3.10 Label window

Downloaded from Arrow.com.

3.10 Checking Memory Contents

Specifying a label name, you can check in the Memory window the content of memory where the label is registered. For example, you can check the content of memory corresponding to _main in byte unit, as shown below.

Choose CPU –> Memory from the View menu or click the Memory toolbar button  to display the Display Address dialog box.

Enter "_main" in the edit box of the Display Address dialog box.

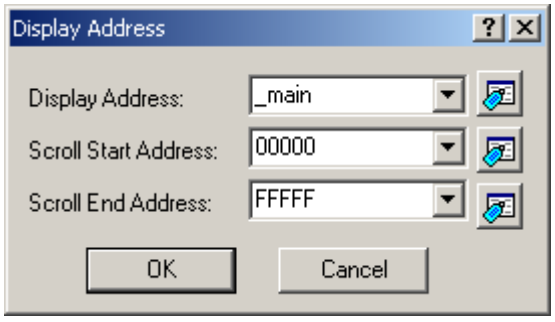
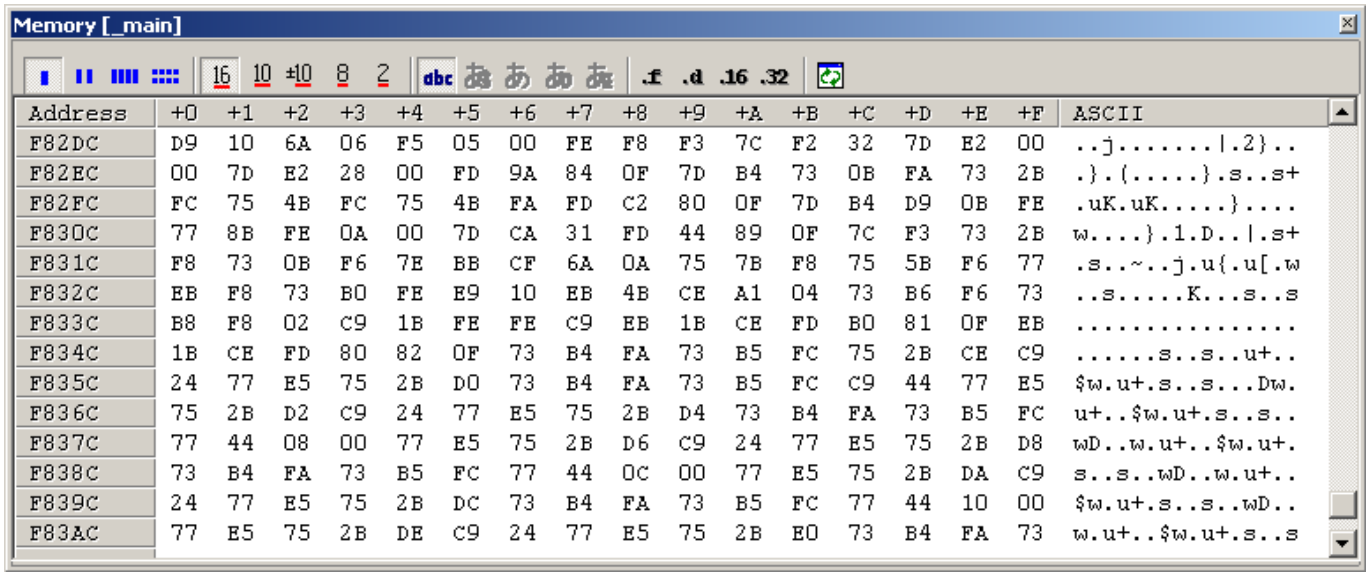


Figure 3.11 Display Address dialog box

Click the OK button. The Memory window will be displayed, showing a specified memory area.



Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	ASCII
F82DC	D9	10	6A	06	F5	05	00	FE	F8	F3	7C	F2	32	7D	E2	00	..j.....|.2}..
F82EC	00	7D	E2	28	00	FD	9A	84	0F	7D	B4	73	0B	FA	73	2B	.}.(.....}.s..s+
F82FC	FC	75	4B	FC	75	4B	FA	FD	C2	80	0F	7D	B4	D9	0B	FE	.uK.uK.....}....
F830C	77	8B	FE	0A	00	7D	CA	31	FD	44	89	0F	7C	F3	73	2B	w....}.1.D..|.s+
F831C	F8	73	0B	F6	7E	BB	CF	6A	0A	75	7B	F8	75	5B	F6	77	.s..~..j.u{.u[.w
F832C	EB	F8	73	B0	FE	E9	10	EB	4B	CE	A1	04	73	B6	F6	73	..s.....K...s..s
F833C	B8	F8	02	C9	1B	FE	FE	C9	EB	1B	CE	FD	B0	81	0F	EB	.....
F834C	1B	CE	FD	80	82	0F	73	B4	FA	73	B5	FC	75	2B	CE	C9	......s..s..u+..
F835C	24	77	E5	75	2B	D0	73	B4	FA	73	B5	FC	C9	44	77	E5	$w.u+.s..s...Dw.
F836C	75	2B	D2	C9	24	77	E5	75	2B	D4	73	B4	FA	73	B5	FC	u+..$w.u+.s..s..
F837C	77	44	08	00	77	E5	75	2B	D6	C9	24	77	E5	75	2B	D8	wD..w.u+..$w.u+.
F838C	73	B4	FA	73	B5	FC	77	44	0C	00	77	E5	75	2B	DA	C9	s..s..wD..w.u+..
F839C	24	77	E5	75	2B	DC	73	B4	FA	73	B5	FC	77	44	10	00	$w.u+.s..s..wD..
F83AC	77	E5	75	2B	DE	C9	24	77	E5	75	2B	E0	73	B4	FA	73	w.u+..$w.u+.s..s

Figure 3.12 Memory window

Downloaded from Arrow.com.

3.11 Referencing Variables

When single-stepping a program, you can see how the values of the variables used in the program will change as you step through source lines or instructions. For example, following the procedure described below, you can look at the long-type array 'a' that is declared at the beginning of a program.

Click the left-hand side of the array 'a' displayed in the Editor window and place the cursor there. Select Instant Watch with the right mouse button. The dialog box shown below will be displayed.

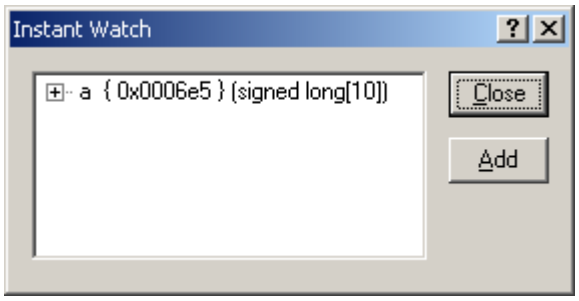


Figure 3.13 Instant Watch dialog box

Click the Add button to add a variable to the Watch window.

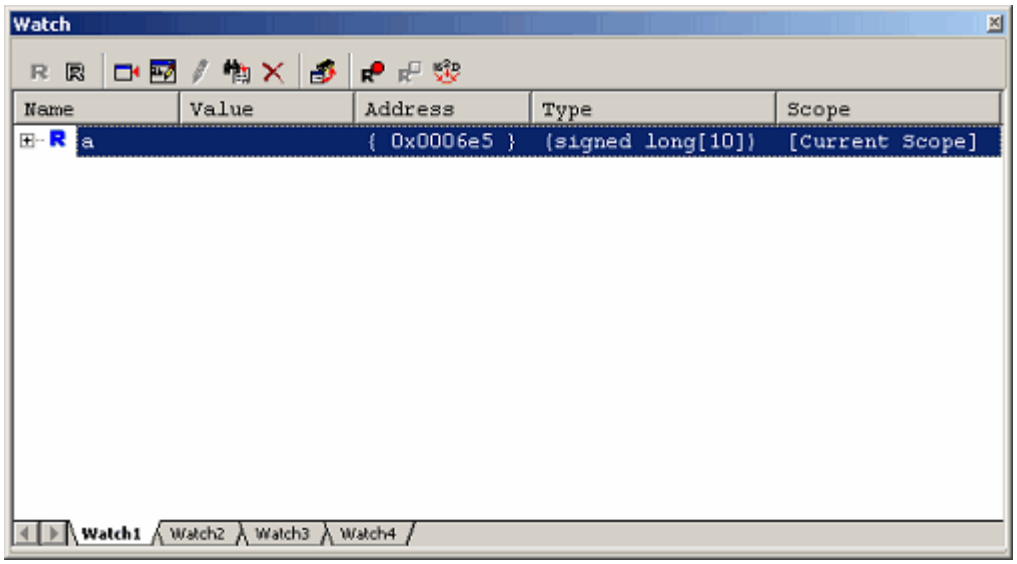


Figure 3.14 Watch window (array display)

Or you can specify a variable name to add a variable to the Watch window. Click the right mouse button in the Watch window and choose Add Watch from the pop-up menu. The dialog box shown below will be displayed.

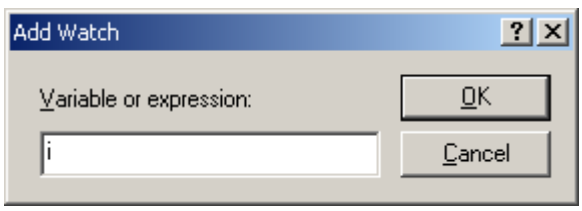


Figure 3.15 Add Watch dialog box

Downloaded from Arrow.com.

Enter a variable 'i' in the Variable or Expression edit box and click the OK button.
An int-type variable 'i' will be displayed in the Watch window.

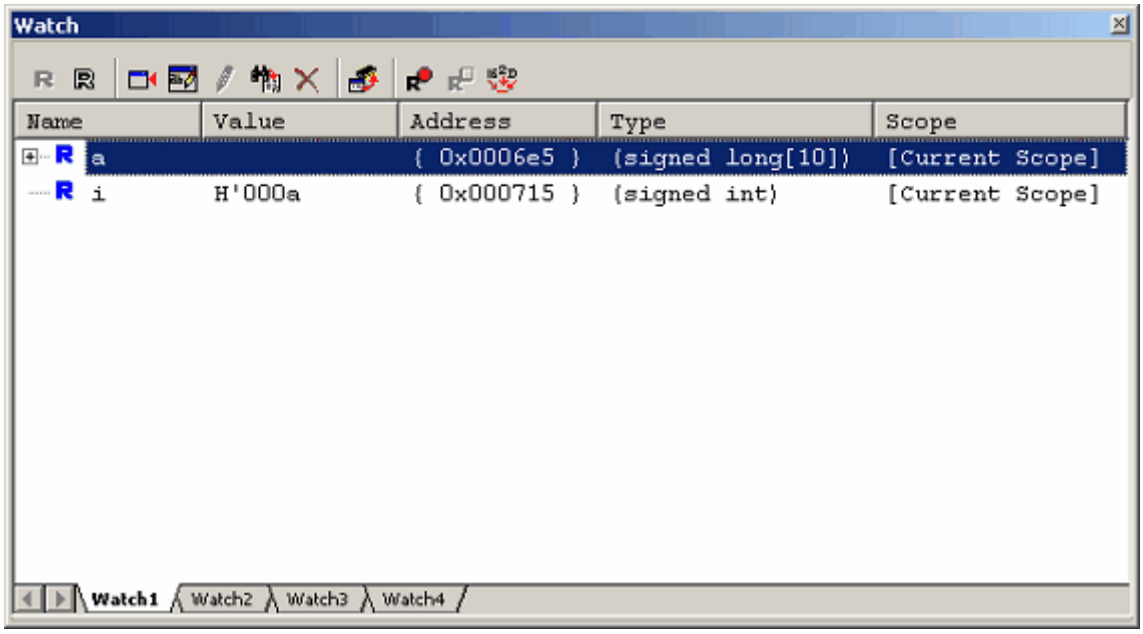


Figure 3.16 Watch window (showing a variable)

Clicking the "+" mark shown to the left of the array 'a' in the Watch window, you can look at each element of the array 'a.'

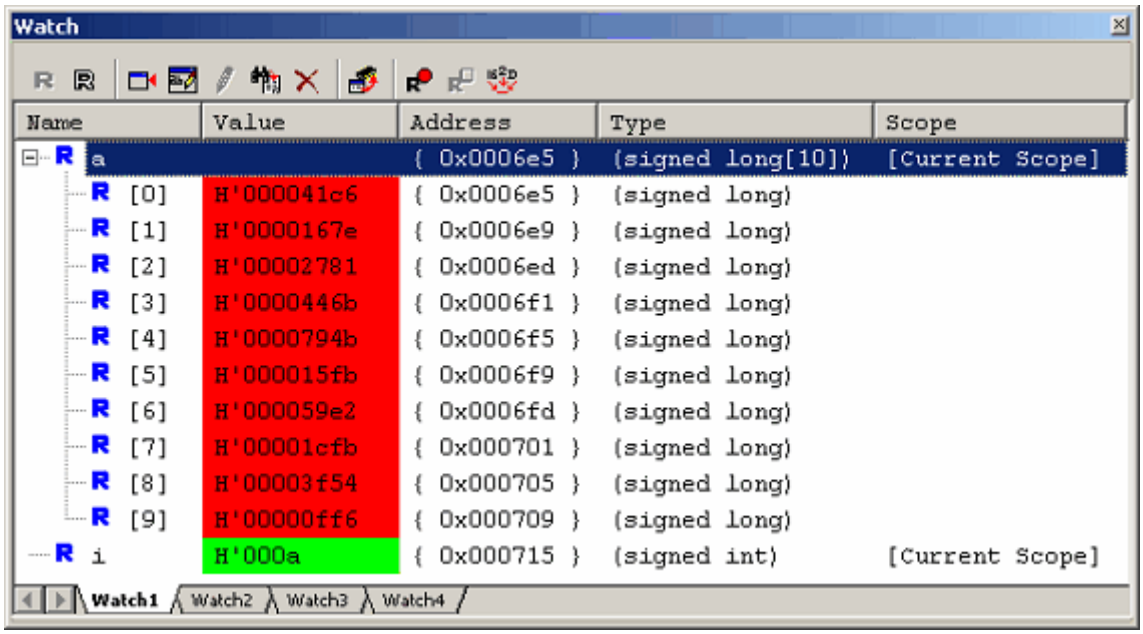


Figure 3.17 Watch window (showing array elements)

Downloaded from Arrow.com.

3.12 Showing Local Variables

Using the Locals window you can display the local variables included in a function. As an example, let's check the local variables of the tutorial function. This function declares four local variables 'a,' 'j,' 'i' and 'p_sam.'

Choose Symbols –> Locals from the View menu or click the Locals toolbar button  to display the Locals window.

The Locals window shows the local variables and the values of the function indicated by the current program counter (PC).

If no variables exist in the function, no information is displayed in the Locals window.

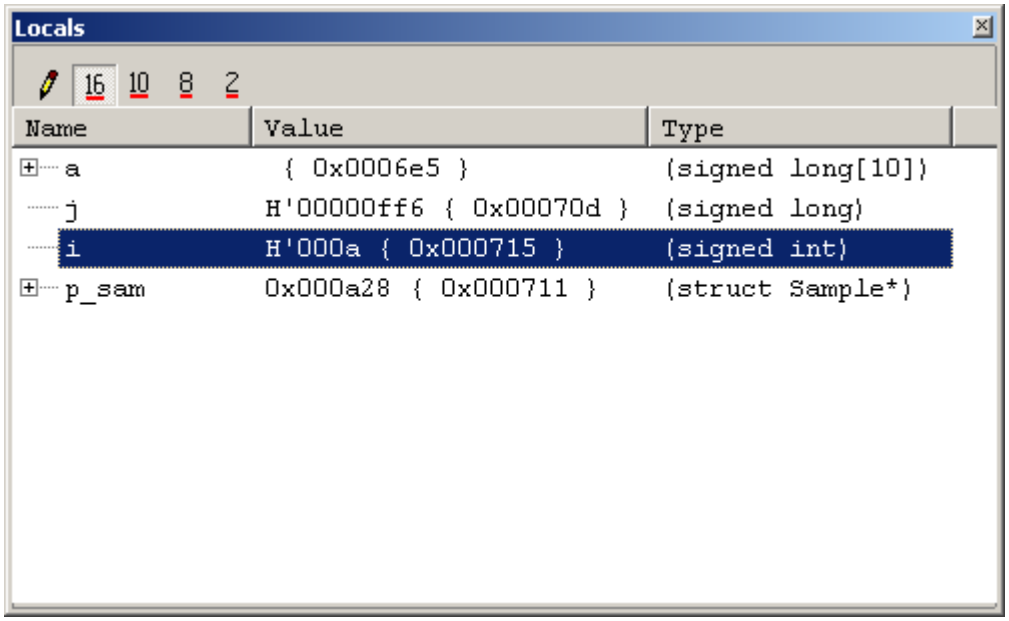


Figure 3.18 Locals window

Click the "+" mark shown to the left of the array 'a' in the Locals window to display the elements comprising the array 'a'.

Look at the elements of the array 'a' before and after the sort function is executed to confirm that random data is sorted in descending order.

3.13 Single-Stepping a Program

The High-performance Embedded Workshop provides various step commands that will prove useful in debugging a program.

Table 3.1 Step Options

Item No.	Command	Description
1	Step In	Executes a program one statement at a time (including statements in a function).
2	Step Over	Executes a program one statement at a time by 'stepping over' a function call if any.
3	Step Out	After exiting a function, stops at the next statement of a program that called the function.
4	Step...	Single-step a program a specified number of times at a specified speed.

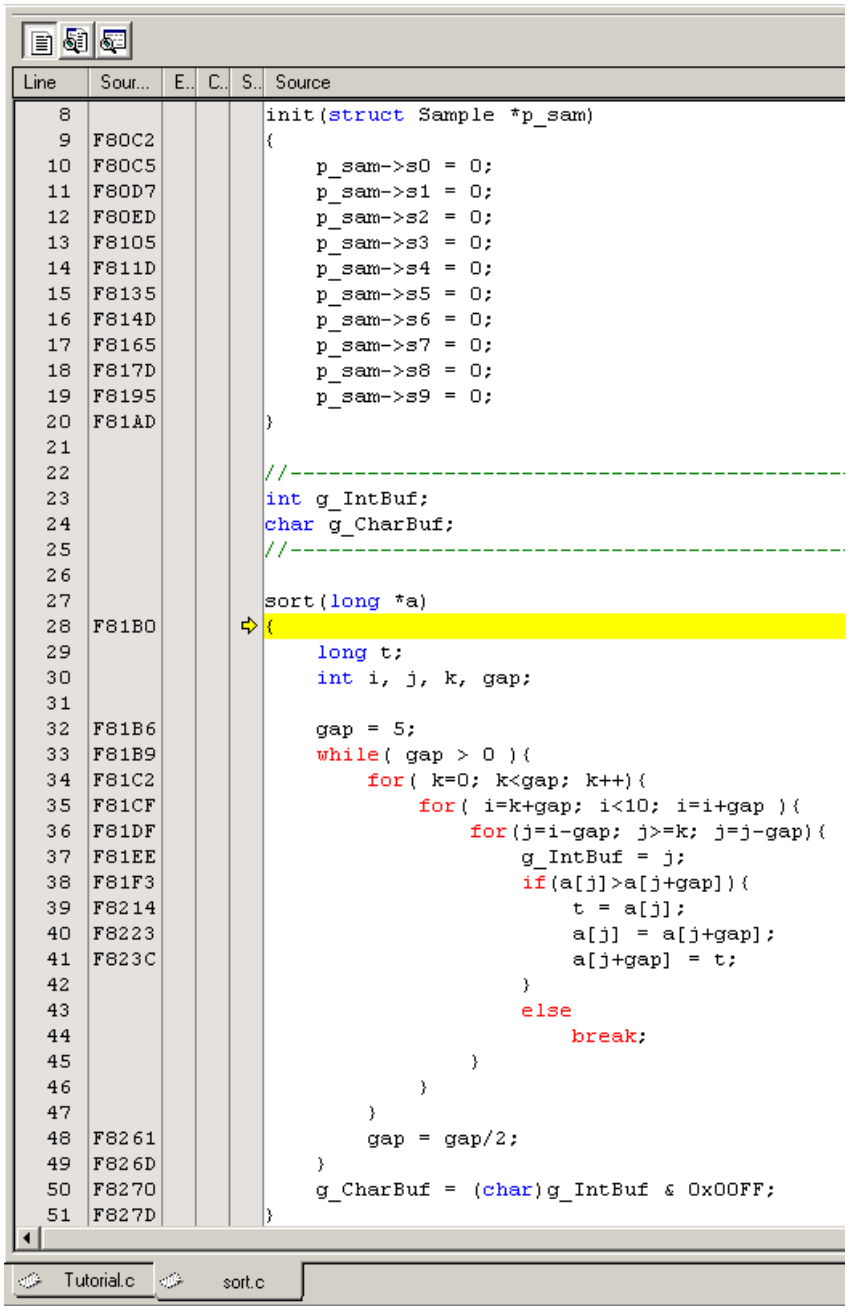
Downloaded from Arrow.com.

3.13.1 Executing Step In Command

The Step In command 'steps in' a called function and stops at the first statement of the called function.
To enter the sort function, choose Step In from the Debug menu or click the Step In button in the toolbar.



Figure 3.19 Step In button



```
Line  Sour...  E..  C..  S..  Source
   8                         init(struct Sample *p_sam)
   9  F80C2                  {
  10  F80C5                      p_sam->s0 = 0;
  11  F80D7                      p_sam->s1 = 0;
  12  F80ED                      p_sam->s2 = 0;
  13  F8105                      p_sam->s3 = 0;
  14  F811D                      p_sam->s4 = 0;
  15  F8135                      p_sam->s5 = 0;
  16  F814D                      p_sam->s6 = 0;
  17  F8165                      p_sam->s7 = 0;
  18  F817D                      p_sam->s8 = 0;
  19  F8195                      p_sam->s9 = 0;
  20  F81AD                  }
  21
  22                         //-----------------------------------------
  23                         int g_IntBuf;
  24                         char g_CharBuf;
  25                         //-----------------------------------------
  26
  27                         sort(long *a)
  28  F81B0                  {
  29                             long t;
  30                             int i, j, k, gap;
  31
  32  F81B6                      gap = 5;
  33  F81B9                      while( gap > 0 ){
  34  F81C2                          for( k=0; k<gap; k++){
  35  F81CF                              for( i=k+gap; i<10; i=i+gap ){
  36  F81DF                                  for(j=i-gap; j>=k; j=j-gap){
  37  F81EE                                      g_IntBuf = j;
  38  F81F3                                      if(a[j]>a[j+gap]){
  39  F8214                                          t = a[j];
  40  F8223                                          a[j] = a[j+gap];
  41  F823C                                          a[j+gap] = t;
  42                                             }
  43                                             else
  44                                                 break;
  45                                     }
  46                                 }
  47                             }
  48  F8261                      gap = gap/2;
  49  F826D                  }
  50  F8270                  g_CharBuf = (char)g_IntBuf & 0x00FF;
  51  F827D              }
```

Tutorial.c sort.c

Figure 3.20 Editor window (Step In)

The highlighting in the Editor window moves to the first statement of the sort function.

Downloaded from Arrow.com.

3.13.2 Executing the Step Out Command

The Step Out command exits a called function by executing it quickly and stops at the next statement of a program from which the function was called.

To exit the sort function, choose Step Out from the Debug menu or click the Step Out button in the toolbar.



Figure 3.21 Step Out button

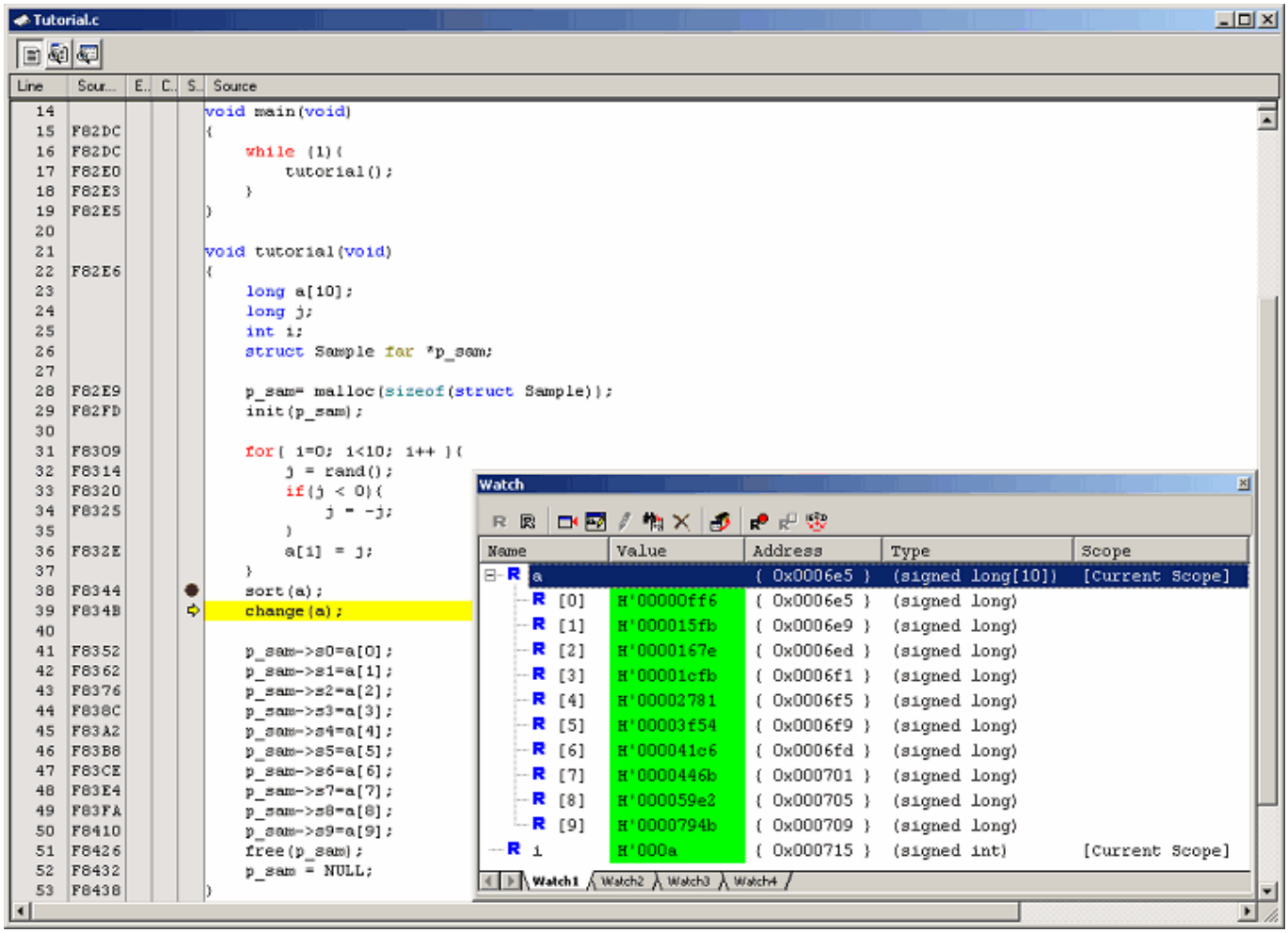


Figure 3.22 Editor window (Step Out)

The data of the variable 'a' displayed in the Watch window will be sorted in ascending order.

Downloaded from Arrow.com.

3.13.3 Executing the Step Over Command

The Step Over command executes the whole of a function call as one step and then stops at the next statement of the main program.

To execute all statements in the change function at a time, choose Step Over from the Debug menu or click the Step Over button in the toolbar.



Figure 3.23 Step Over button

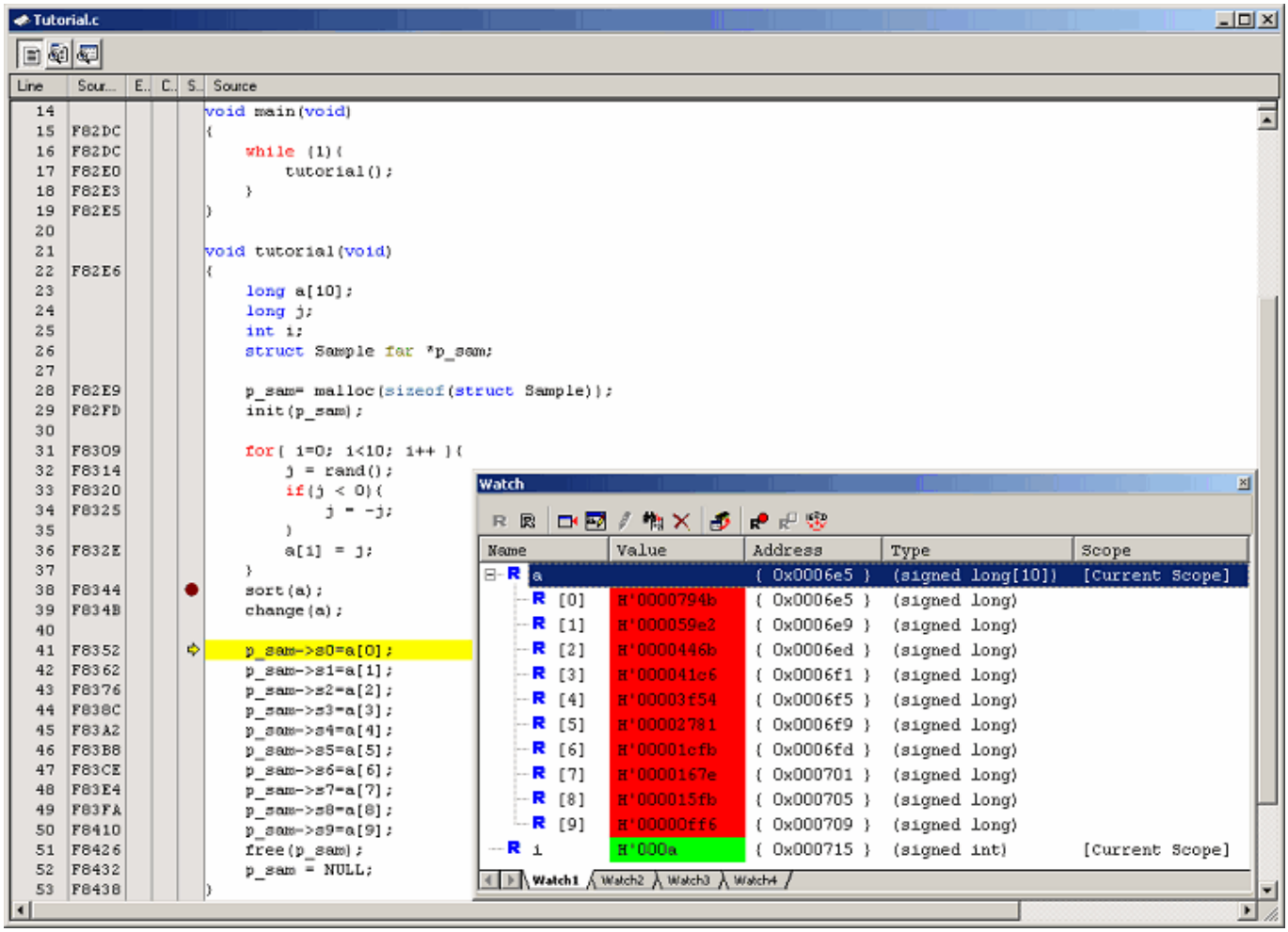


Figure 3.24 Editor window (Step Over)

The data of the variable 'a' displayed in the Watch window will be sorted in descending order.

Downloaded from Arrow.com.

3.14 Forcibly Breaking a Program

The High-performance Embedded Workshop permits you to forcibly break a program.

Clear all breakpoints.

To execute the rest of the tutorial function, choose Go from the Debug menu or click the Go button in the toolbar.



Figure 3.25 Go button

Since the program is executing an infinite loop process, choose Stop Program from the Debug menu or click the Stop button in the toolbar.



Figure 3.26 Stop button

Downloaded from Arrow.com.

3.15 Hardware Break Facility

Hardware breaks cause the program to stop when it executes a specified address (instruction fetch) or reads or writes to a specified memory location (data access).

3.15.1 Stopping a Program when It Executes a Specified Address

The Editor window permits you to set an instruction fetch event easily. For example, you can set an instruction fetch event at a place where the sort function is called.
Double-click a row in the Event column corresponding to the source line that includes a sort function call.

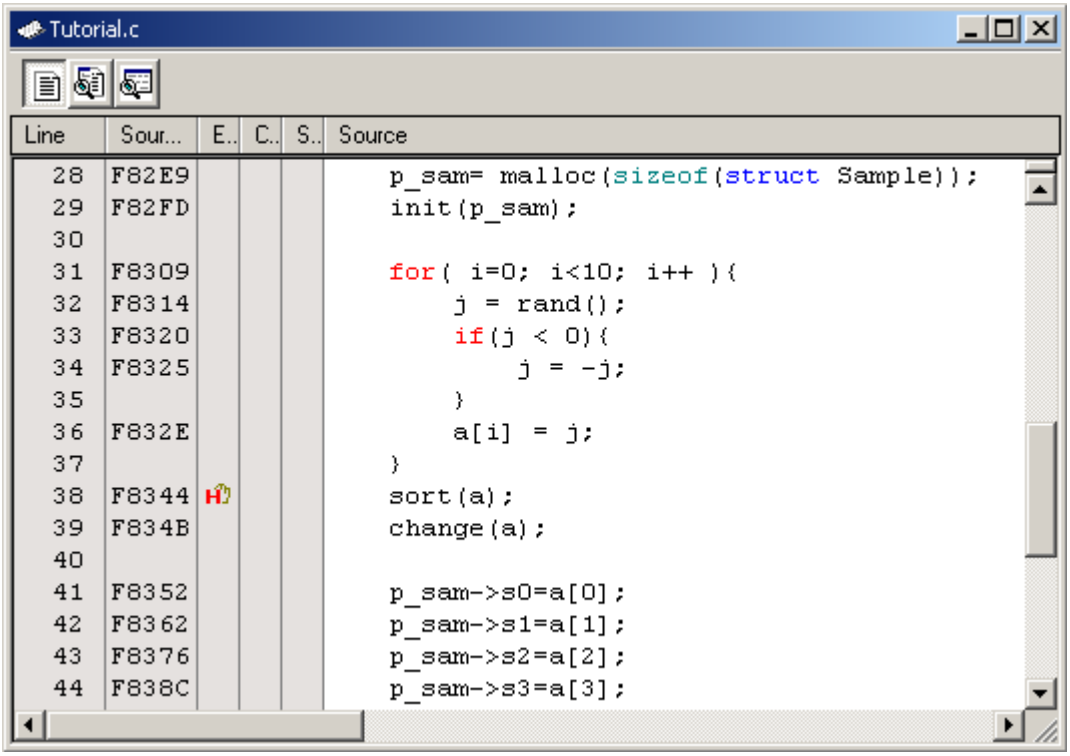


Figure 3.27 Editor window (setting a hardware breakpoint)

The source line that includes the sort function will be marked with , indicating that a hardware breakpoint that will cause a program to stop when it fetches an instruction has been set there.

Downloaded from Arrow.com.

3.16 Stopping a Program when It Accesses Memory

To stop a program when it reads or writes a value to a global variable, set up a hardware break as described below.

Choose Event -> Hardware Break from the View menu to display the Hardware Break dialog box.

Open the OR page of the Hardware Break dialog box. In the Editor window, select a global variable that you want to be the object of a hardware break so that a program is made to stop when it reads or writes a value to the variable, and drag-and-drop the selected variable into the OR page.

Then click the Apply button.

When you run a program, it will stop running when a value is read or written to the global variable you have set.

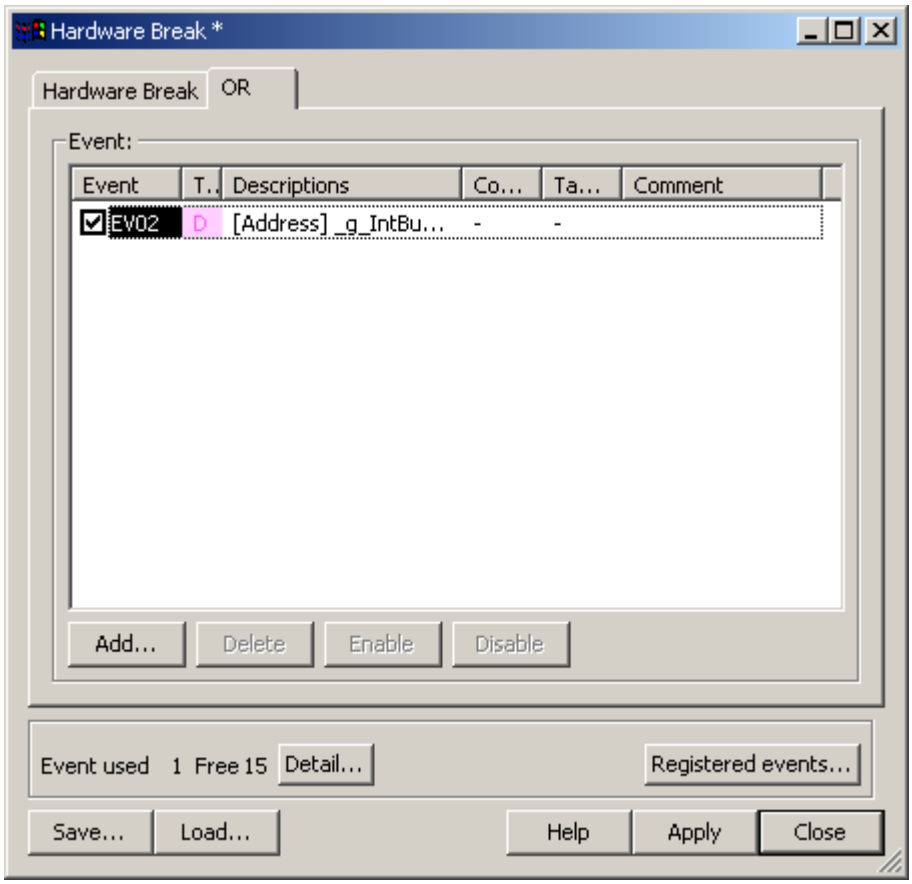


Figure 3.28 Hardware Break dialog box

Notes: (1) Only the global variables that are 1 byte or 2 bytes in size can be set.
(2) Local variables cannot be set.

Downloaded from Arrow.com.

3.17 Trace Facility

The trace facility of the E100 emulator has a special memory known as "trace memory" that can hold an execution record of up to 4M bus cycles, which is always updated during program execution. The content of trace memory is displayed in the Trace window.

Choose Code –> Trace from the View menu or click the Trace toolbar button .

The Trace window shown below will be displayed.

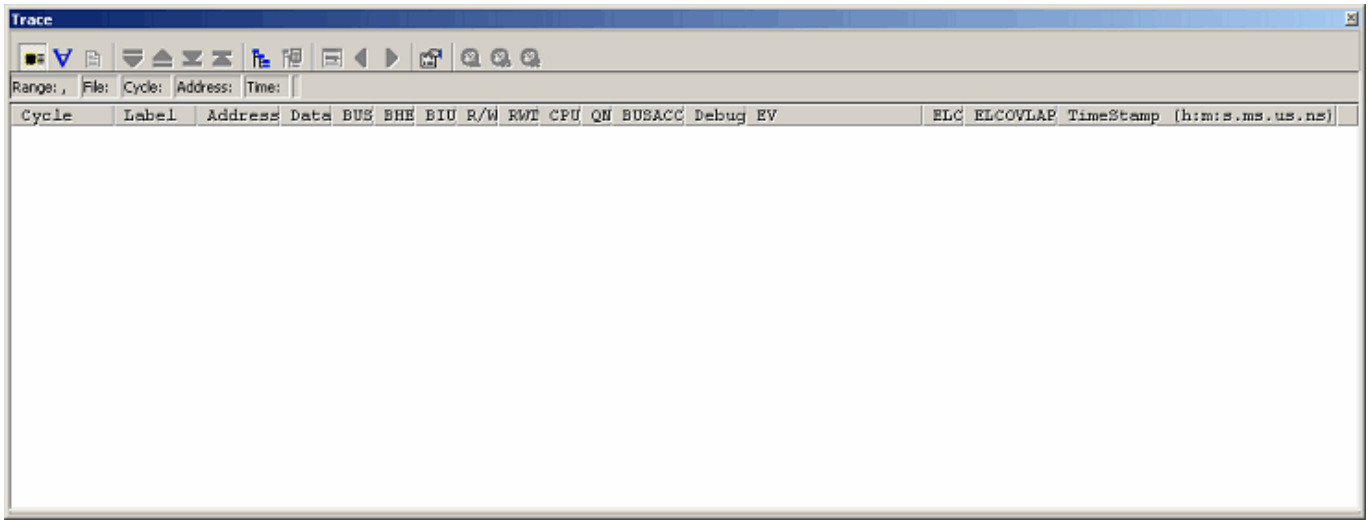


Figure 3.29 Trace window

The following outlines the trace facility and describes how to set.

Downloaded from Arrow.com.

3.17.1 Showing the Trace Information Acquired by Fill Until Stop

The free trace facility acquires trace information successively from when the user program starts running till when it breaks.

(1) Clear all break conditions. Click the right mouse button anywhere in the Trace window and choose Acquisition from the pop-up menu that is displayed. The Trace conditions dialog box shown below will be displayed. Check to see that the selected trace mode is Fill until stop. Click the Close button.

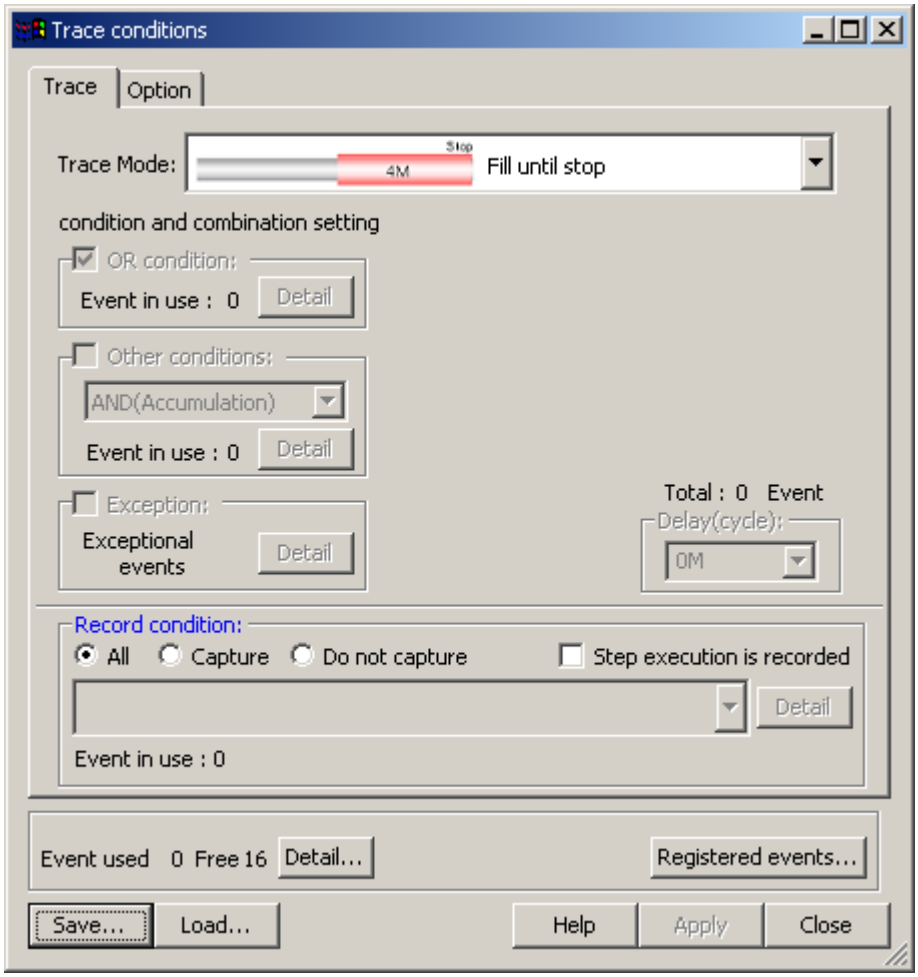
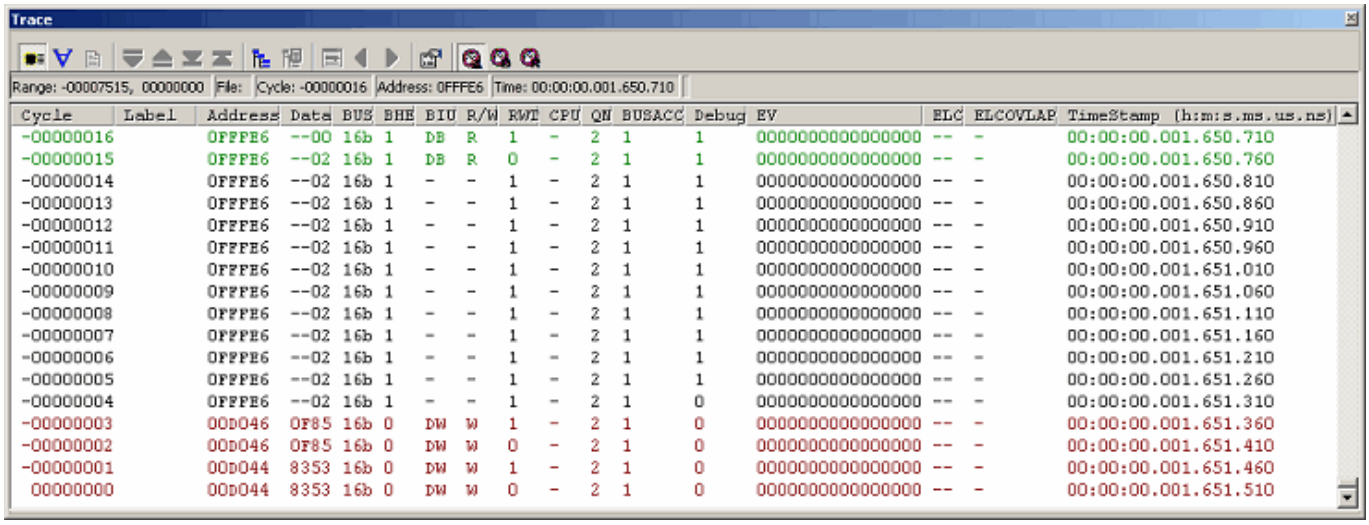


Figure 3.30 Trace conditions dialog box (free trace)

(2) Set a software break in a line of the tutorial function where p_sam ->s0=a[0]; is written.

Downloaded from Arrow.com.

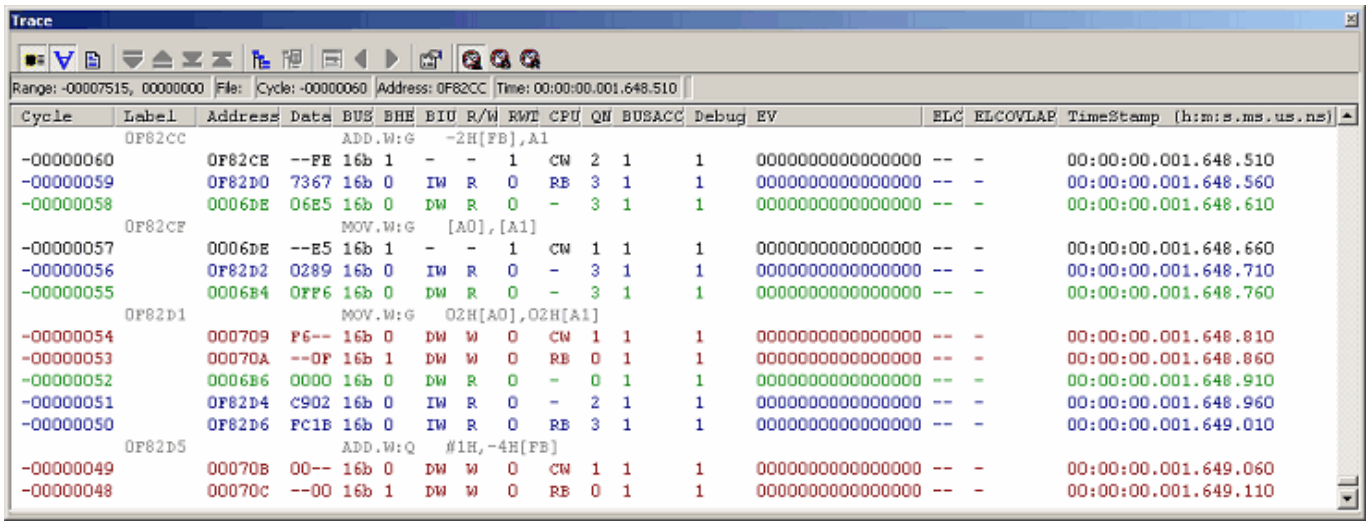
(3) Choose Reset Go from the Debug menu. Processing will be halted by a break, and the trace information from start to break will be displayed in the Trace window.



Cycle	Label	Address	Data	BUS	BHE	BIU	R/W	RWT	CPU	QN	BUSACC	Debug	EV	ELC	ELCOVLAP	TimeStamp (h:m:s.ms.us.ns)
-00000016		0FFFE6	--00	16b	1	DB	R	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.710
-00000015		0FFFE6	--02	16b	1	DB	R	0	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.760
-00000014		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.810
-00000013		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.860
-00000012		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.910
-00000011		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.960
-00000010		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.010
-00000009		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.060
-00000008		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.110
-00000007		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.160
-00000006		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.210
-00000005		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.260
-00000004		0FFFE6	--02	16b	1	-	-	1	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.310
-00000003		00D046	0F85	16b	0	DW	W	1	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.360
-00000002		00D046	0F85	16b	0	DW	W	0	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.410
-00000001		00D044	8353	16b	0	DW	W	1	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.460
00000000		00D044	8353	16b	0	DW	W	0	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.510

Figure 3.31 Trace window (free trace)

(4) A mixed display of bus, disassemble and/or source display is possible. Choosing Display Mode –> DIS from the pop-up menu, you can display trace information in a bus and disassemble mixed mode.



Cycle	Label	Address	Data	BUS	BHE	BIU	R/W	RWT	CPU	QN	BUSACC	Debug	EV	ELC	ELCOVLAP	TimeStamp (h:m:s.ms.us.ns)
	0F82CC			ADD.W:G		-2H[FB],A1										
-00000060		0F82CE	--FE	16b	1	-	-	1	CW	2	1	1	0000000000000000	--	-	00:00:00.001.648.510
-00000059		0F82D0	7367	16b	0	IW	R	0	RB	3	1	1	0000000000000000	--	-	00:00:00.001.648.560
-00000058		0006DE	06E5	16b	0	DW	R	0	-	3	1	1	0000000000000000	--	-	00:00:00.001.648.610
	0F82CF			MOV.W:G		[A0],[A1]										
-00000057		0006DE	--E5	16b	1	-	-	1	CW	1	1	1	0000000000000000	--	-	00:00:00.001.648.660
-00000056		0F82D2	0289	16b	0	IW	R	0	-	3	1	1	0000000000000000	--	-	00:00:00.001.648.710
-00000055		0006B4	0FF6	16b	0	DW	R	0	-	3	1	1	0000000000000000	--	-	00:00:00.001.648.760
	0F82D1			MOV.W:G		02H[A0],02H[A1]										
-00000054		000709	F6--	16b	0	DW	W	0	CW	1	1	1	0000000000000000	--	-	00:00:00.001.648.810
-00000053		00070A	--0F	16b	1	DW	W	0	RB	0	1	1	0000000000000000	--	-	00:00:00.001.648.860
-00000052		0006B6	0000	16b	0	DW	R	0	-	0	1	1	0000000000000000	--	-	00:00:00.001.648.910
-00000051		0F82D4	C902	16b	0	IW	R	0	-	2	1	1	0000000000000000	--	-	00:00:00.001.648.960
-00000050		0F82D6	FC1B	16b	0	IW	R	0	RB	3	1	1	0000000000000000	--	-	00:00:00.001.649.010
	0F82D5			ADD.W:Q		#1H,-4H[FB]										
-00000049		00070B	00--	16b	0	DW	W	0	CW	1	1	1	0000000000000000	--	-	00:00:00.001.649.060
-00000048		00070C	--00	16b	1	DW	W	0	RB	0	1	1	0000000000000000	--	-	00:00:00.001.649.110

Figure 3.32 Trace window (bus and disassemble mixed display)

Downloaded from Arrow.com.

(5) Furthermore, choosing Display Mode –> SRC from the pop-up menu, you can display trace information in a bus, disassemble and source mixed mode.

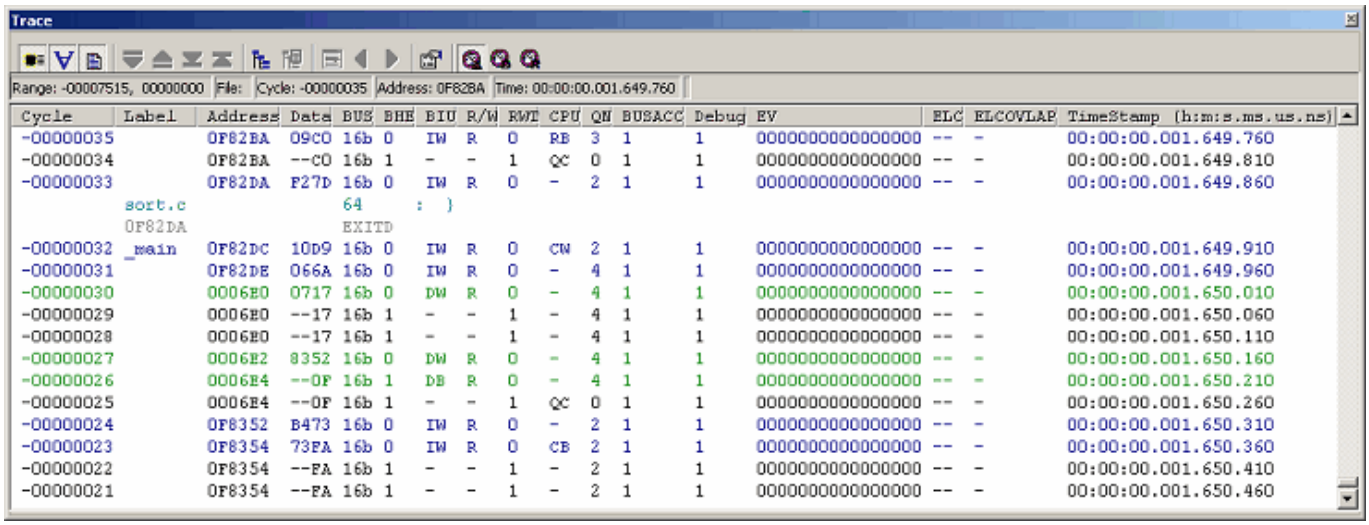


Figure 3.33 Trace window (bus, disassemble and source mixed display)

Downloaded from Arrow.com.

3.17.2 Showing the Trace Information Acquired by Fill around TP

The point & delay facility stops acquiring trace information a specified number of cycles after a trace point is encountered. This facility allows you to keep track of program flow from trace information without having to break the user program.

(1) If any break conditions are set, clear all of them.

(2) Choose Fill around TP for trace mode in the Trace conditions dialog box. In the Delay Value (Cycles) column, specify 4M. (Up to 4M cycles of trace information from where a trace point is encountered will be acquired.)

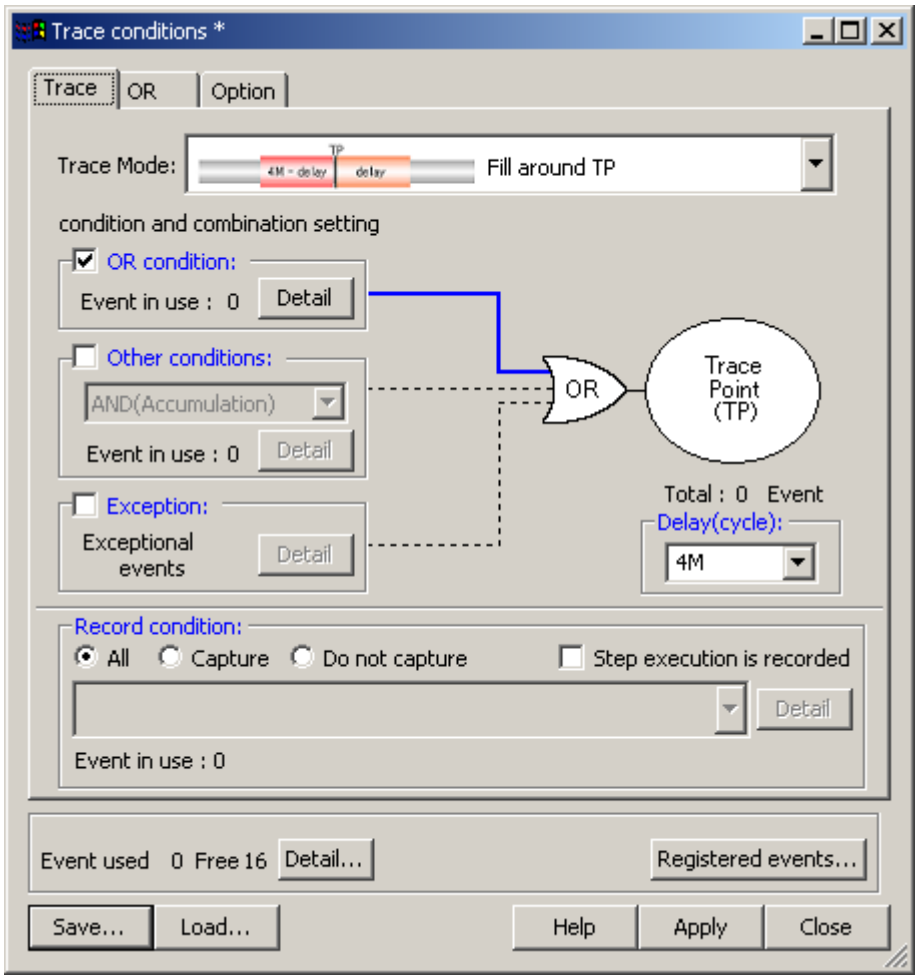


Figure 3.34 Trace conditions dialog box (Fill around TP)

Downloaded from Arrow.com.

(3) Next, set a trace point at which the debugger starts acquiring trace information. Open the OR page of the Trace conditions dialog box. Select the main function in the Editor window and drag-and-drop it into the OR page. Click the Apply button and then the Close button.
Thus, the debugger will start acquiring trace information from when the main function is executed.

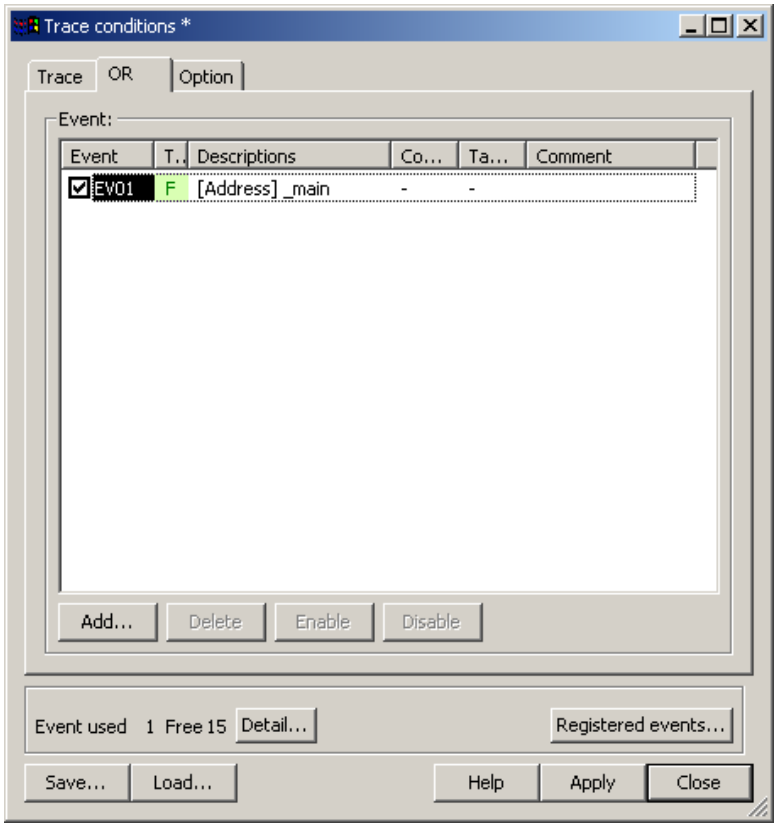


Figure 3.35 Trace conditions dialog box (OR page)

(4) Choose Reset Go from the Debug menu. A short time after a trace point is reached, the trace content shown below will be displayed in the Trace window.

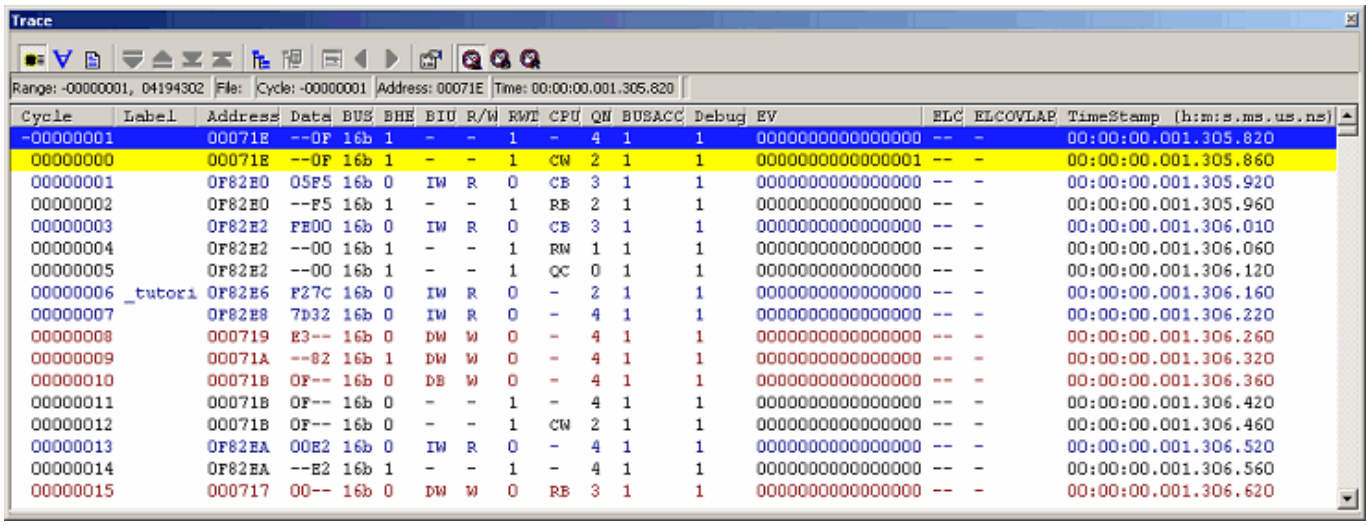


Figure 3.36 Trace window (Fill around TP)

Downloaded from Arrow.com.

3.17.3 Showing a Function Execution History

A function execution history can be displayed from the acquired trace information.

(1) Clear all break conditions. Click the right mouse button anywhere in the Trace window and choose Acquisition from the pop-up menu that is displayed. The Trace conditions dialog box will be displayed. Switch the trace mode to Fill until stop and click the Apply button. Then click the Close button.

(2) Set a software break in a line of the tutorial function where p_sam–>s0=a[0]; is written.

(3) Choose Reset Go from the Debug menu. Processing will be halted by a break, and the trace information from start to break will be displayed in the Trace window.

(4) Click the right mouse button anywhere in the Trace window and choose Function Execution History -> Function Execution History from the pop-up menu that is displayed.

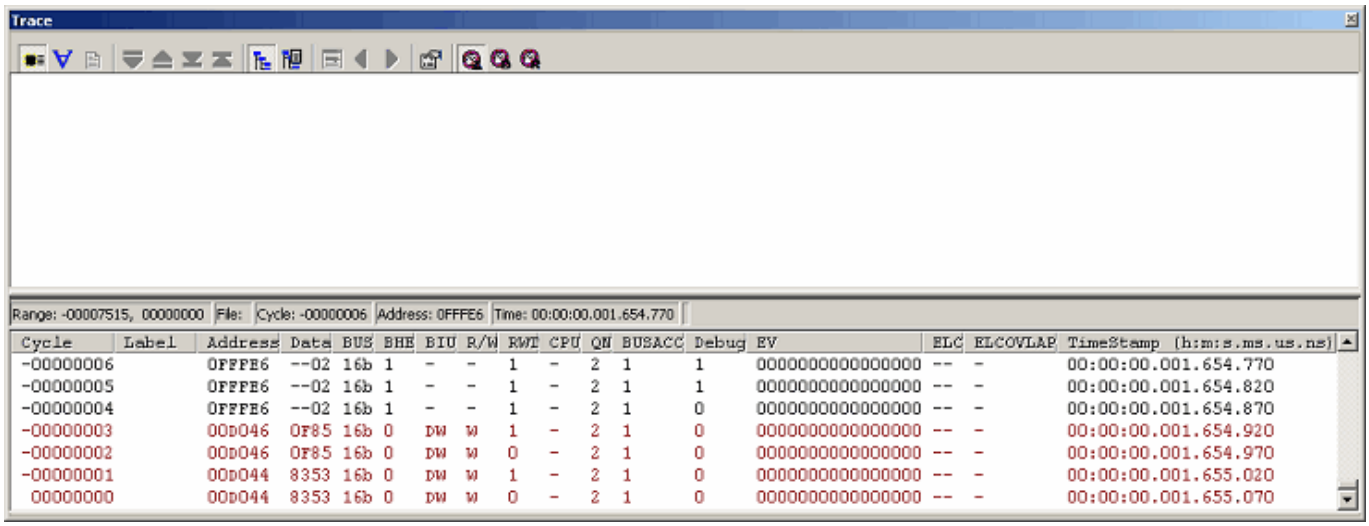


Figure 3.37 Trace window (function execution history–before analysis)

Downloaded from Arrow.com.

(5) Click the right mouse button anywhere in the displayed function execution history window and choose Analyze Execution History from the pop-up menu. A function execution history will be displayed in the upper pane of the Trace window.

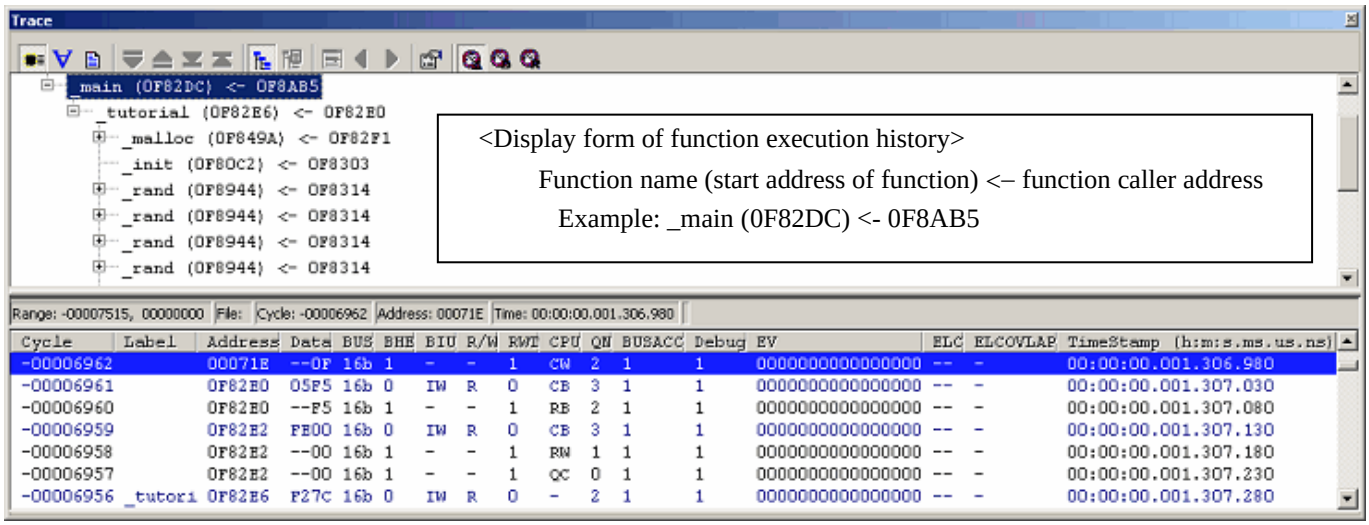


Figure 3.38 Trace window (function execution history–after analysis)

(6) Double-click any function in the displayed function execution history, and the trace information corresponding to that function will be displayed in the lower pane of the Trace window.

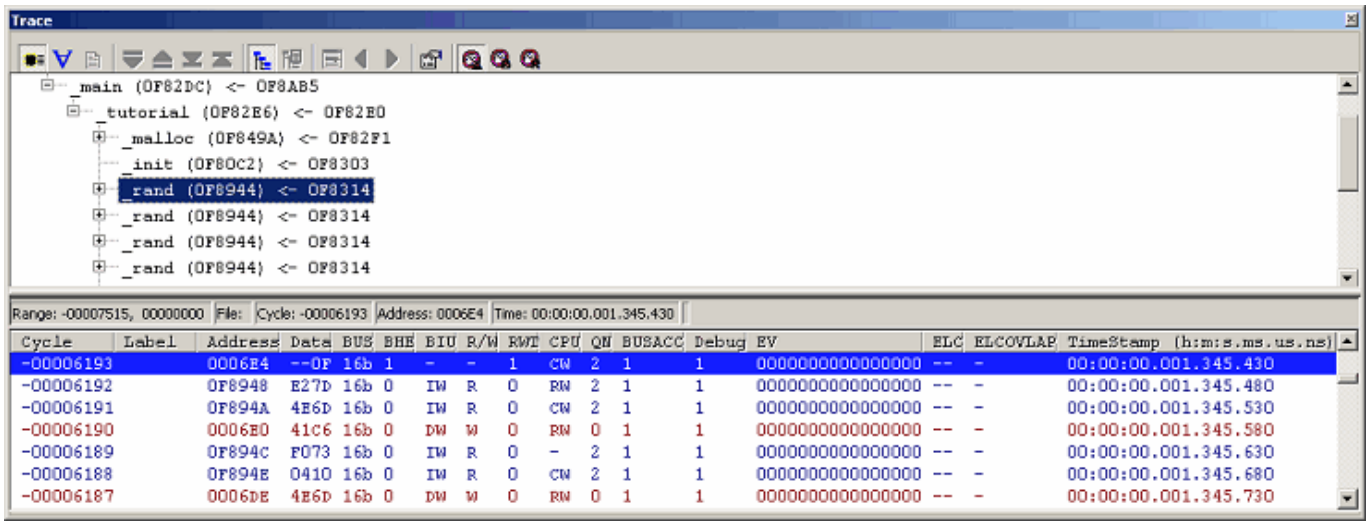


Figure 3.39 Trace window (function execution history)

Downloaded from Arrow.com.

3.17.4 Filter Facility

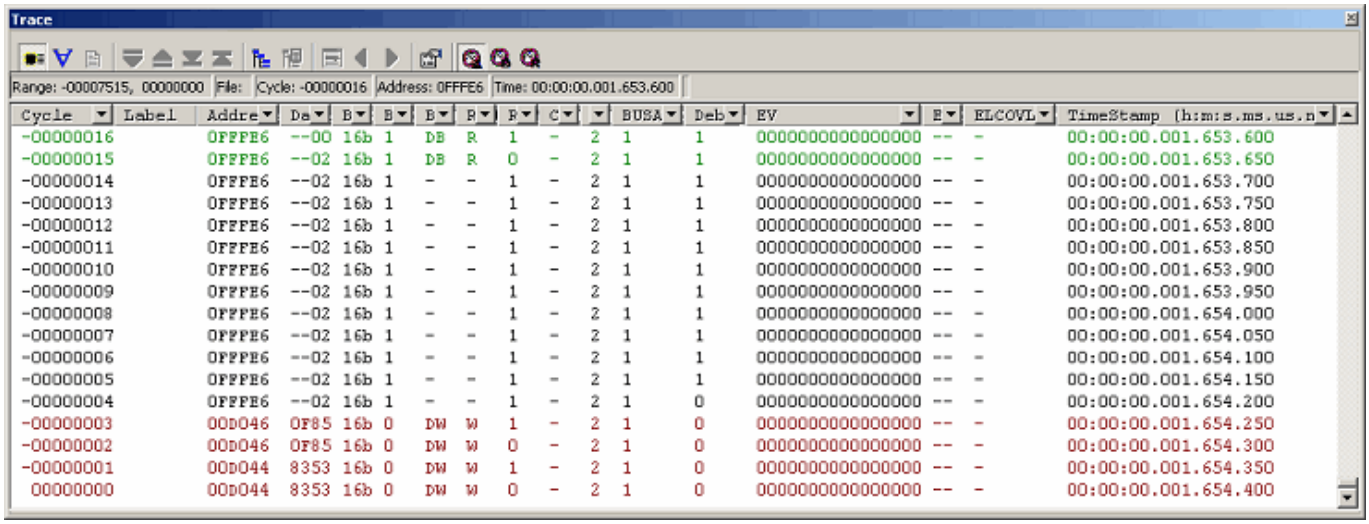
Use the filter facility to extract only the necessary cycles from the acquired trace information.
The filter facility does this by filtering the trace information in software that was acquired by hardware.
Unlike the "Capture/Do not Capture conditions" where you set acquisition conditions before getting trace information, this facility allows you to change filter settings for the acquired trace information any number of times without having to reexecute. Therefore, the necessary information can be extracted easily.

(1) Clear all break conditions. Click the right mouse button anywhere in the Trace window and choose Acquisition from the pop-up menu that is displayed. The Trace conditions dialog box will be displayed. Check to see that the selected trace mode is Fill until stop. Click the Close button.

(2) Set a software break in a line of the tutorial function where p-sam–>s0=a[0]; is written.

(3) Choose Reset Go from the Debug menu. Processing will be halted by a break, and the trace information from start to break will be displayed in the Trace window.

(4) Choose Auto Filter from the pop-up menu of the Trace window. The columns for which filtering can be applied will be marked by a ▼ button.



Cycle	Label	Addre	Da	B	B	B	R	R	C		BUSA	Deb	EV	E	ELCOVL	TimeStamp (h:m:s.ms.us.n
-00000016		0FFFE6	--00	16b	1	DB	R	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.653.600
-00000015		0FFFE6	--02	16b	1	DB	R	0	-	2	1	1	0000000000000000	--	-	00:00:00.001.653.650
-00000014		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.653.700
-00000013		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.653.750
-00000012		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.653.800
-00000011		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.653.850
-00000010		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.653.900
-00000009		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.653.950
-00000008		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.654.000
-00000007		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.654.050
-00000006		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.654.100
-00000005		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.654.150
-00000004		0FFFE6	--02	16b	1	-	-	1	-	2	1	0	0000000000000000	--	-	00:00:00.001.654.200
-00000003		00D046	0F85	16b	0	DW	W	1	-	2	1	0	0000000000000000	--	-	00:00:00.001.654.250
-00000002		00D046	0F85	16b	0	DW	W	0	-	2	1	0	0000000000000000	--	-	00:00:00.001.654.300
-00000001		00D044	8353	16b	0	DW	W	1	-	2	1	0	0000000000000000	--	-	00:00:00.001.654.350
00000000		00D044	8353	16b	0	DW	W	0	-	2	1	0	0000000000000000	--	-	00:00:00.001.654.400

Figure 3.40 Trace window (Auto Filter)

Downloaded from Arrow.com.

(5) Click the button in the R/W column and choose R from the pop-up menu.

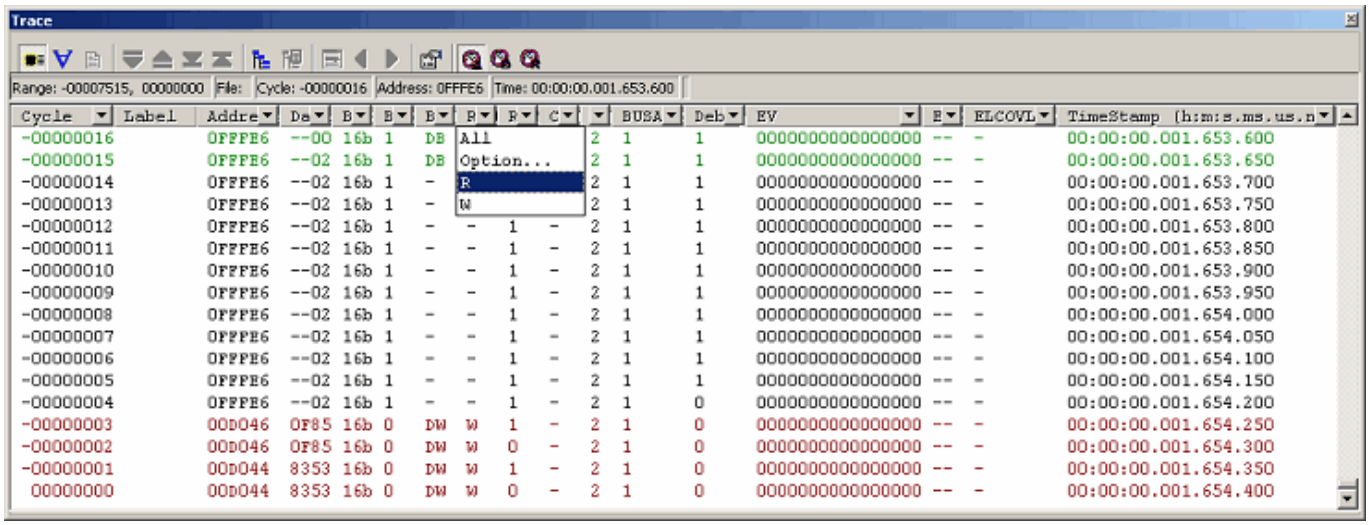


Figure 3.41 Trace window (Auto Filter)

(6) That way, the trace information for only R in the R/W column can be displayed.

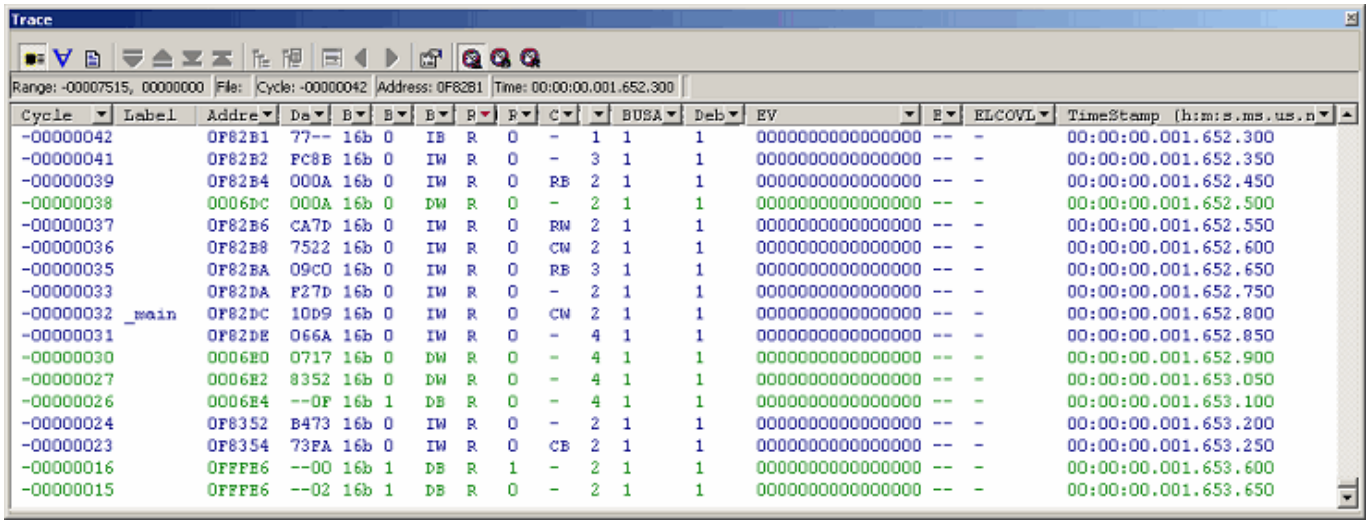


Figure 3.42 Trace window (Auto Filter)

Notes:

(1) The filter function does not affect the trace memory, so that its content remains intact.

(2) The filter can be used when the selected trace mode is Fill until stop, Fill until full or Fill around TP.

Downloaded from Arrow.com.

3.18 Stack Trace Facility

Using stack information, it is possible to show which function is the caller to the function where the current PC exists.
Set a software breakpoint in any line of the sort function by double-clicking at its corresponding row in the S/W Breakpoints column.

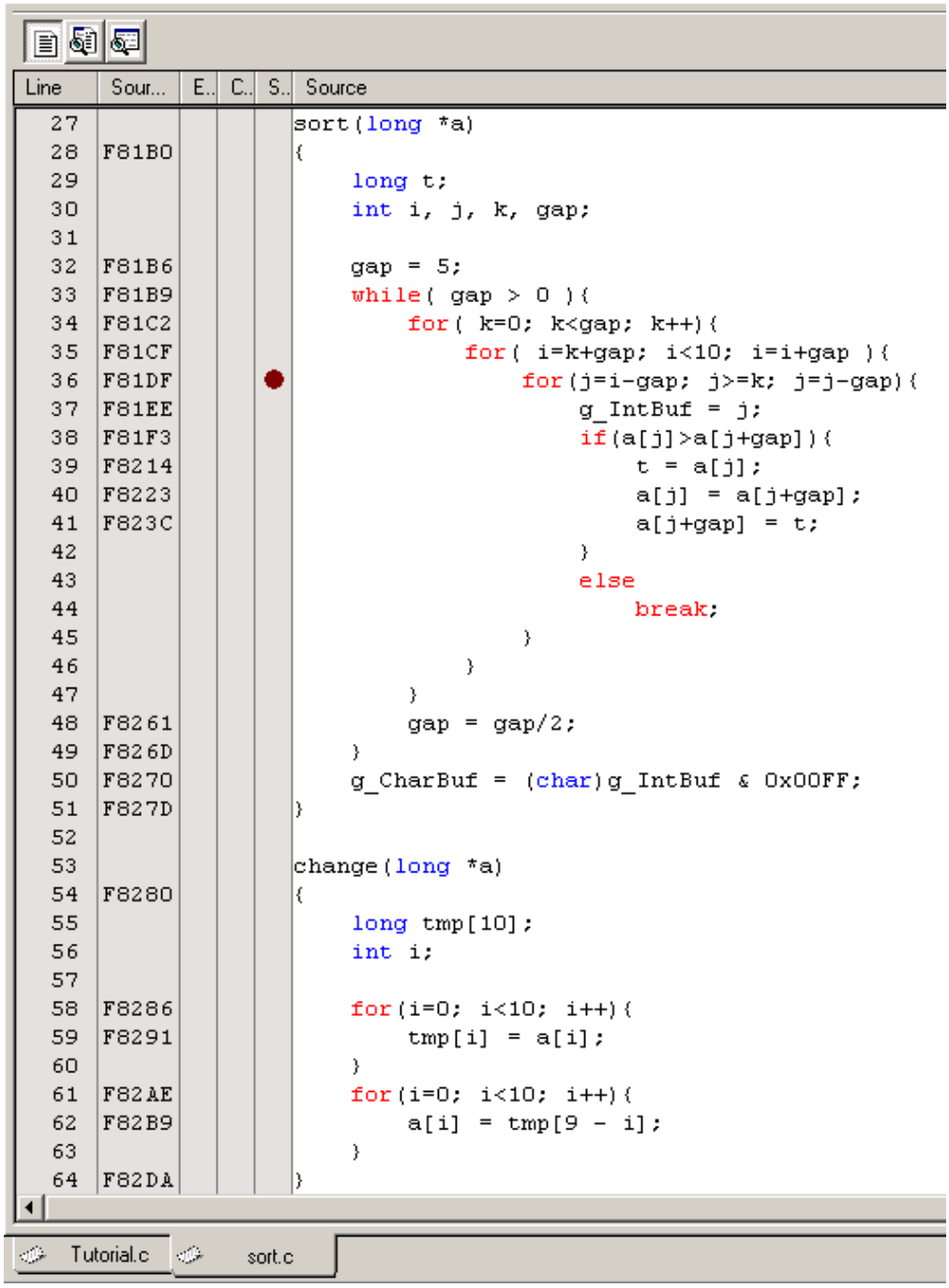


Figure 3.43 Editor window (setting a software breakpoint)

Choose Reset Go from the Debug menu.

Downloaded from Arrow.com.

After a break, choose Code –> Stack Trace from the View menu to open the Stack Trace window.

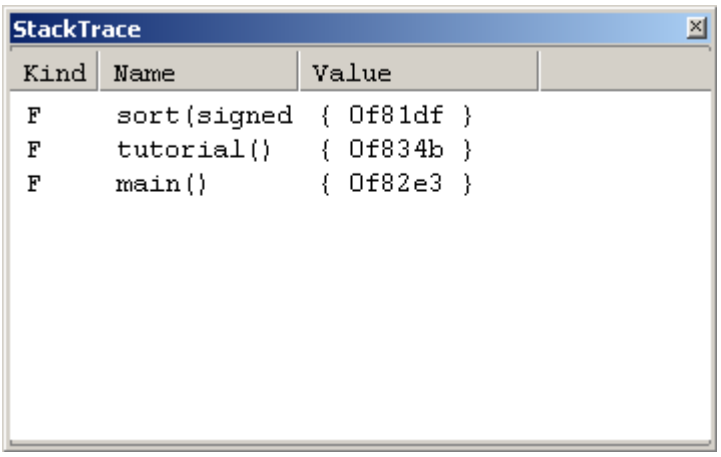


Figure 3.44 Stack Trace window

You will see that the current PC exists within the sort() function, and that the sort() function is called from the tutorial() function.
Clear the software breakpoint that you have set in a line of the sort function by double-clicking at its corresponding row in the S/W Breakpoints column again.

3.19 What Next?

In this tutorial, we have introduced to you several features of the E100 emulator and how to use the High-performance Embedded Workshop.
The emulation facility that the E100 emulator provides allows you to perform advanced debugging. Once the conditions that cause hardware or software problems to occur are exactly separated and identified by that debugging, you can examine those problems effectively.

Downloaded from Arrow.com.

4. Preparing to Debug

4.1 Starting the High-performance Embedded Workshop

Follow the procedure described below to start the High-performance Embedded Workshop.

(1) Connect the host machine and the E100 Emulator and user system. Then turn on the power to the E100 Emulator and user system.

(2) From Programs on the Start menu, choose Renesas -> High-performance Embedded Workshop -> High-performance Embedded Workshop.
The Welcome! dialog box shown below will appear.

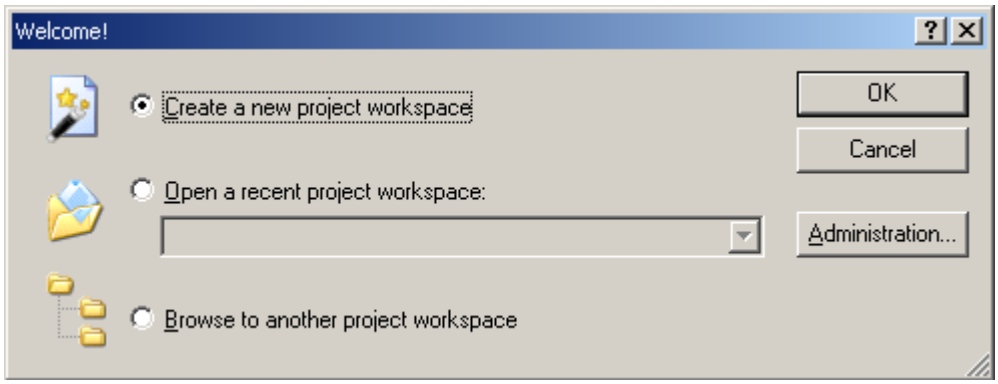


Figure 4.1 Welcome! dialog box

Select a startup method from the following.

- Create a new project workspace
- Open a recently used project workspace
  Select this option when you use an existing workspace.
  A history of the workspace you open will be displayed.
- Browse another project workspace
  Select this option when you use an existing workspace.
  This is the option available to choose when the workspace you opened has no history recorded.

Downloaded from Arrow.com.

4.2 Creating a New Workspace (Toolchain Unused)

The procedure for creating a new project workspace differs depending on whether you use a toolchain or not.
The E100 Emulator has no toolchains included in it. You can use a toolchain in an environment in which the C/C++ compiler package is installed.
Follow the procedure described below to create a new workspace.

(1) In the Welcome! dialog box, select the radio button titled "Create a new project workspace" and click the OK button.

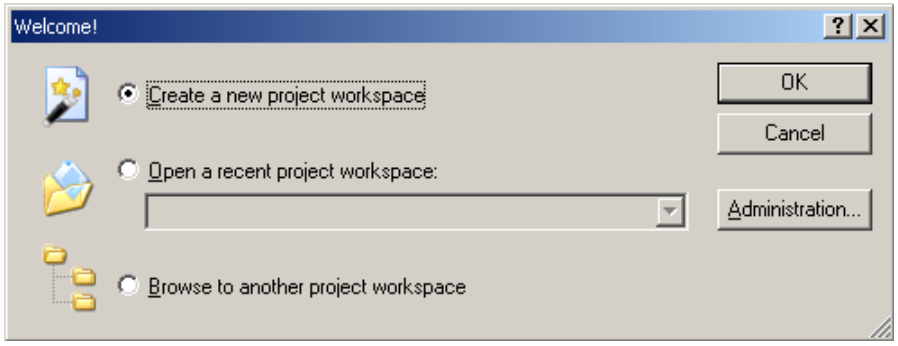


Figure 4.2 Welcome! dialog box

(2) Project Generator will start.

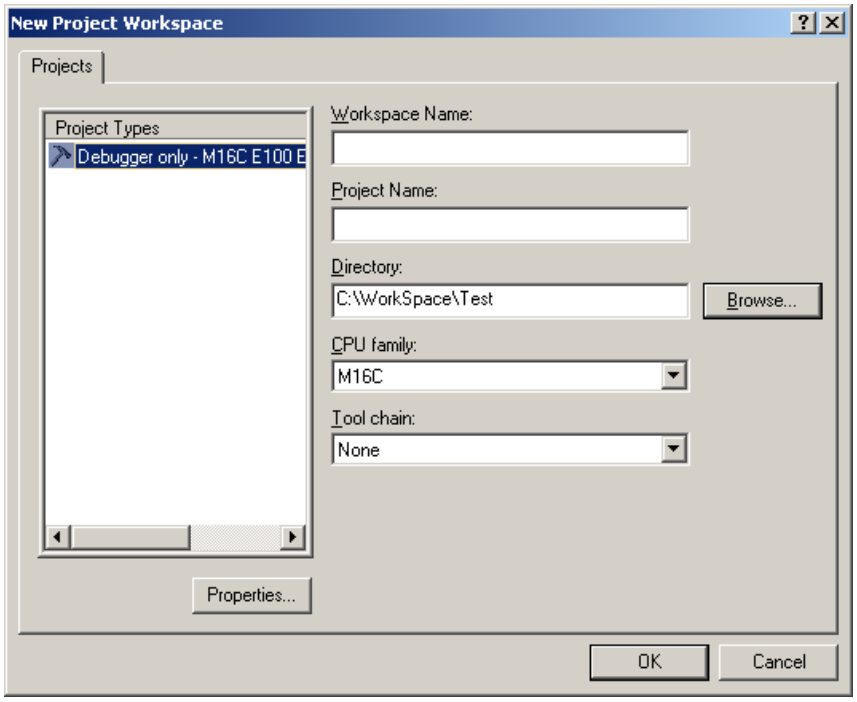


Figure 4.3 New Project Workspace dialog box

Workspace Name:	Enter a workspace name here.
Project Name:	Enter a project name here. If the same name as a workspace name is good, you do not need to enter it.
Directory:	Enter a directory in which you want a workspace to be created. Or you can click the Browse button and select a workspace directory from the ensuing list.
CPU family:	Select the CPU family of the MCU you are using.

Downloaded from Arrow.com.

The other list boxes are used for setting up a toolchain. If no toolchains are installed, the information specific to the CPU family is displayed here. Click the OK button.

(3) Select the debugger target.

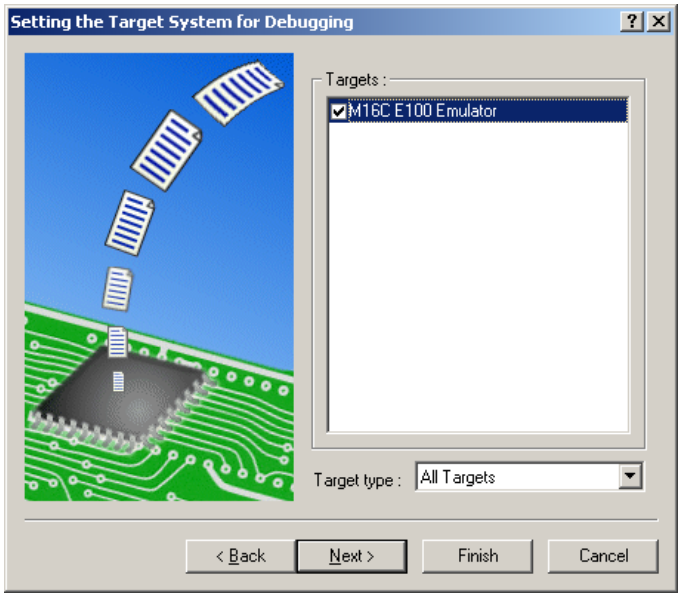


Figure 4.4 Setting the Target System for Debugging dialog box

Select the target platform you use by placing a check mark in its check box and click the Next button.

(4) Set a configuration name. Configuration refers to the file in which the High-performance Embedded Workshop status other than the emulator is saved.

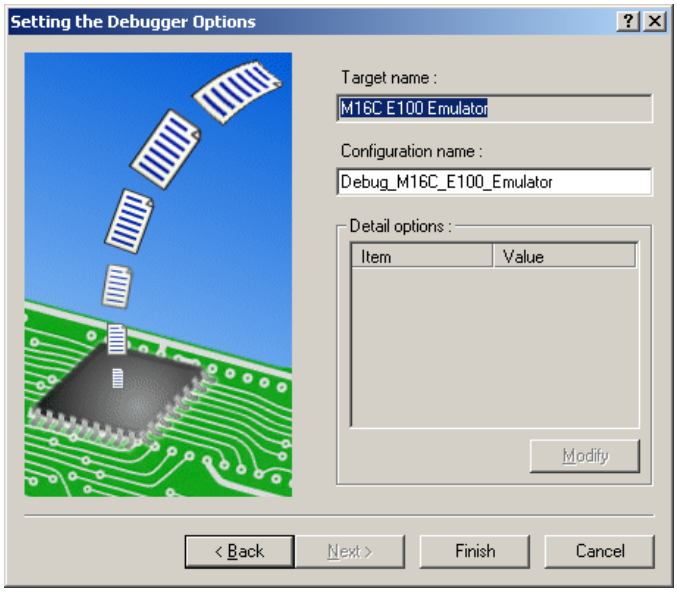


Figure 4.5 Setting the Debugger Options dialog box

If you have selected two or more target platforms, click the Next button and then set a configuration name for each target platform selected.

Downloaded from Arrow.com.

When you have finished setting configuration names, emulator-related settings are completed.
Click the Finish button, and the Summary dialog box will be displayed. Clicking the OK button in it starts the High-performance Embedded Workshop.

(5) After starting the High-performance Embedded Workshop, connect the E100 Emulator.

4.3 Creating a New Workspace (Toolchain Used)

Follow the procedure described below to create a new workspace.

(1) In the Welcome! dialog box, select the radio button titled "Create a new project workspace" and click the OK button.

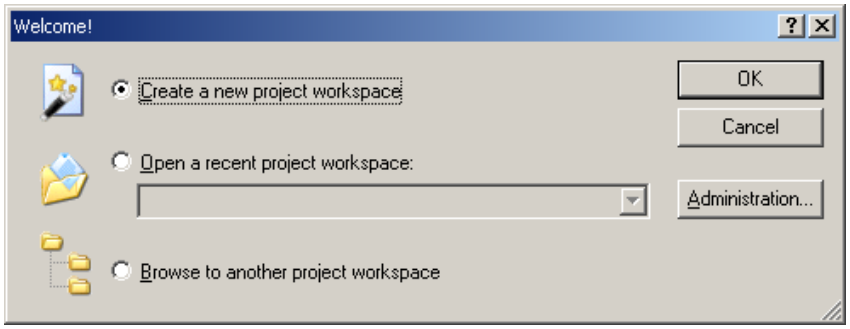


Figure 4.6 Welcome! dialog box

(2) Project Generator will start.

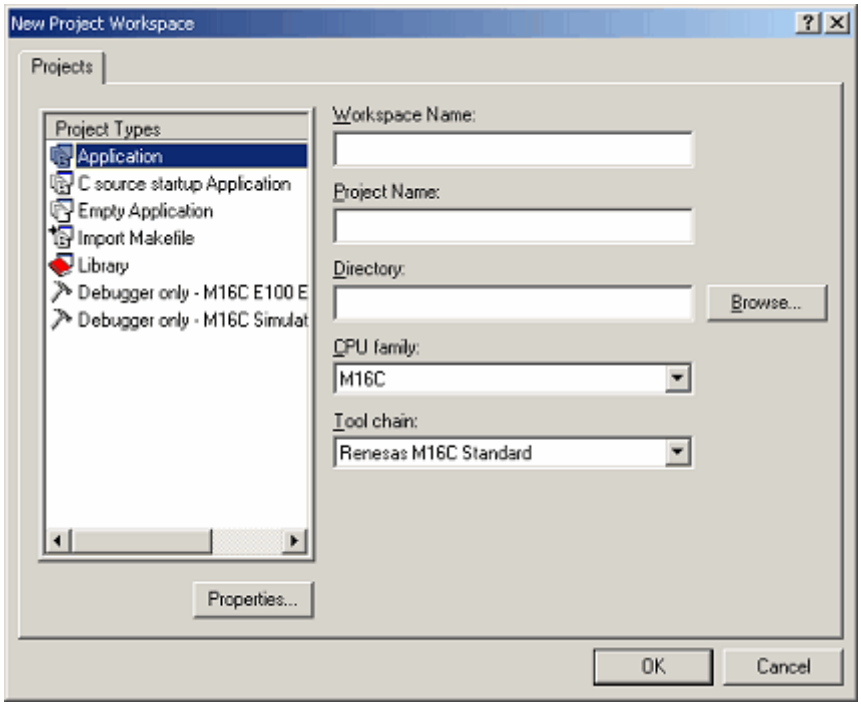


Figure 4.7 New Project Workspace dialog box

Downloaded from Arrow.com.

Workspace Name:	Enter a workspace name here.
Project Name:	Enter a project name here. If the same name as a workspace name is good, you do not need to enter it.
Directory:	Enter a directory in which you want a workspace to be created. Or you can click the Browse button and select a workspace directory from the ensuing list.
CPU family:	Select the CPU family of the MCU you are using.
Toolchain:	To use a toolchain, select the appropriate toolchain here. If you do not use, select None.

The other list boxes are used for setting up a toolchain. If no toolchains are installed, the information specific to the CPU family is displayed here. Click the OK button.

(3) Set the CPU and options for the toolchain and make other necessary settings.

(4) Select the debugger target.

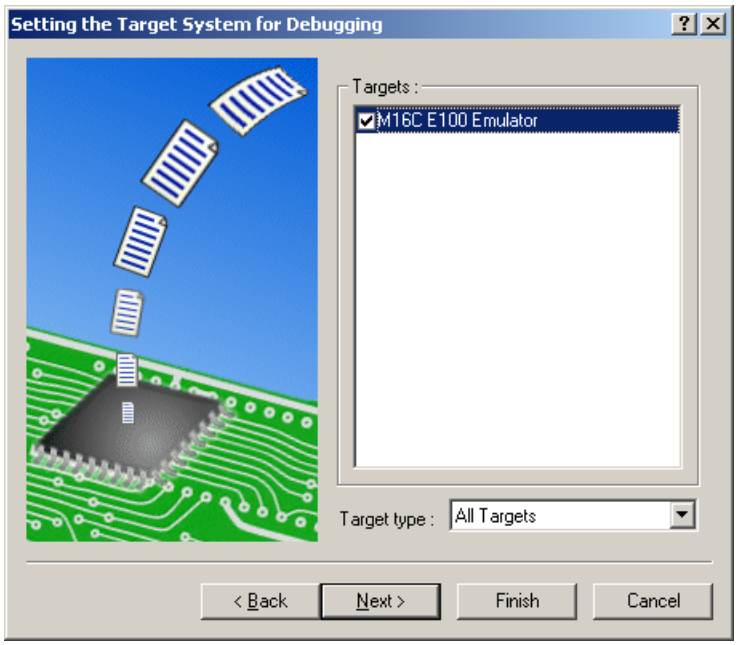


Figure 4.8 Setting the Target System for Debugging dialog box

Select the target platform you use by placing a check mark in its check box and click the Next button.

Downloaded from Arrow.com.

(5) Set a configuration name.

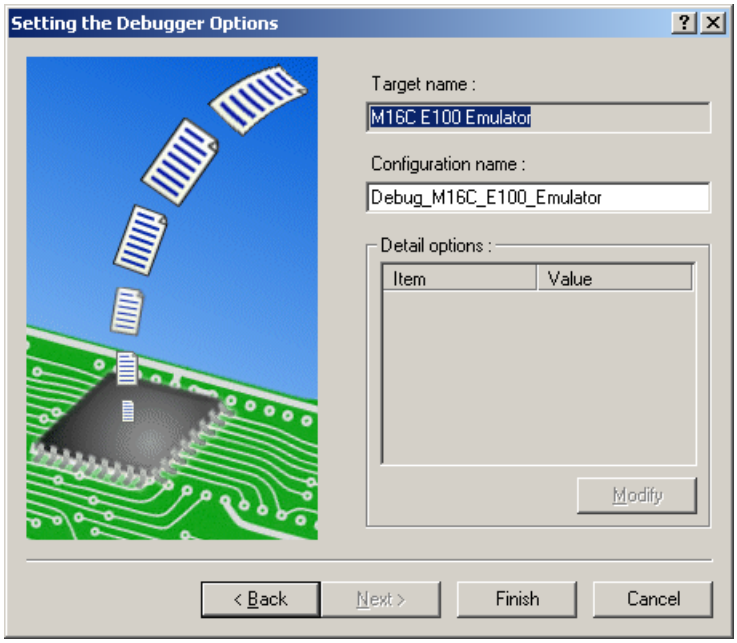


Figure 4.9 Setting the Debugger Options dialog box

If you have selected two or more target platforms, click the Next button and then set a configuration name for each target platform selected. When you have finished setting configuration names, emulator-related settings are completed. Click the Finish button, and the Summary dialog box will be displayed. Clicking the OK button in it starts the High-performance Embedded Workshop.

(6) After starting the High-performance Embedded Workshop, connect the E100 Emulator.

Downloaded from Arrow.com.

4.4 Opening an Existing Workspace

Follow the procedure described below to open an existing workspace.

(1) In the Welcome! dialog box, select the radio button titled "Browse to another project workspace" and click the OK button.

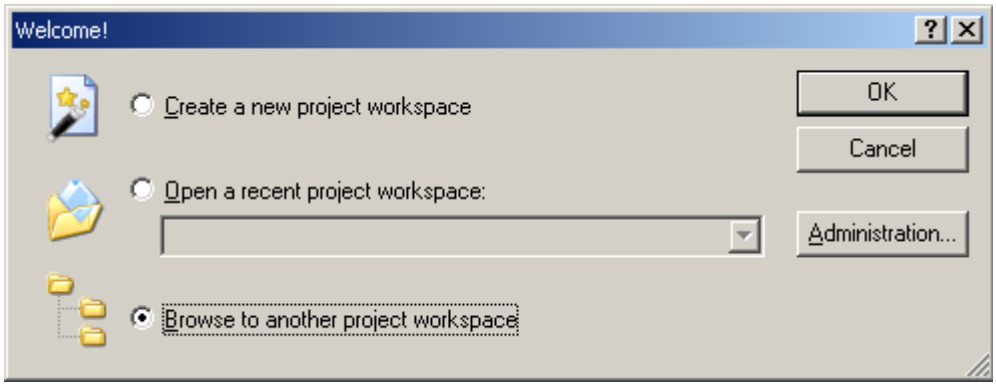


Figure 4.10 Welcome! dialog box

(2) The Open Workspace dialog box shown below will appear.

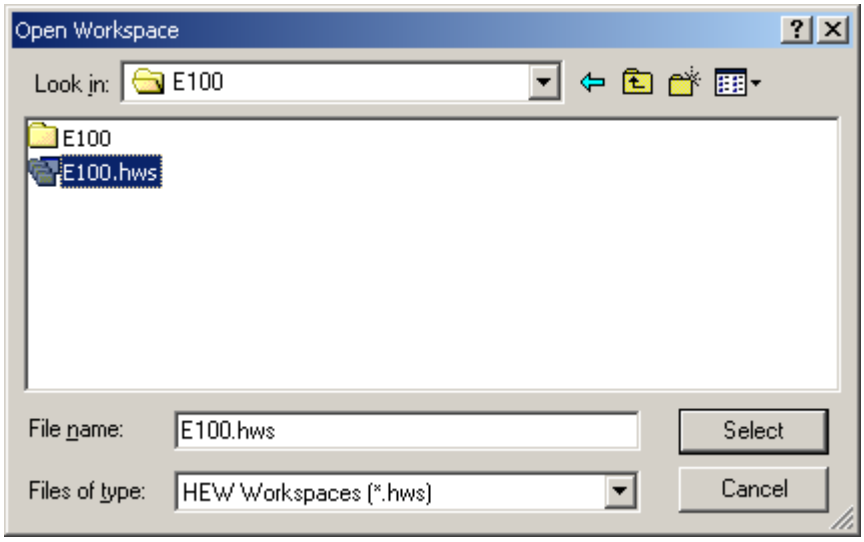


Figure 4.11 Open Workspace dialog box

Specify the directory in which workspaces are created, select a workspace file (extension ".hws") and click the Select button.

(3) The High-performance Embedded Workshop will start, and the state of the selected workspace in which it was saved will be restored. If the saved state of the selected workspace is one in which it was connected to the emulator, the workspace is automatically connected to the emulator. If the saved state of the selected workspace is one in which it was not connected to the emulator and you want to connect it, refer to "4.5.1 Connecting the Emulator".

Downloaded from Arrow.com.

4.5 Connecting the Emulator

4.5.1 Connecting the Emulator

There are following methods for connecting the emulator.

(1) Setting up the emulator at startup before connecting

Choose Debug Settings from the Debug menu to open the Debug Settings dialog box. In this dialog box, you can register download modules and the command chain to be automatically executed. When you are finished filling in the Debug Settings dialog box, the emulator will be connected.

(2) Loading a session file

Switching to the session file that has emulator usage settings preregistered in it helps you connect the emulator easily.

4.5.2 Reconnecting the Emulator

While the emulator is disconnected, you can reconnect it following one of the procedures described below.

(1) Choose Connect from the Debug menu.

(2) Click the Connect tool button [].

(3) Enter the connect command in the Command Line window.

Downloaded from Arrow.com.

4.6 Disconnecting the Emulator

4.6.1 Disconnecting the Emulator

To disconnect the emulator while it is active, follow one of the procedures described below.

(1) Choose Disconnect from the Debug menu.

(2) Click the Disconnect tool button [].

(3) Enter the disconnect command in the Command Line window.

4.7 Quitting the High-performance Embedded Workshop

Choosing Exit from the File menu lets you close the High-performance Embedded Workshop itself.
Before it closes, a message box will be displayed asking you whether you want to save the session. To save the session, click the Yes button.

Downloaded from Arrow.com.

4.8 Setting Up the Debug

Register download modules, set up automatic execution of command line batch files and set download options, etc.

4.8.1 Specifying a Download Module

Choose Debug Settings from the Debug menu to open the Debug Settings dialog box.

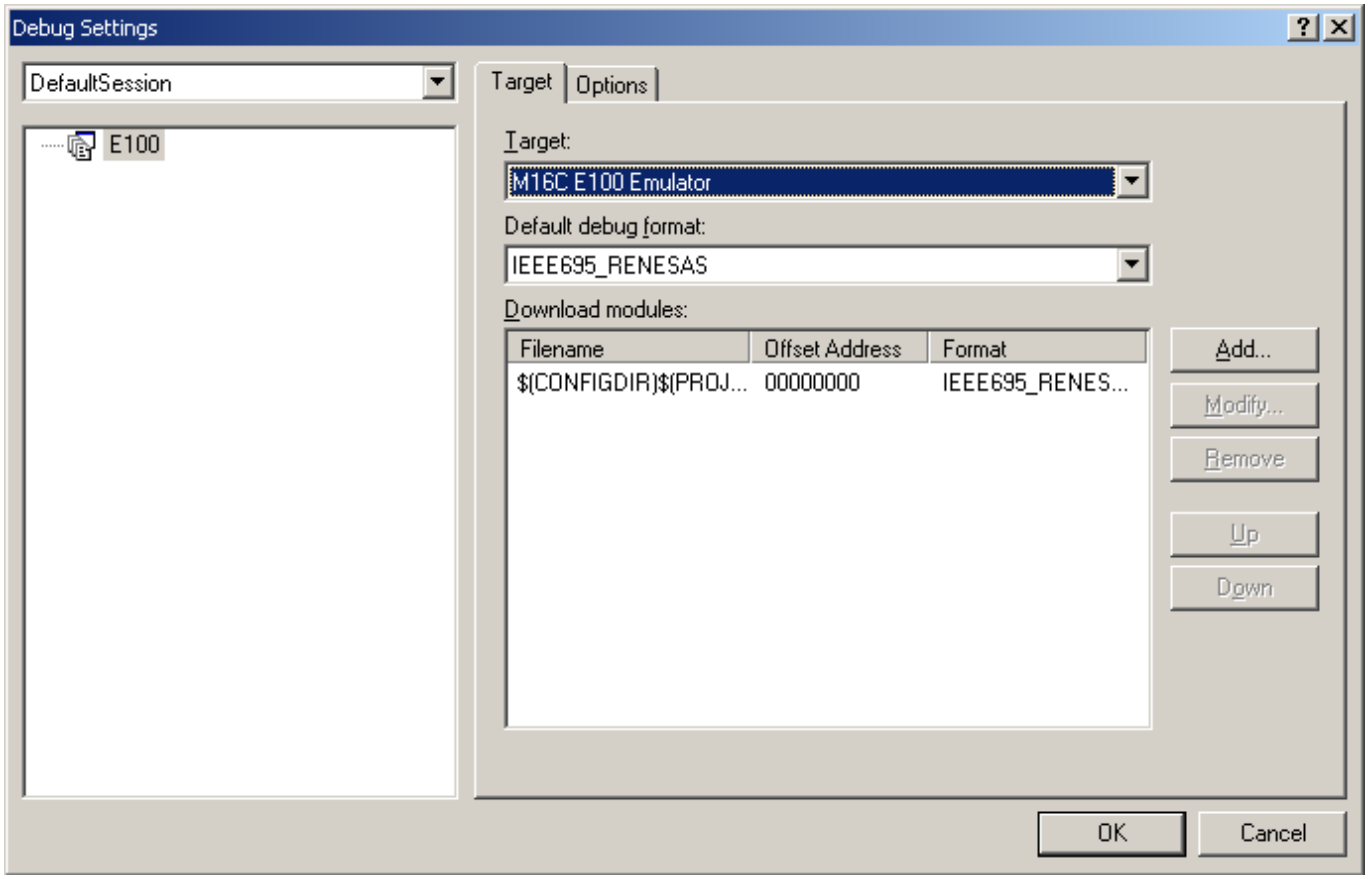


Figure 4.12 Debug Settings dialog box

In the Target drop-down list box, select the product name you want to connect.
In the Default debug format drop-down list box, select the format of the load module you want to download. Then register the load module corresponding to the selected format in the Download modules list box.

CAUTION
At this point in time, no programs are downloaded yet.
For details on how to download, refer to "5.2.1 Downloading a Program".

Downloaded from Arrow.com.

4.8.2 Setting Up Automatic Execution of Command Line Batch Files

Click the Options tab of the dialog box.

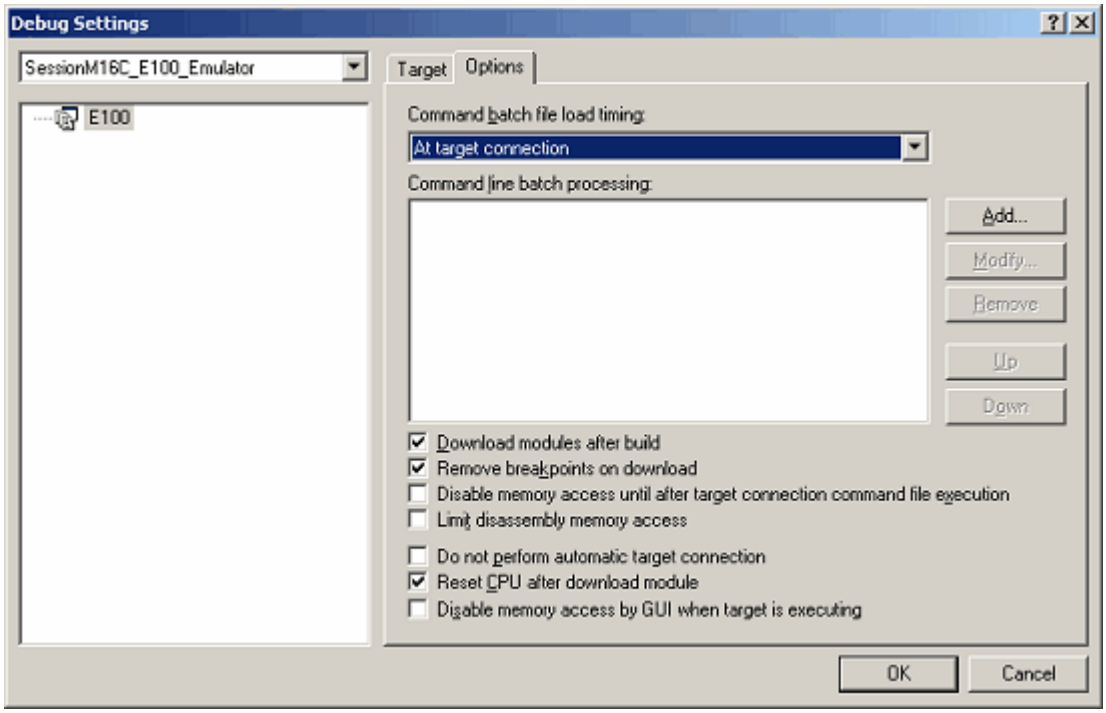


Figure 4.13 Debug Settings dialog box

Here, register a command chain that is automatically executed with specified timing.
Select your desired timing from the following four choices:

- When the emulator is connected
- Immediately before download
- Immediately after download
- Immediately after reset

In the Command batch file load timing drop-down list box, select the timing with which you want a command chain to be executed.

Downloaded from Arrow.com.

5. Debugging Functions

The E100 emulator supports the functions listed in the table below.

Table 5.1 List of Debug Functions

Item No.	Item			Specification
1	Software break			4,096 points
2	Event	Number of event points		Maximum number of effective points: 16
		Content of event		Executed address detection
				Data access detection
				Interrupt generation/exit detection
				External trigger detection
		Task ID		Can be set separately for each event
		Number of times an event occurred		Maximum 255 times
3	Exception detection			Violation of access protection
				Read from uninitialized memory
				Stack access violation
				Performance overflow
				Realtime profile overflow
				Trace memory overflow
				Task stack access violation
				OS dispatch
4	Hardware break	Hardware breakpoints	Event combination	OR, AND (Accumulation), AND (Simultaneous), subroutine, sequential and state transition
			Exception detection	See item No. 3
		Delay		Maximum 65,535 bus cycles
5	Trace	Trace size		Maximum 4M cycles
		Trace mode	Fill until stop	Continues collecting until program stops running
			Fill until full	Stops collecting traces when trace memory is full
			Fill around TP	Stops collecting traces after retarded for delay cycles from when trace point is reached
			Repeat fill until stop	Collects a total of 512 cycles before and after trace point
			Repeat fill until full	Collects a total of 512 cycles before and after trace point
		Trace point	Event combination	OR, AND (Accumulation), AND (Simultaneous), subroutine, sequential and state transition
			Exception detection	See item No. 3
		Delay		Maximum 4M bus cycles
		Trace Capture/Do not Capture		Capture/Do not Capture by event - Between two events - Duration of an event - Duration of an event occurring in a subroutine
				Data access instruction extraction
6	Performance	Content of measurement		Measures maximum, minimum and average execution time in up to 8 sections and pass counts
				Timeout detection
		Resolution		10 ns to 1.6µs
		Measurement mode	Event combination	Between two events, Event period and Interrupt-disabled range between two events
7	RAM monitor			512 bytes × 32 blocks - Shows last read/write accesses performed - Comes with initialization-omitted detect function
8	Profile			128 Kbytes × 8 blocks (1-Mbyte space)
				Cumulative time and pass count overflow detection
9	Coverage			C0 level code coverage 256 Kbytes × 8 blocks (2-Mbyte space)
				C0+C1 level code coverage 128 Kbytes × 8 blocks (1-Mbyte space)
				Address range and source file specification
				Data coverage 64 Kbytes × 8 blocks (512-Kbyte space) Address range, section specification and task stack

Downloaded from Arrow.com.

5.1 Setting Up the Emulation Environment

When the emulator is connected, the Device setting and the Configuration properties dialog boxes are displayed. Here, select the general options associated with the emulator. Note that the target MCU to be debugged, etc. can be set only once at startup.

5.1.1 Setting Up the Emulator at Startup

When the emulator starts, the following three dialog boxes are displayed.

(1) Device setting dialog box

Use this dialog box to select the target MCU and establish communication.
This dialog box can be redisplayed by selecting Emulator -> Device setting from the Setup menu after starting the emulator. In this case, however, be aware that changes of the settings after starting the emulator are not reflected immediately and will be set as the initial value when reconnecting the emulator.

(2) Configuration properties dialog box

This dialog box is displayed after the Device setting dialog box. Use this dialog box to make settings related to the emulator and debug functions.
This dialog box can be re-opened by selecting Emulator -> System from the Setup menu after the emulator has been booted up. Some options in this dialog box can have their settings changed after startup. The changeable options are displayed as in active use, while the unchangeable options are inactive (grayed out), with their set contents only displayed.

(3) Connecting dialog box

This dialog box shows the progress of boot-up processing.

Downloaded from Arrow.com.

5.1.2 Setting Up the Target MCU

(1) Selecting the target MCU

On the Device page of the Device setting dialog box, specify the target MCU to be emulated. For details, refer to the hardware manual supplied with each product.

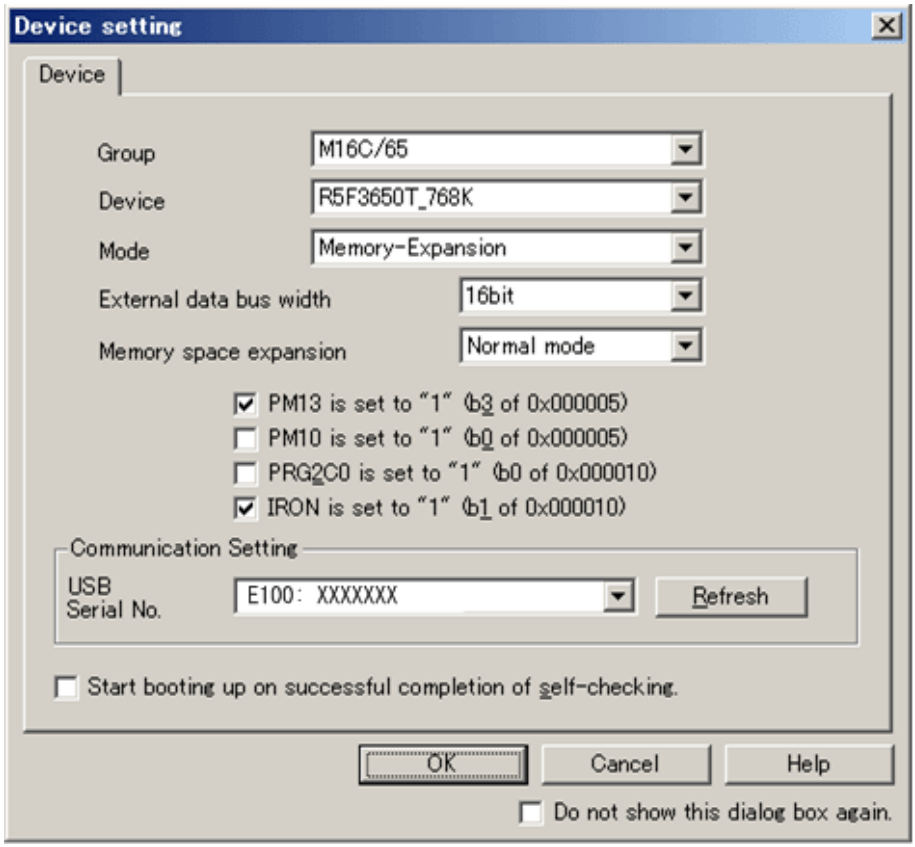


Figure 5.1 Device setting dialog box (Device page)

The target MCU you have set here cannot be changed after the emulator is connected. To change the target MCU, you need to disconnect the emulator and connect it again.

(2) Selecting an operation mode

Select one from the following options:
Single-Chip mode, Memory expansion mode, Microprocessor mode

CAUTION
Options are different depending on the target MCU that you select.

(3) Selecting an external data bus width

You can set this when the operation mode you have selected is Memory expansion mode or Microprocessor mode.
Select one from the following options:
8 bits, 16 bits (initial value)

Downloaded from Arrow.com.

(4) Selecting a memory expansion space

You can set this when the operation mode you have selected is Memory expansion mode or Microprocessor mode.
Select one from the following options:
Normal Mode (initial value), 4MB Mode

(5) Using PM13 (b3 of 0x000005) as set to 1

To switch the setting of the CS2 area, specify the PM13 (third bit of processor mode register 1) setting. When using the user program with PM13 set to 1, select this check box.

(6) Using PM10 (b0 of 0x000005) as set to 1

To expand the internal reserved area, specify the PM10 (zeroth bit of processor mode register 1) setting. When using the user program with PM10 set to 1, select this check box.

(7) Using PRG2C0 (b0 of 0x000010) as set to 1

To enable or disable the program ROM 2 area, specify the PRG2C0 (zeroth bit of program 2 area control register) setting. When using the user program with PRG2C0 set to 1, select this check box.

(8) Using IRON (b1 of 0x000010) as set to 1

To enable or disable the program ROM 1 area, specify the IRON (first bit of program 2 area control register) setting. When using the user program with IRON set to 1, select this check box.

(9) Setting up communication
You can select another target emulator connected via USB.
The 'USB Serial No.' list box shows unique identity information on the emulator connected via USB. Clicking on the Refresh button updates the information.

(10) Performing self-checking

If you click on the OK button with the 'Start booting up on successful completion of self-checking.' checkbox selected, hardware self-checking proceeds after connection to the emulator has been made according to the communication condition you have set.
The results are shown on completion of self-checking.
If the results are normal, boot-up processing continues. If an error is found, boot-up processing stops.

Downloaded from Arrow.com.

5.1.3 Setting Up the System

On the System page of the Configuration Properties dialog box, set up the entire emulator system.
This dialog box is displayed following the Device setting dialog box at startup.
Although this dialog box can be redisplayed after startup, you cannot change some settings in it. These settings can only be changed at startup.

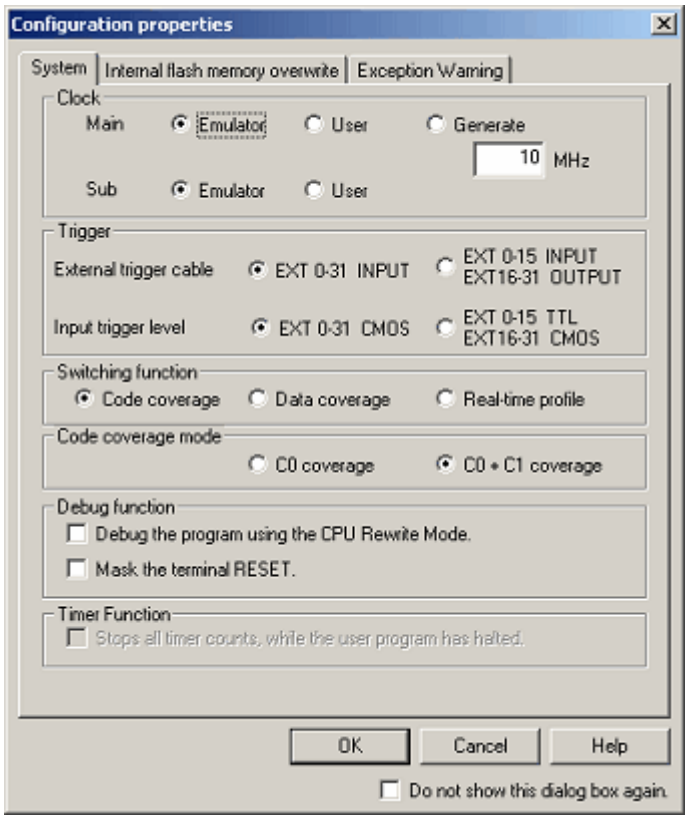


Figure 5.2 Configuration properties dialog box (System page)

(1) Selecting the operating clock

In the Clock section on the System page, select the clocking sources supplied to the main clock and sub-clock.
The main clock can be selected from three choices: Emulator, User and Generate. (By default, Emulator is selected.)
Select Emulator when the main clock is supplied from an internal source or User when the main clock is supplied from an external source. To use a user-defined clock, select Generate and set the clock frequency to be used in the frequency input text box.
The clock frequency can be set in the range 1.0 to 99.9 MHz in 0.1 MHz increments. The clock frequency for Generate can be set only once at startup.

Selection of the sub-clock is displayed only when sub-clocks are supported. It can be selected from Emulator or User. (By default, Emulator is selected.)

CAUTION
The frequency accuracy for Generate is ±5%. Please make sure that final evaluation is performed using a resonator or oscillator module of the frequency used for the actual target board that is mounted on-board.

Downloaded from Arrow.com.

(2) Selecting the direction of external trigger cable

For External trigger cable, select whether EXT pins 16–31 are directed for input or output. EXT pins 0–15 are fixed for input. Select this option from the following:

- EXT 0–31 INPUT (default)
- EXT 0–15 INPUT, EXT 16–31 directed for OUTPUT

The setting of this option is reflected at only startup. If you set this option again in the present dialog box after startup, what you have set has no effect.

(3) Selecting a trigger input level

For Trigger input level, select CMOS level or TTL level. Select this option from the following:

- EXT 0–31 chosen to be CMOS (default)
- EXT 0–15 chosen to be TTL, EXT 16–31 chosen to be CMOS

(4) Selecting a switching function

The code coverage, data coverage and realtime profile functions cannot be used at the same time. Select one function from them.
Initially, code coverage is selected.
The setting of this option can be changed even after startup.
When the code coverage function is selected, measurements are performed at the coverage level selected in Code coverage mode.

(5) Selecting a code coverage mode

Select a code coverage mode.
C0: Instruction coverage rate
C0 + C1: Instruction coverage rate + Branch coverage rate
You can measure up to 2 Mbytes when using the C0 level coverage, and up to 1 Mbytes when using the C0 + C1 level coverage
The initial value is C0 coverage.
The setting of this option is reflected at only startup.
This option is available only when the Code coverage is selected in Switching function.
If you use the code coverage function, select the mode in this option at startup.

(6) Debugging the program using the CPU Rewrite Mode

Select whether you want to debug the program using the CPU Rewrite Mode.

(7) Masking the terminal RESET

Select whether you want the input signal to the RESET pin of the target system to be masked.

Downloaded from Arrow.com.

(8) Stopping all timer counts while the user program is halted

Select whether you want to stop all timer counts while the user program is halted.

CAUTION

The option "Stops all timer counts, while the user program has halted." is inactive (grayed out) with M16C/64 Group, M16C/64A Group, M16C/65 Group, M16C/50 Series and R8C/3x Series MCUs.

Downloaded from Arrow.com.

5.1.4 Creating a Memory Map

On the Memory map page of the Configuration properties dialog box, set an emulation memory allocation. You can specify 4 areas. (In a unit of 4KB)

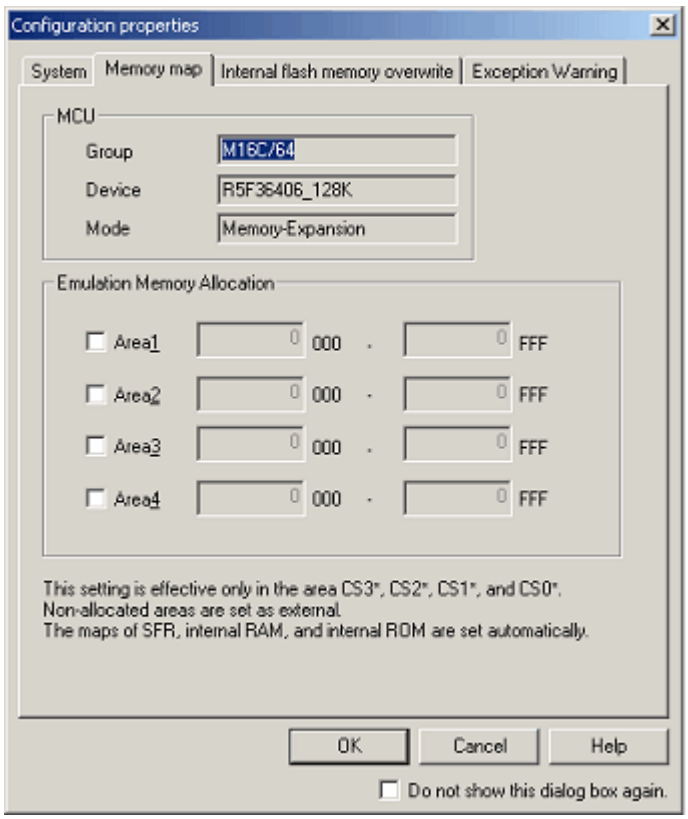


Figure 5.3 Configuration properties dialog box (Memory map page)

The MCU group box displays the device selected in the Device setting dialog box. You cannot change it on this page. The Memory map page does not appear when single-chip mode has been selected.

(1) Allocating emulation memory

You can allocate emulation memory for up to 4 areas.

Select a check box of the area to be used, and enter the start address and end address.

The addresses can be set in a unit of 4KB. Therefore, the low 12 bits of the start address and end address are fixed.

Downloaded from Arrow.com.

5.1.5 Setting Up Flash ROM Overwrite

On the Internal flash memory overwrite page of the Configuration properties dialog box, set up the overwriting of flash ROM blocks, block by block.

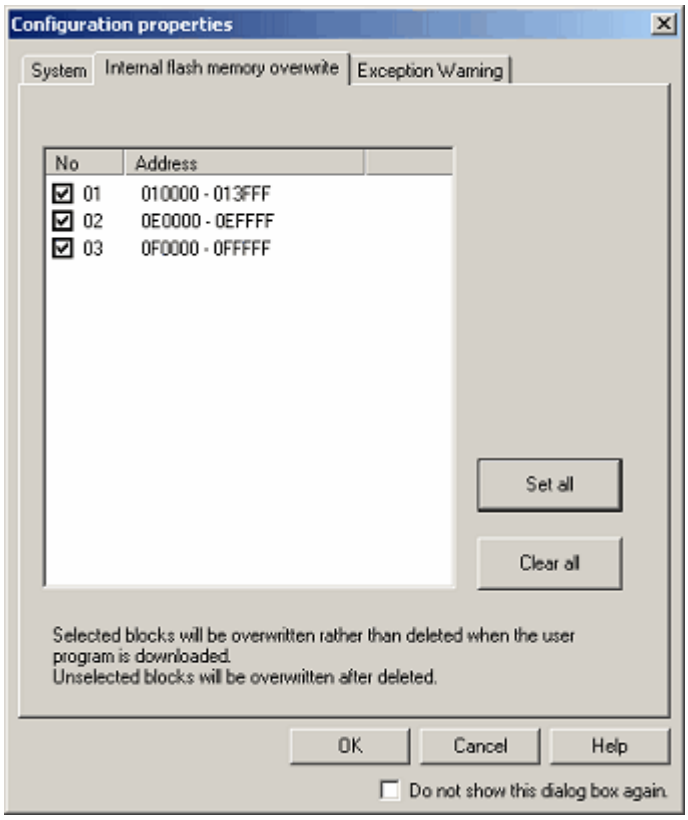


Figure 5.4 Configuration properties dialog box (Internal flash memory overwrite page)

Block-by-block settings matched to the selected target MCU are automatically displayed in the list.
The blocks selected by placing a check mark in the respective check boxes are overwritten (merged), without being erased, when a program is downloaded.

Downloaded from Arrow.com.

5.1.6 Setting the Warning of Exceptional Events

On the Exception Warning page of the Configuration properties dialog box, set whether or not to display warnings of exceptional events in the Status window and status bar balloon.

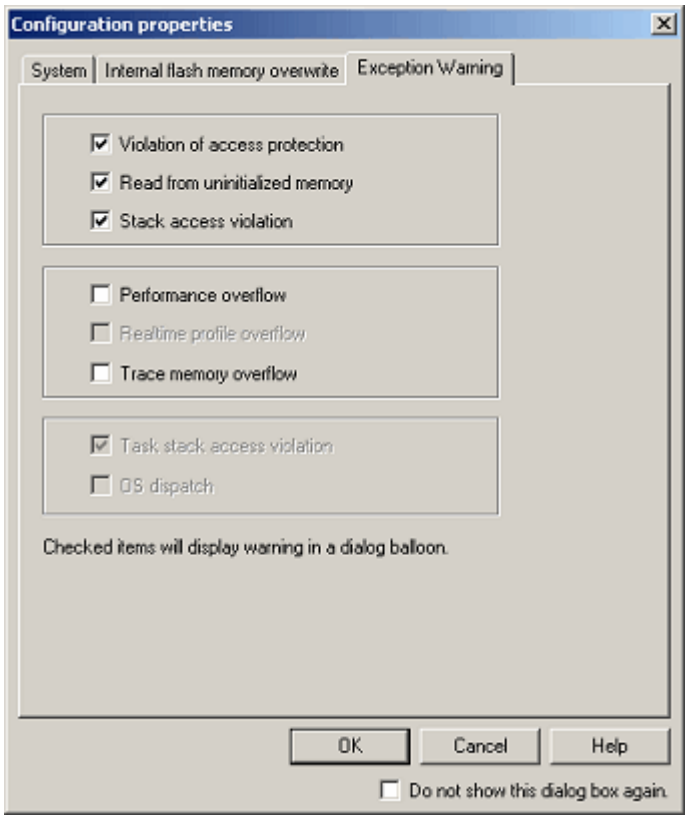


Figure 5.5 Configuration properties dialog box (Exception Warning page)

The 'Violation of access protection' , 'Read from uninitialized memory' and 'Stack access violation' checkboxes are initially selected.
When a load module including OS has been downloaded, the 'Task stack access violation' checkbox is also initially selected.
Other items are not selected.
If you deselect a checkbox, this item will be shown as '-' in the Status window.

Downloaded from Arrow.com.

5.1.7 Showing Progress in Boot-up Processing

You can confirm the progress of boot-up processing by checking the Connecting dialog box.
The Connecting dialog box continues displaying progress information from when boot-up processing starts till when it ends.
While the Device setting and the Configuration properties dialog boxes are displayed, you cannot manipulate this dialog box.

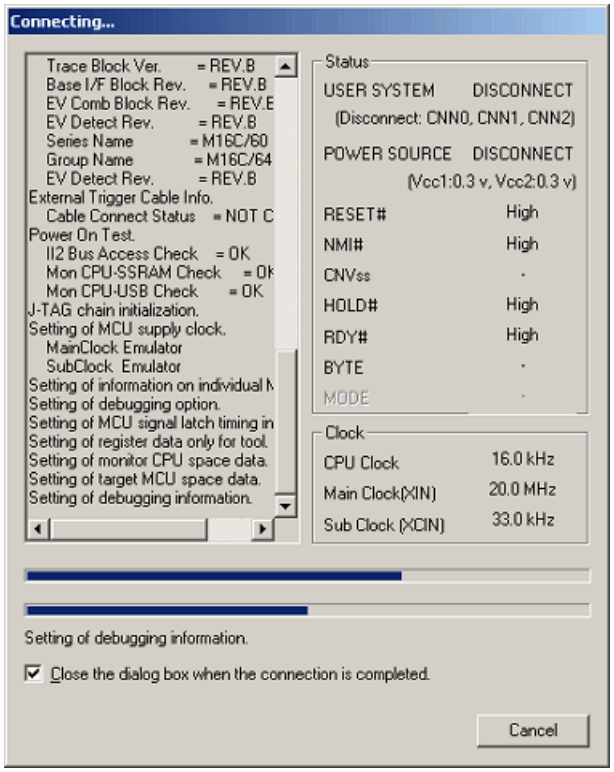


Figure 5.6 Connecting dialog box

(1) Showing the history of processing

The history display area on the left-hand side of the dialog box shows the history of completed processing.
The contents shown here are recorded in a trouble report. To check the content of a trouble report, select Technical Support -> Create Bug Report from the Help menu.

(2) Showing the pin states

The pin states are displayed after the completion of the emulator's startup process.
If the contents set in the Device setting dialog box and the pin states shown here do not match, a warning message is displayed in the history display area.

(3) Showing the clocks

The clocks are displayed after the completion of the emulator's startup process.
Only the actually operating clocks are displayed here.

Downloaded from Arrow.com.

(4) Showing progress with progress bars

The upper progress bar shows the progress of the entire boot-up processing.
The lower progress bar shows the progress of each individual processing.
The content of the currently executed processing is displayed below the bar.

(5) Aborting a connection

Clicking the Cancel button aborts boot-up processing.

Downloaded from Arrow.com.

5.2 Downloading a Program

5.2.1 Downloading a Program

Download the load module to be debugged.

To download a program, choose Download from the Debug menu and select your desired load module from the ensuing list, or right-click a load module in Download modules of the Workspace window and then choose Download from the pop-up or pop-up menu.

CAUTION

Before a program can be downloaded, you must have it registered as a load module in the High-performance Embedded Workshop. For details on how to register, refer to "4.8 Setting Up the Debug".

5.2.2 Showing the Source Code

Follow the procedure described below to show the source code.

- Double-click a source file in the Workspace window.
- Right-click in the source file and choose Open from the pop-up menu.

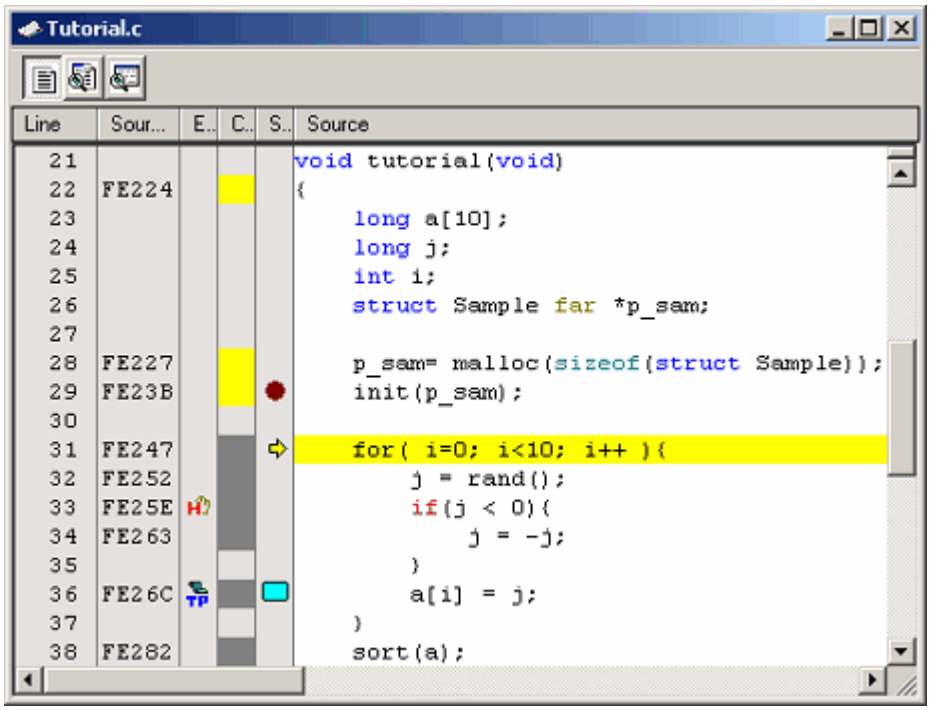


Figure 5.7 Editor window

Shown at left edge of this window are the line information consisting of the following:

Downloaded from Arrow.com.

(1) Line column

Shows the line numbers corresponding to lines in the source file.

(2) Source Address column

When a program is downloaded, this column shows the addresses corresponding to lines in the current source file. This function will prove convenient when you determine where you want the PC value or breakpoint to be set.

(3) Event column

This column shows the following:

Table 5.2 Event column list

	Hardware breakpoint is set
	Trace point (fetch condition) is set

A hardware breakpoint can be inserted by double-clicking in the event column.
Trace points are displayed when fetch conditions are set.
[*] after the title on the title bar of the dialog boxes of Hardware break, Trace conditions or Performance Analysis Conditions shows that some setting is under editing. If you are doing some editing work, you cannot change the settings from the event column of the Editor window.

(4) Code Coverage column

Shows C0 code coverage information graphically.

(5) S/W Breakpoints column

This column shows the following:

Table 5.3 Software breakpoint column list

	Bookmark is set
	Software break is set
	PC position

Downloaded from Arrow.com.

5.2.3 Turning columns in all source files off

(1) From the Editor window

1. Right-click in the Editor window and choose Define Column Format from the pop-up menu.

2. The Global Editor Column States dialog box will be displayed.

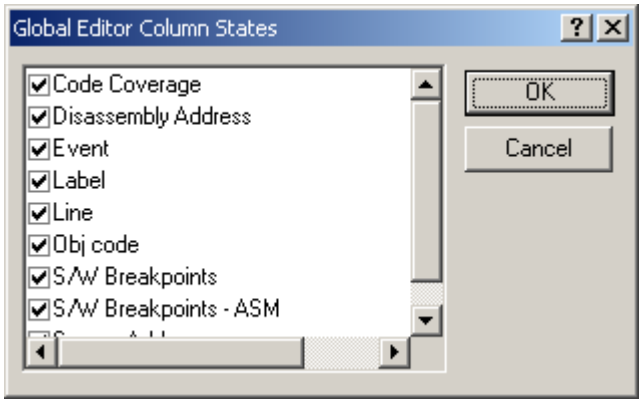


Figure 5.8 Global Editor Column States dialog box

3. Deselect the check box of the column you want to turn off. Click the OK button, and the new column settings you have set will take effect.

5.2.4 Turning columns in one source file off

(1) From the Editor window

1. Right-click in the Editor window and choose Columns from the pop-up menu.

2. Cascaded menu items will be displayed. The currently enabled columns have a check mark attached to the left of the respective names.

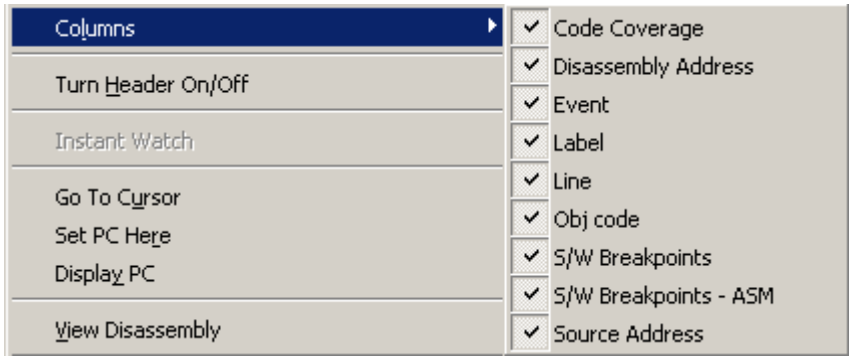


Figure 5.9 Pop-up menu window

3. Clicking a column name lets you enable or disable the column alternately.

Downloaded from Arrow.com.

5.2.5 Showing Assembly Language Code

While a source file is open, click the right mouse button in the Editor window and choose View Disassembly from the pop-up menu. The Disassembly window will be displayed.
The display start address in the Disassembly window is the one that corresponds to the cursor position in the Editor window.
You also can use the Disassembly View button in the Editor window to display disassembled codes.
If no source files exist, one of the following methods may be used to display disassembled codes.

– Click the Disassembly toolbar button [].

– Choose Disassembly from the View menu.
– Use the "Ctrl + D" accelerator.

In this case, the Disassembly window opens at the current PC position.

Mixed mode display where all source lines are displayed beginning with that address is also supported as an option. To display disassembled codes in mixed mode, click the Show in Mixed Mode button.

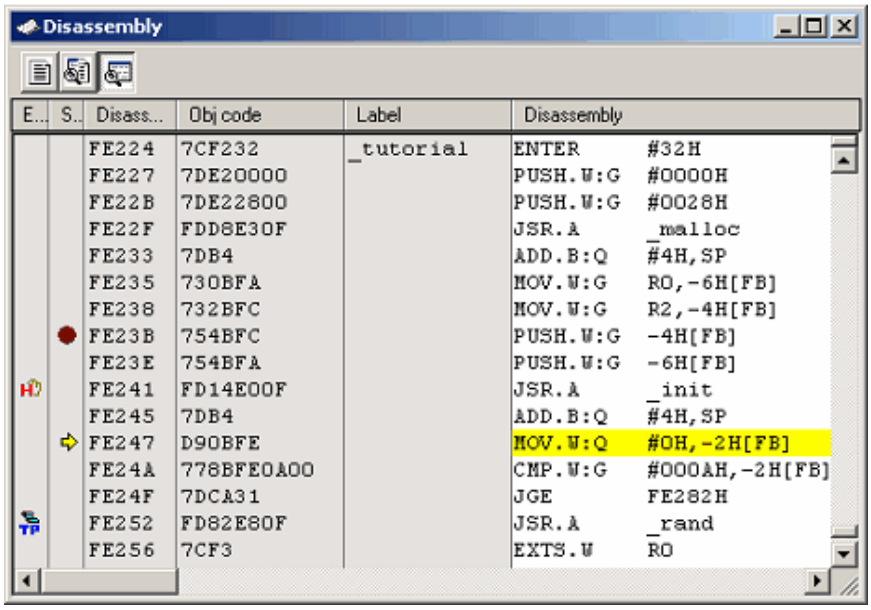


Figure 5.10 Disassembly window

Shown at left edge of this window are the line information consisting of the following:

(1) Event column

This column shows the following:

Table 5.4 Event column list

	Hardware breakpoint is set.
	Trace point (fetch condition) is set.

A hardware breakpoint can be inserted by double-clicking in the event column.
Trace points are displayed when fetch conditions are set.

Downloaded from Arrow.com.

(2) S/W Breakpoints - ASM column

This column shows the following:

Table 5.5 Software breakpoint – ASM column list

	Software break is set.
	PC position

(3) Disassembly Address column

Shows disassembly addresses. Double-clicking here brings up an Address Specification dialog box. In this dialog box, enter the address from which you want a disassembly display to start.

(4) Obj code column

Shows object codes.

(5) Label

Shows a label. This column is unusable unless any module is downloaded.

5.2.6 Correcting Assembly Language Codes

Double-click an instruction you want to correct in the Disassembly window or choose Edit from the pop-up menu, and a dialog box labeled "Assemble" will be displayed. Use this dialog box to correct assembly language.



Figure 5.11 Assembler dialog box

The dialog box shows the address, instruction code and mnemonic of a selected instruction.
Enter a new instruction (or edit the old instruction) in the Mnemonic edit box. When done, hit the Enter key. The memory content will be overwritten with the new instruction code, and the pointer is moved to the next instruction.
Clicking the OK button overwrites the memory content with the new instruction code and closes the dialog box.

CAUTION
Assembly language codes are displayed from the current memory content. When you correct memory contents, new assembly language codes are displayed in the Disassembly window and the Assembler dialog box. However, the source file being displayed in the Editor window remains unchanged. The same applies when the source file includes assembler language.

Downloaded from Arrow.com.

5.3 Displaying Memory Contents in Real Time

5.3.1 Displaying Memory Contents in Real Time

To monitor memory contents while the user program is running, use the RAM Monitor window.
The RAM monitor function permits the memory content and access status in an allocated monitor area to be recorded and inspected in real time without obstructing execution of the user program.
The RAM Monitor window shows access statuses (read, write, non-initialized or uninspected) in different colors.

(1) Allocating a RAM monitor area

A 16-Kbyte RAM monitor area is provided.
This RAM monitor area can be allocated to selected contiguous addresses or divided 32 blocks in 512-byte units.
With initial settings, a maximum 16 Kbytes of area from the beginning address of the internal RAM is allocated as a RAM monitor area.

(2) Monitor display

The access statuses are displayed in different background colors depending on access attributes, as listed below. (The background colors are customizable.)
The read and write accesses show the last accesses made.
Error detection can be displayed by choosing Show Error Detection from the pop-up menu. In this case, the read and write accesses are not displayed.

Table 5.6 Access attribute and background color

	Access attribute		Background color
1	Read access		Green
2	Write access		Red
3	When errors detected	Non-initialized memory (an area not written to has been read)	Yellow
4		Uninspected memory (an initialized area has not been read)	Sky blue
5	No access		White

CAUTION
The contents shown in the RAM Monitor window are the data acquired from bus accesses. Therefore, changes made by accessing memory via other than the user program, as when memory is rewritten directly from external I/O, are not reflected in the displayed memory content.

Downloaded from Arrow.com.

(3) Detecting reading from non-initialized areas

If a memory area that has not been written to is read, the emulator detects "reading from a non-initialized area" and outputs an error.
To view errors of this type, choose Error Detection Display from the pop-up menu.
Non-initialized memory areas are shown in yellow.
This error detection can be an exceptional event used as a condition of a hardware breakpoint or trace point (also refer to "Detecting exceptional events").

(4) Detecting uninspected areas

If an initialized memory area has not been read, the emulator detects "an uninspected area" and outputs an error.
To view errors of this type, choose Error Detection Display from the pop-up menu.
Uninspected memory areas are shown in sky blue.

5.3.2 Setting RAM Monitor Update Intervals

Choose Update Interval Setting from the pop-up menu of the RAM Monitor window. The Update Interval Setting dialog box shown below will appear.

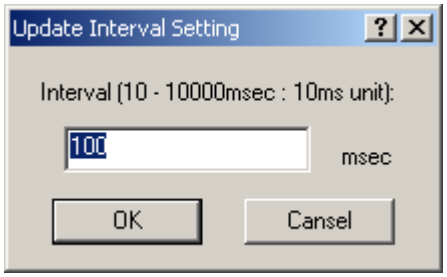


Figure 5.12 Update Interval Setting dialog box

The Update Interval can be specified separately for each window.
The initial value is 100 ms.

5.3.3 Clearing RAM Monitor Access History

Choose Access Data Clear from the pop-up menu of the RAM Monitor window. The history of all accesses made to the RAM monitor area will be cleared.

CAUTION
If this function is executed while the user program is running, the user program's realtime capability may be lost because a memory dump occurs.

5.3.4 Clearing RAM Monitor Error Detection Data

Choose Error Detection Data Clear from the pop-up menu of the RAM Monitor window. The detected data of all uninitialized memory and unreferred memory of the RAM monitor area will be cleared.

Downloaded from Arrow.com.

5.4 Showing the Current Status

5.4.1 Showing the Emulator Status

To know the current status of the emulator, display the Status window.

To open the Status window, choose CPU -> Status from the View menu, or click the View Status toolbar button [].

This window does not update the displayed status during program execution.

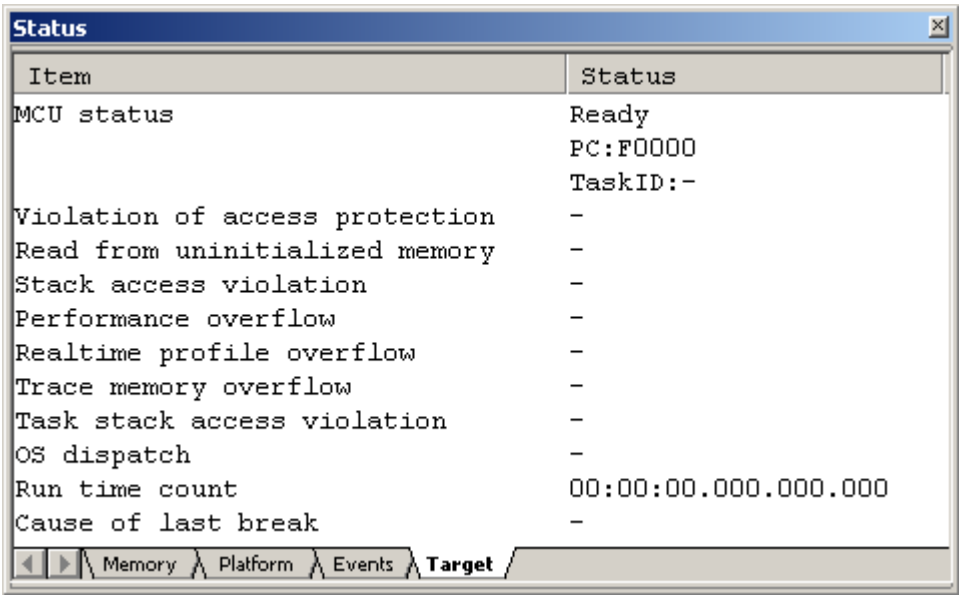


Figure 5.13 Status window

The Status window has the following four sheets.

Table 5.7 sheet list of status window

Sheet name	Description
Memory	Shows information relating to memory resources.
Platform	Shows information relating to the emulator and debugging.
Events	Shows information relating to events.
Target	Shows information relating to the target MCU.

Downloaded from Arrow.com.

5.4.2 Showing the Emulator Status in the Status Bar

The status of the emulator can be displayed in the status bar.

By right clicking on the status bar, the items are shown. Check the items you want to show in the status bar.

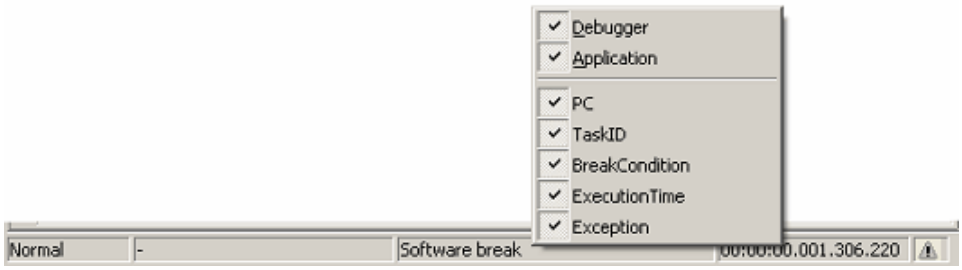


Figure 5.14 Status bar

Table 5.8 Items list of the emulator status shown in the status bar

Item	Description
PC	PC value During execution: PC value During Break: Normal
Task ID	Task ID, task entry label
BreakCondition	Break factors of the user program
ExecutionTime	Result of time measurement
Exception	Status of the exceptional event

(1) When more than one break factors occur.

When you click on the status bar area where the break factor is shown, a balloon is shown.

You can check the break factors being occurred in the balloon.

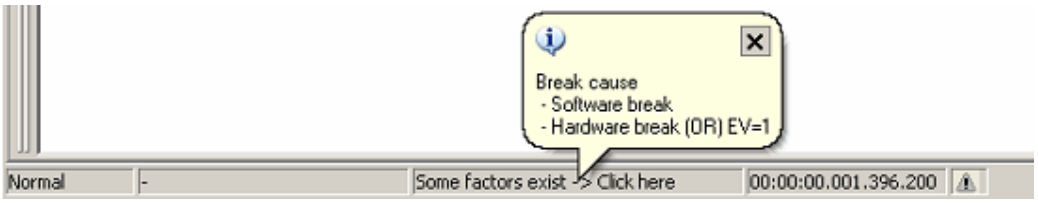


Figure 5.15 Example of break factors display when break factors occur

(2) When an exceptional event occurs.

When an exceptional event occurs, a warning is displayed in a status bar balloon.

The exceptional event that is not checked on the Exception Warning page of the Configuration Properties dialog box is not shown.

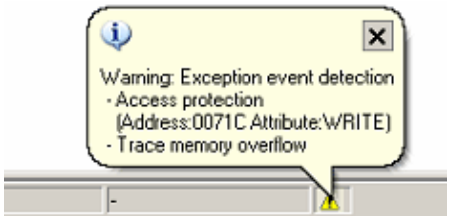


Figure 5.16 Example of warning display when exceptional events occur

Downloaded from Arrow.com.

5.5 Periodically Reading Out and Showing the Emulator Status

5.5.1 Periodically Reading Out and Showing the Emulator Information

To know the changing emulator information whether the user program is running or remains idle, use the Extended Monitor window.

The extended monitor function only monitors the signals output from the user system or MCU, and does not affect execution of the user program.

To open the Extended Monitor window, choose CPU -> Extended Monitor from the View menu, or click the Extended Monitor toolbar button [].

The displayed items are updated at an interval of about 1,000 ms during user program execution or about 5,000 ms during a break.

CAUTION

(1) "CPU Clock" can be measured only when a user program is being executed.

(2) The display contents of the status differ depending on products.

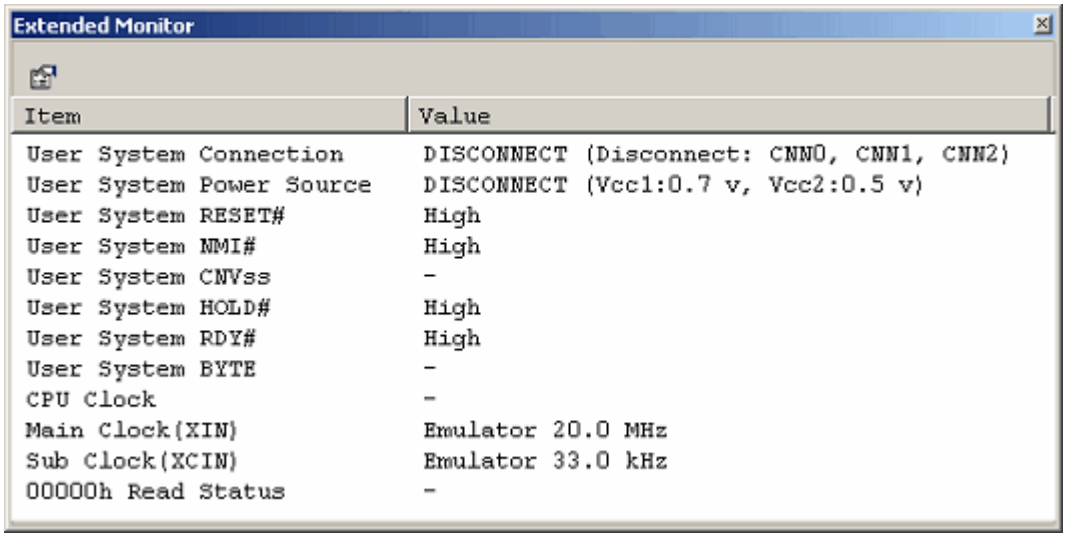


Figure 5.17 Extended Monitor window

Downloaded from Arrow.com.

5.5.2 Selecting the Items to Be Displayed

Choose Properties from the pop-up menu of the Extended Monitor window, and the Extended Monitor Configuration dialog box will be displayed.

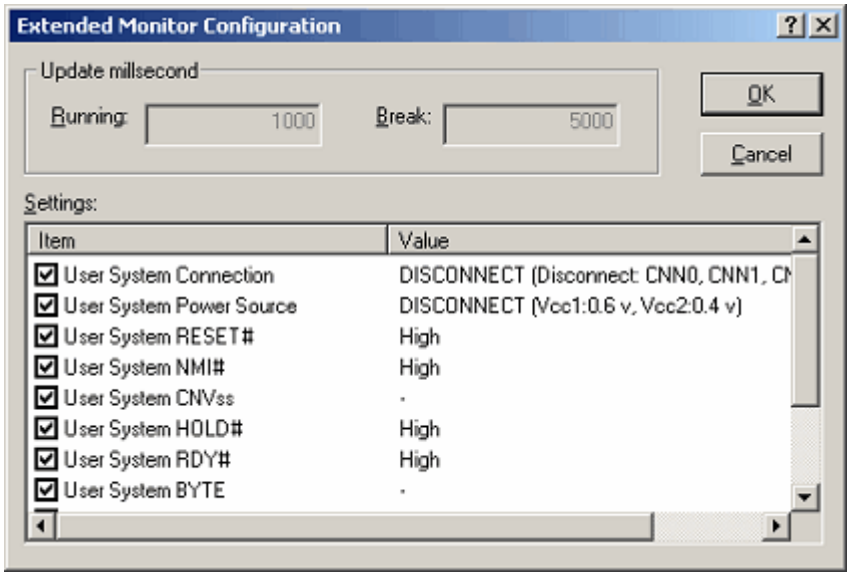


Figure 5.18 Extended Monitor Configuration dialog box

This dialog box permits you to set each item you want to be displayed in the Extended Monitor window.

Downloaded from Arrow.com.

5.6 Using Software Breakpoints

5.6.1 Using Software Breakpoints

A software break causes the user program to stop running by rewriting the instruction code at a specified address with a BRK instruction to generate a BRK interrupt. In that sense, this is a pre-execution break function.
4096 breakpoints can be set.
If multiple software breakpoints are set, the program breaks at any one of those breakpoints reached.

(1) When stopped at a software breakpoint

When the program you have created is run and the address you have set as a software breakpoint is reached, a message "Software Break" is displayed on the Debug sheet of the Output window, with the program made to stop there. At this time, the Editor or the Disassembly window is updated, and the position at which the program has stopped is marked with an arrow [⇨] in the S/W Breakpoints column.

CAUTION
When a break occurs, the program stops immediately before executing the line or instruction at which a software breakpoint is set. If Go or Step is selected after the program has stopped at that software breakpoint, the program restarts from the line marked with an arrow.

Downloaded from Arrow.com.

5.6.2 Adding/Removing Software Breakpoints

Follow one of the following methods to add or remove software breakpoints.

- From the Editor or the Disassembly window
- From the Breakpoints dialog box (only removing)
- From the command line

(1) From the Editor or the Disassembly window

1. Check to see that the Editor or the Disassembly window that is currently open includes the position at which you want to set a software breakpoint.

2. In the S/W Breakpoints column, double-click the line where you want the program to stop.

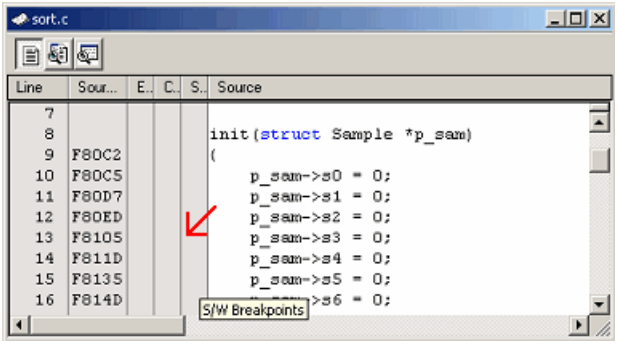


Figure 5.19 Editor window

Or you use the method described below to set a breakpoint. Select Toggle Breakpoint from the pop-up menu or press the F9 key on the keyboard.

3. When a software breakpoint is set, a red circle [●] is displayed at the corresponding position in the S/W Breakpoints column of the Editor or the Disassembly window.

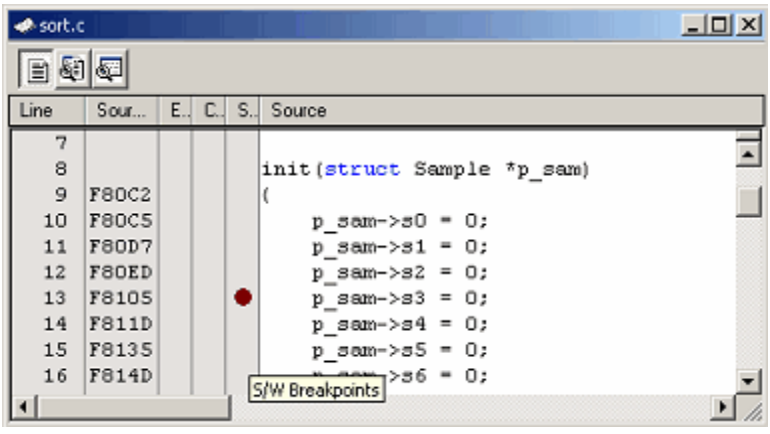


Figure 5.20 Editor window

Double-clicking one more time removes the breakpoint.

Downloaded from Arrow.com.

5.6.3 Enabling/Disabling Software Breakpoints

Follow one of the following methods to enable or disable software breakpoints.

- From the Editor or the Disassembly window
- From the Breakpoints dialog box
- From the command line

(1) From the Editor or the Disassembly window

1. Place the cursor at the line where a software breakpoint exists and then select Enable/Disable Breakpoint from the pop-up menu. Or press the Ctrl and F9 keys together.

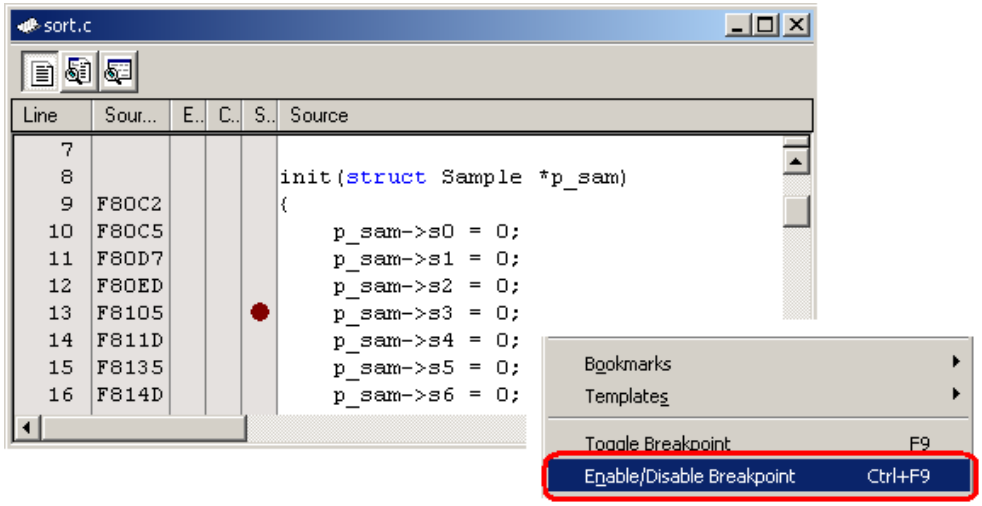


Figure 5.21 Editor window and pop-up menu

2. The software breakpoint is enabled or disabled alternately.

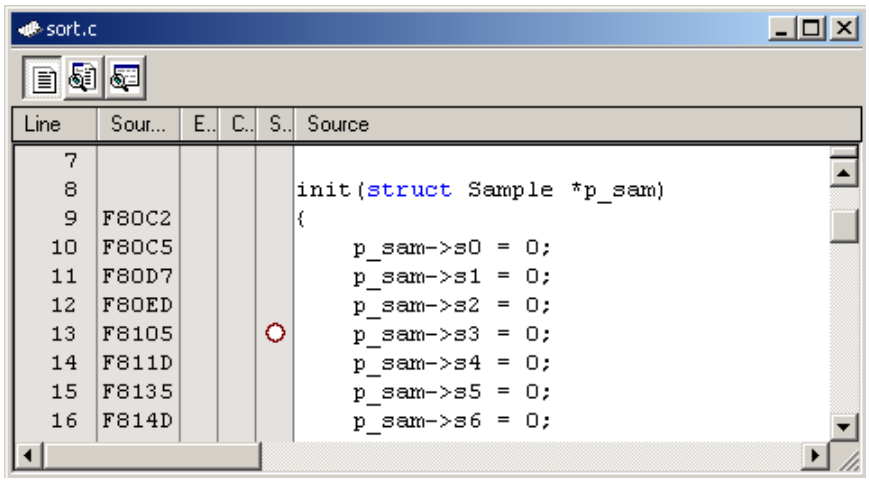


Figure 5.22 Editor window

Downloaded from Arrow.com.

(2) From the Breakpoints dialog box

1. Select Source Breakpoints from the Edit menu to bring up the Breakpoints dialog box. In this dialog box, you can alternately enable or disable a currently set breakpoint, as well as remove it.

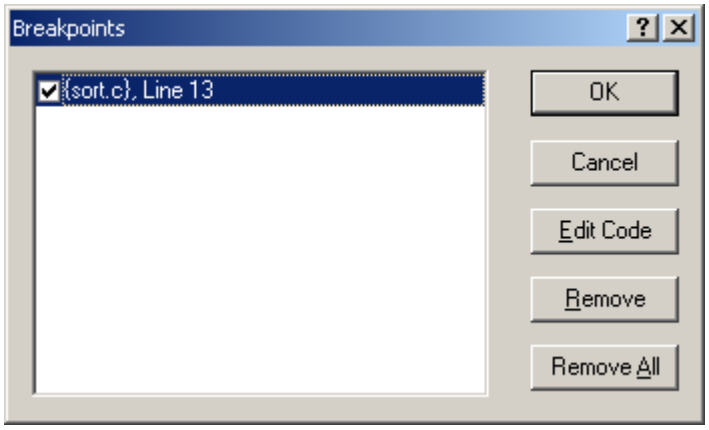


Figure 5.23 Breakpoints dialog box

Downloaded from Arrow.com.

5.7 Using Events

5.7.1 Using Events

An event refers to a combination of phenomena that occur during program execution.
The E100 emulator permits you to use the event you have set as a condition of the break, trace or performance function.
Events can be set at up to 16 points at the same time.
These 16 points can be located at any desired positions.
The events you have created can be registered for reuse at a later time.

(1) Types of events

There are following types of events.

Table 5.9 Event types list

Instruction fetch	An event is detected when an instruction at the specified address is executed by CPU. An event is detected not in the cycles prefetched by an instruction queue but in the cycles executed by the CPU.
Data access	An event is detected when a specified address or specified address range is accessed under a specified condition.
Interrupt	Detection is made of an interrupt generation and interrupt termination.
Trigger input	An event is detected when the signal fed in from external trigger signal input cable is in a specified state.

(2) Event combination

One of the following combinatorial conditions can be specified using two or more events in combination.

Table 5.10 Event combinations list

OR	Condition is met when any one of the specified events occurs.
AND (Accumulation)	Condition is met when all of the specified events occur irrespective of the time axis.
AND (Simultaneous)	Condition is met when all of the specified events occur at the same time.
Subroutine	Condition is met when a specified event occurs within a specified address range.
Sequential	Condition is met when a specified event occurs in a specified order.
State transitions	Condition is met when an event occurs under the condition specified in a state transition diagram.

Downloaded from Arrow.com.

5.7.2 Adding Events

Follow one of the following methods to add events.

- Create a new event
- Add by dragging and dropping from another window
- Add from the command line

(1) Creating a new event

[When creating an event from any setup dialog box]

1. Click the Add button or choose a line where you want to input and double-click.

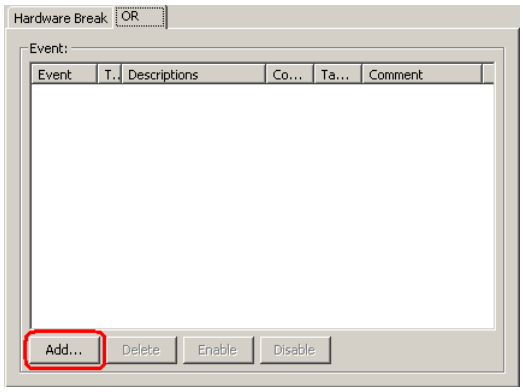


Figure 5.24 Hardware Break dialog box

2. The Event dialog box shown below will be displayed. In this dialog box, set detail event conditions and then click the OK button.

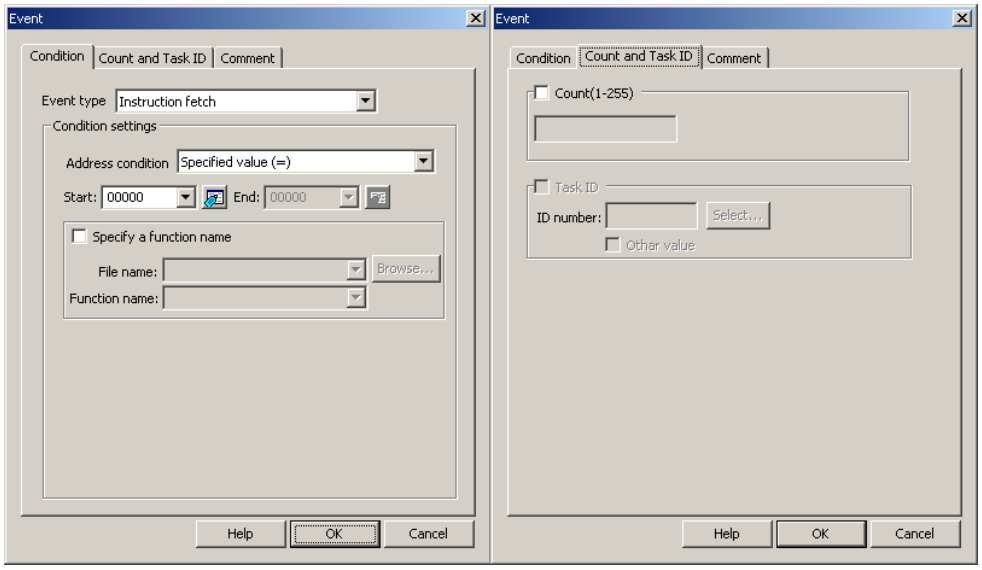


Figure 5.25 Event dialog box

Downloaded from Arrow.com.

3. An event will be added at the specified position.

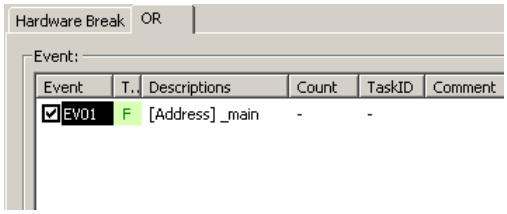


Figure 5.26 Hardware Break dialog box

4. If events exceed 16 points when you created an event, an error is displayed. If you created an event exceeding 16 points, the event you have added has no effect.

[When adding an event from the Registered Events dialog box]

1. Click the Add button in the Registered Events dialog box.

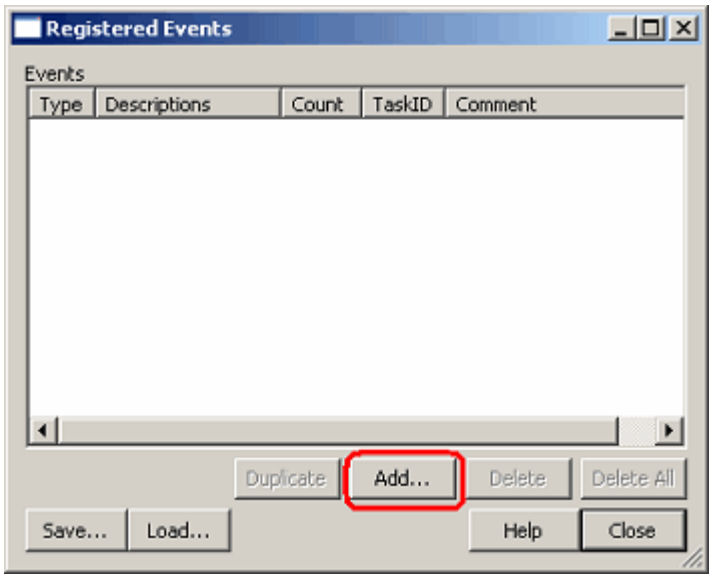


Figure 5.27 Registered Events dialog box

Downloaded from Arrow.com.

2. The Event dialog box shown below will be displayed. In this dialog box, set detail event conditions. Enter a comment if any necessary. Then click the OK button.

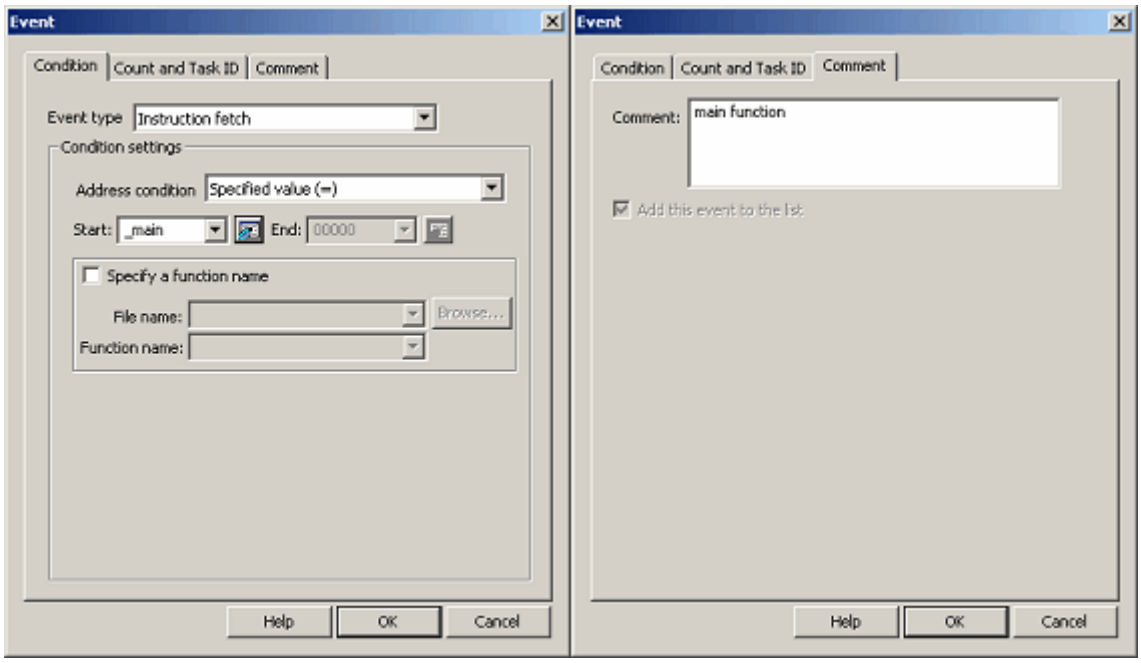


Figure 5.28 Event dialog box

3. An event will be added to the list of registered events.

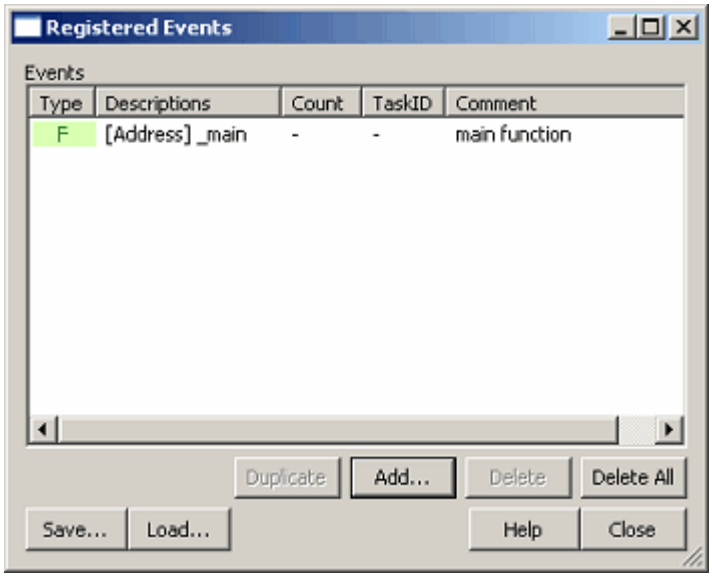


Figure 5.29 Registered Events dialog box

Downloaded from Arrow.com.

(2) Adding an event from the event column of the Editor window

[When adding a hardware breakpoint]

1. Select the HW Break Point from the pop-up menu displayed by double-clicking or right clicking anywhere in the event column of the Editor window.
   You can set a hardware breakpoint based on a fetch to that address as a condition. => Instruction fetch condition

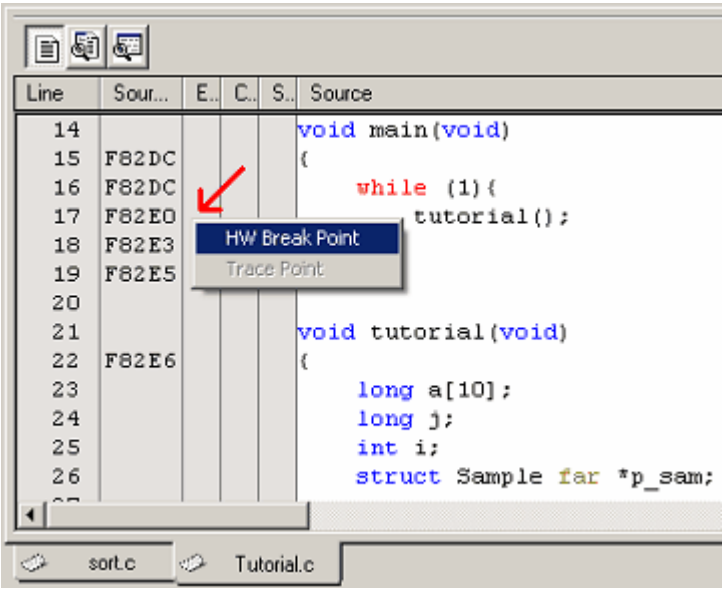


Figure 5.30 Editor window

2. If there is room for event counts, the event you have added from the Editor window is added to the other events as an OR condition. If there is no room, an error message is displayed.

CAUTION
If you are doing some editing work in the Hardware Break dialog box, you cannot set hardware breaks from the event column of the Editor window.

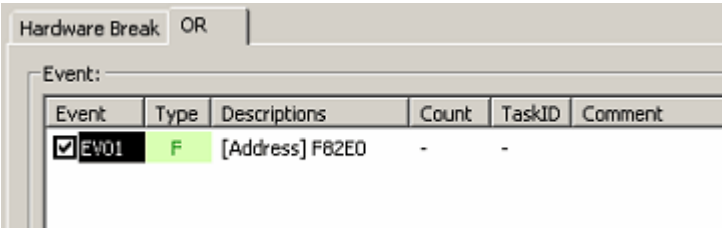


Figure 5.31 Hardware Break dialog box

Downloaded from Arrow.com.

[When adding a trace point]

1. Select the Trace Point from the pop-up menu displayed by double-clicking or right clicking anywhere in the event column of the Editor window.
   You can set a trace point based on a fetch to that address as a condition. => Instruction fetch condition

   Double-click the instruction fetch event in the Event column of the Editor window to delete it.

CAUTION

No trace points can be set from the event column of the Editor window in the following cases.

- While editing the contents in the Trace conditions dialog box
- When selecting the Fill until stop or Fill until full of the trace mode

(3) Adding events by dragging and dropping

[When dragging and dropping the variable and function names in the Editor window]

1. Dragging and dropping a variable name into the Event column, you can set an event based on an access to that variable as a condition. => Data access condition
   At this time, the size of the variable is automatically set to be a condition of a data access event.
   Only global or static variables of 1 or 2 bytes in size can be registered as an event. Static variables in functions cannot be registered as an event.
2. Dragging and dropping a function name into the Event column, you can set an event based on an instruction fetch to the start address of that function as a condition.

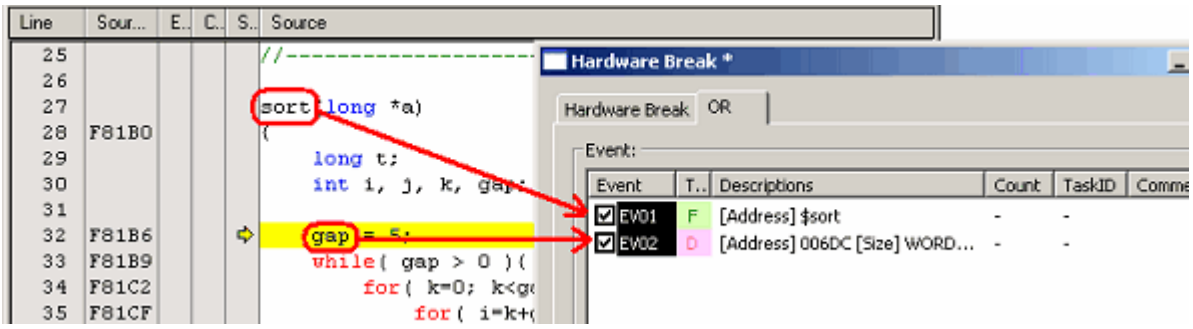


Figure 5.32 Editor window and Hardware Break dialog box

[When dragging and dropping the address range in the Memory window]

Select memory content in the Memory window and drag and drop it into the Event column. That way, you can set a data access event based on the address range of the selected memory content as a condition. => Data access condition

[When dragging and dropping the label in the Label window]

You can set an event based on a fetch to that label as a condition. => Instruction fetch condition

Downloaded from Arrow.com.

5.7.3 Removing Events

Follow one of the following methods to remove events.

[When deleting an event from any setting dialog box]

1. To remove one point, select a line you want to remove in the event setting area and then click the Delete button (You can use the keys Ctrl + Del instead of clicking the Delete button).
   The selected event will be removed from the event setting area.

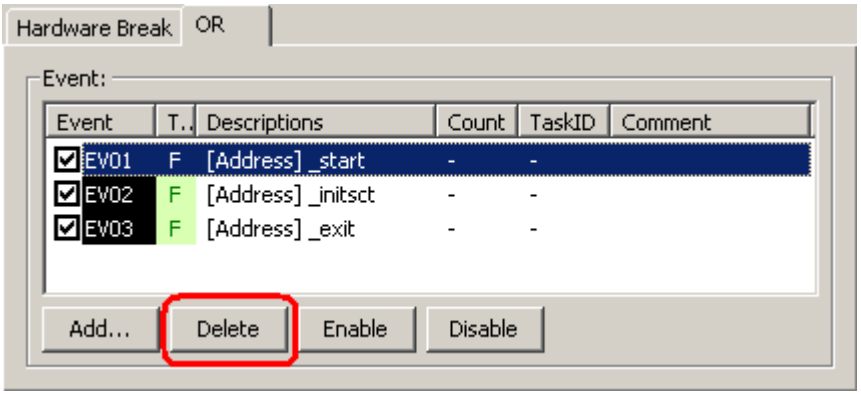


Figure 5.33 Hardware Break dialog box

2. To remove multiple events, hold down the Shift or the Ctrl key while you select lines you want to remove in the event setting area and then click the Delete button (You can use the keys Ctrl + Del instead of clicking the Delete button).
   The selected events will be removed from the event setting area.

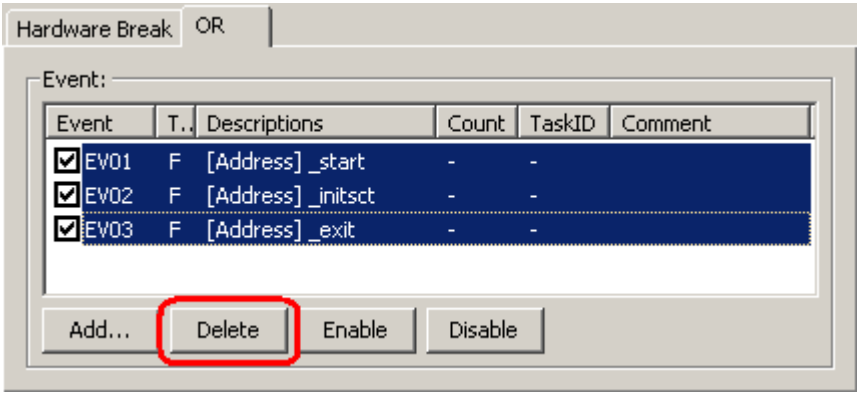


Figure 5.34 Hardware Break dialog box

Downloaded from Arrow.com.

[When deleting an event from the Registered Events dialog box]

To remove one point, select a line you want to remove in the Registered Events dialog box and then click the Delete button (You can use the keys Ctrl + Del instead of clicking the Delete button).
The selected event will be removed from the list of registered events.
To delete all events, click the Delete All button.

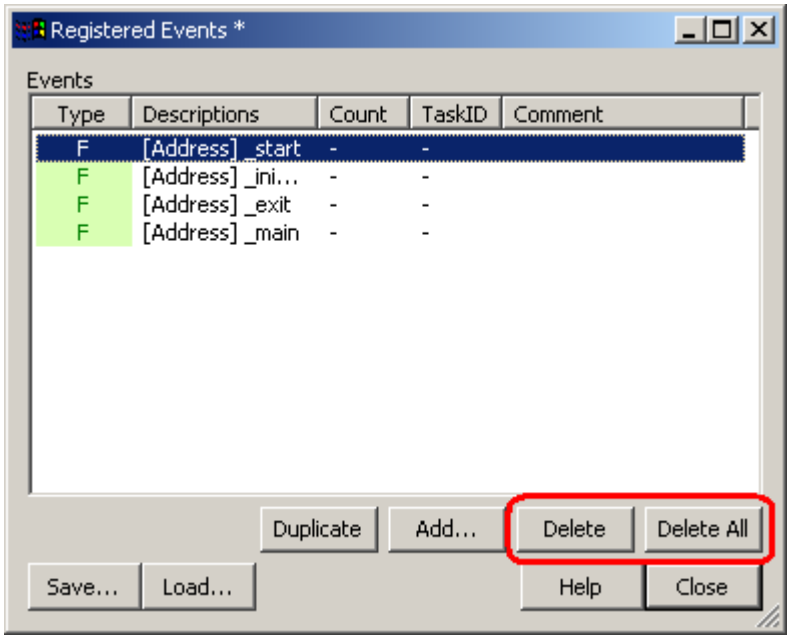


Figure 5.35 Registered Events dialog box

Downloaded from Arrow.com.

5.7.4 Registering Events

"Registering an event" refers to placing an event into the list of registered events. A registered event can be reused at a later time. Follow one of the following methods to register an event. Up to 256 events can be registered.

(1) Registering events

[When creating an event from the Event dialog box]

1. Display the Comments page of the Event dialog box and select the "Add this event to the list" check box. Then click the OK button.

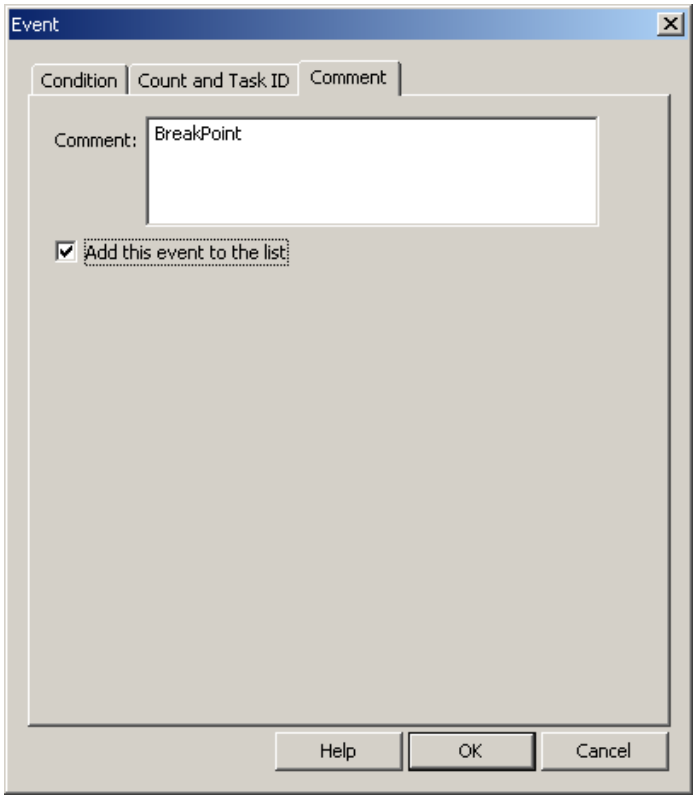


Figure 5.36 Event dialog box

2. An event is added at the specified position while at the same time registered in the Registered Events.

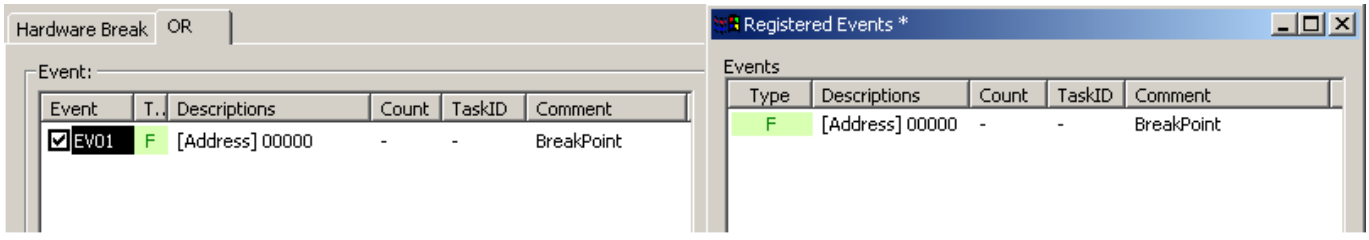


Figure 5.37 Hardware Break dialog box and Registered Events dialog box

Downloaded from Arrow.com.

[When registering an event by dragging and dropping]

The event you have created can be registered in the Registered Events by dragging and dropping it into the list.

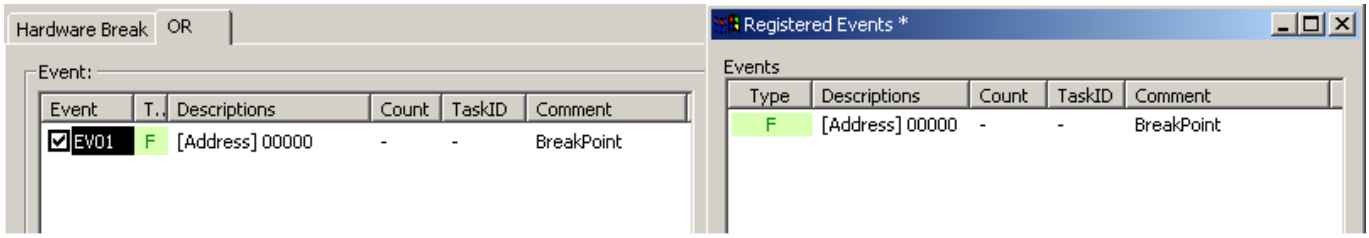


Figure 5.38 Hardware Break dialog box and Registered Events dialog box

[When registering an event from the Registered Events dialog box]

Click the Add button to create an event. The events you create here are added to the Registered Events.

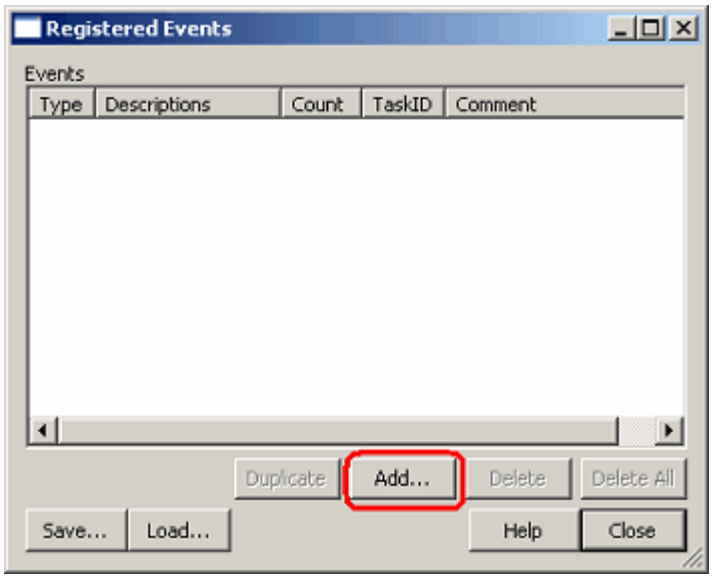


Figure 5.39 Registered Events dialog box

(2) Attaching comments

Attach a comment to the registered event as necessary. Check the Registered Events dialog box to know the registered contents and comments.

Downloaded from Arrow.com.

5.7.5 Entering Events Each Time or Reusing Events

There are following two methods to set events in any function concerned.
One method is to create events in the respective setting dialog boxes each time.
The other method is to choose one condition you want to use from the registered event list and drag and drop it into the condition area in which you want to set the event.
The former method is referred to here as entering events each time, and the latter as reusing events.

[Entering events each time]

This is the condition used only once. The event you created is used "without ever being registered."
After the event is used (i.e., changed or removed), its setting becomes nonexistent.
The events you create by only double-clicking in the Event column of the Editor window are the one that is entered each time.

[Reusing events]

Any event registered in the Registered Events dialog box can be reused by dragging and dropping it into the condition setting area of any function concerned.

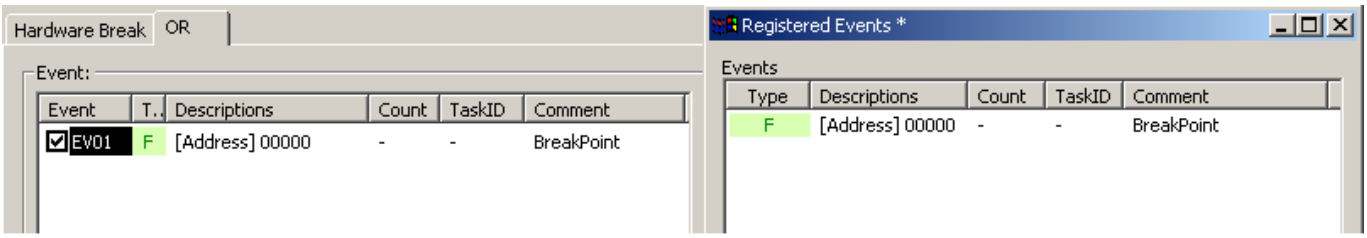


Figure 5.40 Schematic of event reuse

(1) Dragging and dropping into multiple functions

One event in the Registered Events can be dragged and dropped into multiple functions.
If the content of an event is altered after being dragged and dropped, the alteration you made is not reflected on the registered event list side.

(2) Registering duplicates in the registered event list

Even the events that have the same contents set can be registered in the list overlapping one another.

Downloaded from Arrow.com.

5.7.6 Applying Events

To enable the setting of an event after you have created it, click the Apply button. The content of what you have set has no effect until you click the Apply button.

[*] after the title on the title bar of the dialog boxes of Hardware break, Trace conditions or Performance Analysis Conditions shows that some setting is under editing. If you are doing some editing work, you cannot change the settings from the event column of the Editor window or the command line.

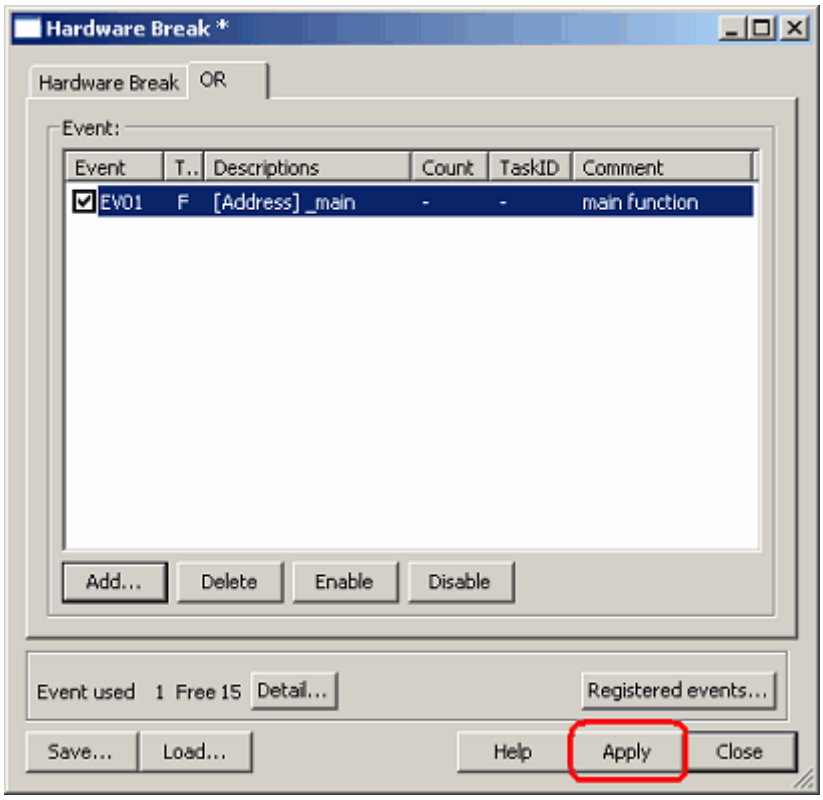


Figure 5.41 Applying the setting

Downloaded from Arrow.com.

5.8 Setting Hardware Break Conditions

5.8.1 Setting Hardware Break Conditions

A hardware break causes the user program to stop running a specified number of cycles after a set event or phenomenon is detected (i.e., a hardware breakpoint is encountered). Up to 16 events can be specified as hardware breakpoint conditions.

5.8.2 Setting Hardware Breakpoints

(1) Setting Hardware Breakpoints

For hardware breakpoints, you can set an OR condition, other condition (AND (Accumulation), AND (Simultaneous), subroutine, sequential or state transition) and detection of exceptional events.
The OR condition, other condition and the detection of exceptional events can be set all at the same time, or only one at a time.

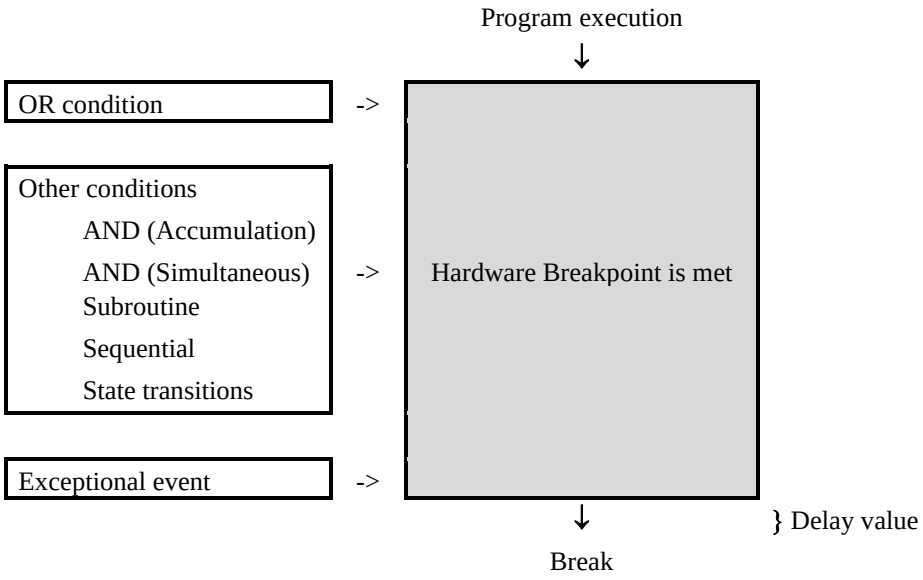


Figure 5.42 Outline of the hardware break

Downloaded from Arrow.com.

(2) Setting OR conditions

You can choose to enable or disable the OR condition. By default, the OR condition is enabled.
To disable the OR condition, deselect the check box to the left of "OR Condition."
If you add an event by double-clicking in the Editor window while the OR condition is disabled, the OR condition is automatically enabled.
If you reenable the OR condition when it is disabled, the previously set event is restored with its OR condition check box selected.
However, if a maximum of 16 events is exceeded when you have reenabled for an event, the event is restored with its OR condition check box unselected (disabled).

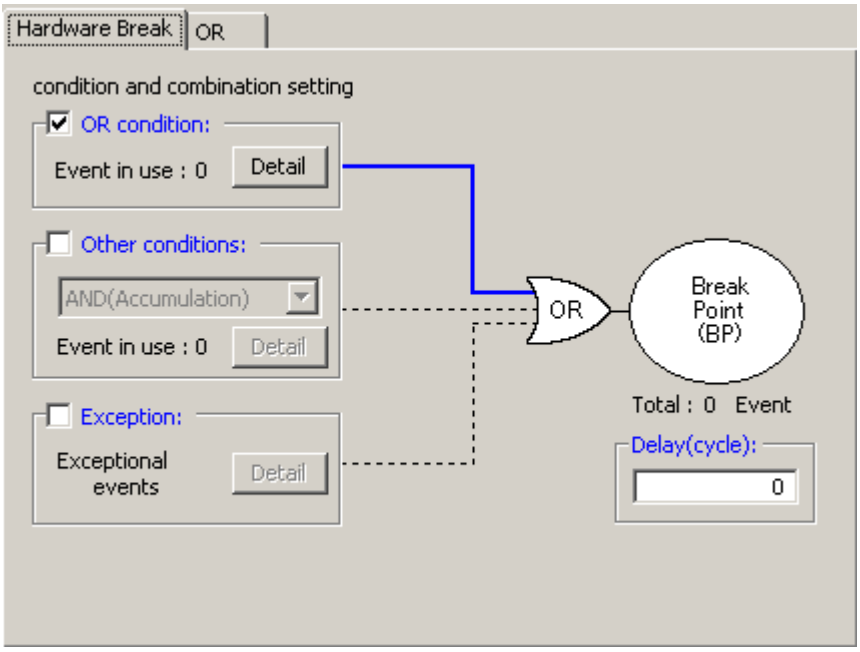


Figure 5.43 Hardware Break dialog box

Table 5.11 OR condition

	Type	Description
1	OR condition	Breakpoint is encountered when any one of the set events occurs.

Downloaded from Arrow.com.

(3) Setting other conditions

You can select one of five choices available: AND (Accumulation), AND (Simultaneous), Subroutine, Sequential and State Transition. To set any condition, select the check box to the left of "Other Conditions." By default, other conditions are disabled (the check box to the left of "Other Conditions" is unselected).

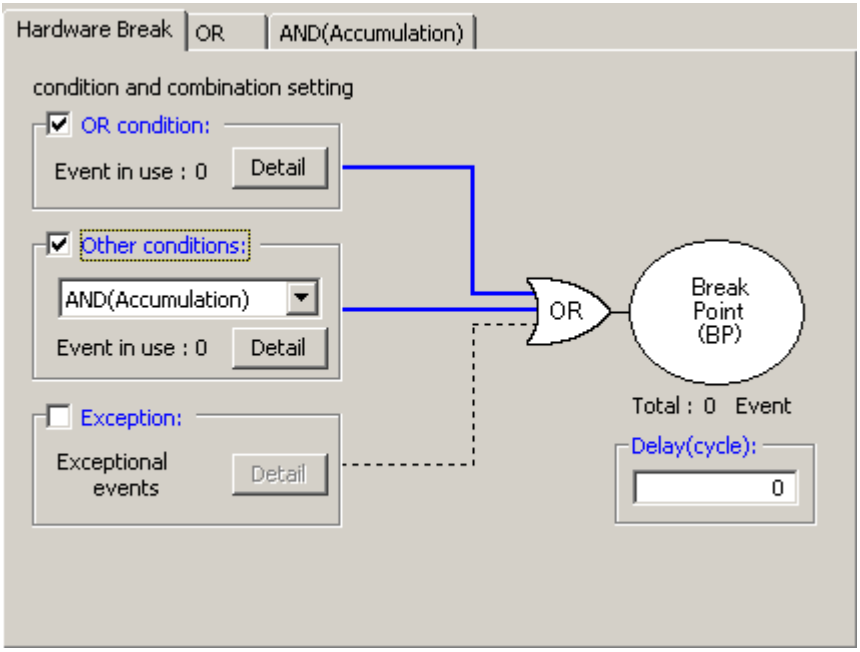


Figure 5.44 Hardware Break dialog box

Table 5.12 Other conditions

	Type	Description
1	AND (Accumulation)	Breakpoint is encountered when all of the set events occur irrespective of the time axis.
2	AND (Simultaneous)	Breakpoint is encountered when all of the set events occur at the same time.
3	Subroutine	Breakpoint is encountered when a specified event occurs within a specified address range (Subroutine, function).
4	Sequential	6 steps, (forward direction) + reset point Breakpoint is encountered when a set event occurs in a specified order.
5	State transitions	3 steps, 9 paths + reset point Breakpoint is encountered when a set event occurs in a specified order.

The events shown in the list of each condition can be deleted by the keys Ctrl + Del.

CAUTION

When a time-out condition is set in State transitions (Hardware break point) dialog box, the time to make transition from a set state to another then back to the original set state must be 10 µs or more. Transition time of less than 10 µs will result in an incorrect timeout detection.

Downloaded from Arrow.com.

(4) Detection of exceptional events

Specify whether you want detection of following exceptional events to be used as a breakpoint.

- Violation of access protection
- Read from uninitialized memory
- Stack access violation
- Performance overflow
- Realtime profile overflow
- Trace memory overflow
- Task stack access violation
- OS dispatch

(5) Specifying a delay value

The program breaks a specified number of cycles after a breakpoint is encountered. A breakpoint delay value can be set in the range from 0 to 65,535 bus cycles (default = 0).

5.8.3 Saving/Loading the Set Contents of Hardware Breaks

(1) Saving hardware break settings

Click the Save button of the Hardware Break dialog box. The Save dialog box will be displayed.
Specify a file name to which you want break settings to be saved. The file name extension is .hev. If omitted, the extension .hev is automatically attached.

(2) Loading the set contents of hardware breaks

Click the Load button of the Hardware Break dialog box. The Load dialog box will be displayed. Specify the file name you want to load.
When you load a file, the hardware break settings you had before you have loaded the file are discarded and the hardware breaks are reset with the loaded settings.
Click the Apply button of the Hardware Break dialog box to confirm the hardware break settings you have loaded.

Downloaded from Arrow.com.

5.9 Looking at Trace Information

5.9.1 Looking at Trace Information

A trace is the function to acquire bus information every cycle and store it in trace memory during user program execution. Using a trace you can track the flow of application execution or examine the points at which problems occurred.

The E100 emulator allows you to acquire up to 4M bus cycles.

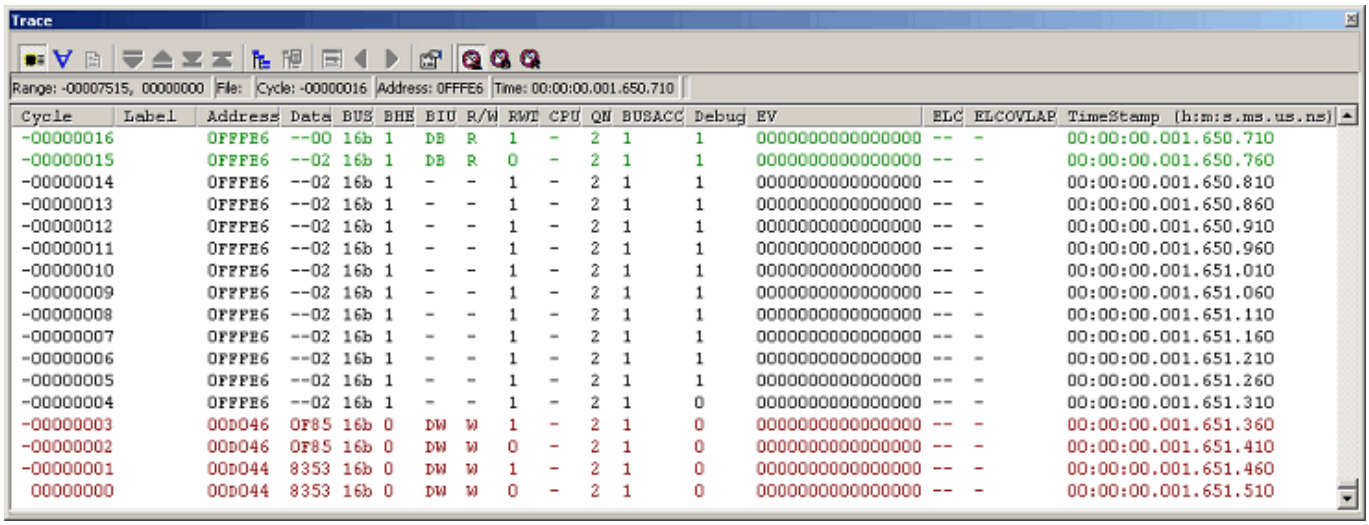
When program execution stops (for an exception break, forced halt or breakpoint), the contents stored in trace memory at the time the program has stopped are displayed as the trace result even when no trace points are encountered yet.

5.9.2 Acquiring Trace Information

The E100 emulator operates in such a way that when no trace information acquisition conditions are set, it by default traces all bus cycles to get trace information unconditionally. (Trace mode = Fill until stop)

In free mode, at the same time the user program starts running the emulator starts tracing bus cycles to get trace information, and when the user program stops the emulator stops tracing.

The acquired trace information is displayed in the Trace window.



Cycle	Label	Address	Data	BUS	BHE	BIU	R/W	RWT	CPU	QN	BUSACC	Debug	EV	ELC	ELCOVLAP	TimeStamp [h:m:s.ms.us.ns]
-00000016		0FFFE6	--00	16b	1	DB	R	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.710
-00000015		0FFFE6	--02	16b	1	DB	R	0	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.760
-00000014		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.810
-00000013		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.860
-00000012		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.910
-00000011		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.960
-00000010		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.010
-00000009		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.060
-00000008		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.110
-00000007		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.160
-00000006		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.210
-00000005		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.260
-00000004		0FFFE6	--02	16b	1	-	-	1	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.310
-00000003		00D046	0F85	16b	0	DW	W	1	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.360
-00000002		00D046	0F85	16b	0	DW	W	0	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.410
-00000001		00D044	8353	16b	0	DW	W	1	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.460
00000000		00D044	8353	16b	0	DW	W	0	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.510

Figure 5.45 Trace window

Downloaded from Arrow.com.

The following items of information are displayed. (This applies for bus display.)

Table 5.13 Display items

Column	Description
Cycle	Cycle numbers stored in trace memory. The last cycle acquired is numbered 0, and the older cycles are assigned smaller numbers –1, –2, etc. sequentially retracing the past. If a delay count is set, the cycle in which a trace stop condition is met is numbered 0 and the cycles that were executed until the condition is met (cycles during a delay period) are assigned larger numbers +1, +2, etc. sequentially toward the last cycle acquired.
Label	Labels corresponding to addresses (displayed only when labels are set).
Address	Addresses of the address bus. [CAUTION] When using 4M mode, the address b31 will be "1b" when bank 0-6 is accessed. b30-28 shows the bank being accessed when b31 is "1b".
Data	Data of the data bus. Displayed in hexadecimal.
BUS	Shows the external data bus width, indicated as "8b" when the bus is 8 bits wide or "16b" when 16 bits wide.
BHE	Shows the state (0 or 1) of BHE (Byte High Enable) signal. When this signal is '0,' it means that an odd address is being accessed.
BIU	Shows the state between the BIU (Bus Interface Unit) and the memory and I/O. - No change DMA Data access such as DMA, etc. requested from other than the CPU INT INTACK sequence start IB Instruction code read (in bytes) requested from the CPU DB Data access (in bytes) requested from the CPU IW Instruction code read (in words) requested from the CPU DW Data access (in words) requested from the CPU
R/W	Shows the data bus state, indicated as "R" when in a read state, "W" when in a write state or "–" when no accesses made.
RWT	The signal indicating the valid position of bus cycle. When valid, this signal is '0.' The Address, Data and BIU lines are valid when this signal is '0.'
CPU	Shows the state between the CPU and the BIU (Bus Interface Unit). - No change CB Op-code read (in bytes) RB Operand read (in bytes) QC Instruction queue buffer clear CW Op-code read (in words) RW Operand read (in words)
QN	Shows the number of bytes stored in the instruction queue buffer. Displayed in the range from 0 to 4. [CAUTION] When stopping the user program by using software break, QN (the number of bytes stored in the instruction queue buffer) from the next cycle after an occurrence of software break is not displayed correctly.
BUSACC	When memory access is performed by a debugger operation during user program execution, shows "0" during the emulator is occupying the MCU bus. [CAUTION] Execution of the user program is temporarily stopped during such access to memory.
Debug	When memory access is performed by a debugger operation during user program execution, shows "0" during the emulator is occupying the MCU bus. [CAUTION] Execution of the user program is temporarily stopped during such access to memory.
EV	The event No. when a set event occurred. To show EV column, you need to select the EV number on the Option page of the Trace conditions dialog box displayed from the menu of the Trace window.
TID	Task ID (when RTOS is used). Example display: A task ID (task entry label) is displayed like 1 (_Task1). To show Task ID column, you need to select the Task ID on the Option page of the Trace conditions dialog box displayed from the menu of the Trace window.

Downloaded from Arrow.com.

EXT	Shows the signal fed in from the external trigger cable, indicated as "1" when the signal is high or "0" when the signal is low. To show EXT column, you need to select the External trigger on the Option page of the Trace conditions dialog box displayed from the menu of the Trace window.
ELC	Shows the module number of the linked module destination in event link. [CAUTION] "--" is displayed with M16C/64 Group, M16C/64A Group, M16C/65 Group, M16C/50 Series and R8C/3x Series MCUs.
ELCOVLAP	Shows if there is an overlap in event link. [CAUTION] "-" is displayed with M16C/64 Group, M16C/64A Group, M16C/65 Group, M16C/50 Series and R8C/3x Series MCUs.
TimeStamp	Shows an elapsed time since the target program started. Each time the user program starts running, the timestamp starts counting from 0. [CAUTION] When the counter overflows, the time is not displayed correctly.

The unnecessary columns in the Trace window can be hidden. To hide a column, right-click in the header column and select the column you want to hide from the pop-up menu.

5.9.3 Setting Trace Information Acquisition Conditions

The trace buffer is limited in size, so that when the buffer is filled, the old trace data is overwritten with new data sequentially beginning with the oldest.

Setting trace information acquisition conditions, you can acquire only the useful trace information, making effective use of the trace buffer.

To set trace information acquisition conditions, use the Trace conditions dialog box that is displayed when you choose Acquisition from the pop-up menu of the Trace window.

(1) Setting trace modes

First, select a trace mode.

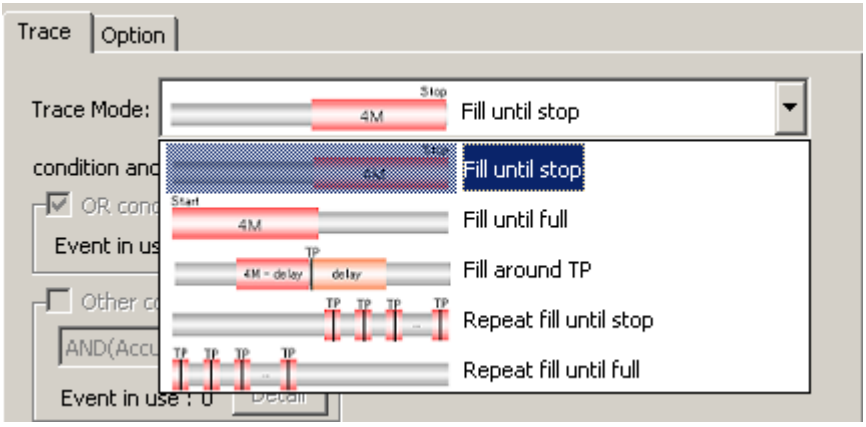


Figure 5.46 Trace conditions dialog box

(2) Setting trace points

If you selected Fill around TP, Repeat fill until stop or Repeat fill until full for the trace mode, set a trace point.

For trace points, you can set event based on conditions and exceptional events.

For Fill around TP, furthermore, you can set a delay value.

Downloaded from Arrow.com.

(3) Setting Capture/Do not Capture

If the selected trace mode is Fill until stop, Fill until full or Fill around TP, you can specify Capture/Do not Capture conditions in the Record condition group box.

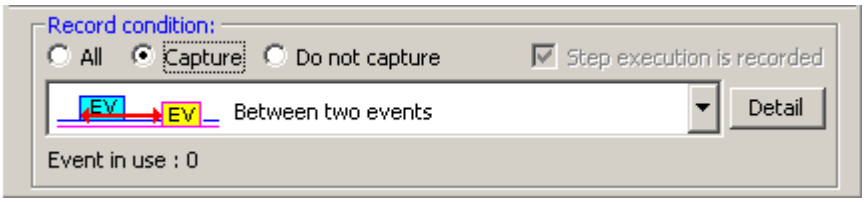


Figure 5.47 Record condition group box

You can choose to extract only the necessary portions of trace information specified by events or delete the unnecessary portions.

(4) Recording step execution

If the selected trace mode is Fill until stop, you can record step execution. To record step execution, select the Step execution is recorded check box in the Record condition group box.

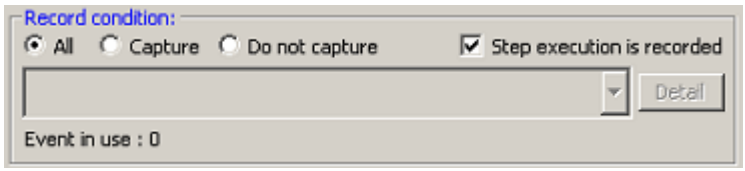


Figure 5.48 Recording step execution

The recordable modes of step execution are Step In, Step Over and Step Out.

(5) Setting trace acquisition methods

Use the Options page of the Trace conditions dialog box to set the acquisition method associated with the entire trace. By default, External Trigger is selected for trace acquisition.

Downloaded from Arrow.com.

5.9.4 Setting Trace Modes

(1) Setting trace modes

Following five trace modes are available.

Table 5.14 Trace modes

	Stop mode	Description
1	Fill until stop	Trace acquisition continues until the program stops running.
2	Fill until full	Trace acquisition stops when the trace memory is filled.
3	Fill around TP	Trace acquisition stops a specified number of cycles after a trace point is encountered. A delay value can be specified in a range of up to the maximum value of trace capacity.
4	Repeat fill until stop	Each time a trace point is encountered, acquisition is made for a total of 512 cycles* before and after that point, and acquisition continues that way until the program stops running.
5	Repeat fill until full	Each time a trace point is encountered, acquisition is made for a total of 512 cycles* before and after that point, and acquisition continues that way until the trace memory is filled.

CAUTION

Recording is made in units of total 512 cycles, consisting of 1 cycle at the line where a trace point is met and 255 cycles before that point and 256 cycles after that point.

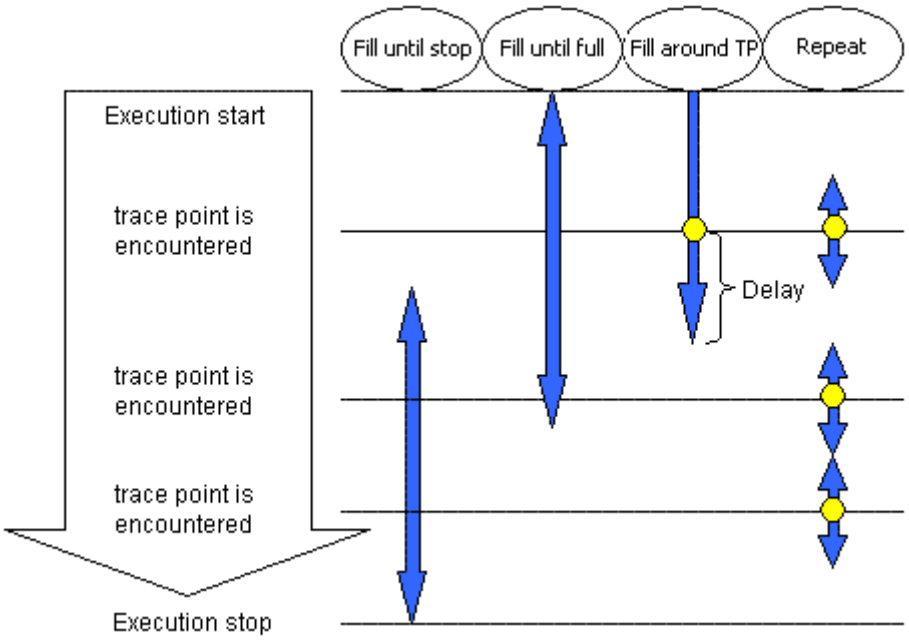


Figure 5.49 Differences between the trace modes

Specifiable conditions vary depending on trace mode, as summarized in the table below.

Downloaded from Arrow.com.

1. Fill until stop

The trace memory can hold up to 4M bus cycles. When the buffer is filled, the oldest data of the acquired trace information is overwritten with new data. That way, the emulator continues acquiring trace information.

Table 5.15 Specifiable conditions: Fill until stop

Trace point setting	Delay specification	Capture/Do not Capture condition setting	Step execution recording
-	-	Possible	Possible

2. Fill until full

When the trace memory of the emulator main unit overflows during trace acquisition, the emulator stops acquiring trace information.

Table 5.16 Specifiable conditions: Fill until full

Trace point setting	Delay specification	Capture/Do not Capture condition setting	Step execution recording
-	-	Possible	-

3. Fill around TP

Trace acquisition is halted a specified number of cycles delayed after a trace point is encountered. In this mode, the user program continues running and only trace acquisition is halted. Sophisticated conditions can be set using a maximum of 16 event points. A delay value can be chosen to be 0, 1M, 2M, 3M or 4M cycles.

Table 5.17 Specifiable conditions: Fill around TP

Trace point setting	Delay specification	Capture/Do not Capture condition setting	Step execution recording
Possible	Possible	Possible	-

4. Repeat fill until stop

Each time a trace point is encountered, a total of 512 cycles before and after that point are acquired, and acquisition continues that way. Acquisition continues until it is halted by a break or forced stop. The positions where trace points are encountered can be checked in the Trace window.

Table 5.18 Specifiable conditions: Repeat fill until stop

Trace point setting	Delay specification	Capture/Do not Capture condition setting	Step execution recording
Possible	-	-	-

Downloaded from Arrow.com.

5. Repeat fill until full

Each time a trace point is encountered, a total of 512 cycles before and after that point are acquired, and acquisition continues that way. When the trace memory overflows, acquisition is halted. The positions where trace points are encountered can be checked in the Trace window.

Table 5.19 Specifiable conditions: Repeat fill until full

Trace point setting	Delay specification	Capture/Do not Capture condition setting	Step execution recording
Possible	-	-	-

CAUTION

If trace points are encountered in consecutive cycles in the repeat fill until stop or repeat fill until full mode, only one first cycle is highlighted in yellow as a trace point.

5.9.5 Setting Trace Points

(1) Setting trace points

For trace points, you can set an OR condition, other condition (AND (Accumulation), AND (Simultaneous), subroutine, sequential or state transition) and detection of exceptional events.

The OR condition, other condition and the detection of exceptional events can be set all at the same time, or only one at a time.

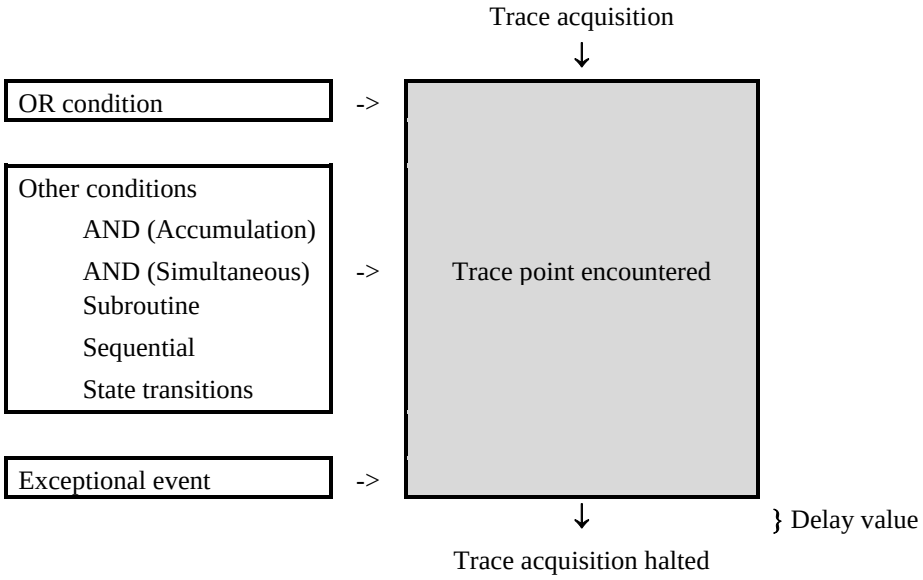


Figure 5.50 Outline of the trace point

Downloaded from Arrow.com.

(2) OR condition

You can choose to enable or disable the OR condition. By default, the OR condition is enabled.
If you reenable the OR condition when it is disabled, the previously set event is restored with its OR condition check box selected. However, if a maximum of 16 points is exceeded when you have reenabled for an event, the event is restored with its OR condition check box unselected (disabled).

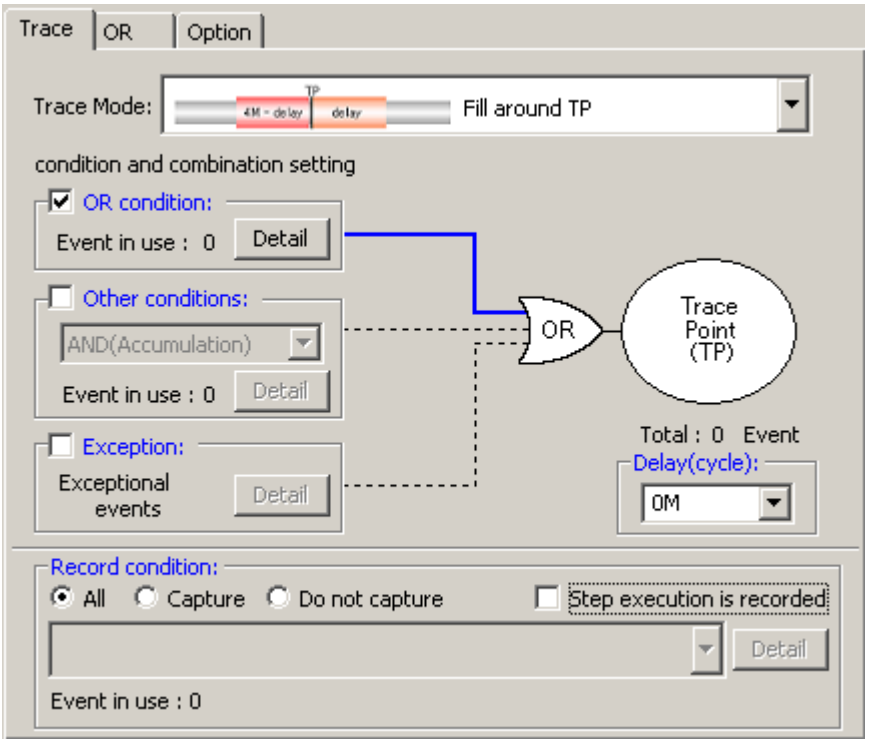


Figure 5.51 Trace conditions dialog box

Table 5.20 OR condition

	Type	Description
1	OR condition	Trace point is encountered when any one of the set events occurs.

Downloaded from Arrow.com.

(3) Other conditions

You can select one of five choices available: AND (Accumulation), AND (Simultaneous), Subroutine, Sequential and State Transition. To set any condition, select the check box to the left of "Other Conditions."
By default, other conditions are disabled (the check box to the left of "Other Conditions" is unselected).

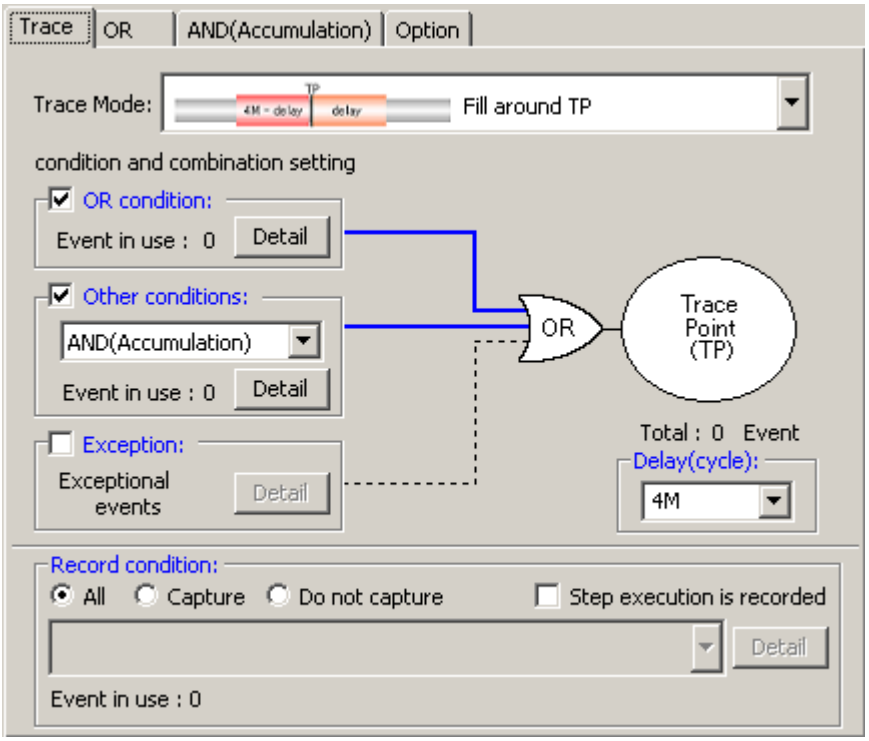


Figure 5.52 Trace conditions dialog box

Table 5.21 Other conditions

	Type	Description
1	AND (Accumulation)	Trace point is encountered when all of the set events occur irrespective of the time axis.
2	AND (Simultaneous)	Trace point is encountered when all of the set events occur at the same time.
3	Subroutine	Trace point is encountered when a specified event occurs within a specified address range (subroutine or function).
4	Sequential	6 steps (forward direction) + reset point Trace point is encountered when a set event occurs in a specified order.
5	State transitions	3 steps, 9 paths + reset point Trace point is encountered when a set event occurs in a specified order.

CAUTION
When a time-out condition is set in State transitions (Trace) dialog box, the time to make transition from a set state to another then back to the original set state must be 10 µs or more. Transition time of less than 10 µs will result in an incorrect timeout detection.

Downloaded from Arrow.com.

(4) Detection of exceptional events

Specify whether you want detection of following exceptional events to be used as a trace point.

- Violation of access protection
- Read from uninitialized memory
- Stack access violation
- Performance overflow
- Realtime profile overflow
- Task stack access violation
- OS dispatch

(5) Specifying a delay value

The program breaks a specified number of cycles delayed after a trace point is encountered.
A trace-point delay value can be selected from 0M, 1M, 2M, 3M or 4M bus cycles (default: 0M).
Select your desired delay value in the delay value setting column.

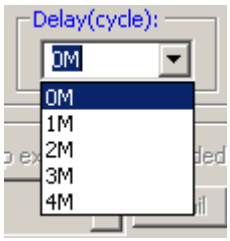


Figure 5.53 Trace conditions dialog box

Downloaded from Arrow.com.

5.9.6 Setting Capture/Do not Capture Conditions

If the selected trace mode is Fill until stop, Fill until full or Fill around TP, you can specify Capture/Do not Capture conditions. You can choose to extract only the necessary portions of trace information specified by events or delete the unnecessary portions.

(1) Capture/Do not Capture conditions

There are following types of conditions.

Table 5.22 Capture/Do not Capture conditions

Type			Description
Extraction		Between two events	Cycles extracted begin when a start event occurs and end with the next to last end event occurs (the cycle where an end event occurs is not extracted).
		Duration of an event	Only cycles where a specified event occurred are extracted.
		Duration of an event occurring in a subroutine	Only cycles where a specified event occurred in a specified address range (subroutine or function) are extracted.
		Instruction accessing specific data	Instructions that accessed specified data are detected.
Deletion		Between two events	Cycles extracted begin when a start event occurs and end with the next to last end event occurs (the cycle where an end event occurs is not extracted).
		Duration of an event	Only cycles where a specified event occurred are deleted.
		Duration of an event occurring in a subroutine	Only cycles where a specified event occurred in a specified address range (subroutine or function) are deleted.

Select the conditions you want to set from the list box that is displayed when you select Extract or Delete in the Record condition group box of the Trace conditions dialog box.

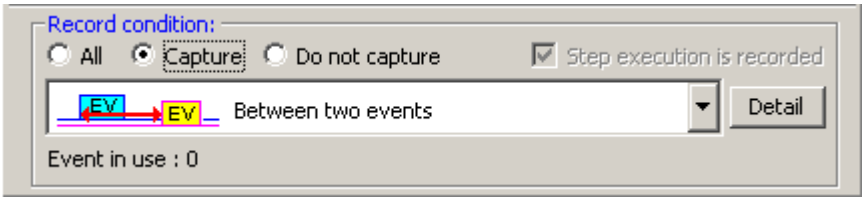


Figure 5.54 Record condition group box

Then click the Detail button. A page in which you can set events will be displayed.

Downloaded from Arrow.com.

CAUTION

When you specify extraction or deletion conditions, you cannot select DIS (disassemble display) and SRC (source display) from Display Modes in the Trace window.

When you use a data access event for extraction or deletion, be sure to specify the MCU bus for the access type.

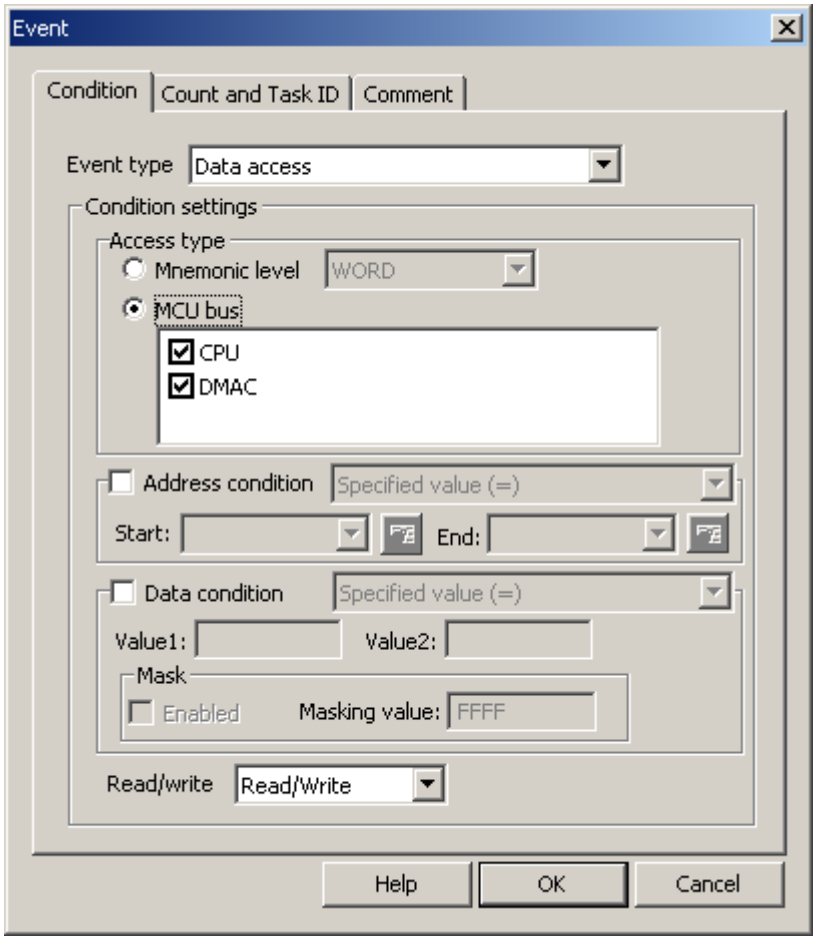


Figure 5.55 Event dialog box

Downloaded from Arrow.com.

5.9.7 Selecting the Content of Trace Acquisition

Select the content of trace information you want to be captured into trace memory. Use the Options page of the Trace conditions dialog box to make this selection.

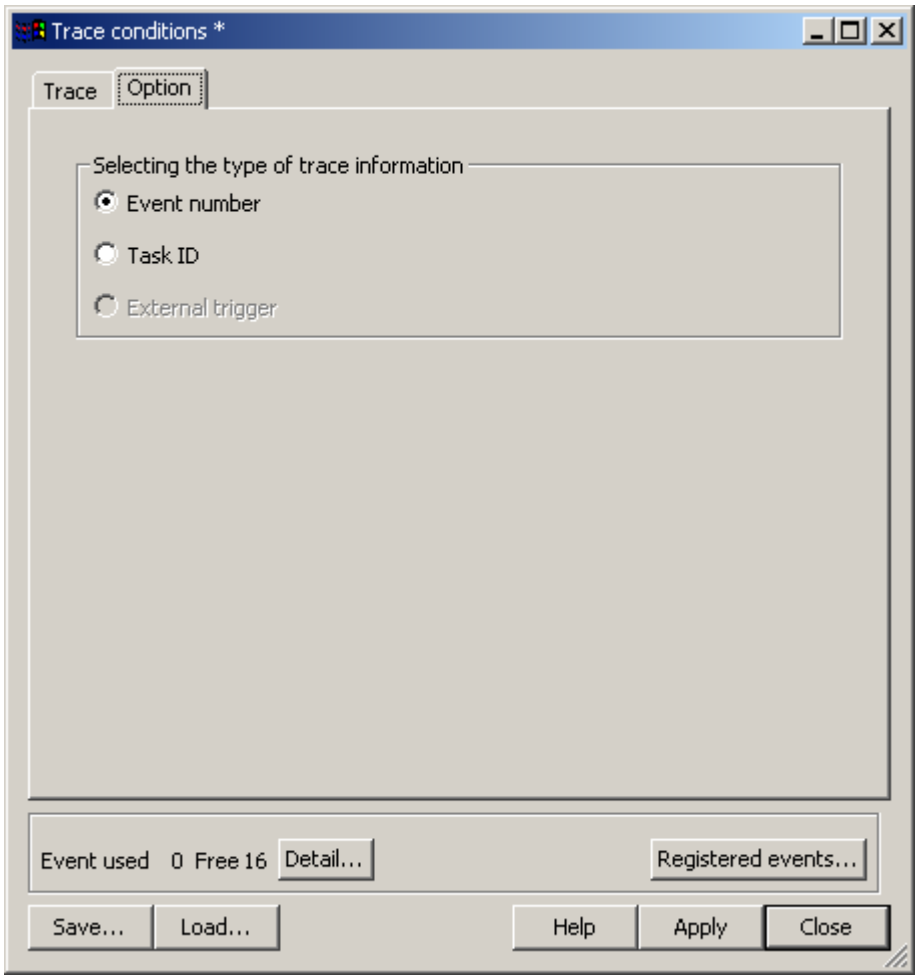


Figure 5.56 Trace conditions dialog box

Select which signal you want to be acquired from three choices available: Event Number, Task ID or External trigger. By default, the Event number is selected.

CAUTION

If you want a history of trace execution to be displayed in trace acquisitions carried out by running a realtime OS program, be sure to select Task ID.

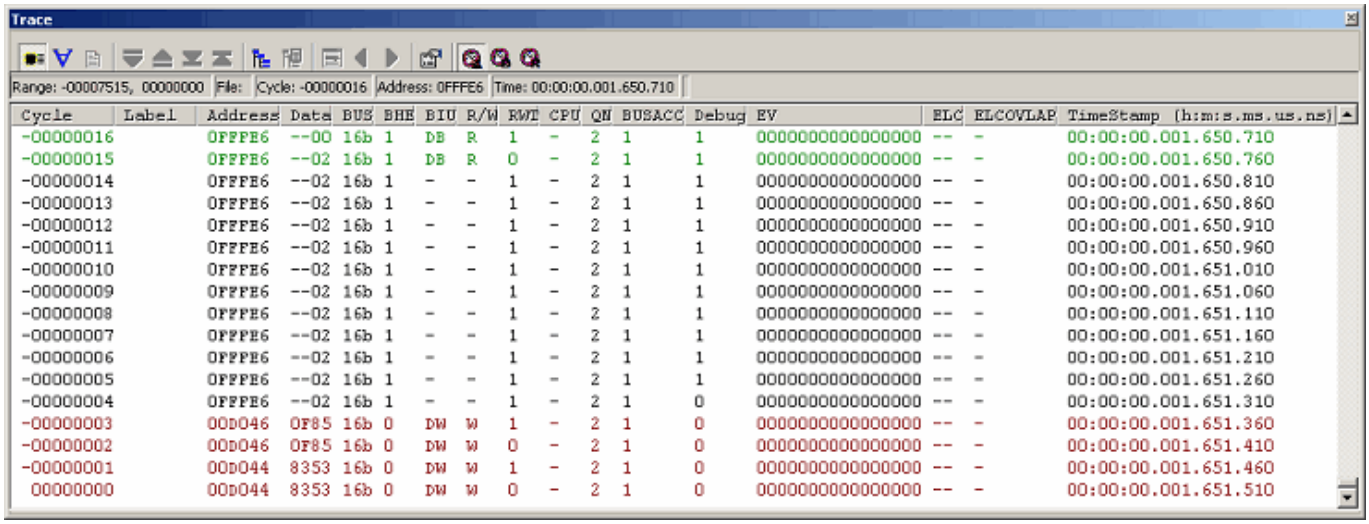
Downloaded from Arrow.com.

5.9.8 Showing Trace Results

To check trace results, look at the Trace window. Trace results can be shown in one of the following display modes.
These display modes can be switched using Display Modes on the pop-up menu of the Trace window.
There are five trace result display modes: Bus Display, Disassembled Display, Source Display and Mixed Display.

(1) Bus Display Mode

In the pop-up menu, select Display Modes -> BUS. Bus information on each cycle traced are displayed. (Default display mode)

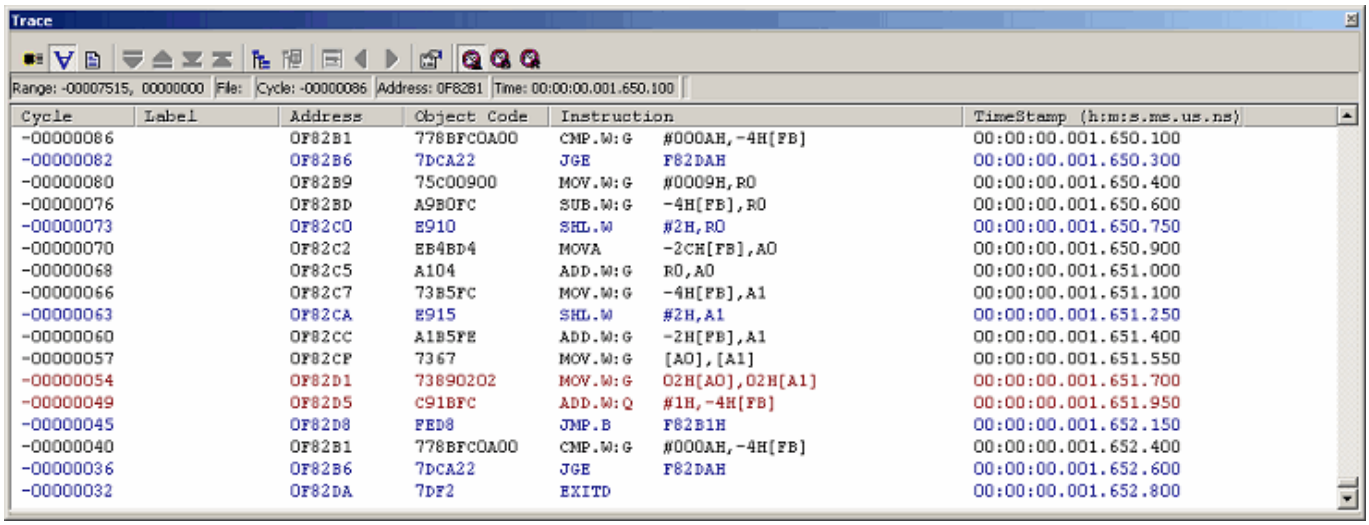


Cycle	Label	Address	Data	BUS	BHE	BIU	R/W	RWT	CPU	QN	BUSACC	Debug	EV	ELC	ELCOVLAP	TimeStamp (h:m:s.ms.us.ns)
-00000016		0FFFE6	--00	16b	1	DB	R	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.710
-00000015		0FFFE6	--02	16b	1	DB	R	0	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.760
-00000014		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.810
-00000013		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.860
-00000012		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.910
-00000011		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.650.960
-00000010		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.010
-00000009		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.060
-00000008		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.110
-00000007		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.160
-00000006		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.210
-00000005		0FFFE6	--02	16b	1	-	-	1	-	2	1	1	0000000000000000	--	-	00:00:00.001.651.260
-00000004		0FFFE6	--02	16b	1	-	-	1	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.310
-00000003		00D046	0F85	16b	0	DW	W	1	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.360
-00000002		00D046	0F85	16b	0	DW	W	0	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.410
-00000001		00D044	8353	16b	0	DW	W	1	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.460
00000000		00D044	8353	16b	0	DW	W	0	-	2	1	0	0000000000000000	--	-	00:00:00.001.651.510

Figure 5.57 Trace window

(2) Disassembled Display Mode

From the pop-up menu, choose Display Modes -> DIS. This display mode allows you to inspect the machine language instructions executed.



Cycle	Label	Address	Object Code	Instruction		TimeStamp (h:m:s.ms.us.ns)
-00000086		0F82B1	778BFC0A00	CMP.W:G	#000AH,-4H[FB]	00:00:00.001.650.100
-00000082		0F82B6	7DCA22	JGE	F82DAH	00:00:00.001.650.300
-00000080		0F82B9	75C00900	MOV.W:G	#0009H,R0	00:00:00.001.650.400
-00000076		0F82BD	A9B0FC	SUB.W:G	-4H[FB],R0	00:00:00.001.650.600
-00000073		0F82C0	E910	SHL.W	#2H,R0	00:00:00.001.650.750
-00000070		0F82C2	EB4BD4	MOVA	-2CH[FB],A0	00:00:00.001.650.900
-00000068		0F82C5	A104	ADD.W:G	R0,A0	00:00:00.001.651.000
-00000066		0F82C7	73B5FC	MOV.W:G	-4H[FB],A1	00:00:00.001.651.100
-00000063		0F82CA	E915	SHL.W	#2H,A1	00:00:00.001.651.250
-00000060		0F82CC	A1B5FE	ADD.W:G	-2H[FB],A1	00:00:00.001.651.400
-00000057		0F82CF	7367	MOV.W:G	[A0],[A1]	00:00:00.001.651.550
-00000054		0F82D1	73890202	MOV.W:G	02H[A0],02H[A1]	00:00:00.001.651.700
-00000049		0F82D5	C91BFC	ADD.W:Q	#1H,-4H[FB]	00:00:00.001.651.950
-00000045		0F82D8	FED8	JMP.B	F82B1H	00:00:00.001.652.150
-00000040		0F82B1	778BFC0A00	CMP.W:G	#000AH,-4H[FB]	00:00:00.001.652.400
-00000036		0F82B6	7DCA22	JGE	F82DAH	00:00:00.001.652.600
-00000032		0F82DA	7DF2	EXITD		00:00:00.001.652.800

Figure 5.58 Trace window

Downloaded from Arrow.com.

(3) Source Display Mode

From the pop-up menu, choose Display Modes -> SRC. This display mode allows you to inspect the source program's execution path.

The execution path can be verified by stepping through the source within trace data forward or backward from the current trace cycle.

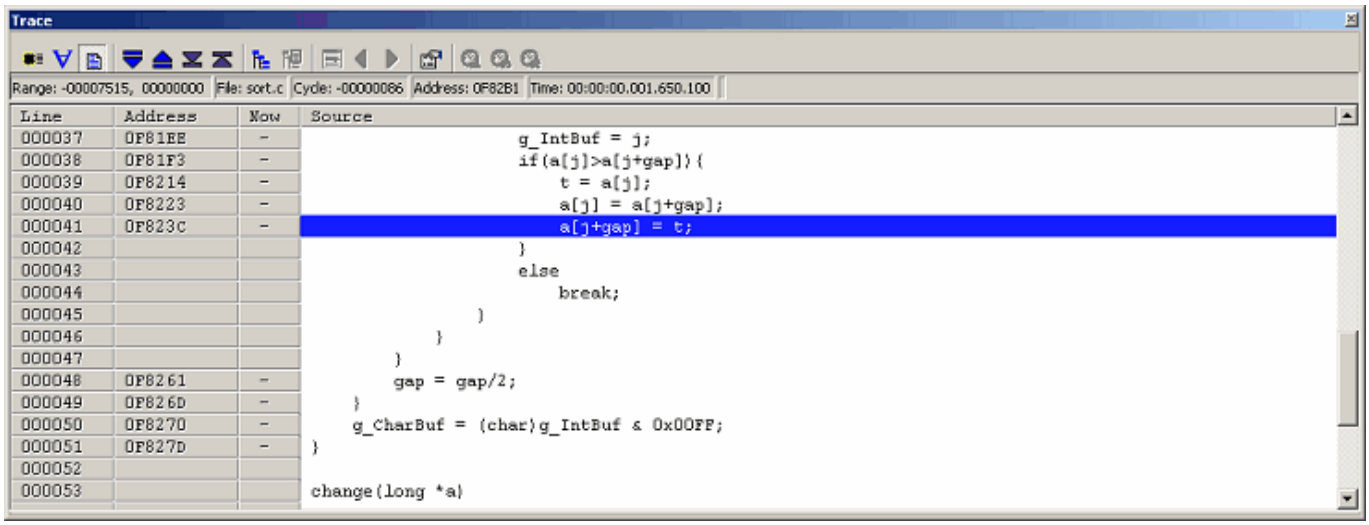


Figure 5.59 Source Display screen

(4) Mixed Display Mode

This display mode provides a mixed display of bus, disassemble or source display.

After choosing Display Modes -> BUS from the pop-up menu, select Display Modes -> DIS. That way, you can produce a bus and disassemble mixed display.

In the same way, you can produce a bus and source, a disassemble and source or a bus, disassemble and source mixed display.

To revert to a bus only display after viewing a bus and disassemble mixed display, choose Display Modes –> DIS from the pop-up menu again.

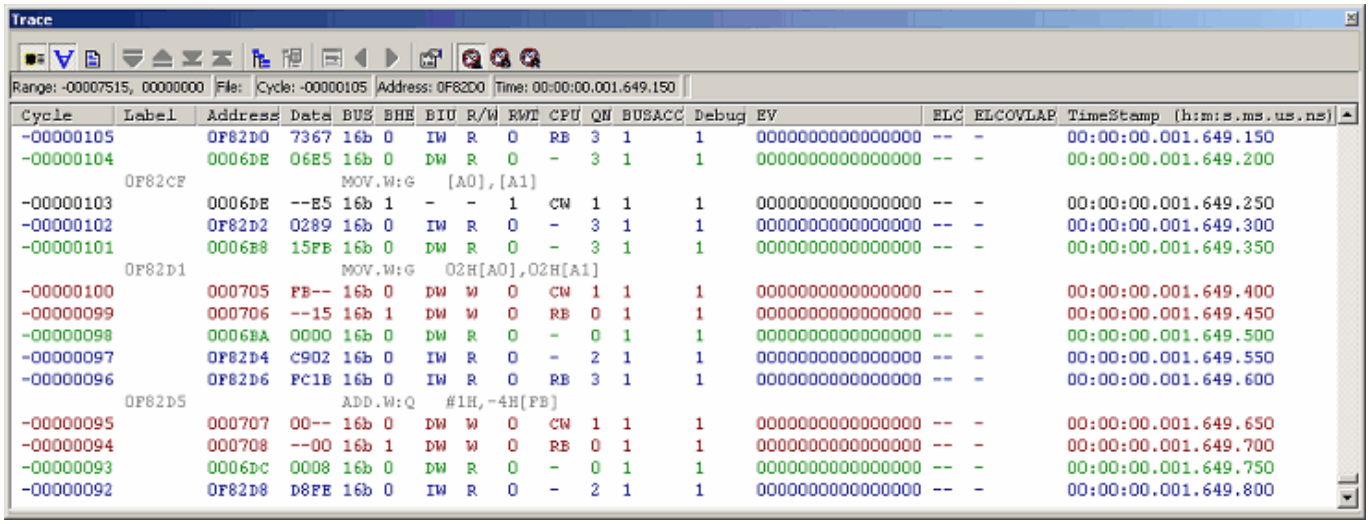


Figure 5.60 Trace window

Downloaded from Arrow.com.

5.9.9 Filtering Trace Information

Use the filter function to extract only the necessary records from the acquired trace information. The filter function filters the trace information in software that was acquired by hardware.

Unlike the "Capture/Do not Capture conditions" where you set acquisition conditions before getting trace information, this function permits you to change filter settings for the acquired trace information any number of times.

Therefore, the necessary information can be extracted easily, with data analysis significantly facilitated.

The filter function does not affect the trace memory, so that its content remains intact.

The filter can be used when the selected trace mode is Fill until stop, Fill until full or Fill around TP and the selected display mode is Bus or Disassembled.

(1) Auto-filter function

To use the filter function, choose Auto Filter from the pop-up menu of the Trace window. When Auto Filter is turned on, each column of the Trace window is marked with an auto-filter arrow [▼].

Click any arrow [▼] and select the necessary condition from the ensuing drop-down list. That way, you can easily filter the records to get those that meet the condition. Selecting Option in the drop-down list brings up the Option dialog box. In this dialog box, you can set detail conditions.

Some columns such as Address and Data have only Option provided in the drop-down list because they do not have other inherent items. Selecting All returns you to a non-filter state.

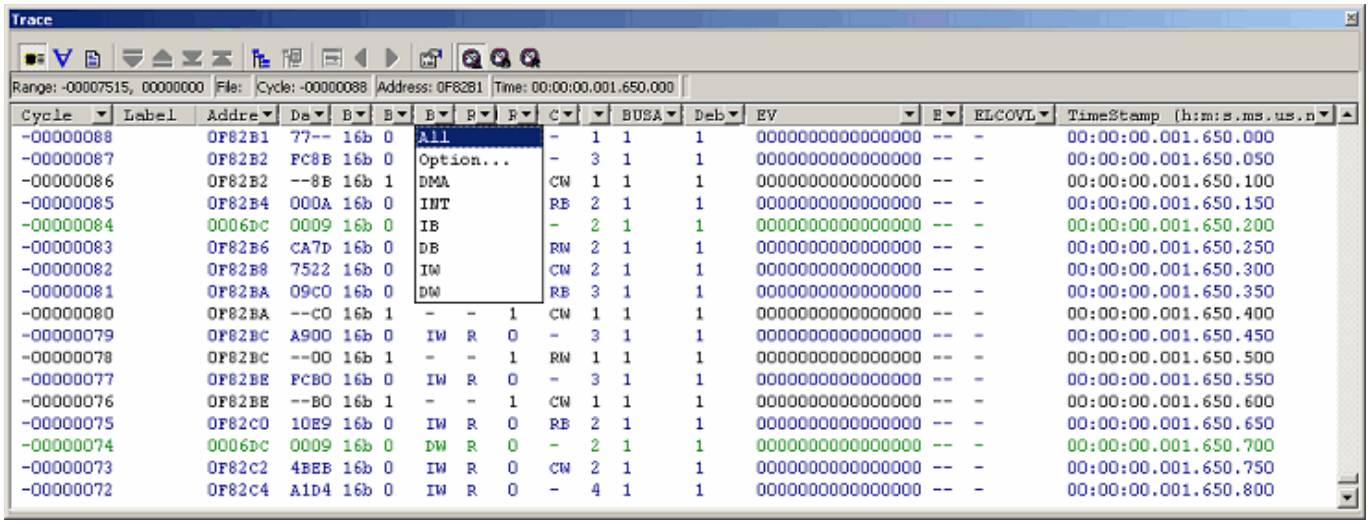


Figure 5.61 Trace window

If after filtering records in bus display mode you switch to disassemble-only or source-only display, Auto Filter is deselected. Similarly, when after filtering records in disassembled display mode you switch to bus-only or source-only display, Auto Filter is deselected.

Downloaded from Arrow.com.

If there are multiple items you can specify in the Option dialog box, these items can be used as an OR condition with which to filter.

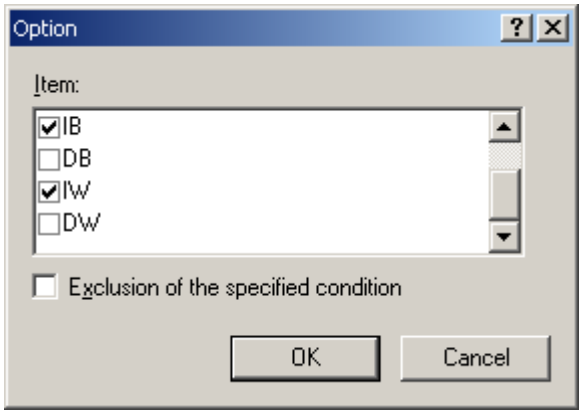


Figure 5.62 Option dialog box

5.9.10 Searching for Trace Records

You can search the acquired trace information for a specific trace record.

To search for trace records, use the Find dialog box. To open it, choose Find -> Find from the pop-up menu of the Trace window or click the Find button  in the toolbar.

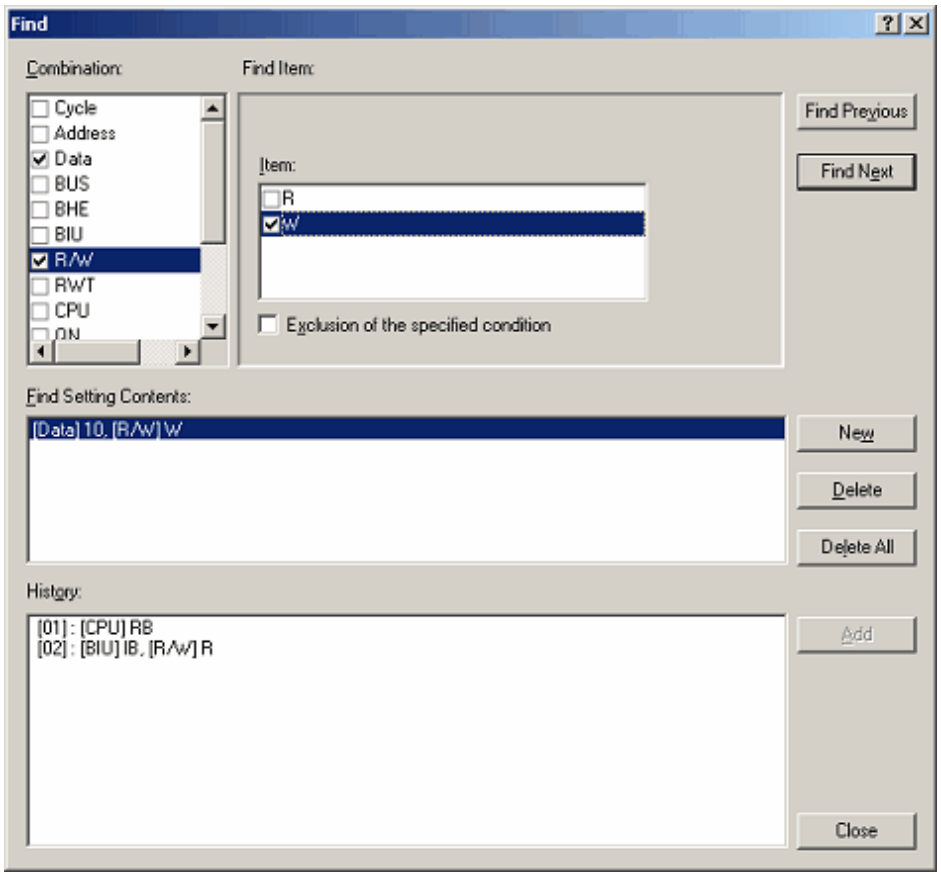


Figure 5.63 Find dialog box

Downloaded from Arrow.com.

Select the conditions you want to search for in the Combination column and select the check boxes.
In the Find Item column, you can select the items that correspond to the selected conditions.
If you checked more than one condition in the combination column, set items for each condition. The items you have set are searched for as multiple AND conditions.
The conditions you have set are shown in the Find Setting Contents.
After setting search conditions, click the Find Previous or the Find Next button to start a search. Trace records are searched in forward or searched in reverse from the line you have clicked in the Trace window (the line highlighted in blue).
When a matching trace record is found by a search, the relevant line in the Trace window is highlighted. If no matching trace records are found, a message dialog box is displayed.
When an instance of the trace record was successfully found, choose Find Previous or Find Next from the pop-up menu. The next instance of the trace record will be searched for.

(1) Search history

The conditions once searched are left as a history in the history column while the High-performance Embedded Workshop remains active.
The next time you perform a search, choose the line you want to search from this history and click the Add button. That way, you can search trace information with that condition again.
The search history contains a history of up to 10 last searches performed.

(2) OR search

You can perform a search using two or more search conditions as OR conditions.
To set OR conditions, begin by setting the first condition (shown on the first line in the search content setting column) and then click the New button.
Then enter the second condition. At this time, the second condition is added to the second line in the search content setting column.
In this case, the conditions on the first and second lines in the search content setting column can be used as OR conditions for a search performed.
Up to 16 conditions (16 lines) can be set.

CAUTION
The conditions set on one and the same line in the search content setting column comprise AND conditions.

5.9.11 Saving Trace Information to Files

To save trace information to a file, choose File -> Save from the pop-up menu or click the Save button  in the toolbar.
The trace information displayed in the Trace window is saved in binary or text format.

(1) Saving in binary format

To save trace information in binary format, choose "Trace Data File: Memory Image (*.rtt)" in the Save As Type list box of the dialog box that is displayed when you choose File -> Save from the pop-up menu.
When saved in binary format, all cycles are saved. This type of file can be loaded into the Trace window.

(2) Saving in text format

To save trace information in text format, choose "Text Files: Save Only (*.txt)" in the Save As Type list box of the dialog box that is displayed when you choose File -> Save from the pop-up menu.
When saved in text format, a range of cycles to be saved can be specified. This type of file can only be saved and cannot be loaded into the Trace window.

Downloaded from Arrow.com.

5.9.12 Loading Trace Information from Files

To load trace information from a file, choose File -> Load from the pop-up menu or click the Load button  in the toolbar.
Specify a trace information file saved in binary format. The current trace result is overwritten.
Before loading a file saved in binary format, switch to the trace mode in which mode you saved trace information. Do this switching in the Trace conditions dialog box that is displayed when you choose Acquisition from the pop-up menu of the Trace window.
If the current trace mode differs from the one in which mode you saved trace information, an error results. Trace information files saved in text format cannot be loaded into the Trace window.

5.9.13 Temporarily Stopping Trace Information Acquisition

To temporarily stop acquiring trace information during user program execution, choose Trace -> Stop from the pop-up menu of the Trace window or click the Stop button  in the toolbar.
Trace acquisition will be aborted, with the trace display updated. Use this function when you only want to stop acquiring trace information and check the trace information without stopping program execution.

5.9.14 Restarting Trace Information Acquisition

If after temporarily stopping acquisition of trace information during user program execution you want to start acquiring trace information again, choose Trace -> Restart from the pop-up menu of the Trace window or click the Restart button  in the toolbar.

5.9.15 Switching Timestamp Display

The timestamp displayed in the Trace window can be switched to absolute time, differential time or relative time. In the initial state, the timestamp is displayed in absolute time.

(1) Absolute time

From the pop-up menu, choose Time -> Absolute Time or click the Absolute Time button  in the toolbar. The timestamp will be displayed by an absolute time since program execution started.

(2) Differential time

From the pop-up menu, choose Time -> Differences or click the Differences button  in the toolbar. The timestamp will be displayed by a differential time from the preceding cycle.

(3) Relative time

From the pop-up menu, choose Time -> Relative Time or click the Relative Time button  in the toolbar. The timestamp will be displayed by a relative time from a specified cycle.

Downloaded from Arrow.com.

5.9.16 Showing the History of Function Execution

To show the history of function execution from the acquired trace information, choose Function Execution History -> Function Execution History from the pop-up menu or click the Function Execution History button  in the toolbar.

An upper pane of the window will be displayed. (Initially, this window is blank.)

When you choose Analyze Execution History from the pop-up menu or click the Analyze Execution History button  in the toolbar, the emulator starts analyzing the execution history from the end of the trace result and shows the result in a tree structure.

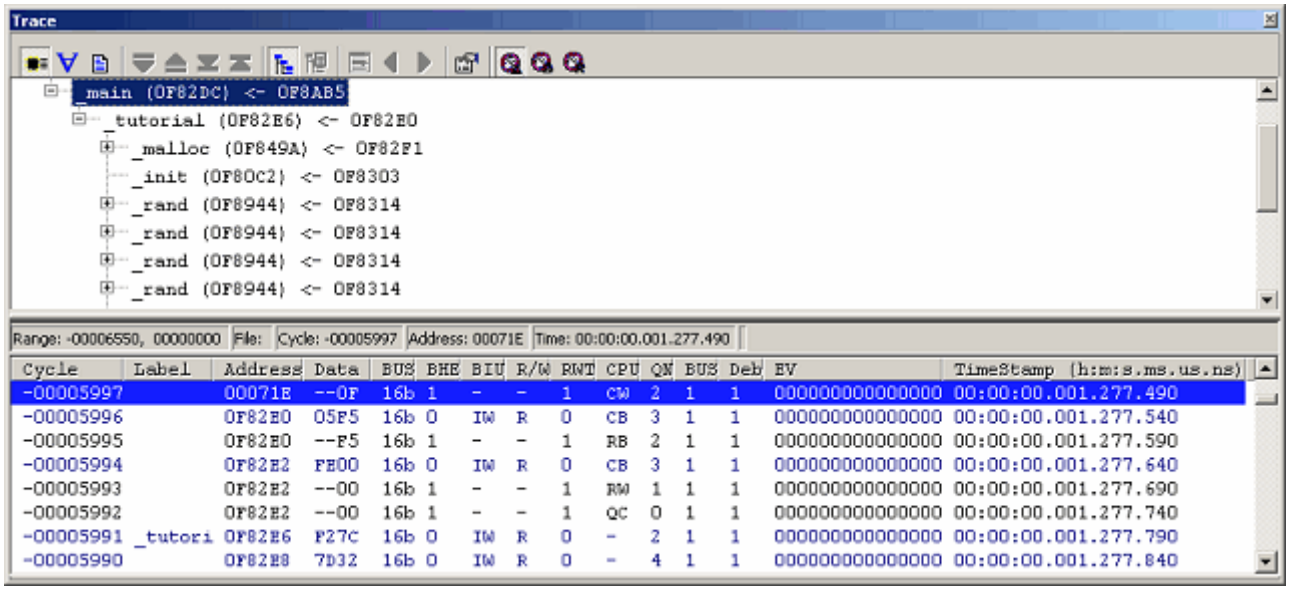


Figure 5.64 Trace window

The lower pane of the window shows the trace result beginning with the cycle in which the function selected in the upper pane was called.

The lower pane of the window can show trace results in disassemble, source or mixed mode.

CAUTION

If trace extraction or deletion conditions are specified, the function execution history cannot be displayed.

If repeat (free) or repeat (full) mode is specified, the function execution history cannot be displayed.

Downloaded from Arrow.com.

5.9.17 Showing the History of Task Execution

The history of task execution can only be displayed when you are debugging a realtime OS program.

Furthermore, to show the history of task execution, you need to select Task ID on the Options page of the Trace conditions dialog box that is displayed when you choose Acquisition from the pop-up menu of the Trace window.

To show the history of function execution from the acquired trace information, choose Show Function Execution History from the pop-up menu click the Show Function Execution History button  in the toolbar.

An upper pane of the window will be displayed. (Initially, this window is blank.)

When you choose Analyze Execution History from the pop-up menu that is displayed when you right-click in the upper pane or click the Analyze Execution History button  in the toolbar, the emulator shows the history of task execution.

When showing the history of task execution, note that the functions called from within tasks are not displayed in a tree structure. Only the order in which the functions were executed is displayed.

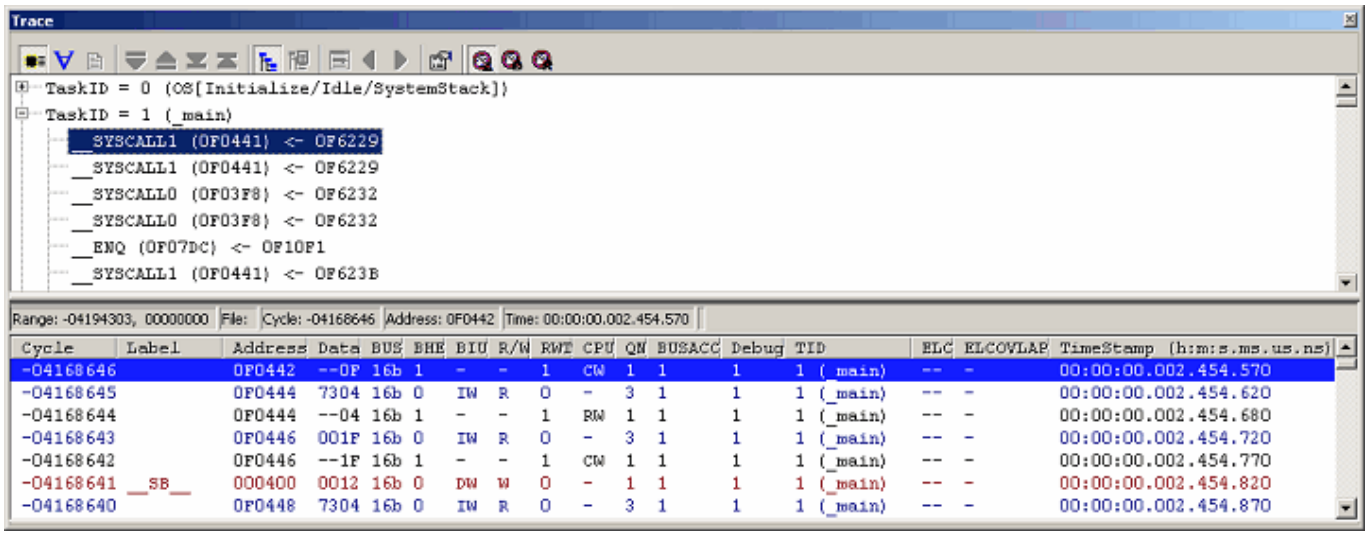


Figure 5.65 Trace window

The lower pane of the window shows the trace result beginning with the cycle in which the task selected in the upper pane was called.

The lower pane of the window can show trace results in disassemble, source or mixed mode.

CAUTION

If trace extraction or deletion conditions are specified, the task execution history cannot be displayed.

If repeat (free) or repeat (full) mode is specified, the task execution history cannot be displayed.

Downloaded from Arrow.com.

5.10 Measuring Performance

5.10.1 Measuring Performance

The performance function measures a maximum, minimum, average and total execution time and a pass count in each of up to eight specified sections of the user program and then shows a time ratio relative to the total execution time (Go–Break) numerically as percentage and graphically.

Since the performance function uses the emulator's performance measurement circuit to measure the execution time, it does not obstruct the execution of the user program.

Performance measurement conditions cannot be manipulated during program execution.

5.10.2 Showing the Result of Performance Measurement

Measurement results are displayed in the Performance Analysis window.

To open the Performance Analysis window, choose Performance –> Performance Analysis from the View menu or

click the Performance Analysis toolbar button [].

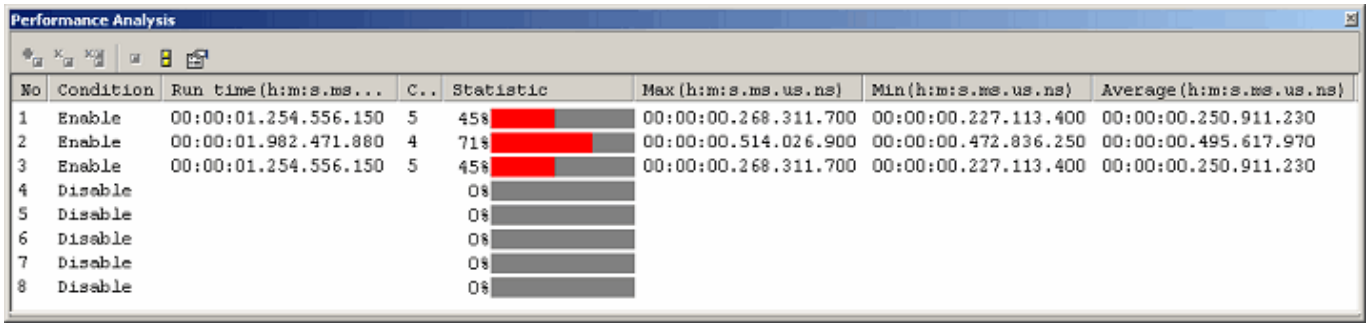


Figure 5.66 Performance Analysis window

The Performance Analysis window shows a ratio of execution time conforming to the conditions you set in the immediately preceding program execution numerically as percentage and graphically.

The unnecessary columns in this window can be hidden.

To hide any column, right-click in the header column and select the column you want to hide from the pop-up menu.

To redisplay any hidden column, select that column from the pop-up menu again.

Downloaded from Arrow.com.

The contents displayed in this window are listed below.

Table 5.23 Columns and contents

Column	Description
No	Numbers assigned to 1–8 measurement sections set in the Performance Analysis Conditions dialog box. Click Settings on the pop-up menu to open the Performance Analysis Conditions dialog box.
Condition	Indicated as Enable when measurement conditions are set in the Performance Analysis Conditions dialog box. Otherwise, indicated as Disable.
Run time (h:m:s.ms.us.ns)	Cumulative execution time. It shows a cumulative time of measured execution time.
Count	Shows the number of times measured.
Statistic	Shows a ratio of cumulative execution time relative to Go–Break execution time. [Ratio calculation formula] (Cumulative execution time / Go–Break cumulative execution time) * 100
Max (h:m:s.ms.us.ns)	Maximum execution time per measurement performed
Min (h:m:s.ms.us.ns)	Minimum execution time per measurement performed
Average (h:m:s.ms.us.ns)	Average execution time per measurement performed

5.10.3 Setting Performance Measurement Conditions

In the Performance window, select a line of the section No. in which you want to set conditions and choose Set from the pop-up menu. The Performance Analysis Conditions dialog box will be displayed.

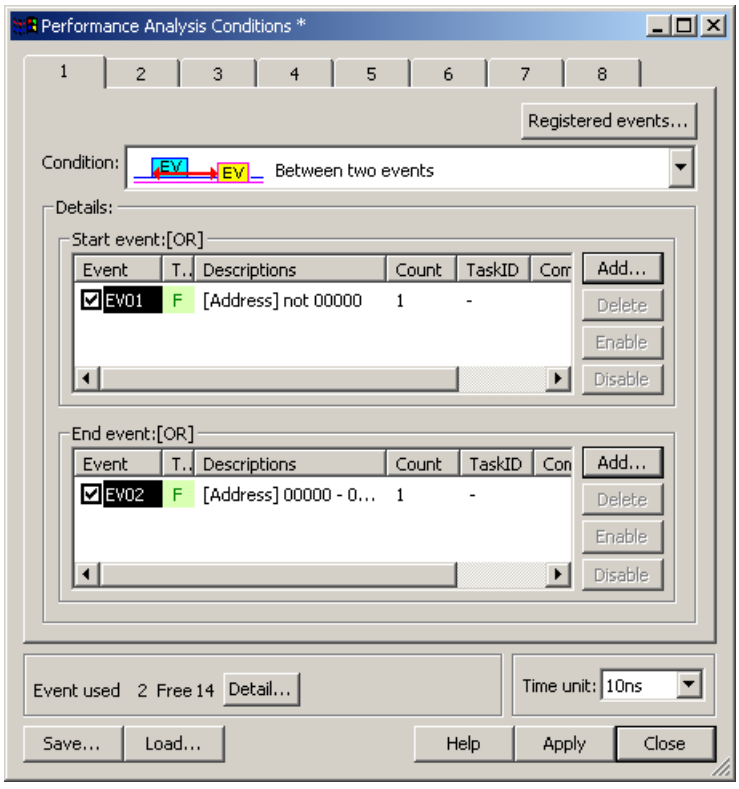


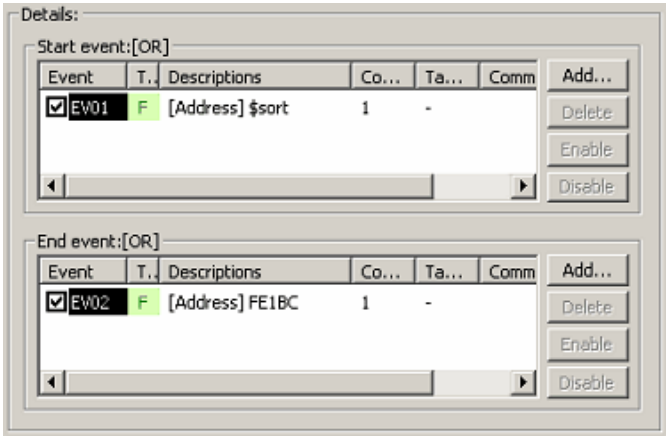
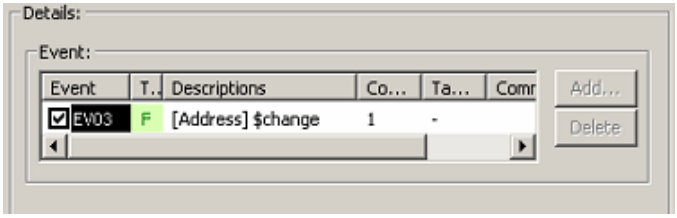
Figure 5.67 Performance Analysis Conditions dialog box

Downloaded from Arrow.com.

(1) Setting measurement conditions

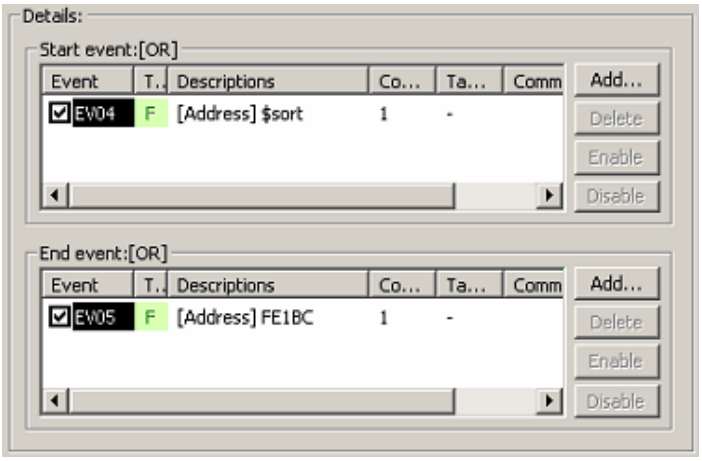
A measurement condition can be selected from the following four modes. Select one measurement condition for one section. Use events to set a section. Event counts are fixed to 1. Even when an event count is set to other than 1, it is handled as 1.

Table 5.24 Measurement condition modes

	[Disabled] Not measured.
EV EV	[Between two events]  Figure 5.68 Between two events Measurement is taken of time from when a start event occurs to when an end event occurs. Specifically, measurement is taken of an execution time and execution count in the range set by a start event and an end event. The measurement of time starts when a start event occurs and is aborted when an end event occurs. The execution count is incremented by one each time a start event and an end event occur in pairs within the set range. Start event: One or multiple events can be set. End event: One or multiple events can be set.
EV EV	[Event cycle counting]  Figure 5.69 Event cycle counting Measurement is taken of periods in which an event occurs. Namely, measurement is taken of an event occurrence period and execution count. The time from when an event occurs to when the next event occurs is measured as one instance of measurement. The execution count is incremented by one each time an event occurs. Event: Only one event point can be set.

Downloaded from Arrow.com.

Table 5.25 Measurement condition modes (Continued)

EV EV	[Interrupt-disabled range between two events]  Figure 5.70 Interrupt-disabled range between two events Measurement is taken of an interrupt disabled section from when a start event occurs to when an end event occurs. Specifically, measurement is taken of an interrupt disabled time and an interrupt disabled count within the range set by a start event and an end event. The measurement of time starts at the same time an interrupt is disabled and is aborted at the same time the interrupt is reenabled. The count is incremented by one each time an interrupt is disabled. Start event: One or multiple events can be set. End event: One or multiple events can be set.

[CAUTION]
To measure an execution time of a function (maximum, minimum or average execution time of a function), use Between two events.
Set a fetch to the beginning address of the function as a start event and a fetch to the exit of the function (where return statement is written) as an end event. If there are more than one exit, set fetch conditions as an end event for each exit.

(2) Selecting the measurement interval

This setting is applied in common to all of 8 sections. The measurement interval can be selected from the following options:
10 ns (default), 20 ns, 40 ns, 80 ns, 160 ns, 1.6 µs
The maximum measurement time varies with the measurement interval you set.

5.10.4 Starting Performance Measurement

When the user program is run, performance measurement is automatically started according to the performance measurement conditions set.
When the user program is halted, the measurement result is displayed in the Performance Analysis window.
When the user program is rerun without changing measurement conditions after being halted, the measured time in this instance is added to the previously measured value.
To perform a measurement over again, clear the measurement result before running the program.

Downloaded from Arrow.com.

5.10.5 Clearing Performance Measurement Conditions

Select the measurement condition you want to clear in the Performance Analysis window and then choose Set from the pop-up menu to display the Performance Analysis Conditions dialog box. In the Performance Analysis Conditions dialog box, disable the condition you want to clear.



Figure 5.71 Performance Analysis Conditions dialog box

5.10.6 Clearing the Performance Measurement Result

Select the section you want to clear in the Performance Analysis window and then choose Clear Data from the pop-up menu. The measurement result of the selected section will be cleared. To clear all measurement results, choose Clear All Data from the pop-up menu.

5.10.7 About the Maximum Measurement Time of Performance

(1) Maximum measurement time

The timer used for performance measurement is comprised of a 40-bit counter.
The maximum measurement time varies with the measurement interval selected.
To select the measurement interval, use the Measurement Unit list box of the Performance Analysis Conditions dialog box.
The measurable maximum times are listed in the table below.

Table 5.26 Measurable maximum time

No.	Resolution	Measurable maximum time
1	10ns	Approx. 3 hours, 03 minutes, 15 seconds
2	20ns	Approx. 6 hours, 06 minutes, 30 seconds
3	40ns	Approx. 12 hours, 13 minutes, 00 seconds
4	80ns	Approx. 24 hours, 26 minutes, 00 seconds
5	160ns	Approx. 48 hours, 52 minutes, 01 seconds
6	1.6µs	Approx. 488 hours, 40 minutes, 18 seconds

CAUTION

Note that performance measurement produces an error equal to ±1 resolution (when resolution = 20 ns, ±20 ns).

(2) Maximum measurement count

Execution counts are measured using a 32-bit counter. Measurement can be taken of up to a count of 4,294,967,295.

Downloaded from Arrow.com.

5.11 Acquiring Code Coverage

5.11.1 Acquiring Code Coverage

Code coverage is the function to indicate the 'digestion' degree of test, i.e., "to what degree tests have been carried out on software code (pass)."
Instruction execution information is displayed at C/C++ and assembly-language levels.
This function allows the emulator to acquire instruction execution information from a program without causing it to break. Therefore, the realtime operation of the user program will not be affected.
The coverage result is updated upon a break.

The E100 emulator supports C0 (instruction) coverage and C1 (branch) coverage.

Table 5.27 Code coverage definition

C0: Instruction coverage	All statements within the code are executed at least once.
C1: Branch coverage	All branches within the code are executed at least once.

The E100 emulator comes with up to a 2-Mbyte code coverage memory when using the C0 + C1 level coverage, and up to a 1-Mbyte code coverage memory when using the C1 level coverage.
With initial settings, the code coverage memory is allocated automatically to addresses in the ROM and RAM areas in this order.

5.11.2 Opening the Code Coverage Window

Choose Code -> Code Coverage from the View menu or click the Code Coverage toolbar button [].

The Code Coverage window is initially empty.

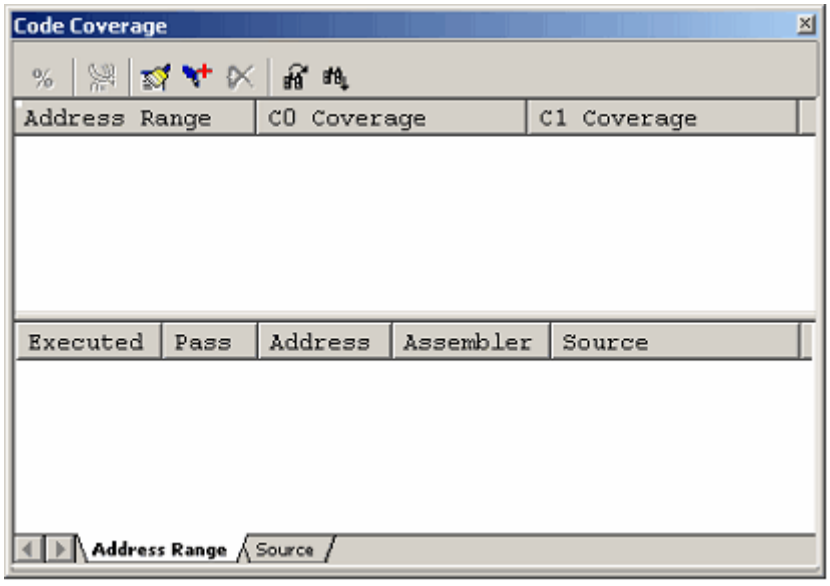


Figure 5.72 Code Coverage window

Downloaded from Arrow.com.

(1) Measurement method

The Code Coverage window consists of two sheets.

Table 5.28 Sheets of the Code Coverage window

Sheet name	Description
Address Range sheet	Measurement is performed on any address range.
Source sheet	Measurement is performed on a specified source file

The respective sheets permit multiple ranges to be registered.
Up to two instances of the Code Coverage window can be opened at the same time.

5.11.3 Allocating Code Coverage Memory (Hardware Resource)

(1) Memory allocation

Before code coverage can be measured, code coverage memory must be allocated to the addresses at which to be measured. Coverage data can be obtained from only the address range that has had memory allocated.
To allocate code coverage memory, use the Coverage Memory Allocation dialog box.
To open it, choose Hardware Settings from the pop-up menu of the Code Coverage window.

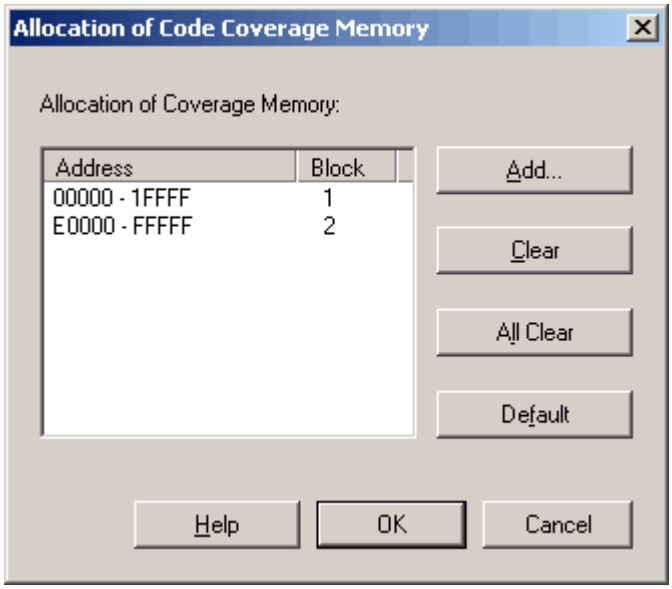


Figure 5.73 Allocation of Code Coverage Memory dialog box

When using the C0 level coverage and C1 level coverage, you can specify any of 1–8 blocks (maximum 2 Mbytes) each beginning with the 256-Kbyte boundary and any of 1–8 blocks (maximum 1 Mbyte) each beginning with the 128-Kbyte boundary as a code coverage measurement area respectively.
Contiguous blocks or noncontiguous blocks, either one, can be set.
With initial settings, the coverage memory is allocated to addresses in the ROM and RAM areas.

Downloaded from Arrow.com.

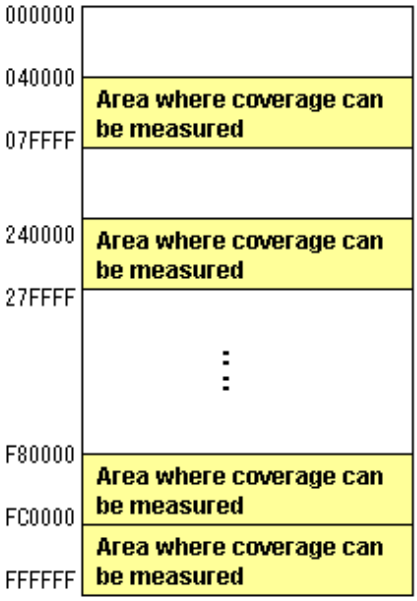


Figure 5.74 Schematic of coverage memory allocation

(2) Changing memory allocation

If coverage memory allocation is changed, the coverage data acquired from the addresses before being changed is retrieved from coverage memory into a coverage-only buffer.

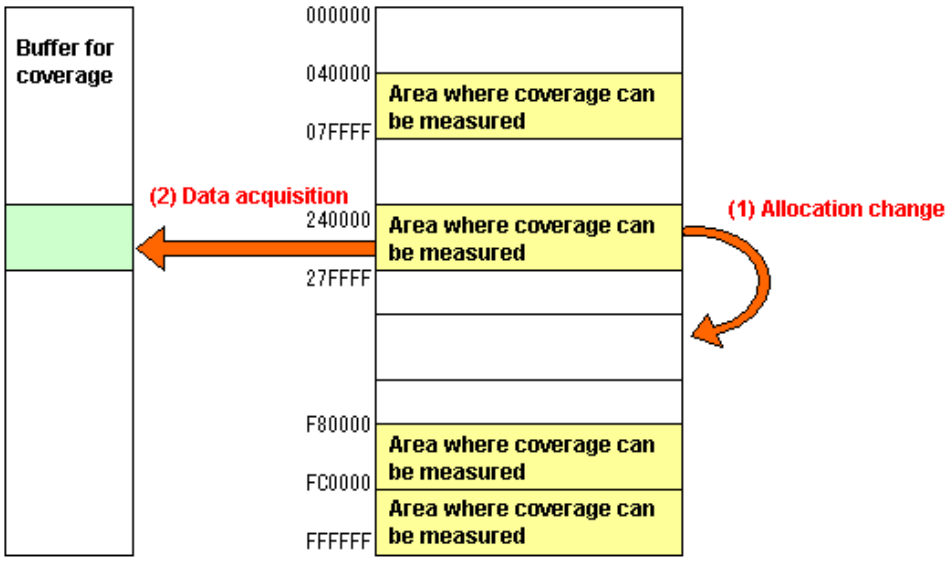


Figure 5.75 Schematic of coverage memory allocation change

The data accumulated in a coverage-only buffer is retained until the user clears it. However, data is not updated for the areas that have no coverage memory allocated.
The coverage information shown in the Code Coverage window includes the content of the coverage-only buffer.

Downloaded from Arrow.com.

5.11.4 Code Coverage in an Address Range

The Address Range sheet shows the code coverage information (C0 coverage and C1 coverage) acquired by the emulator from a user-specified address range.

Multiple address ranges can be registered.

An address range exceeding 2 Mbytes or even an area that has no coverage memory allocated can be specified. However, the coverage information on areas that have no coverage memory allocated is not updated

Areas where coverage information is not updated are displayed in gray.

An example display is shown below.

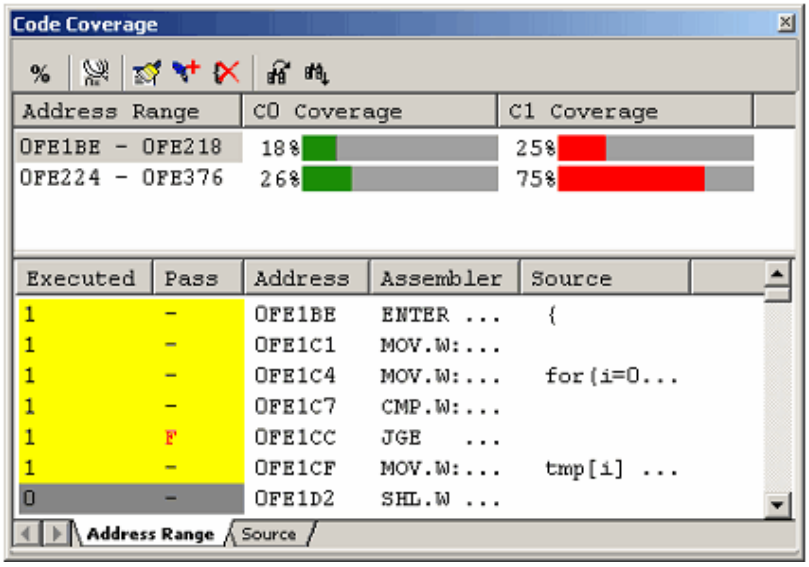


Figure 5.76 Code Coverage window (address specification)

The Code Coverage window is vertically divided into two by the splitter.

The upper pane shows the address ranges to be measured, C0 coverage and C1 coverage.

Table 5.29 Contents shown in the upper pane of the Code Coverage window

[Address Range]	Address range in which coverage is measured
[C0 Coverage]	C0 coverage in percentage and graph
[C1 Coverage]	C1 coverage in percentage and graph

The lower pane shows detailed information of the address range selected in the upper pane (assembly-language level).

Table 5.30 Contents shown in the lower pane of the Code Coverage window

[Executed]	1: The instruction was executed 0: The instruction was not executed
[Pass]	Condition for execution of a conditional branch instruction T: The condition was satisfied. F: The condition was not satisfied. T/F: The condition was satisfied in one case and not satisfied in another.
[Address]	Instruction address
[Assembler]	Disassembled program
[Source]	C/C++ or assembler source program

Downloaded from Arrow.com.

The acquired coverage information is accumulated in memory until the user clears it.
When you double click Assembler code shown in the Address Range sheet, the corresponding source code is shown in the Editor window.
Be sure that source code is not displayed in the cases listed below.

- A source file that corresponds the assembler line does not exist.
- A source line that corresponds the assembler line does not exist.
- Where no debug information is included, such as where the assembler line is a library.

5.11.5 Adding Address Ranges

Follow the procedure described below to add address ranges.

(1) From the Address Range sheet of the Code Coverage window

1. Right-click in the upper pane of the Address Range sheet and choose Add Range from the pop-up menu.

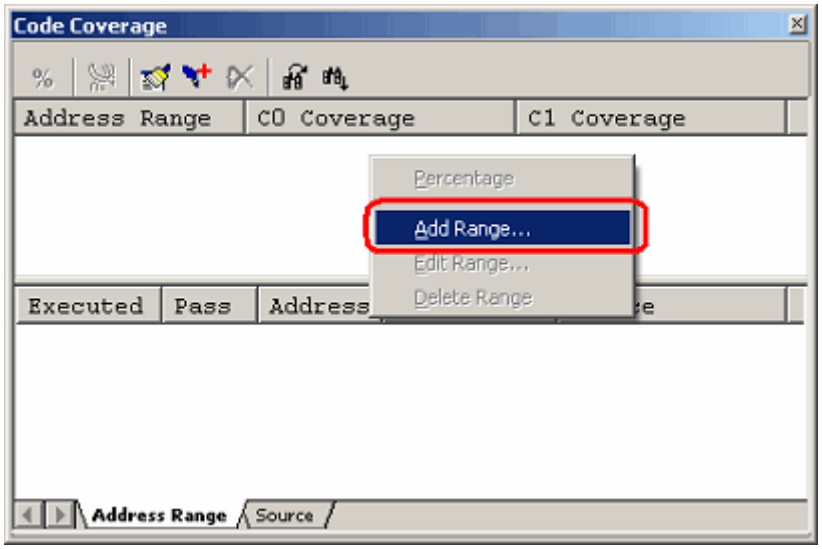


Figure 5.77 Code Coverage window

2. In the Add Address Range dialog box that is displayed, enter an address range.

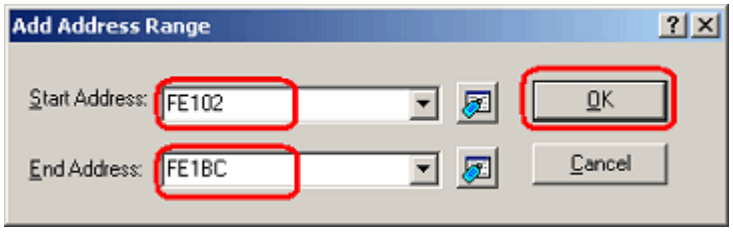


Figure 5.78 Add Address Range dialog box

Downloaded from Arrow.com.

3. The address range you have added will be displayed in the upper pane of the Code Coverage window.

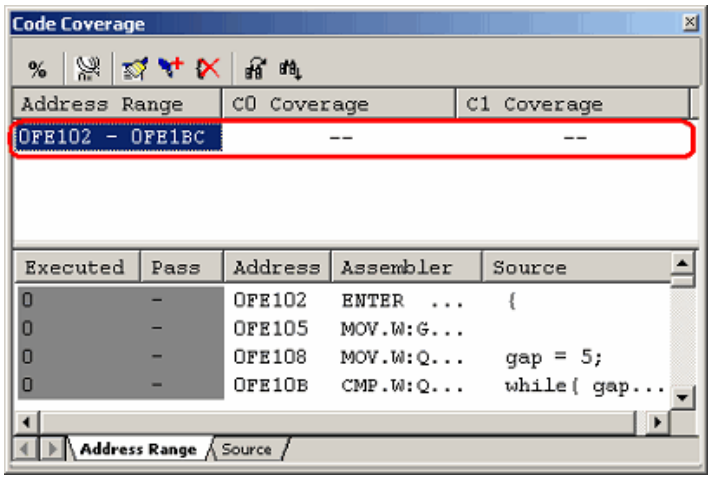


Figure 5.79 Code Coverage window

5.11.6 Changing Address Ranges

Follow the procedure described below to change address ranges.

(1) From the Address Range sheet of the Code Coverage window

1. Select an address range you want to change in the Address Range sheet and while holding it selected, choose Edit Range from the pop-up menu.

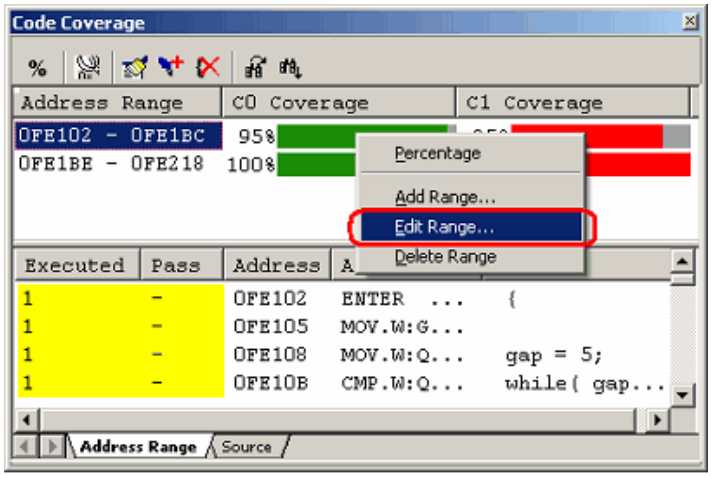


Figure 5.80 Code Coverage window

2. In the Edit Address Range dialog box that is displayed, change the address range.

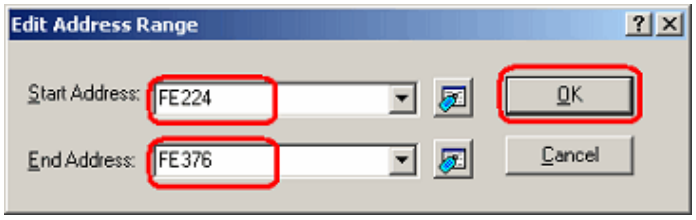


Figure 5.81 Edit Address Range dialog box

Downloaded from Arrow.com.

3. The address range you have changed will be displayed in the upper pane of the Code Coverage window.

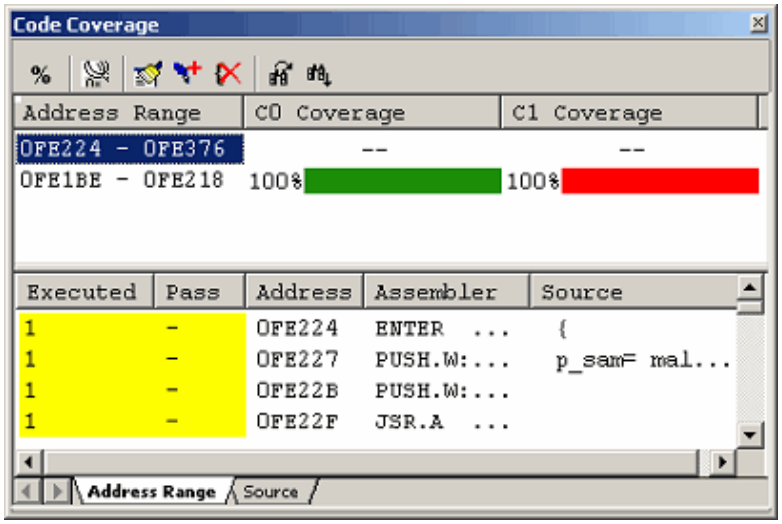


Figure 5.82 Code Coverage window

5.11.7 Removing Address Ranges

Follow the procedure described below to remove address ranges.

(1) From the Address Range sheet of the Code Coverage window

1. Select an address range you want to remove in the Address Range sheet and while holding it selected, choose Delete Range from the pop-up menu.

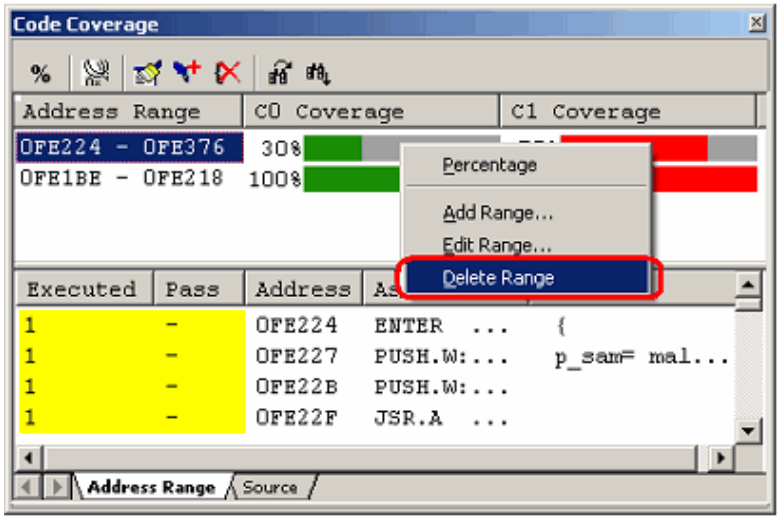


Figure 5.83 Code Coverage window

Downloaded from Arrow.com.

2. A dialog box asking for your confirmation will be displayed.

Choose to save or not save coverage data. To save, specify a file name and then click the OK button. If you do not save, simply click the OK button.

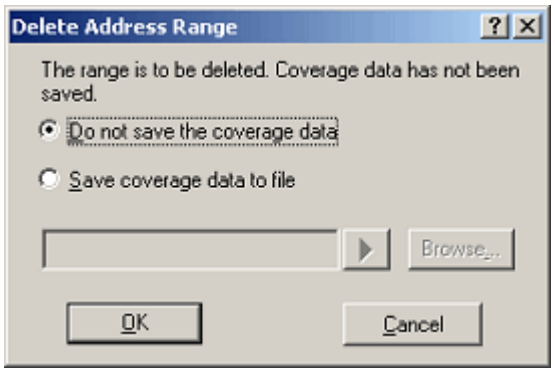


Figure 5.84 Delete Address Range dialog box

3. The address range you have selected will be removed.

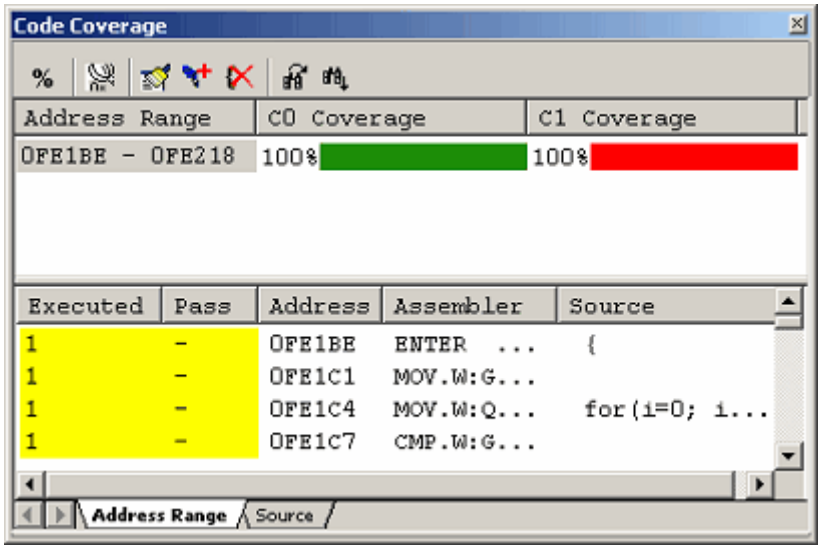


Figure 5.85 Code Coverage window

Downloaded from Arrow.com.

5.11.8 Code Coverage in a Source File

The Source sheet shows the code coverage information (C0 coverage and C1 coverage) acquired by the emulator from a user-specified source file.

Multiple source files can be registered.

A source file exceeding 2 Mbytes in size or even a file that includes an area that has no coverage memory allocated can be specified.

However, the coverage information on areas that have no coverage memory allocated is not updated.

Address lines where coverage information is not updated are displayed in gray.

An example display is shown below.

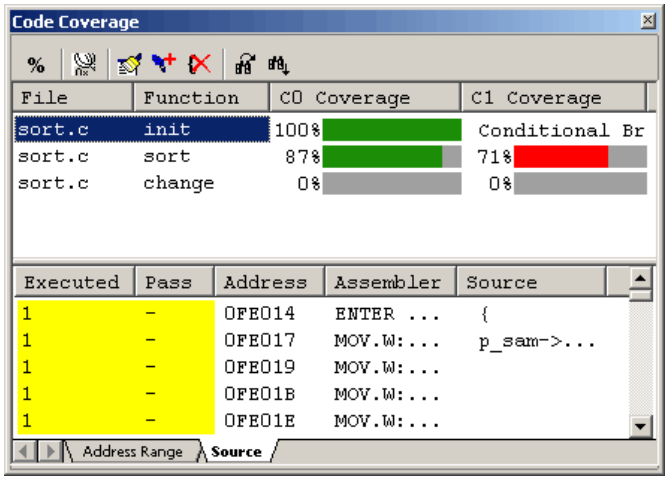


Figure 5.86 Code Coverage window (source file specification):

The Code Coverage window is vertically divided into two by the splitter.

The upper pane shows the address ranges to be measured (file and function names), C0 coverage and C1 coverage.

Table 5.31 Contents shown in the upper pane of the Code Coverage window

[File]	File name
[Function]	Function name
[C0 Coverage]	C0 coverage in percentage and graph
[C1 Coverage]	C1 coverage in percentage and graph

The lower pane shows detailed information of the address range selected in the upper pane (assembly-language level).

Table 5.32 Contents shown in the lower pane of the Code Coverage window

[Executed]	1: Instructions was executed. 0: Instructions was not executed.
[Pass]	Condition for execution of a conditional branch instruction T: The condition was satisfied. F: The condition was not satisfied. T/F: The condition was satisfied in one case and not satisfied in another.
[Address]	Instruction address
[Assembler]	Disassembled program
[Source]	C/C++ or assembler source program

The acquired coverage information is accumulated in memory until the user clears it.

Downloaded from Arrow.com.

5.11.9 Adding Source Files

Follow the procedure described below to add source files.

(1) From the Source sheet of the Code Coverage window

1. Right-click in the upper pane of the Source sheet and choose Add Range from the pop-up menu.

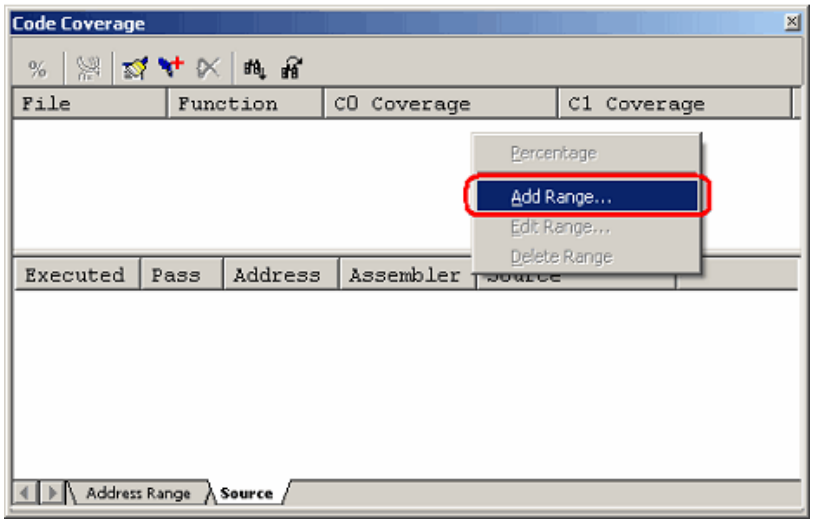


Figure 5.87 Code Coverage window

2. In the Add Source Files dialog box that is displayed, enter a file name.

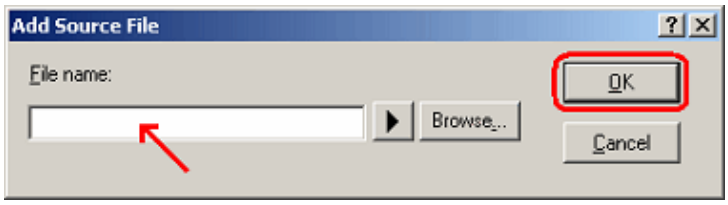


Figure 5.88 Add Source File dialog box

3. The source file you have added and the function names included in it will be displayed in the upper pane of the Code Coverage window.

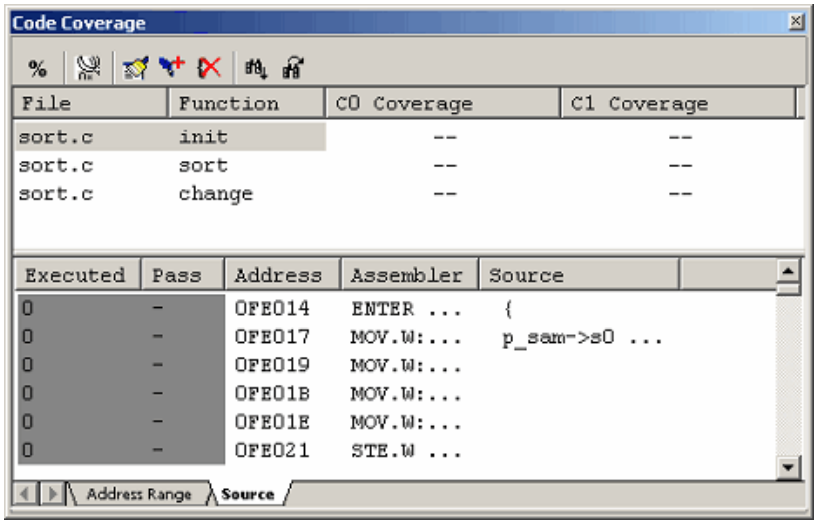


Figure 5.89 Code Coverage window

Downloaded from Arrow.com.

5.11.10 Removing Source Files

Delete source files by the following methods.

(1) From the Source sheet of the Code Coverage window

1. Select a function you want to remove in the upper pane of the Source sheet and while holding it selected, choose Delete Range from the pop-up menu.

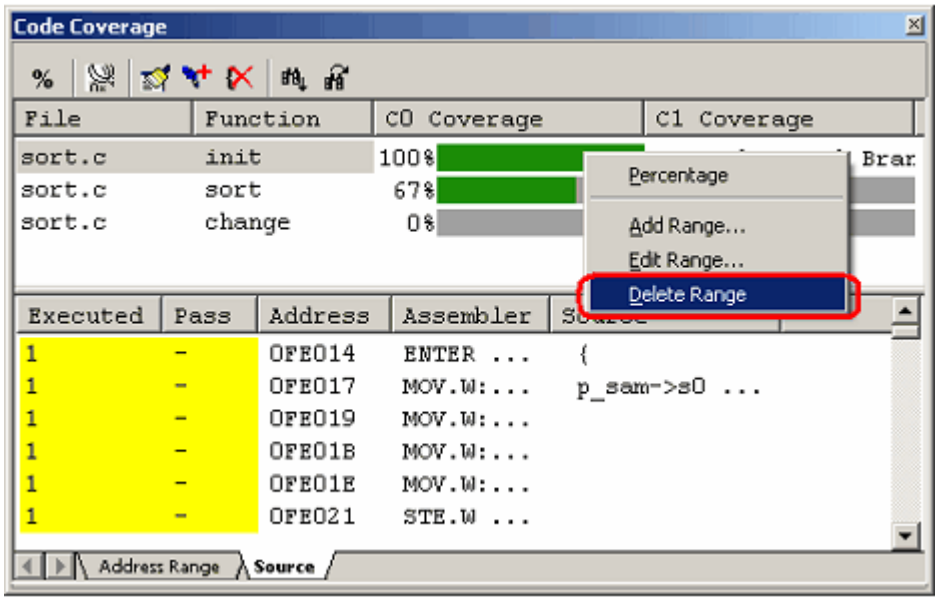


Figure 5.90 Code Coverage window

Downloaded from Arrow.com.

2. A dialog box prompting for your confirmation will be displayed. Choose to save or not save coverage data. To save, specify a file name and then click the OK button. If you do not save, simply click the OK button.

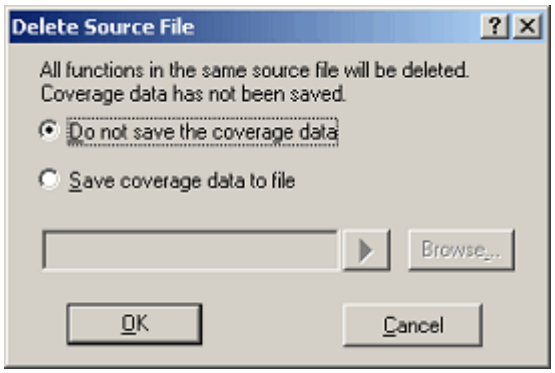


Figure 5.91 Delete Source File dialog box

3. All functions included in the selected source file will be removed.

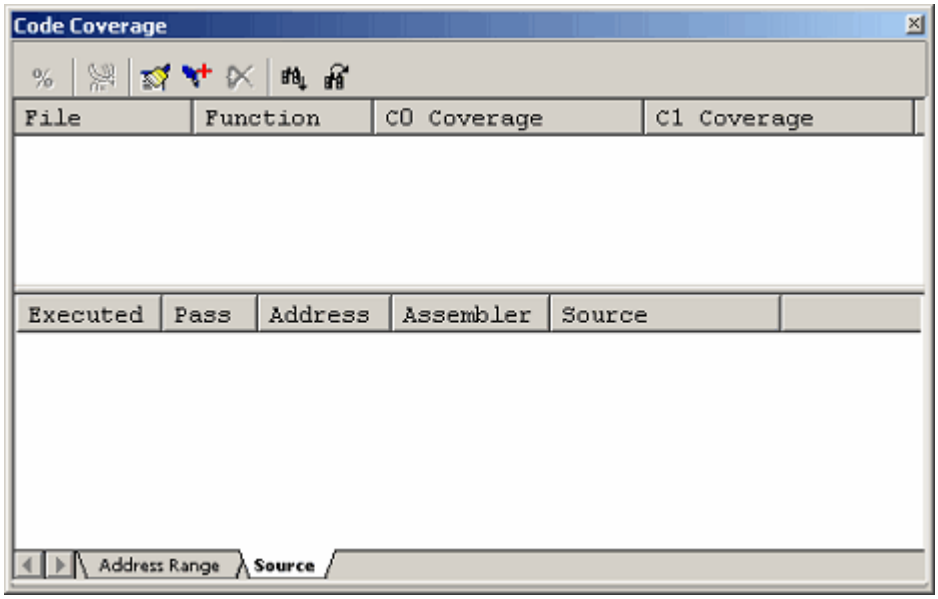


Figure 5.92 Code Coverage window

Downloaded from Arrow.com.

5.11.11 Showing Percentages and Graphs

After the program has stopped, right-click in the upper pane of the Code Coverage window and choose Percentage from the pop-up menu. The emulator will start calculating C0 (instruction) coverage and C1 (branch) coverage for each address range. When the calculation is completed, coverage information is displayed in the upper pane as percentages and graphs.

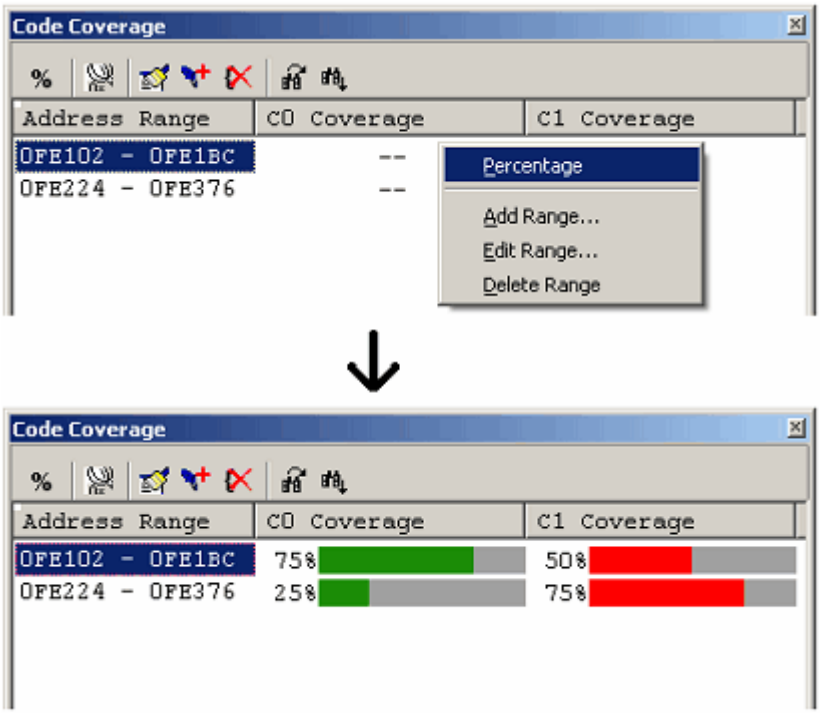


Figure 5.93 Code Coverage window

Downloaded from Arrow.com.

5.11.12 Using the Sort Function

Clicking on a header column in the upper pane of the Code Coverage window allows the coverage data to be sorted.

(1) Clicking on the File column

The data can be sorted by the file name. Lines in the same file are sorted by function name.
Example:

File	Function	C0 Coverage
file1.cpp	func1	40% ■■■■
file1.cpp	func2	10% ■
file1.cpp	func3	80% ■■■■■■■■
file1.cpp	func4	70% ■■■■■■■
file2.cpp	func1	20% ■■
file2.cpp	func2	60% ■■■■■■
file2.cpp	func3	90% ■■■■■■■■■
file3.cpp	func1	0%
file3.cpp	func2	30% ■■■
file3.cpp	func3	10% ■

(2) Clicking on the C0 Coverage column

The data can be sorted by coverage rate.
First clicking on the column sorts the values in decreasing order. Clicking on the column again sorts the values in ascending order.
Example:

File	Function	C0 Coverage
file2.cpp	func3	90% ■■■■■■■■■
file1.cpp	func3	80% ■■■■■■■■
file1.cpp	func4	70% ■■■■■■■
file2.cpp	func2	60% ■■■■■■
file1.cpp	func1	40% ■■■■
file3.cpp	func2	30% ■■■
file2.cpp	func1	20% ■■
file1.cpp	func2	10% ■
file3.cpp	func3	10% ■
file3.cpp	func1	0%

Downloaded from Arrow.com.

(3) Clicking on the C0 Coverage and the File columns in this order

The data for each file is sorted by the coverage rate in descending order.
Example:

```
  File       Function     C0 Coverage
    ---------------------------------------
    file1.cpp func3      80% ■■■■■■■■
    file1.cpp func4      70% ■■■■■■■
    file1.cpp func1      40% ■■■■
    file1.cpp func2      10% ■
    file2.cpp func3      90% ■■■■■■■■■
    file2.cpp func2      60% ■■■■■■
    file2.cpp func1      20% ■■
    file3.cpp func2      30% ■■■
    file3.cpp func3      10% ■
    file3.cpp func1       0%
```

5.11.13 Searching for Unexecuted Lines

Search for unexecuted lines in a selected address range or function. When you click the Find button  in the toolbar, the Find dialog box shown below appears.

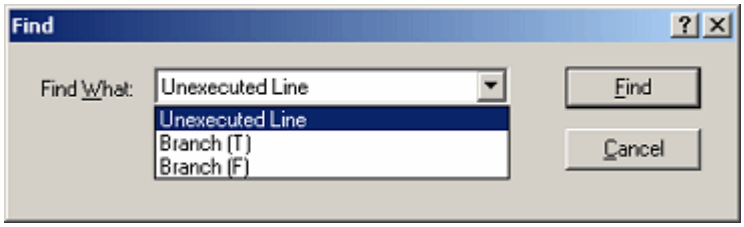


Figure 5.94 Find dialog box

Following three search options are available.

Table 5.33 Search options

Unexecuted Line	Instructions not executed yet
Branch (T)	Branch instructions only tested as TRUE.
Branch (F)	Branch instructions only tested as FALSE.

Clicking the Find Next button () starts a search.
When a matching instruction is found, the line of that instruction is highlighted.
When no matching instructions are found, a message is displayed.

Downloaded from Arrow.com.

5.11.14 Clearing Code Coverage Information

(1) Clearing the code coverage information of the specified range

Selecting Clear Coverage Range from the pop-up menu opens the Clear Address Range dialog box.

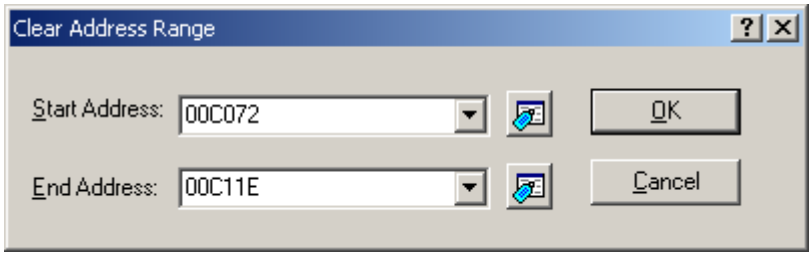


Figure 5.95 Clear Address Range dialog box

Enter the start and end addresses of the range to be cleared. Clicking the OK button clears the coverage information of the selected range.

(2) Clearing all the code coverage information

Selecting Clear the Entire Coverage from the pop-up menu clears all the code coverage information.

5.11.15 Updating Coverage Information

Selecting Refresh from the pop-up menu updates the content of the Code Coverage window.
If Lock Refresh has been selected, the information is not automatically updated when the program breaks. To view the latest information, therefore, you need to update manually.

5.11.16 Preventing Update of Coverage Information

Selecting Lock Refresh from the pop-up menu prevents update of the Code Coverage window while the user program execution is stopped.

Downloaded from Arrow.com.

5.11.17 Saving the Code Coverage Information to a File

You can save the code coverage information of the currently selected sheet to a file.
Selecting Save Data from the pop-up menu opens the Save Coverage Data dialog box.

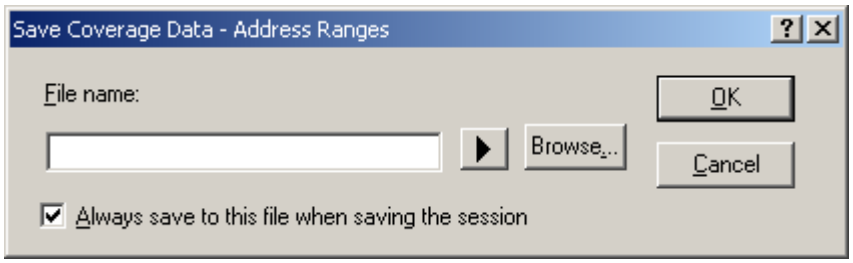


Figure 5.96 Save Coverage Data dialog box

Enter a file name in which you want the information to be saved.
If the file extension is omitted, ".cov" will automatically be added as the file extension.
If you specify an existing file name, the file is overwritten.

5.11.18 Loading Code Coverage Information from a File

You can load a code coverage information file.
Selecting Load Data from the pop-up menu opens the Load Coverage Data dialog box.

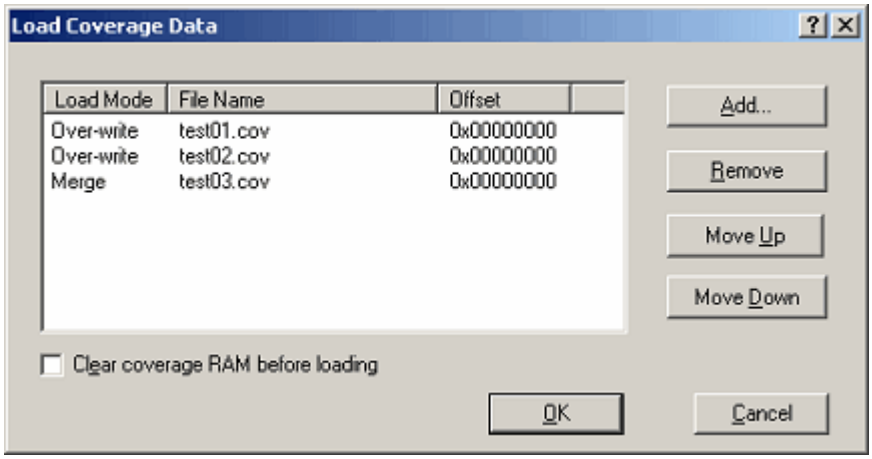


Figure 5.97 Load Coverage Data dialog box

Downloaded from Arrow.com.

Clicking on the Add button opens the Add Coverage Files dialog box shown below.

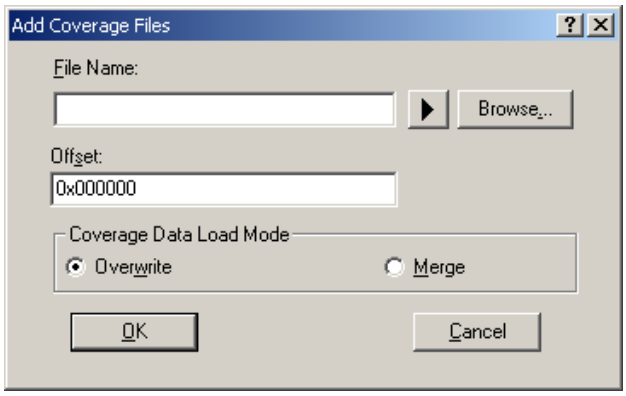


Figure 5.98 Add Coverage Files dialog box

Use this dialog box to specify a coverage information file you want to load. You can also specify a load mode and offset for each file you load.
The only file extension available is ".cov". An error message will appear if any other file extension is entered.
The files you add will be listed in the Load Coverage Data dialog box. The files will be loaded in the order in which they are listed. If necessary, use the Up or Down button to change the order.

CAUTION
If the coverage information file you are loading is of a source file type, you cannot specify an offset.

5.11.19 Coverage Information File Load Modes

Coverage information file load modes are schematically shown below.

(1) When "Overwrite" has been selected

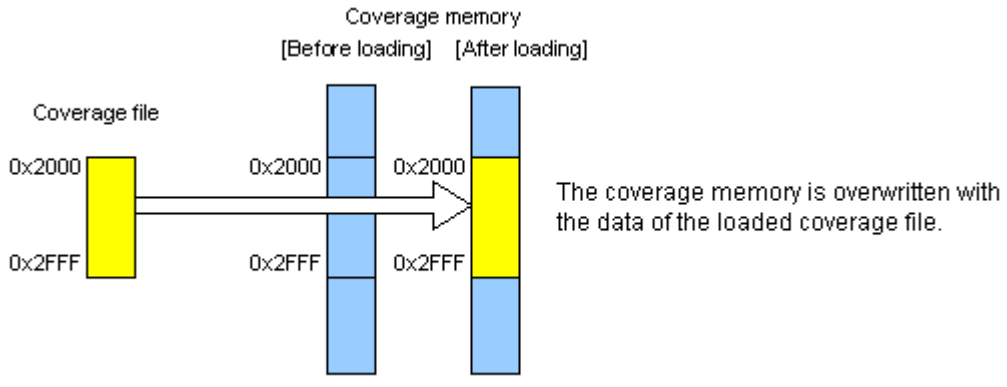


Figure 5.99 Schematic of the overwrite mode

Downloaded from Arrow.com.

(2) When "Merge" has been selected

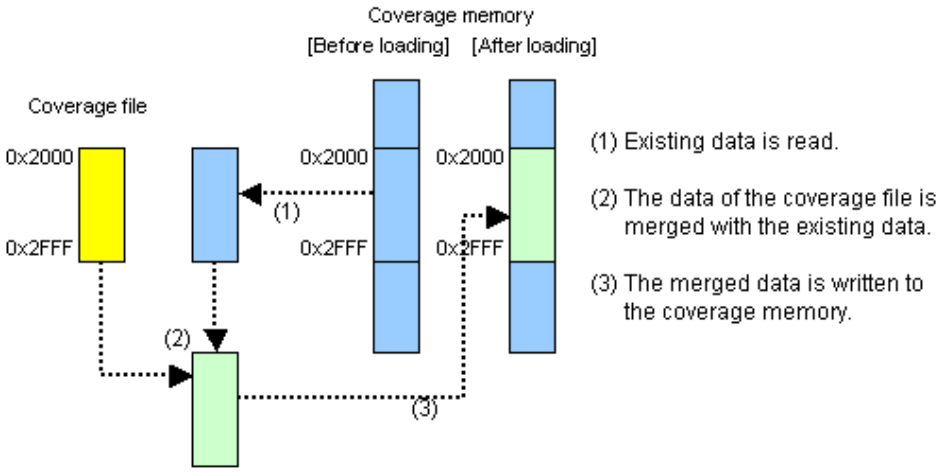


Figure 5.100 Schematic of the merge mode

(3) Application example of merge mode

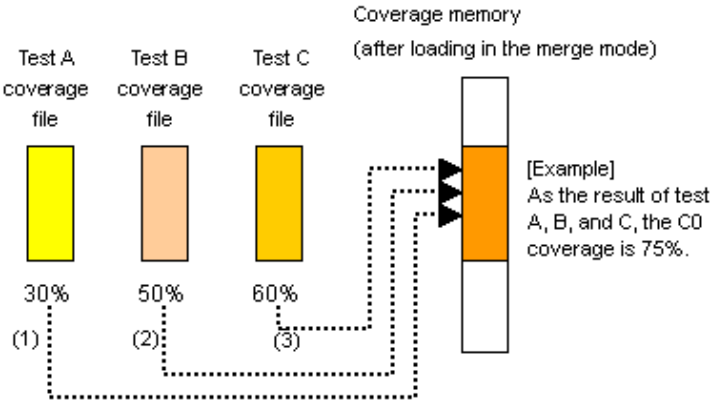


Figure 5.101 Schematic of merge-mode application

[Procedure]

(1) Open the Load Coverage Data dialog box.
To begin with, select the "Clear coverage RAM before loading" check box.
(2) Add a coverage file for test A in the merge mode.
(3) Add a coverage file for test B in the merge mode.
(4) Add a coverage file for test C in the merge mode.
(5) Click the OK button.

You have now finished merging three files.
By calculating percentages in the Code Coverage window, you can view the coverage (in percentage) of those tests as a whole.
Furthermore, the merged data can be saved to a file, so that you can manage those data as a single file.

Downloaded from Arrow.com.

5.11.20 Displaying Code Coverage Information in the Editor Window

When the Editor window is open in the source mode, the results of coverage are displayed in its Code Coverage column.
Part of the Code Coverage column that corresponds to a source line where an instruction has been executed is highlighted.
If the user changes any setting regarding the coverage information in the Code Coverage window, the content of the corresponding Code Coverage column will also be updated.

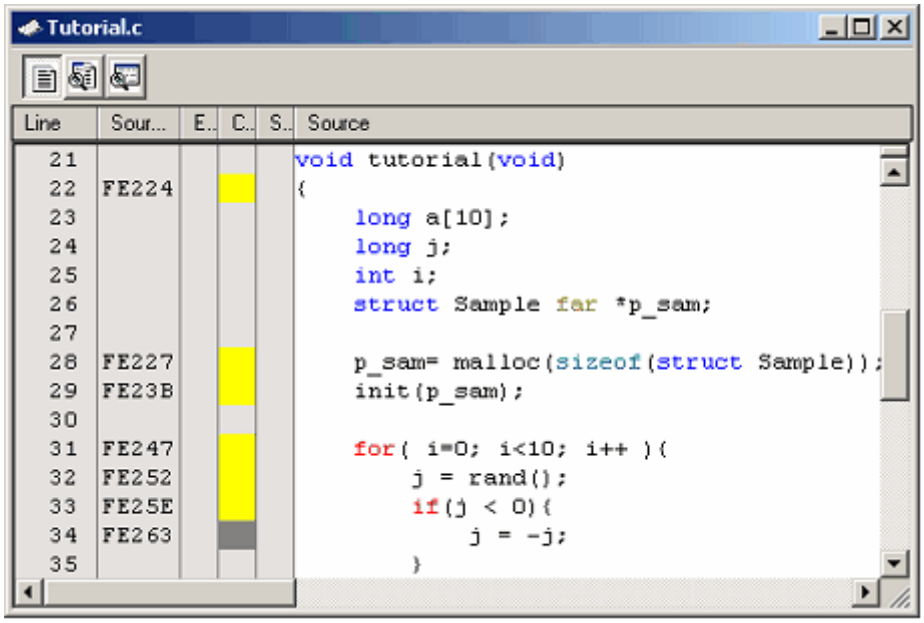


Figure 5.102 Example of code coverage results

Downloaded from Arrow.com.

5.12 Acquiring Data Coverage

5.12.1 Acquiring Data Coverage

The E100 emulator has code coverage, data coverage and realtime profile functions usable exclusively to each other.
To use the data coverage function, choose Data Coverage in the Exclusive Functions section on the System page of the Configuration Properties dialog box.

Data coverage is the function to indicate "what kinds of accesses have been made" to the data area.
This function allows the emulator to acquire access information per byte without causing a program to break. Therefore, the realtime operation of the user program will not be affected.
The coverage result is updated upon a break.

The E100 emulator comes with 512 Kbytes of data coverage memory.
With initial settings, the data coverage memory is automatically assigned at addresses in the ROM and RAM areas in this order.

5.12.2 Opening the Data Coverage Window

Choose Code -> Data Coverage from the View menu or click the Data Coverage toolbar button [].

The Data Coverage window is initially empty.

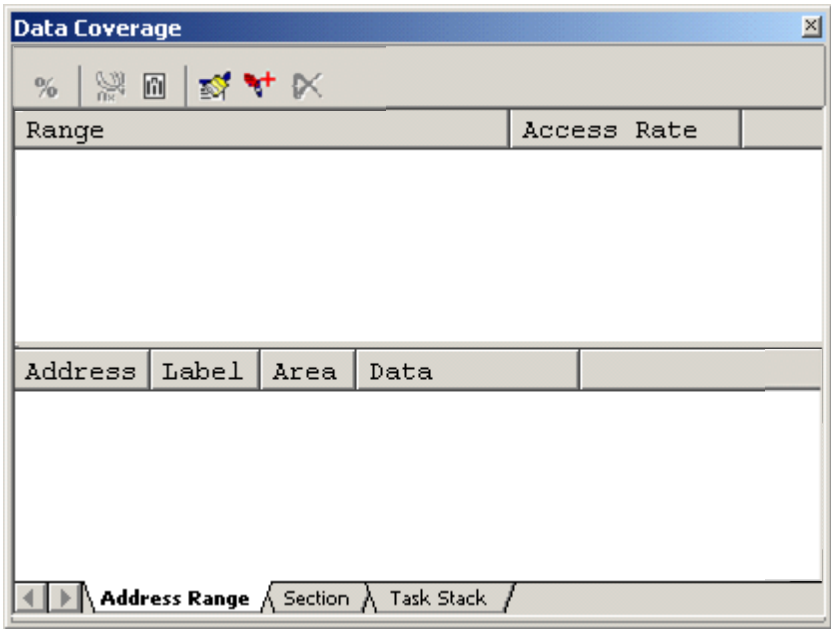


Figure 5.103 Data Coverage window

Downloaded from Arrow.com.

(1) Measurement method

The Data Coverage window consists of three sheets.

Table 5.34 Sheets of the Data Coverage window

Sheet	Description
Address Range	Measurement is performed on any address range.
Section	Measurement is performed on a specified section.
Task Stack	Measurement is made of all task stack areas.

The respective sheets permit multiple ranges to be registered.
The Task Stack sheet supports only automatic registration.
Up to three instances of the Data Coverage window can be opened at the same time.

5.12.3 Allocating Data Coverage Memory (Hardware Resource)

(1) Memory allocation

Before data coverage can be measured, data coverage memory must be assigned to the target address range. Coverage data can be obtained from only the address range that has had memory allocated.
To allocate data coverage memory, use the Allocation of Data Coverage Memory dialog box. To open this dialog box, select [Hardware Settings…] from the pop-up menu of the Data Coverage window.

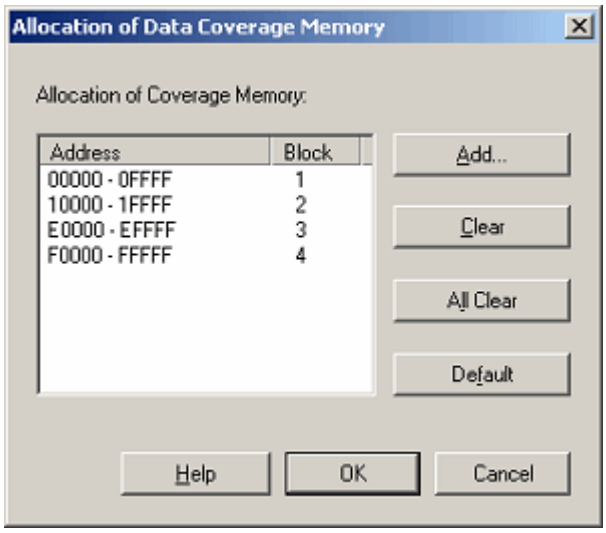


Figure 5.104 Allocation of Data Coverage Memory dialog box

You can specify any of blocks 1 to 8 (up to 512 Kbytes) each beginning with a 64-Kbyte boundary as data coverage measurement areas.
Either contiguous blocks or non-contiguous blocks can be assigned.
With initial settings, the coverage memory is automatically assigned at addresses in the ROM and RAM areas.

Downloaded from Arrow.com.

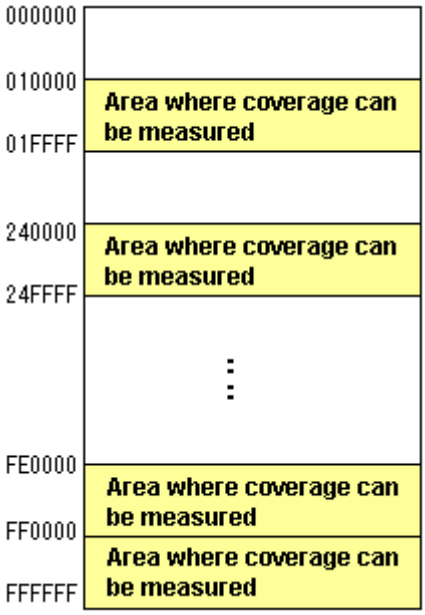


Figure 5.105 Schematic of data coverage memory allocation

(2) Changing memory allocation

If coverage memory allocation is changed, the coverage data acquired from the addresses before being changed is retrieved from coverage memory into a coverage-only buffer.

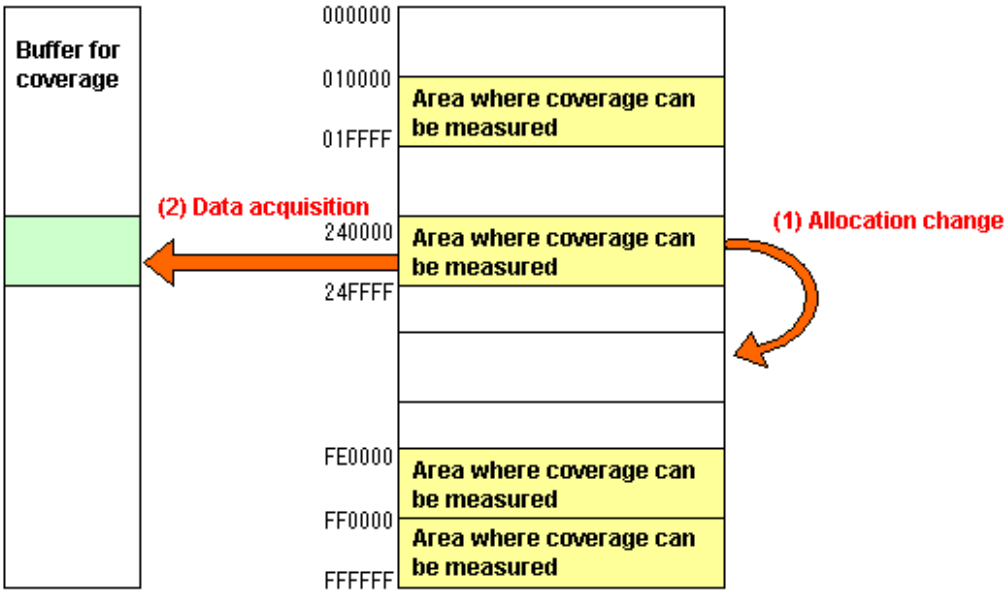


Figure 5.106 Schematic of data coverage memory allocation change

The acquired coverage information is accumulated in the coverage buffer until the user clears it. However, the coverage information on areas that have no coverage memory allocated is not updated.
The coverage information shown in the Data Coverage window includes the content of the coverage-only buffer.

Downloaded from Arrow.com.

5.12.4 Data Coverage in an Address Range

The E100 emulator shows the access information it acquired from a user-specified address range.

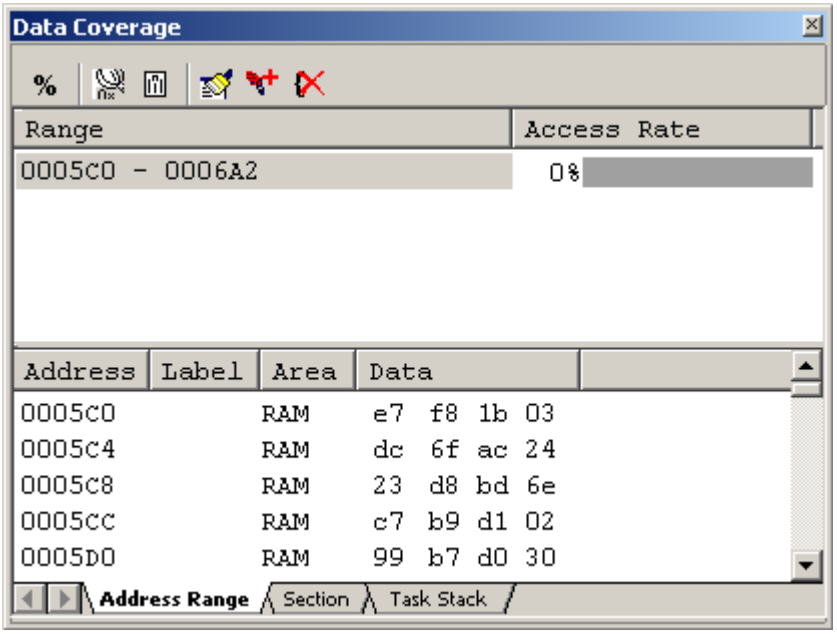


Figure 5.107 Data Coverage window (address specification)

The Data Coverage window is vertically divided into two by the splitter.
The upper pane shows the address ranges to be measured and access rates.

Table 5.35 Contents in the upper pane of the Data Coverage window

[Range]	Address range in which coverage is measured
[Access Rate]	Access rate in percentage and graph

The lower pane shows detailed information of the address range selected in the upper pane.

Table 5.36 Contents shown in the lower pane of the Data Coverage window

[Address]	Address value
[Label]	Label name
[Area]	Memory area (FlashROM, RAM, SFR) This column is blank when the area is unused.
[Data]	Memory data The accessed data has its background displayed in purple.

If located outside the coverage memory allocated area, address lines are displayed in gray. Although the existing coverage information of those addresses is retained, the coverage information will not be updated by program execution.
The acquired coverage information is accumulated in memory until the user clears it.

Downloaded from Arrow.com.

5.12.5 Adding Address Ranges

Follow the procedure described below to add address ranges.

(1) From the Address Range sheet of the Data Coverage window

1. Right-click in the upper pane of the Address Range sheet and choose Add Range from the pop-up menu.

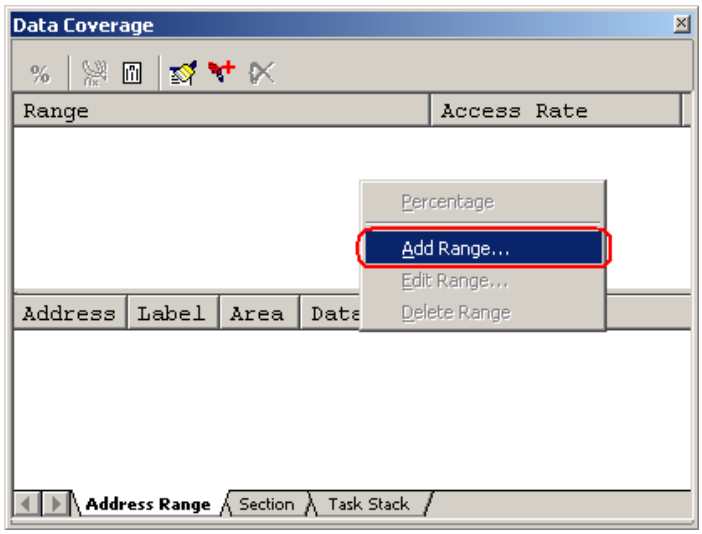


Figure 5.108 Data Coverage window

2. In the Add Address Ranges dialog box that is displayed, enter an address range.

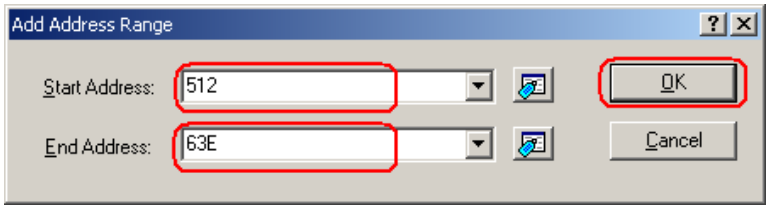


Figure 5.109 Add Address Range dialog box

3. The address range you have added will be displayed in the upper pane of the Data Coverage window.

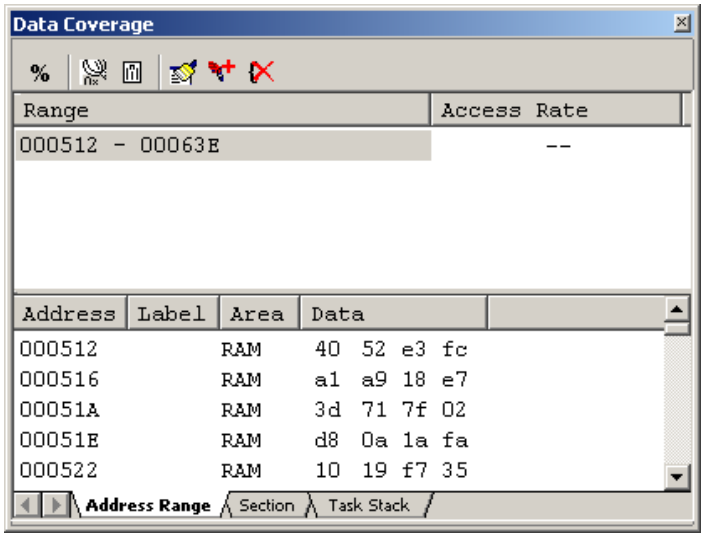


Figure 5.110 Data Coverage window

Downloaded from Arrow.com.

5.12.6 Changing Address Ranges

Follow the procedure described below to change address ranges.

(1) From the Address Range sheet of the Data Coverage window

1. Select an address range you want to change in the Address Range sheet and while holding it selected, choose Edit Range from the pop-up menu.

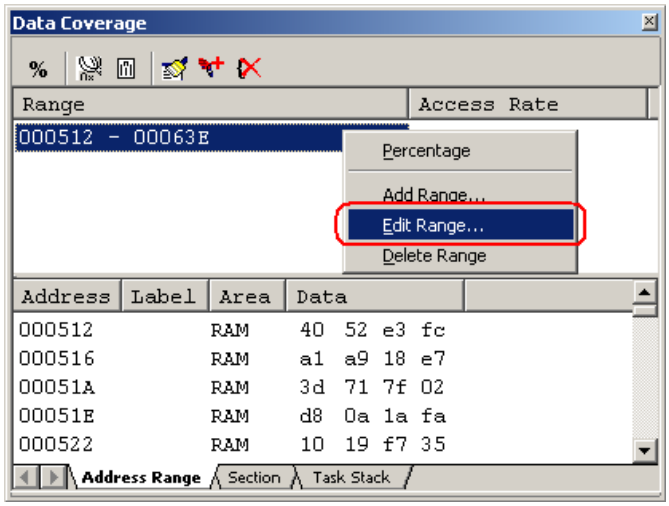


Figure 5.111 Data Coverage window

2. In the Edit Address Range dialog box that is displayed, change the address range.

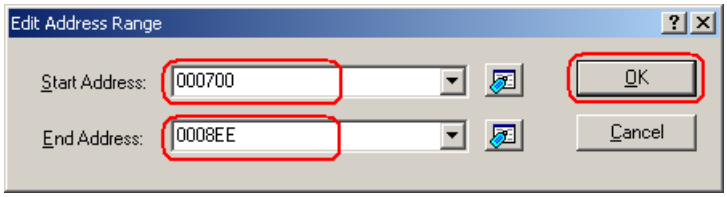


Figure 5.112 Edit Address Range dialog box

3. The address range you have changed will be displayed in the upper pane of the Data Coverage window.

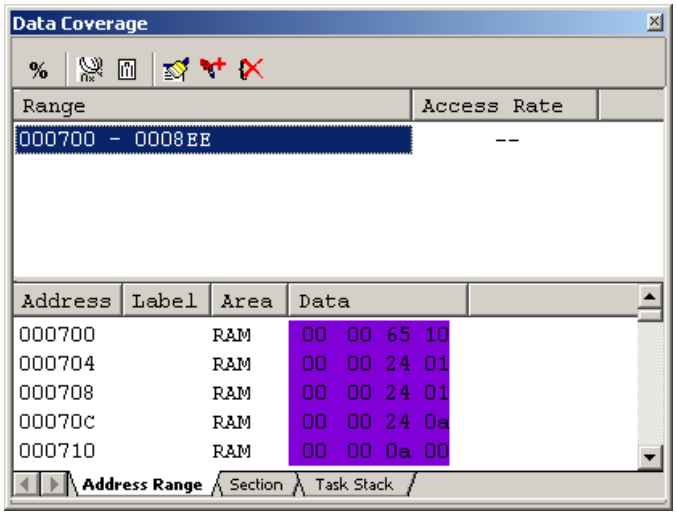


Figure 5.113 Data Coverage window

Downloaded from Arrow.com.

5.12.7 Removing Address Ranges

Follow the procedure described below to remove address ranges.

(1) From the Address Range sheet of the Data Coverage window

1. Select an address range you want to remove in the Address Range sheet and while holding it selected, choose Delete Range from the pop-up menu.

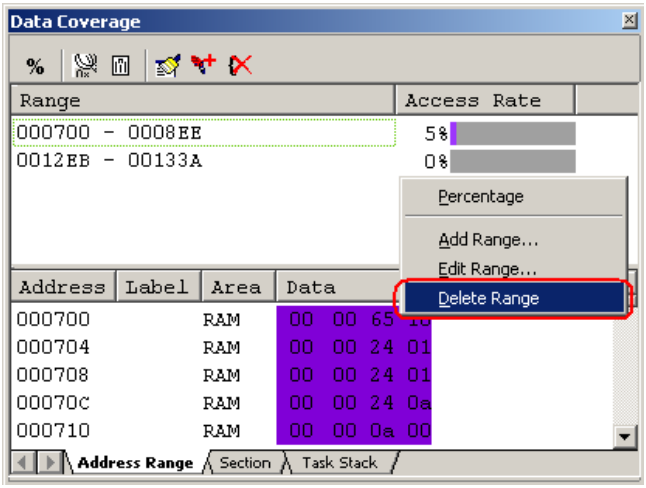


Figure 5.114 Data Coverage window

2. A dialog box prompting for your confirmation will be displayed. Choose to save or not save coverage data. To save, click the Yes button. If you do not save, click the No button.

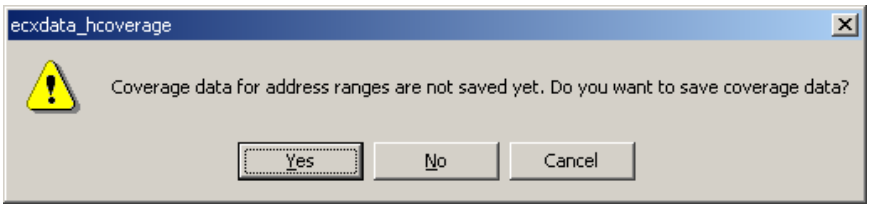


Figure 5.115 Confirmation of Edit Address Range dialog box

3. The address range you have selected will be removed.

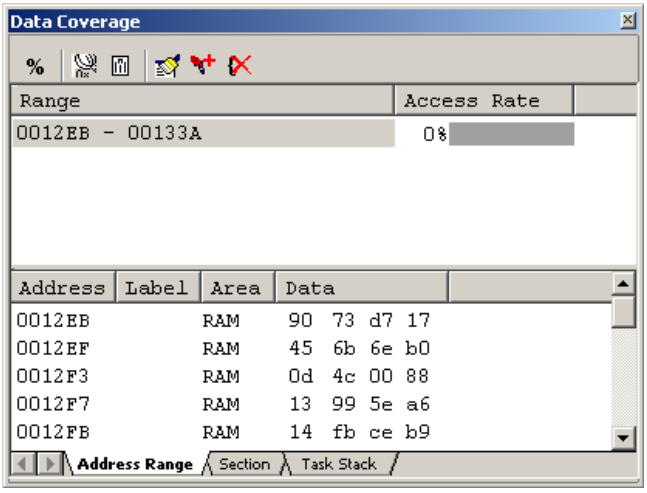


Figure 5.116 Data Coverage window

Downloaded from Arrow.com.

5.12.8 Data Coverage in a Section

The E100 emulator shows the access information it acquired from a user-specified section.

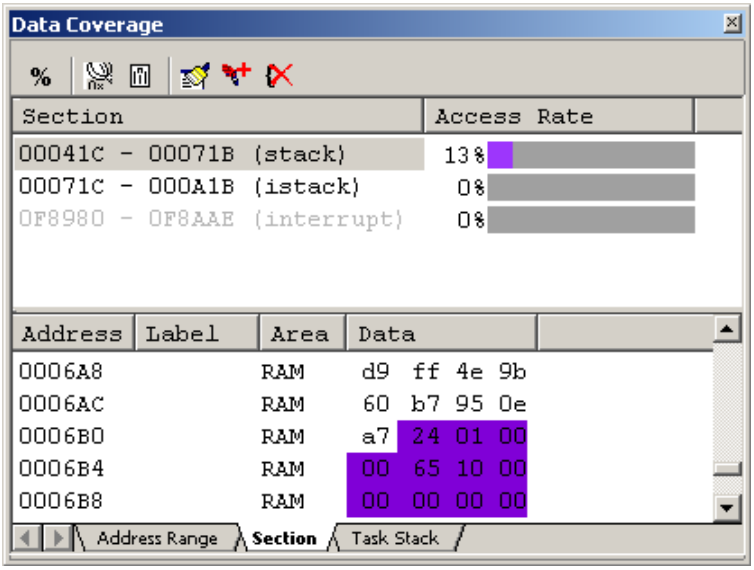


Figure 5.117 Data Coverage window (section name specification)

The Data Coverage window is vertically divided into two by the splitter.
The upper pane shows the address ranges (section names) to be measured and access rates.

Table 5.37 Contents in the upper pane of the Data Coverage window

[Section]	Address range (section) in which coverage is measured
[Access Rate]	Access rate in percentage and graph

The lower pane shows detailed information of the address range selected in the upper pane.

Table 5.38 Contents shown in the lower pane of the Data Coverage window

[Address]	Address value
[Label]	Label name
[Area]	Memory area (FlashROM, RAM, SFR) This column is blank when the area is unused.
[Data]	Memory data The accessed data has its background displayed in purple.

If located outside the coverage memory allocated area, address lines are displayed in gray. Although the existing coverage information of those addresses is retained, the coverage information will not be updated by program execution.
The acquired coverage information is accumulated in memory until the user clears it.

Downloaded from Arrow.com.

5.12.9 Adding Sections

Follow the procedure described below to add sections.

(1) From the Section sheet of the Data Coverage window

1. Right-click in the upper pane of the Section sheet and choose Add Range from the pop-up menu.

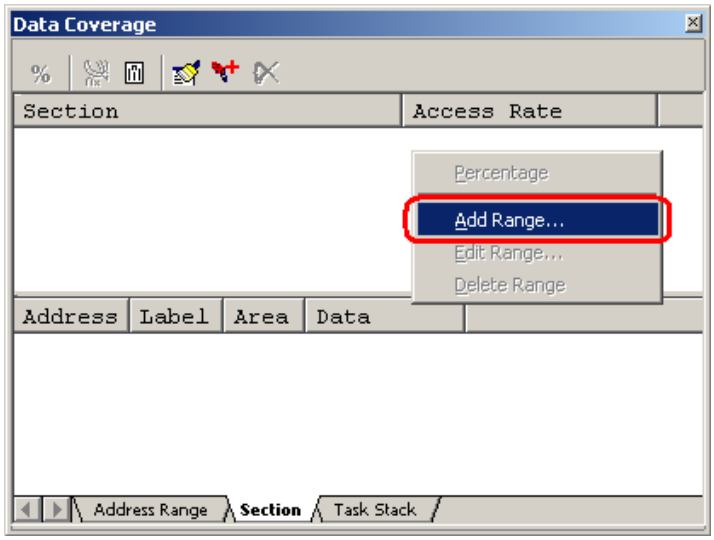


Figure 5.118 Data Coverage window

2. In the Add A Section dialog box that is displayed, enter a section name.



Figure 5.119 Add A Section dialog box

3. The address range (section name) you have added will be displayed in the upper pane of the Data Coverage window.

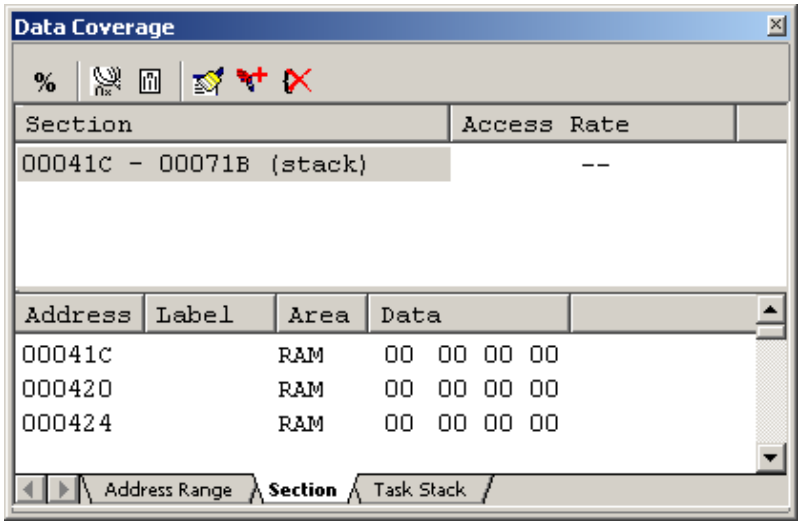


Figure 5.120 Data Coverage window

Downloaded from Arrow.com.

5.12.10 Removing Sections

Follow the procedure described below to remove sections.

(1) From the Section sheet of the Data Coverage window

1. Select a section name you want to remove in the Section sheet and while holding it selected, choose Delete Range from the pop-up menu.

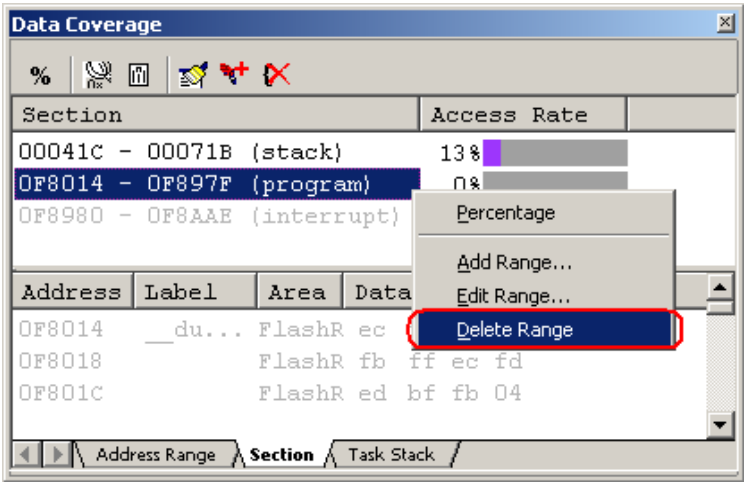


Figure 5.121 Data Coverage window

2. A dialog box prompting for your confirmation will be displayed. Choose to save or not save coverage data. To save, click the Yes button and specify a file name. If you do not save, click the No button.

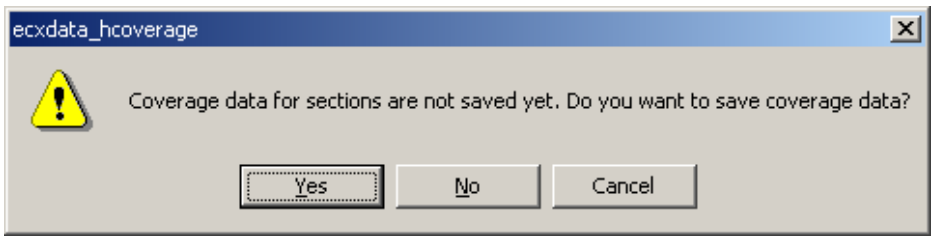


Figure 5.122 Confirmation of Removing Section dialog box

3. The section name you have selected will be removed.

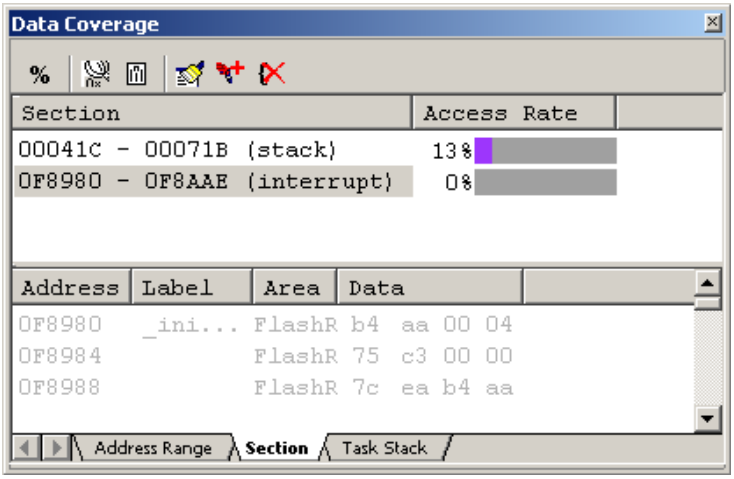


Figure 5.123 Data Coverage window

Downloaded from Arrow.com.

5.12.11 Data Coverage in a Task Stack

The Task Stack sheet shows the access information acquired from a task stack.

Task stacks are automatically registered.

You cannot add, remove or change any task.

If tasks are changed pursuant to alterations of the user program, for example, the window is automatically updated.

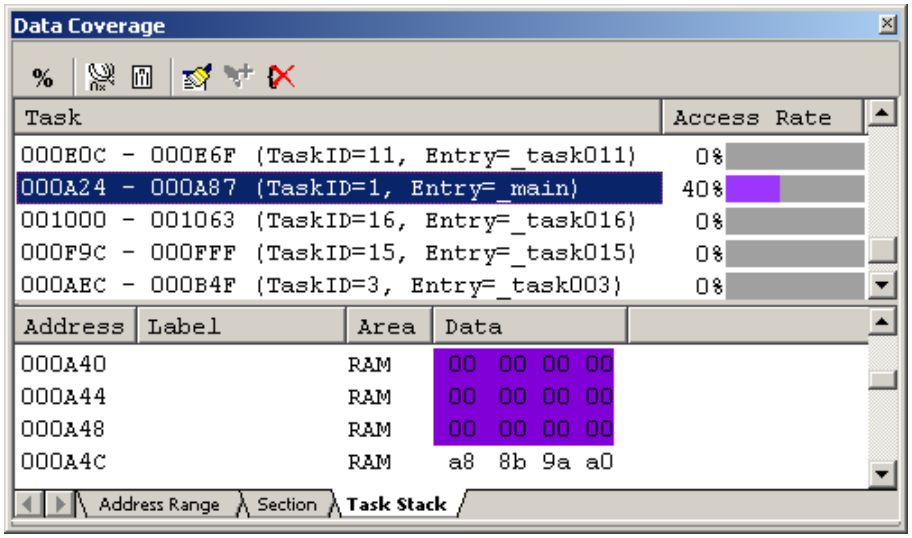


Figure 5.124 Data Coverage window (task stack specification)

The Data Coverage window is vertically divided into two by the splitter.

The upper pane shows the automatically registered task stacks and access rates.

Table 5.39 Contents shown in the upper pane of the Data Coverage window

[Task]	Task stacks (task ID, task entry label)
[Access Rate]	Access rate in percentage and graph

The lower pane shows detailed information of the task stack selected in the upper pane.

Table 5.40 Contents shown in the lower pane of the Data Coverage window

[Address]	Address value
[Label]	Label name
[Area]	Memory area (FlashROM, RAM, SFR) This column is blank when the area is unused.
[Data]	Memory data The accessed data has its background displayed in purple.

If located outside the coverage memory allocated area, address lines are displayed in gray. Although the existing coverage information of those addresses is retained, the coverage information will not be updated by program execution.

The acquired coverage information is accumulated in memory until the user clears it.

Downloaded from Arrow.com.

5.12.12 Clearing Data Coverage Information

(1) Clearing the data coverage information of the specified range

Selecting Clear Coverage Range from the pop-up menu on the Address Range or Section sheet opens the Clear Coverage Range dialog box.

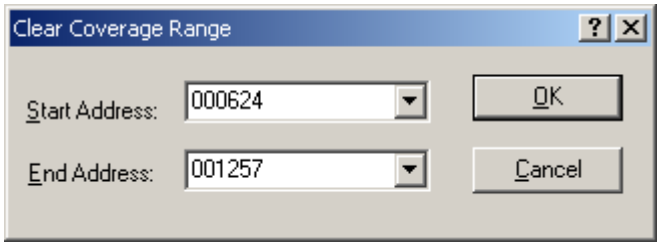


Figure 5.125 Clear Coverage Range dialog box

Enter the start and end addresses of the range to be cleared. Clicking the OK button clears the coverage information of the selected range.

(2) Clearing all data coverage information

Selecting Clear the Entire Coverage from the pop-up menu clears all the data coverage information.

5.12.13 Updating Coverage Information

Selecting Refresh from the pop-up menu updates the content of the Data Coverage window.
If Lock Refresh has been selected, the information is not automatically updated when the program breaks. To view the latest information, therefore, you need to update manually.

5.12.14 Preventing Update of Coverage Information

Selecting Lock Refresh from the pop-up menu prevents update of the Data Coverage window while the user program execution is stopped.

Downloaded from Arrow.com.

5.12.15 Saving the Data Coverage Information to a File

You can save the data coverage information of the currently selected sheet to a file.
Selecting Save Data from the pop-up menu opens the Save Data dialog box.

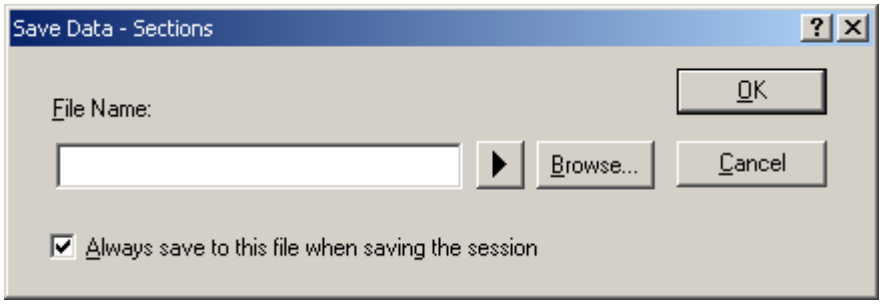


Figure 5.126 Save Data dialog box

Enter a file name in which you want the information to be saved. If a file extension is omitted, ".cdv" will automatically be added as the file extension. If you specify an existing file name, the file is overwritten.

5.12.16 Loading Data Coverage Information from a File

You can load a data coverage information file.
Selecting Load Data from the pop-up menu opens the Load Coverage Data dialog box.

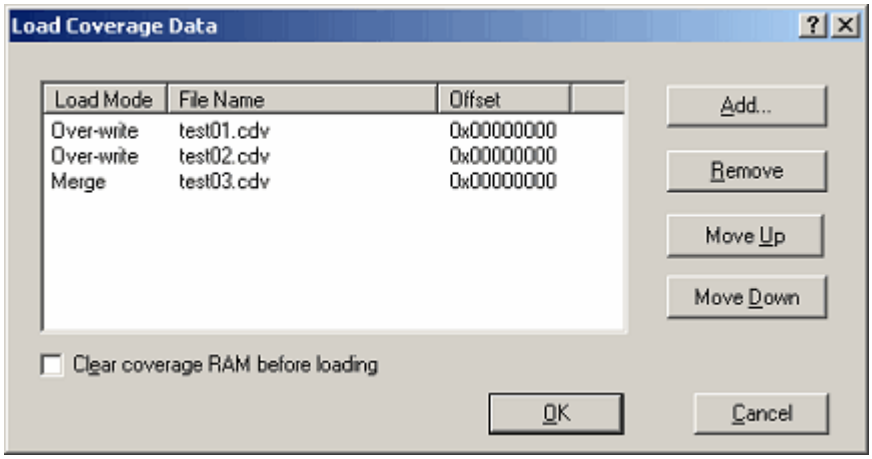


Figure 5.127 Load Coverage Data dialog box

Downloaded from Arrow.com.

Clicking on the Add button opens the Add coverage data file dialog box shown below.

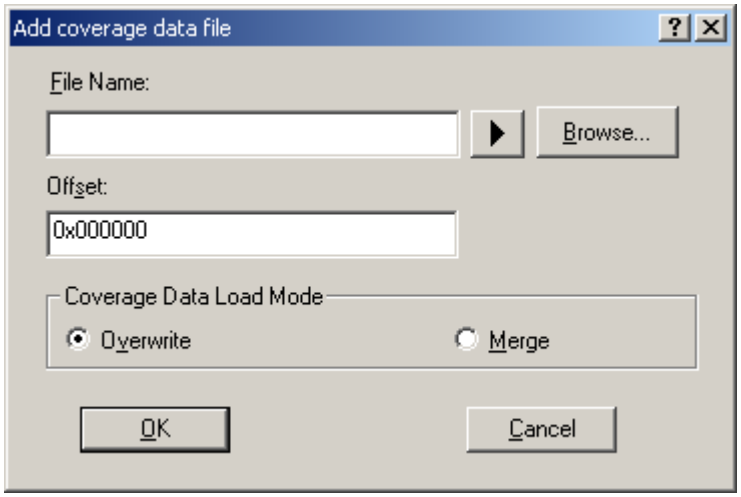


Figure 5.128 Add coverage data file dialog box

Use this dialog box to specify a coverage information file you want to load. You can also specify a load mode and offset for each file you load.
The only file extension available is ".cdv". An error message will appear if any other file extension is entered.
The files you add will be listed in the Load Coverage Data dialog box. The files will be loaded in the order in which they are listed. If necessary, use the Up or Down button to change the order.

Downloaded from Arrow.com.

5.13 Viewing Realtime Profile Information

5.13.1 Viewing Realtime Profile Information

The E100 emulator has code coverage, data coverage and realtime profile functions usable exclusively to each other.
To use the realtime profiling function, choose Realtime Profile in the Switching function section on the System page of the Configuration properties dialog box.
Realtime profiling is the function to measure execution performance within an area allocated to addresses in the profile range, one function or one task at a time. It will help you find the locations and causes of performance degradation in an application program.
Measurements are carried out without obstructing user program execution.
The measurement results are updated when the program breaks.

(1) Function profile

Execution performance is measured one function at a time.
The Realtime profile window shows function names, the start addresses of functions, function sizes, counts and the cumulative execution time, execution rate and average execution time of functions.
The function profile of the E100 emulator does not include the execution time of subroutines in its cumulative display of function execution time.

CAUTION
The function profile is subject to the following limitations:

(a) About the areas to be measured

The E100 emulator can acquire profile information on all functions in areas up to 8 blocks, each in 128 KB units.
Each block you set can be comprised of a contiguous or noncontiguous address area.
No functions can be set that are outside the range of block addresses. In that case, the functions or tasks are displayed in gray.

(b) Limit to the number of functions

Measurement can be taken of up to 8K – 1 (= 8,191) functions.
If the number of functions measured exceeds 8K – 1 (= 8,191), the extra functions are excluded from the subject of measurement. In that case, those functions are displayed in gray. (Function names, address and function sizes are displayed in gray.)

(c) In-line expansion

The functions that are expanded in-line for optimization by the compiler are not displayed in the Realtime Profile window.

(d) Recursive functions

Although the execution time of recursive functions can be measured correctly, they are executed only once.

Downloaded from Arrow.com.

(e) Relationship between Go execution start address and break address within a measurement range and the measurable range

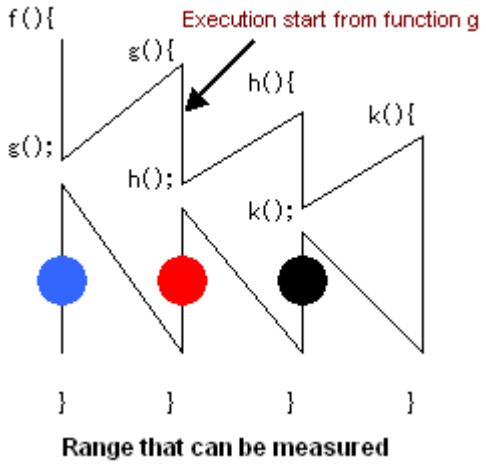


Figure 5.129 Measurable range

Measurable range when the program breaks at the location of a black dot [●]: Execution time and execution count of functions h and k
Measurable range when the program breaks at the location of a red dot [●]: Execution time and execution count of functions h and k
Measurable range when the program breaks at the location of a blue dot [●]: Execution time and execution count of functions h and k
For the function g, the execution time and count in its executed portion can be measured.

Thus, the above is the relationship between break addresses and the measurable range.
Even after the program returned to a high-order function, execution counts of the function from which program execution started cannot be measured.

(f) Function measurement
To measure functions accurately, you need to be in a function to be measured for 100 ns or more after entering the function. Otherwise, the execution time and count may not be measured properly.

(g) Debug information option
To get execution time and execution count of functions, you need to specify a source file that includes the functions for measurement or an option that outputs debug information to the library during compiling. When not specifying the Debug information option, you cannot measure execution time and execution counts of the function.

(h) Maximum execution time and minimum execution time
With the realtime profile, you cannot measure the maximum and minimum execution time of a function. To measure the maximum and minimum execution time of a function, use the Performance Analysis window.

Downloaded from Arrow.com.

(2) Task profile

Execution performance is measured one task at a time.
The Realtime profile window shows task IDs, counts and the cumulative execution time, execution rate and average execution time of tasks.

5.13.2 Setting Realtime Profile Measurement Modes

Choose Set Ranges from the pop-up menu that is displayed when you right-click in the present window.
The Realtime Profile Setting dialog box will be displayed. In the Profile Mode list box of this dialog box, you can select "Function profile" or "Task profile."
When profile modes are changed, all measurement results are cleared.

5.13.3 Measuring Function Profiles

You can measure execution performance one function at a time.

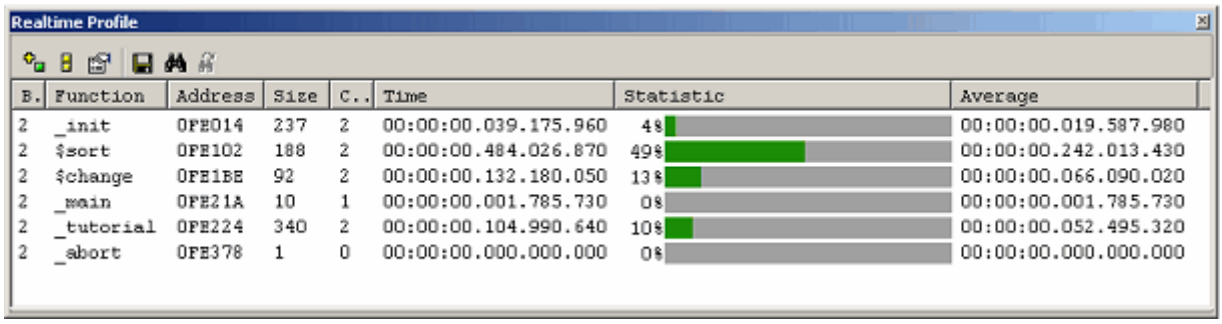


Figure 5.130 Realtime Profile window (function profile)

The following shows detailed information in each column.

Table 5.41 Details on each column

Block	Block number
Function	Function name
Address	Start address of function
Size	Function size
Count	Number of times a function is called
Time	Cumulative time of function execution The timestamp is displayed in the form shown below. Hours:minutes:seconds.milliseconds.microseconds.nanoseconds
Statistic	Ratio of function Time to Go-Break execution time
Average	Average execution time per measurement performed

If located outside the profile memory allocated area, address lines are displayed in gray.
The acquired profile measurement results are accumulated in memory until the user clears them.

Downloaded from Arrow.com.

5.13.4 Setting Function Profile Measurement Ranges

Choose Set Ranges from the pop-up menu that is displayed when you right-click in the present window.
The Realtime Profile Setting dialog box will be displayed. In this dialog box, set a profile measurement range.

[Function mode]

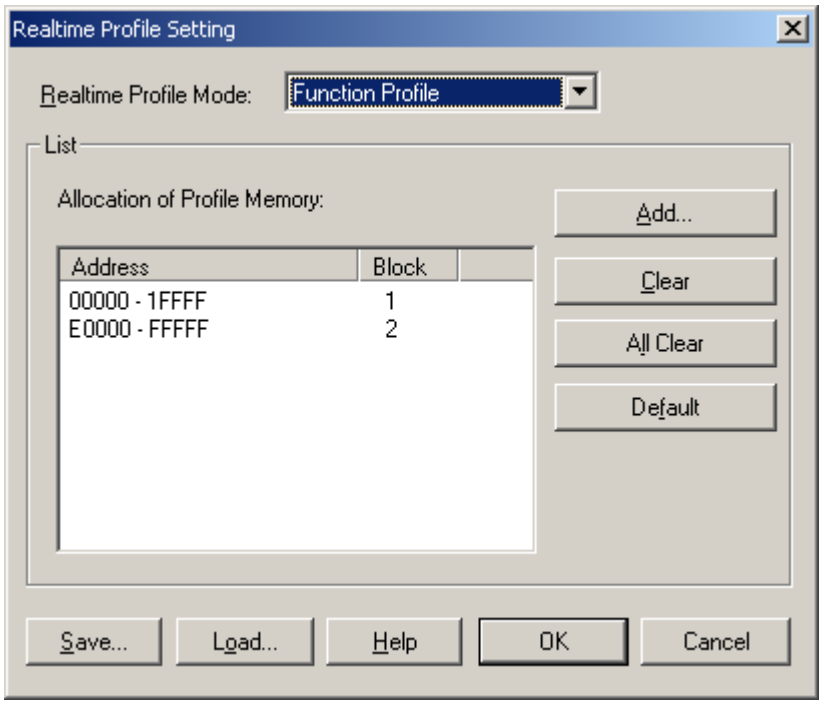


Figure 5.131 Realtime Profile Setting dialog box

(1) Memory allocation

Before function profiles can be measured, profile memory must be allocated to the addresses at which to be measured. Profile data can be obtained from only the address range that has had memory allocated.
You can specify any of blocks 1 to 8 (up to 1 Mbyte) each beginning with a 128-Kbyte boundary as profile measurement areas. Either contiguous or non-contiguous blocks can be assigned.
With initial settings, the profile memory is automatically assigned at addresses in the ROM and RAM areas.

(2) Automatic function detection

When profile memory is assigned at addresses, the E100 emulator automatically detects functions included that address range and registers those functions in the window.

Downloaded from Arrow.com.

5.13.5 Saving Function Profile Measurement Ranges

You can save the current task mode and function profile measurement range (memory allocation state).
Click the Save button of the Realtime Profile Setting dialog box, and the Save As dialog box will be displayed.
Enter a file name in which you want function profile measurement ranges to be saved.
If the file extension is omitted, ".rpf" will automatically be added as the file extension.
If you specify an existing file name, a message is displayed asking you to confirm whether you want the file to be overwritten.

5.13.6 Loading Function Profile Measurement Ranges

You can load a function profile measurement range.
Click the Load button of the Realtime Profile Setting dialog box, and the Open dialog box will be displayed.

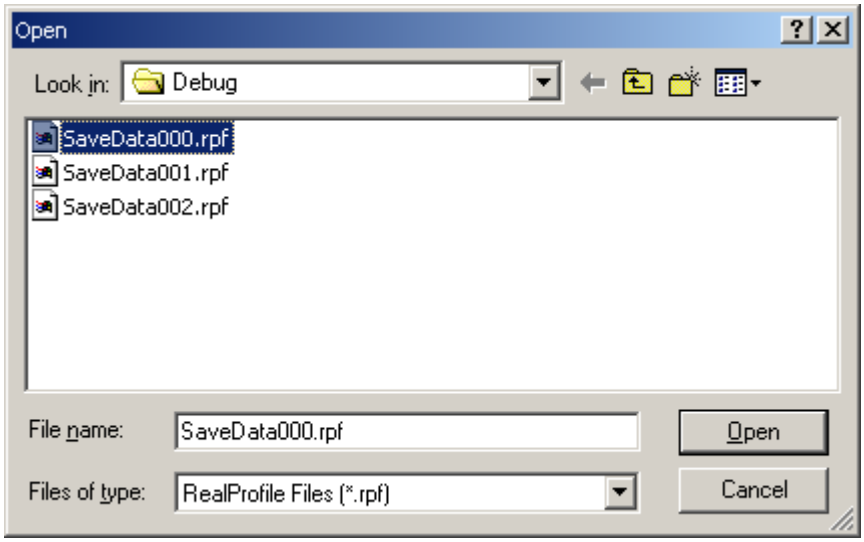


Figure 5.132 Open dialog box

Enter a file name you want to load.
The only file extension available is ".rpf". An error message will appear if any other file extension is entered.
When a file load is complete, the list in the Realtime Profile Setting dialog box is updated.
If task profile information is included in the loaded file, modes in the Realtime Profile Setting dialog box are switched to task mode.

Downloaded from Arrow.com.

5.13.7 Measuring Task Profiles

You can measure execution performance one task at a time.

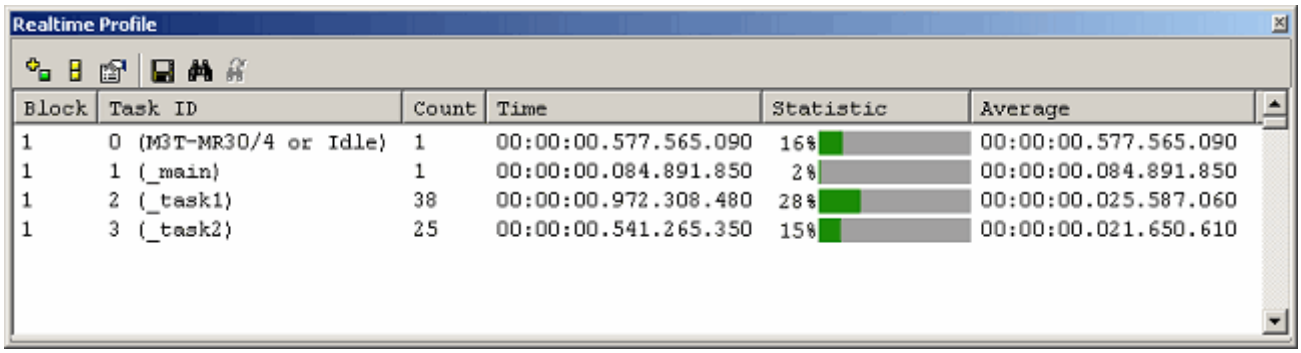


Figure 5.133 Realtime Profile dialog box (task profile):

The following shows detailed information in each column.

Table 5.42 Details on each column

Block	Block number
Task ID	Task ID, entry address
Count	Number of times a task is called
Time	Cumulative time of task execution The timestamp is displayed in the form shown below. Hours:minutes:seconds.milliseconds.microseconds.nanoseconds
Statistic	Ratio of task Time to Go-Break execution time
Average	Average execution time per measurement performed

Disabled tasks are displayed in gray.

The acquired profile measurement results are accumulated in memory until the user clears them.

Downloaded from Arrow.com.

5.13.8 Setting Task Profile Measurement Ranges

Choose Set Range from the pop-up menu that is displayed when you right-click in the present window.
The Realtime Profile Setting dialog box will be displayed. In this dialog box, set a profile measurement range.

[Task mode]

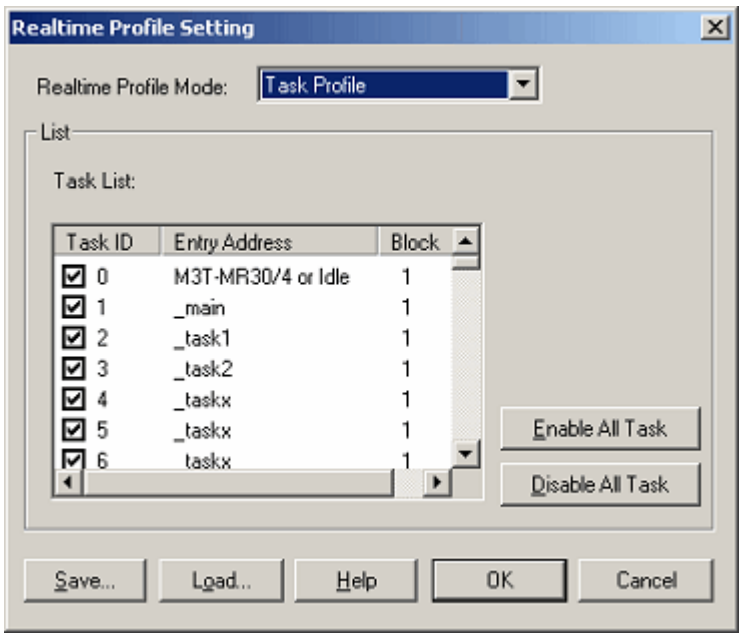


Figure 5.134 Realtime Profile Setting dialog box

(1) Automatic task detection

If you have downloaded a load module that has the OS included in it, the E100 emulator automatically detects a task list.

(2) Selecting tasks

Select the check box of a task ID you want to measure. (By default, all check boxes are selected.)
The selected tasks will automatically be assigned block numbers (1–8).

CAUTION
If measurement blocks are lacking, block numbers become blank, so that no more task IDs can be registered. In that case, deselect the check boxes of unnecessary task IDs.

Downloaded from Arrow.com.

5.13.9 Saving Task Profile Measurement Tasks

You can save the current task mode and measurement tasks (task IDs and enabled/disabled states).
Click the Save button of the Realtime Profile Setting dialog box, and the Save As dialog box will be displayed.
Enter a file name in which you want task profile measurement tasks to be saved.
If the file extension is omitted, ".rpf" will automatically be added as the file extension.
If you specify an existing file name, a message is displayed asking you to confirm whether you want the file to be overwritten.

5.13.10 Loading Task Profile Measurement Tasks

You can load task profile measurement tasks.
Click the Load button of the Realtime Profile Setting dialog box, and the Open dialog box will be displayed.

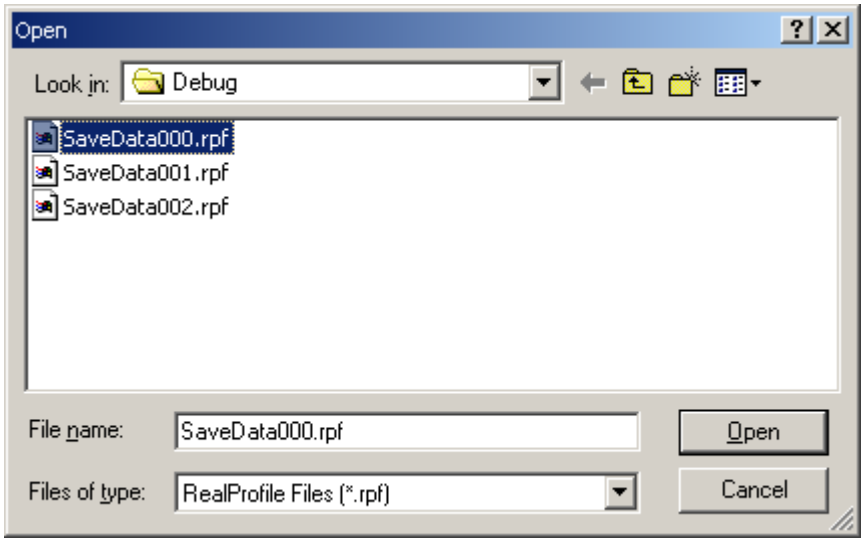


Figure 5.135 Open dialog box

Enter a file name you want to load.
The only file extension available is ".rpf". An error message will appear if any other file extension is entered.
When a file load is complete, the list (task list) in the Realtime Profile Setting dialog box is updated.

If any loaded task IDs are nonexistent, although they are displayed once in the list (task list) in the Realtime Profile Setting dialog box, it is only the existing task IDs that are registered as measurement tasks when you click the OK button.
Reopening the Realtime Profile Setting dialog box, you can check the currently registered measurement tasks.

If function profile information is included in the loaded file, modes in the Realtime Profile Setting dialog box are switched to function mode.

Downloaded from Arrow.com.

5.13.11 Clearing Realtime Profile Measurement Results

Choose Clear from the pop-up menu of the Realtime Profile window, and all measurement results will be cleared.
Unless you choose to Clear, measurement results are accumulated in memory.

5.13.12 Saving Realtime Profile Measurement Results

You can save the current realtime profile measurement results in text format.
Choose Save To File from the pop-up menu of the Realtime Profile window, and the Save As dialog box will be displayed.

Enter a file name in which you want the measurement results to be saved.
If the file extension is omitted, ".txt" will automatically be added as the file extension.
If you specify an existing file name, a message is displayed asking you to confirm whether you want the file to be overwritten.

5.13.13 Setting the Measurement Interval

Choose Properties from the pop-up menu that is displayed when you right-click in the present window.
The Properties dialog box will be displayed.

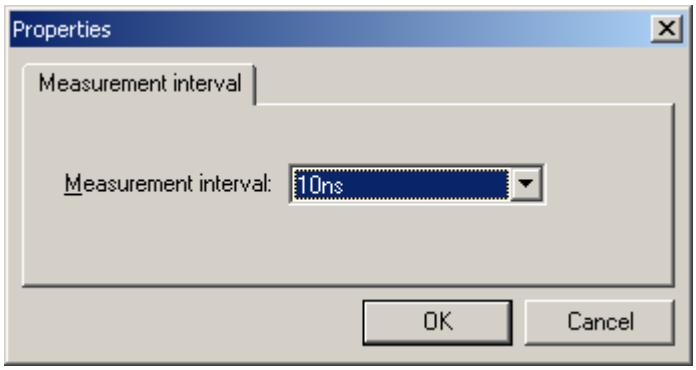


Figure 5.136 Properties dialog box

The measurement interval can be selected from the following options:
10 ns, 20 ns, 40 ns, 80 ns, 160 ns, 1.6 µs

CAUTION
When the currently set measurement interval is changed, the measurement results hitherto accumulated are cleared.

Downloaded from Arrow.com.

5.13.14 Maximum Measurement Time of the Realtime Profile

(1) Maximum measurement time

The timer used for performance measurement is comprised of a 40-bit counter. The maximum measurement time varies with the measurement interval selected.
To select a measurement interval, specify it in the Measurement interval drop-down list of the Properties dialog box.
The measurable maximum times are listed below.

Table 5.43 Maximum measurement time

No.	Resolution	Maximum measurement time
1	10ns	Approx. 3 hours, 03 minutes, 15 seconds
2	20ns	Approx. 6 hours, 06 minutes, 30 seconds
3	40ns	Approx. 12 hours, 13 minutes, 00 seconds
4	80ns	Approx. 24 hours, 26 minutes, 00 seconds
5	160ns	Approx. 48 hours, 52 minutes, 01 seconds
6	1.6us	Approx. 488 hours, 40 minutes, 18 seconds

CAUTION
Note that performance measurement produces an error of ± "2 resolution + 100 ns" (when resolution = 20 ns, ±140 ns) whenever entering functions. If the resolution is 20 ns and you enter functions 10 times, ±1400 ns error occurs.

(2) Maximum measurement count

Execution counts of the realtime profile are measured using a 16-bit counter. Measurement can be taken of up to a count of 65,535.

Downloaded from Arrow.com.

5.14 Detecting Exceptional Events

5.14.1 Detecting Exceptional Events

The E100 emulator permits you to detect various exceptional events that have occurred during user program execution. Exceptional events include an abnormal behavior of the user program, as well as an overflow of the measurement counter of any function involved, etc. Detection of a specified exceptional event can be set as a condition of a breakpoint or trace point.

(1) Exceptional events

The E100 emulator detects the exceptional events listed below.

- Violation of access protection: An error is detected when an access other than a specified access attribute was attempted.
- Read from uninitialized memory: An error is detected when uninitialized area (not write accessed) was accessed for read.
- Stack access violation: An error is detected when the value of the stack register is beyond a boundary of the stack area.
- Performance overflow: An error is detected when the time measurement counter for a section has overflowed.
- Realtime profile overflow: An error is detected when the maximum measurable time or maximum measurable number of passes is exceeded during profile measurement of a function (or a task).
- Trace memory overflow: An error is detected when the trace memory has overflowed.
- Task stack access violation: An error is detected when one task attempts writing to the task stack of another task.
- OS dispatch: An error is detected if a task dispatch has occurred.

5.14.2 Detecting an Access Protect Violation

Violations of access protection such as writing to a ROM area or access to an unused area (for reading, writing, or execution of an instruction) can be detected as an error.

(1) Access attributes

Following attributes can be specified in word units for any area.

Read/Write: Accessible for both read/write
Read Only: Accessible for read only
Write Only: Accessible for write only
Disable: Access prohibited
Disable (OS): Any access except from OS is prohibited (this attribute is automatically assigned when a program including an OS is downloaded).

(2) Protected areas

Any area in the entire memory space may be access protected.
At emulator startup, the whole area is by default assigned a Read/Write access attribute.

Downloaded from Arrow.com.

(3) Methods for setting protection

There are following two methods of specification:

- Automatic setting by section information in a download module
- Specifying the access attribute of any area individually

(4) Detection method

An access protect violation is detected by the emulator's internal resources (blocks 1–16).
The blocks are automatically allocated by the emulator's exclusive algorithm.

CAUTION
Since the emulator's internal resources are limited, not all blocks can be access protected. In that case, reduce the amount of used blocks by "removing blocks" before setting protection again.

		Access attribute		
		Read/Write		
Write access	->	Read Only	->	Detected
Read access	->	Write Only	->	Detected
Read access Write access	->	Disable	->	Detected

Figure 5.137 NG patterns of detection methods

(5) Actions taken when an access protect violation is detected

The following actions can be set:

- Display a warning

Selecting the Access Protect Violation check box on the Exception Warning page of the Configuration properties dialog box, you can display a warning in the Status window and in a status bar balloon.

- Make the detection of an access protect violation a condition of a hardware breakpoint
- Make the detection of an access protect violation a condition of a trace point

Downloaded from Arrow.com.

5.14.3 Setting an Access Protected Area

Follow the procedure described below to set an access protected area.

(1) From the Hardware Break dialog box

1. Select the Exception check box on the Hardware Break sheet and then click the Detail button.

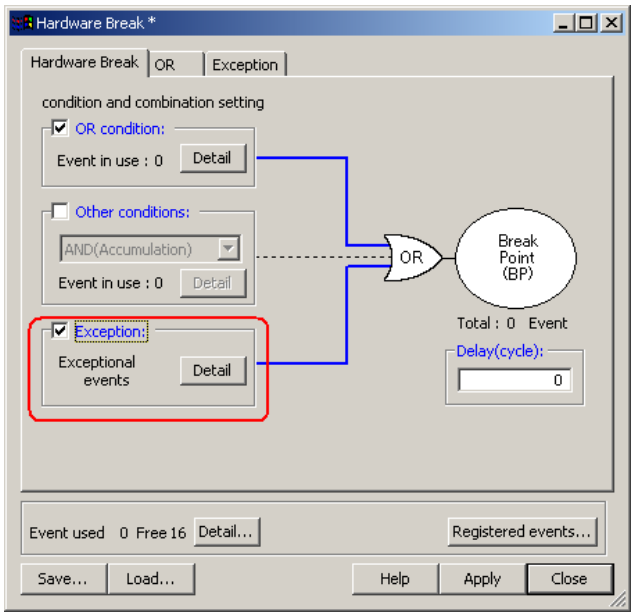


Figure 5.138 Hardware Break dialog box

2. The Exception page shown below will appear. Click the Detail button to the right of the Violation of access protection check box.

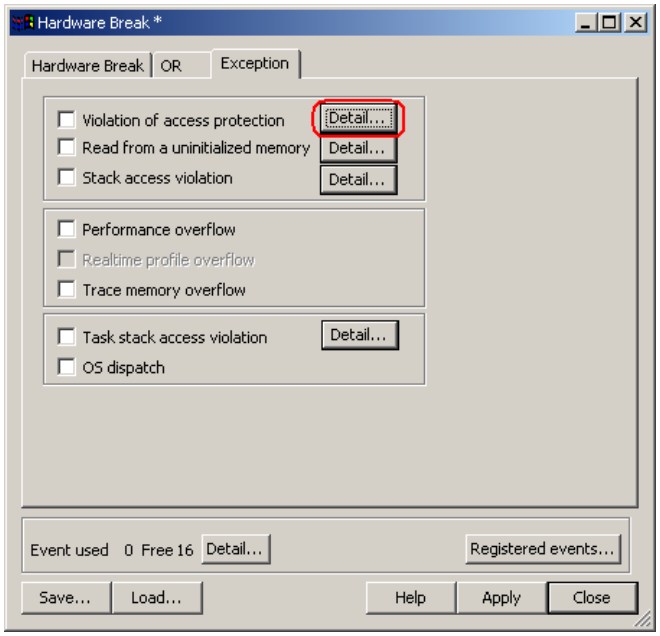


Figure 5.139 Hardware Break dialog box

Downloaded from Arrow.com.

3. The Violation of access protection dialog box shown below will be displayed.
   To have the access attributes automatically set according to the section information in a download module when a program is downloaded, select the check box labeled "Automatically set address areas at downloading."

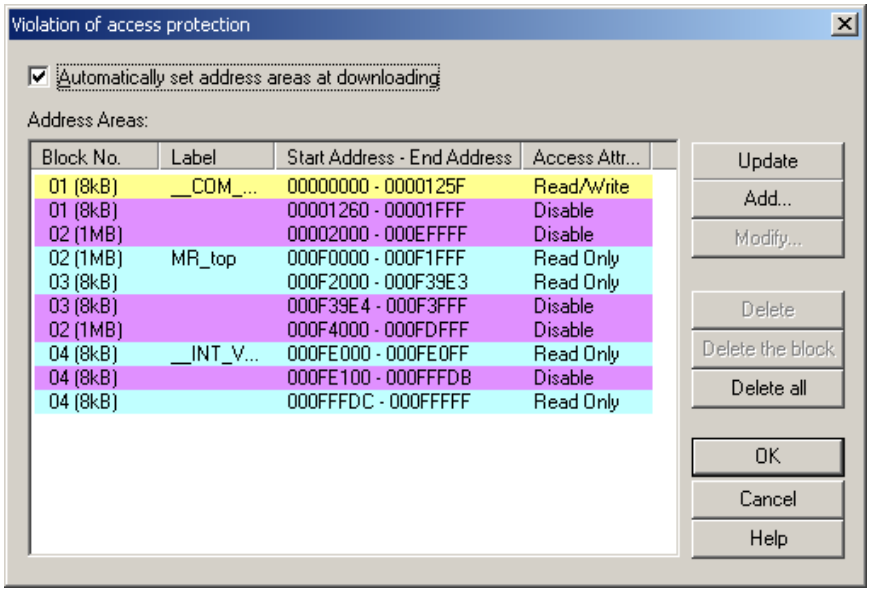


Figure 5.140 Violation of access protection dialog box

4. Click the Update button, and the access attributes will be updated according to the section information in a download module.

5. To add an access attribute manually, click the Add button. The Access protection condition dialog box shown below will appear. Specify any address range and access attribute.

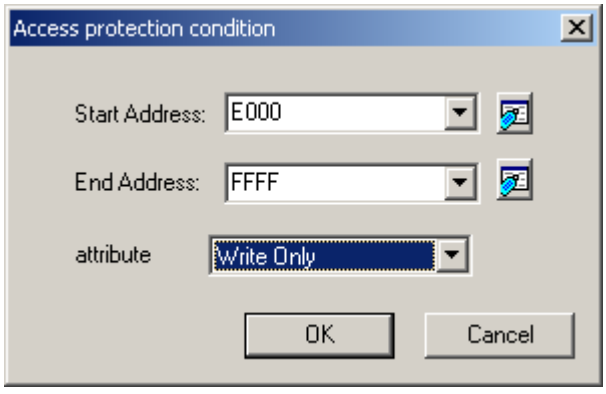


Figure 5.141 Access protection condition dialog box

Downloaded from Arrow.com.

6. The protected area you have added will be displayed in the Address Areas list of the Violation of access protection dialog box.

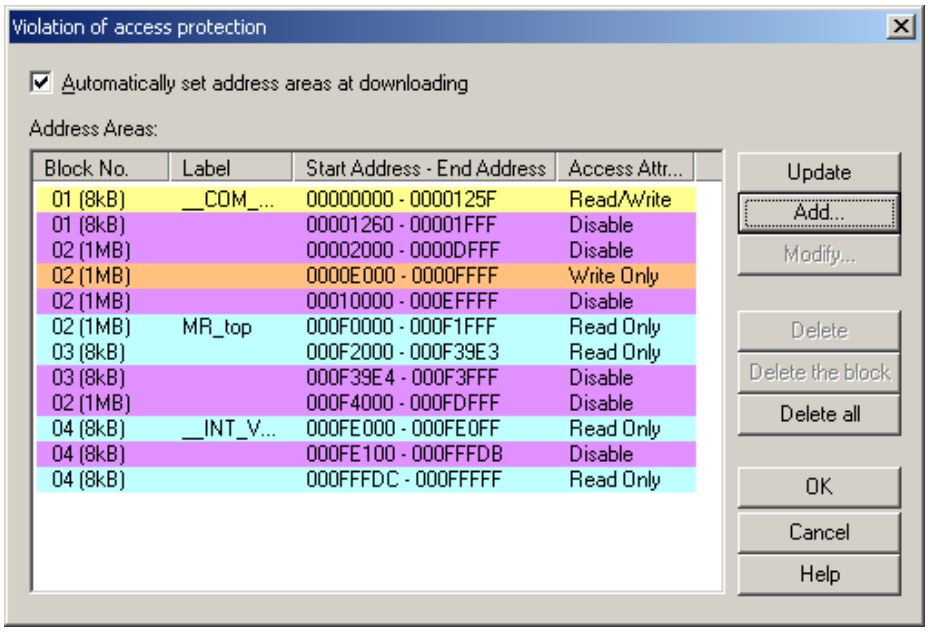


Figure 5.142 Violation of access protection dialog box

(2) From the Trace conditions dialog box

1. In the Trace Mode drop-down list of the Trace sheet, select Fill around TP. Select the Exception check box and then click the Detail button.

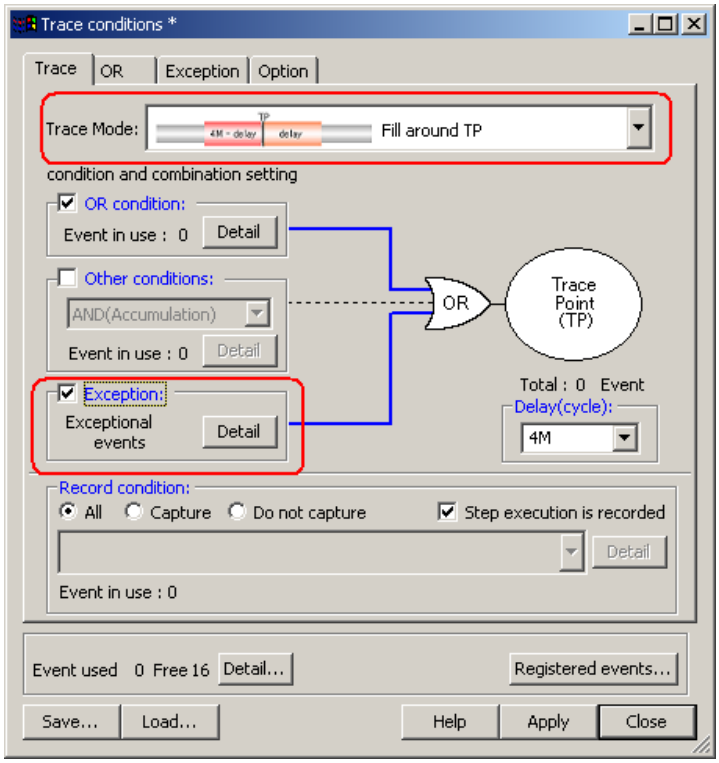


Figure 5.143 Trace conditions dialog box

Downloaded from Arrow.com.

2. The Exception page shown below will appear. Click the Detail button to the right of the Violation of access protection check box.

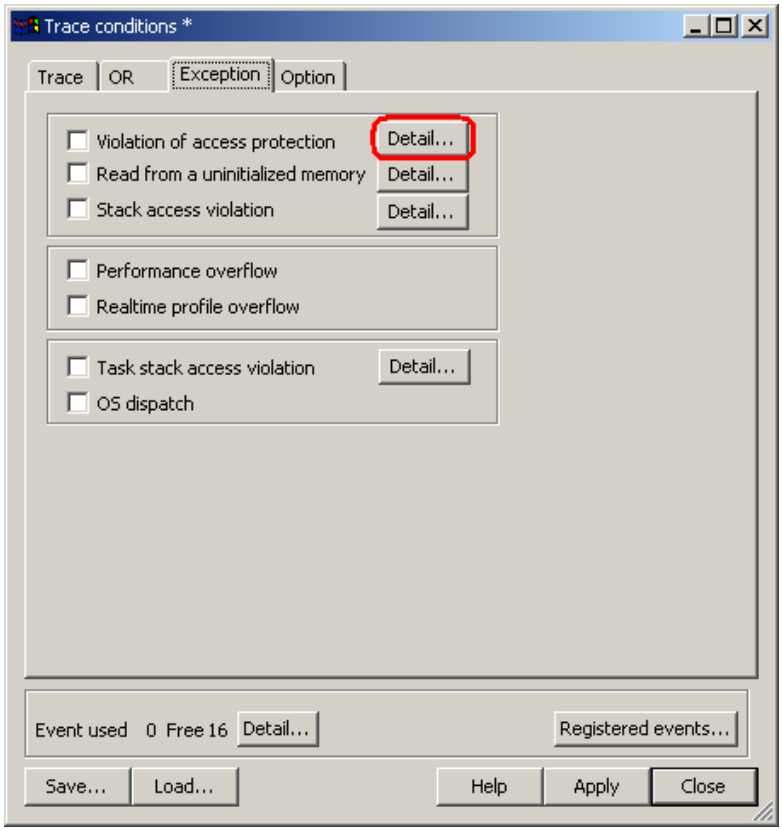


Figure 5.144 Trace conditions dialog box

3. The Violation of access protection dialog box will be displayed.
The rest is the same as you opened it from the Hardware Break dialog box.

Downloaded from Arrow.com.

5.14.4 Detecting Initialization-Omitted

Reading from a non-initialized area, i.e. cases of reading from a memory location to which nothing has been written, can be detected as an error.

In the emulator, the blocks 0-31 (maximum 16 Kbytes) can be specified as a detection area of the initialization-omitted.

(1) Detection method

An initialization-omitted is detected by the RAM monitor function.

Allocate a RAM monitor area to a given address range and enable error detection in that area.

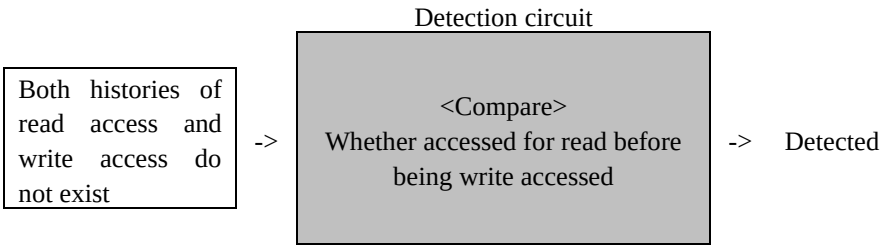


Figure 5.145 Outline of the initialization omitted

(2) Actions taken when an initialization-omitted is detected

The following actions can be set:

- Display a warning

Selecting the Read from uninitialized memory check box on the Exception Warning page of the Configuration properties dialog box, you can display a warning in the Status window and in a status bar balloon.

Color display in the RAM Monitor window

- Set the detection of an initialization-omitted as a condition of a hardware breakpoint
- Set the detection of an initialization-omitted as a condition of a trace point

Downloaded from Arrow.com.

5.14.5 Detecting Stack Access Violation

Setting the size of the stack too small in software development raises the possibility of a program going out of control or malfunctioning. The E100 emulator actively detects abnormal access by the stack pointer.

(1) Setting a stack range

By selecting a stack section it is possible to automatically set a stack range, or you can enter any address range you want.

(2) Initial settings at startup

At startup, stack sections are automatically set. However, because address information is nonexistent, those stack sections do not work until a program is downloaded.

(3) Detection method

The emulator monitors the values of USP and ISP and detects if the value points to a location outside the stack areas.

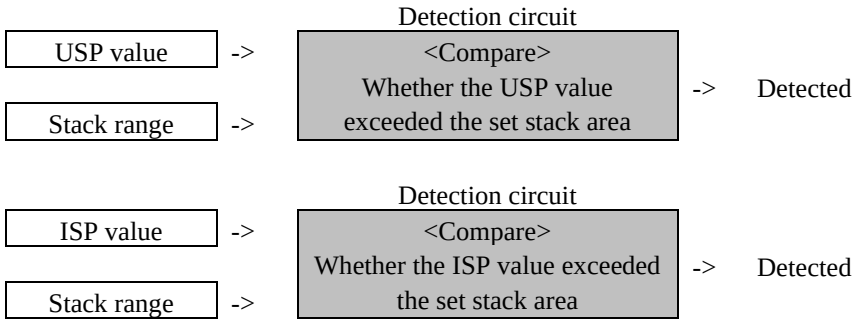


Figure 5.146 Outline of the detection of a stack access violation

The emulator will detect the error if the value of the stack pointer is beyond the stack areas on

1. generation of an interrupt or return from an interrupt handler;
2. calling of a function or return from a function; or
3. the stack pointer pointing to a location outside reserved stack areas.

CAUTION

Detection does not cover cases of corruption of data within a stack area.

(4) Actions taken when a stack access violation is detected

The following actions can be set:

- Display a warning

Selecting the Stack Access Violation check box on the Exception Warning page of the Configuration properties dialog box, you can display a warning in the Status window and in a status bar balloon.

- Set the detection of a stack access violation as a condition of a hardware breakpoint
- Set the detection of a stack access violation as a condition of a trace point

Downloaded from Arrow.com.

5.14.6 Detecting a Performance Overflow

A time in performance measurement coming to exceed the maximum value can be detected as an error.
Timeout case in a performance measurement is referred to as a performance overflow.

(1) Actions taken when a performance overflow is detected

The following actions can be set:

- Display a warning

A warning is displayed in the Performance window.
The result display line of a program section in which a timeout phenomenon occurred is marked with a string "overflow."
Selecting the Performance Overflow check box on the Exception Warning page of the Configuration properties dialog box, you can display a warning in the Status window and in a status bar balloon.

- Set the detection of a performance overflow as a condition of a hardware breakpoint
- Set the detection of a performance overflow as a condition of a trace point

5.14.7 Detecting a Realtime Profile Overflow

A time or number of passes in realtime profile measurement coming to exceed the maximum value can be detected as an error.
Timeout and count-out (count expired) cases in a realtime profile are collectively referred to as a realtime profile overflow.

(1) Actions taken when a realtime profile overflow is detected

The following actions can be set:

- Display a warning

A warning is displayed in the Realtime Profile window.
The function or result display line of a task in which a timeout or count-out phenomenon occurred is marked with a string "overflow".
Selecting the Realtime Profile Overflow check box on the Exception Warning page of the Configuration properties dialog box, you can display a warning in the Status window and in a status bar balloon.

- Set the detection of a realtime profile overflow as a condition of a hardware breakpoint
- Set the detection of a realtime profile overflow as a condition of a trace point

Downloaded from Arrow.com.

5.14.8 Detecting a Trace Memory Overflow

Overflows of the trace memory (4 M cycles) can be detected as errors.

(1) Actions taken when a trace memory overflow is detected

The following actions can be set:

- Display a warning

Selecting the Trace memory overflow check box on the Exception Warning page of the Configuration properties dialog box, you can display a warning in the Status window and in a status bar balloon.

- Set the detection of a trace memory overflow as a condition of a hardware breakpoint

5.14.9 Detecting a Task Stack Access Violation

This facility is only available when a load module that includes an OS has been downloaded. The emulator detects an error when one task attempts writing to the task stack for another task.

(1) Initial settings at startup

At startup, the check box labeled "Automatically set address areas at downloading" is selected (flagged by a check mark). However, because address information is nonexistent, the function does not work until a program is downloaded.

(2) Actions taken when a task stack access violation is detected

The following actions can be set:

- Display a warning

Selecting the Task stack access violation check box on the Exception Warning page of the Configuration Properties dialog box, you can display a warning in the Status window and in a status bar balloon.

- Set the detection of a task stack access violation as a condition of a hardware breakpoint
- Set the detection of a task stack access violation as a condition of a trace point

Downloaded from Arrow.com.

5.14.10 Setting a Task Stack Area

Follow the procedure described below to set a task stack area.

(1) From the Hardware Break dialog box

1. Select the Exception check box on the Hardware Break sheet and then click the Detail button.

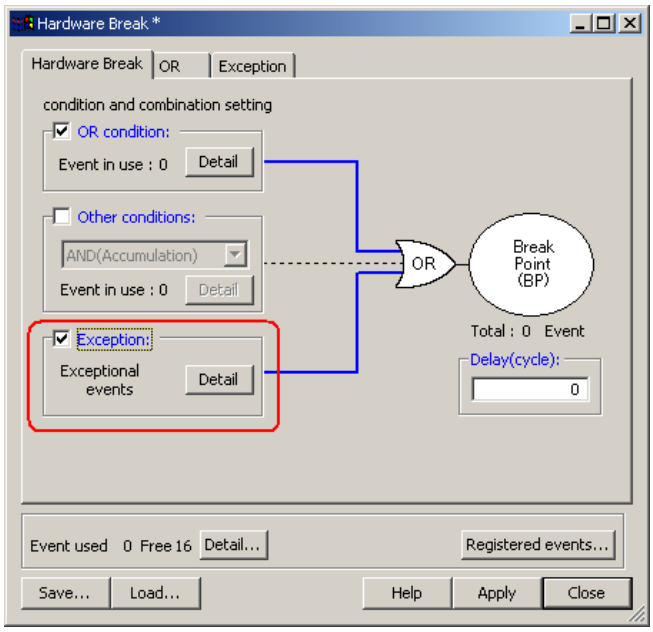


Figure 5.147 Hardware Break dialog box

2. The Exception page shown below will appear. Click the Detail button to the right of the Task stack access violation check box.

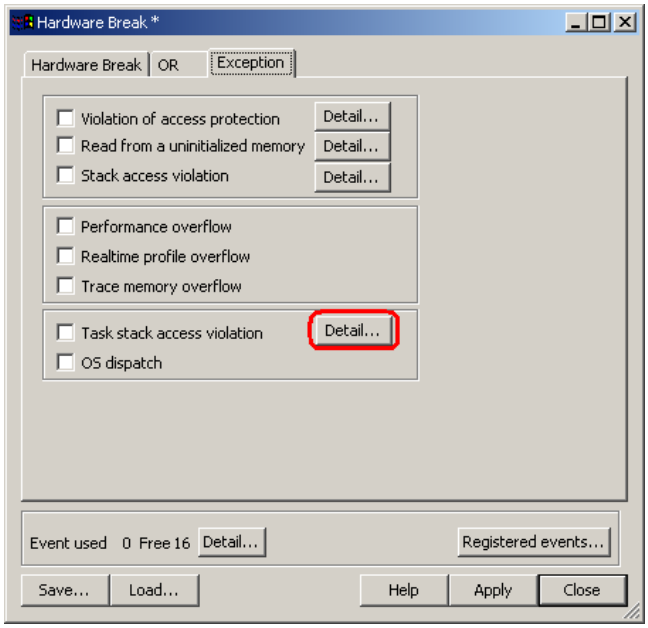


Figure 5.148 Hardware Break dialog box

Downloaded from Arrow.com.

3. The Violation of task stack access dialog box shown below will be displayed. To have the task stack ranges automatically set when a program is downloaded, select the check box labeled "Automatically set address areas at downloading."

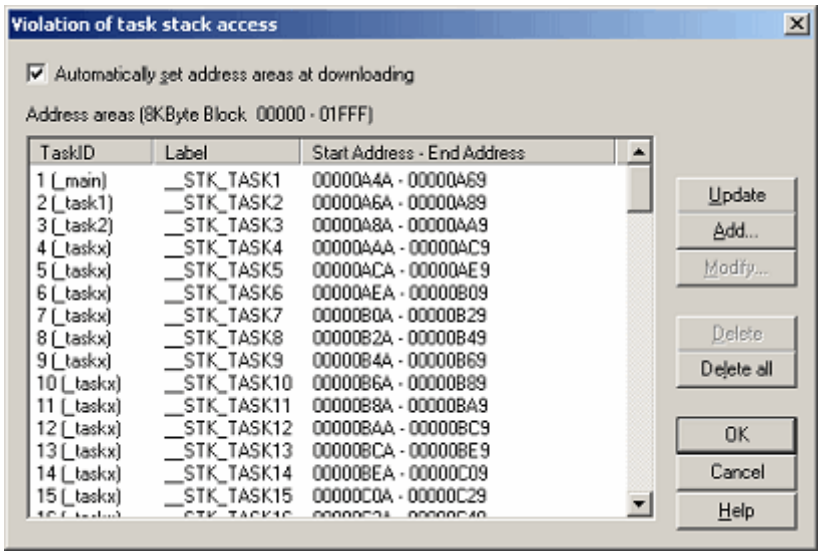


Figure 5.149 Violation of task stack access dialog box

4. Click the Update button, and the task stack ranges will be automatically set.

5. To add a task stack range manually, click the Add button. The Task stack access condition dialog box shown below will appear. Specify any task ID and the address range of a task stack.

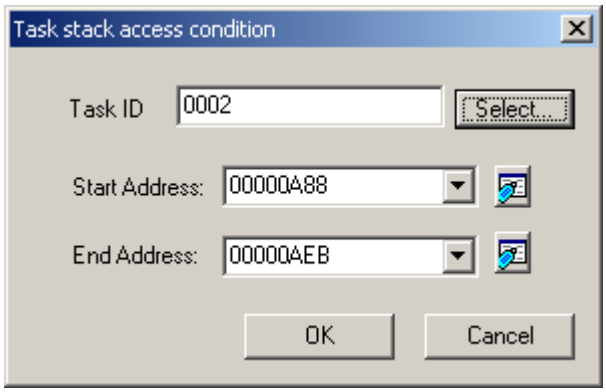


Figure 5.150 Task stack access condition dialog box

6. The task stack ranges you have added will be displayed in the Address Areas list of the Violation of task stack access dialog box.

Downloaded from Arrow.com.

(2) From the Trace conditions dialog box

1. In the Trace Mode drop-down list of the Trace sheet, select Fill around TP. Select the Exception check box and then click the Detail button.

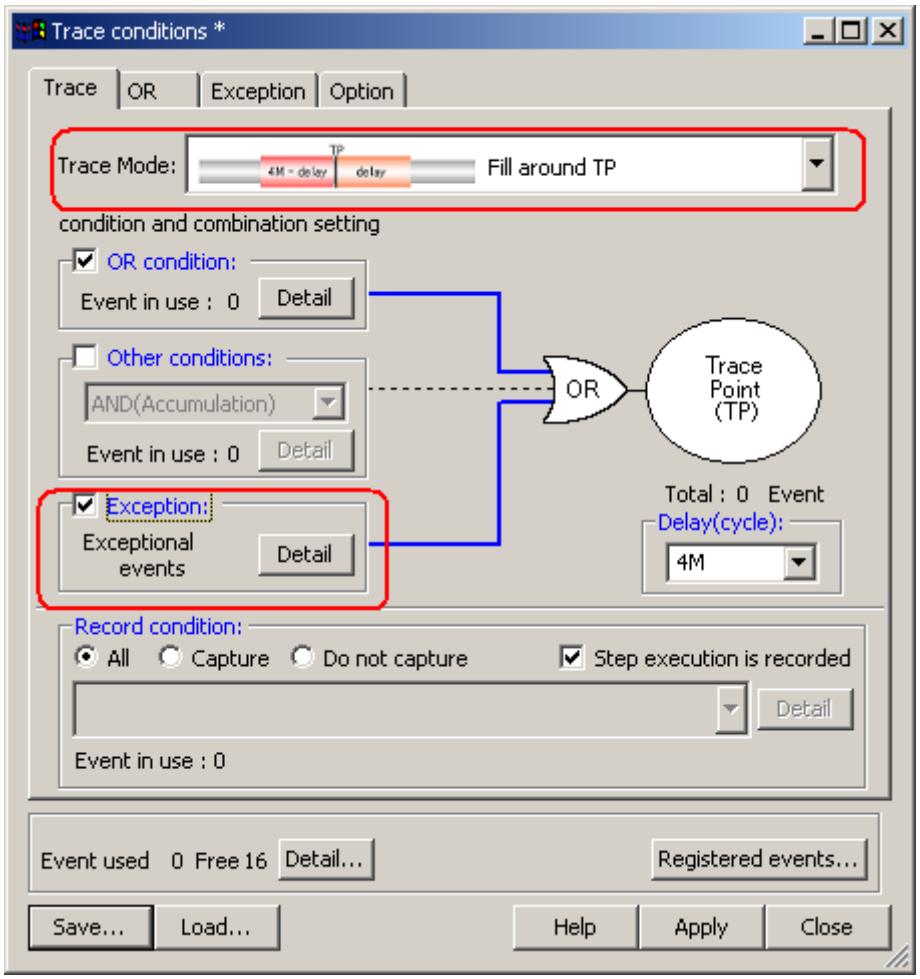


Figure 5.151 Trace conditions dialog box

Downloaded from Arrow.com.

2. The Exception page shown below will appear. Click the Detail button to the right of the Task stack access violation check box.

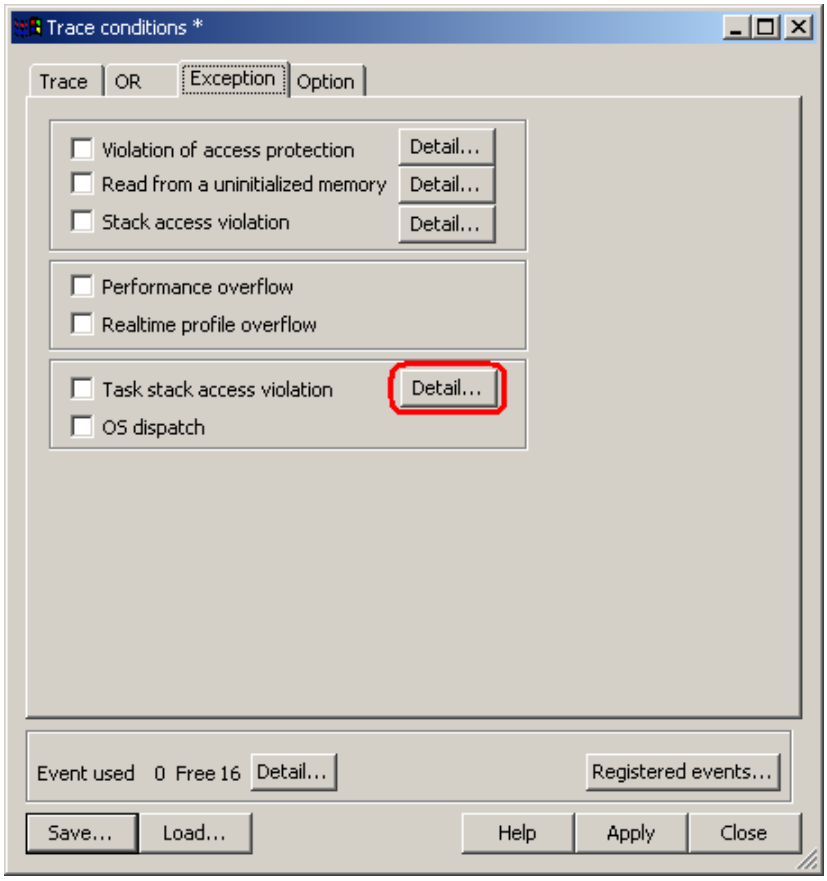


Figure 5.152 Trace conditions dialog box

3. The Violation of task stack access dialog box will be displayed. The rest is the same as you opened it from the Hardware Break dialog box.

5.14.11 Detecting an OS Dispatch

This facility is only available when a load module that includes an OS has been downloaded. The emulator detects the generation of task dispatch as an error.

(1) Actions taken when an OS dispatch is detected

The following actions can be set:

- Display a warning

Selecting the OS dispatch check box on the Exception Warning page of the Configuration properties dialog box, you can display a warning in the Status window and in a status bar balloon.

- Set the detection of an OS dispatch as a condition of a hardware breakpoint
- Set the detection of an OS dispatch as a condition of a trace point

Downloaded from Arrow.com.

5.15 Using the Start/Stop Function

The emulator executes the specified routine of the user program immediately before starting and immediately after halting program execution. This function is used to control the user system in synchronization with execution and halting of the user program.

5.15.1 Opening the Start/Stop Function Setting Dialog Box

The routine executed immediately before starting and immediately after halting the user program execution is specified in the [Start/Stop function setting] dialog box.

To open the Start/Stop function setting dialog box, choose Setup -> Emulator -> Start/Stop function setting… from the menu.

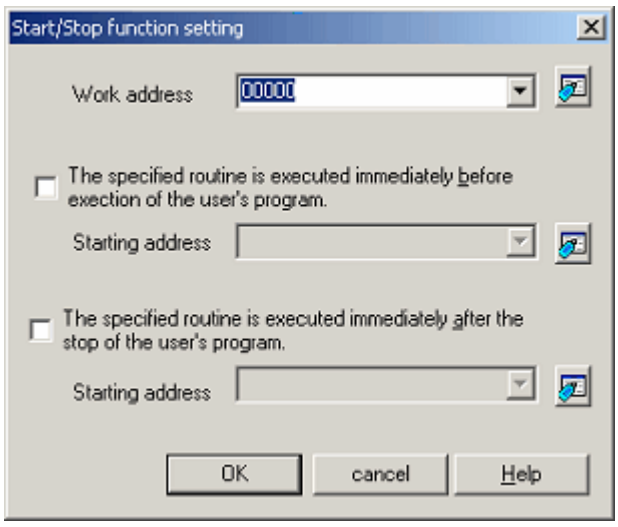


Figure 5.153 Start/Stop function setting dialog box

5.15.2 Specifying the Work Address

Use this command to specify the address of a work area (stack area) for use by a routine to run before the user program execution is started or after user program execution is stopped.

CAUTION

The specified address must be in the RAM area and not used by the user program.

5.15.3 Specifying the Routine to be Executed

It is possible to specify the respective routines immediately before starting and immediately after halting the user program execution.

When The specified routine is executed immediately before execution of the user's program check box is selected, the routine specified in the Starting address combo box, which is below this check box, is executed immediately before starting user program execution.

When The specified routine is executed immediately after the stop of the user's program check box is selected, the routine specified in Starting address combo box, which is below this check box, is executed immediately after halting user program execution.

Downloaded from Arrow.com.

5.15.4 Limitations of the Start/Stop Function

The Start/Stop function is subject to the following limitations.

- While the Start/Stop function is in use, do not use the debug functions listed below.
  (a) Memory setting and download into the program area of a specified routine
  (b) Breakpoint setting in the program area of a specified routine
- While a specified routine is executed, the 4-byte value pointed to by the interrupt stack is used under control on the emulator side.
- The general-purpose registers and flags used in a specified routine are subject to the following limitations.

Table 5.44 Limitations to the registers and flags

Register/flag Name	Limitations
ISP register	When a specified routine has ended, the value of this register must be restored to one that it had when the specified routine started.
U flag	When a specified routine has ended, the value of this flag must always be set to 0.
I flag	Interrupts are disabled while a specified routine is executed.

- When a specified routine is executed, the debug functions listed below have no effect.
  (a) Trace function
  (b) Break-related functions
  (c) RAM monitor function
- When a specified routine is executed, non-maskable interrupts are always disabled.
- The table below shows which state the MCU will be in when the user program starts running after a specified routine is executed.

Table 5.45 MCU Status at start of the user program

MCU Resource	Status
MCU general-purpose registers	These registers are in the state in which they were when the user program last stopped or the MCU registers that were set in the register window by the user. The register contents changed after a specified routine is executed are not reflected.
Memory in MCU space	Memory accesses attempted after a specified routine is executed are reflected.
MCU peripheral functions	Operation of the MCU peripheral functions after execution of the specified routine is continued.

5.15.5 Limitations to the Statements written in a Specified Routine

The statements written in a specified routine are subject to the limitations described below.

- If a stack needs to be used in a specified routine, always be sure to use the user stack.
- To terminate the processing of a specified routine, write a return subroutine instruction.
- Make sure that one session of processing performed by a specified routine is terminated within 10 ms. If, for example, the clock is turned off and kept inactive within a specified routine, then the emulator may become unable to control program execution.
- The values stored in the registers at the time a specified routine starts running are indeterminate. Be sure that the register values are initialized within a specified routine.

Downloaded from Arrow.com.

5.16 Using the Trigger Output Function

The trigger output function allows output of signals through an external trigger cable. Trigger pin numbers 31 to 16 can be used for output. Note, however, that operation of a trigger pin depends on its pin number. Table 5.46 lists the trigger pin numbers and how they operate.

Table 5.46 Trigger Pin Numbers and Operation

No.	Operation
31 to 24	These pins constantly output a signal; either high or low can be selected.
23	A high-level signal is output when a breakpoint is encountered.
22	A high-level signal is output when a trace point is encountered.
21	A high-level signal is output when specific trace data is extracted or discarded.
20 to 16	An event can be specified for each of the signals and a high-level signal is output when that event occurs.

Output is at the power voltage level of the target system. If the MCU in use has two power supplies, the level on VCC1 will be applicable.

5.16.1 Using the External Trigger Cable for Output

You can specify input and output through the external trigger cable on the System page of the Configuration properties dialog box. Select the 'EXT 0-15 INPUT EXT16-31 OUTPUT' radio button for 'External trigger cable'.

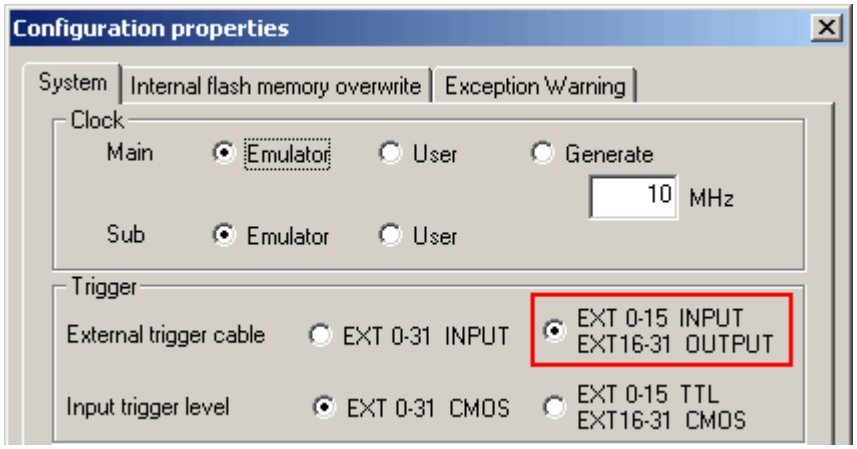


Figure 5.154 Configuration properties dialog box (System page)

Downloaded from Arrow.com.

5.16.2 Opening the Trigger Output Conditions Dialog Box

Choose [Event -> Trigger Output Conditions] from the View menu, or click on the 'Trigger Output Conditions' toolbar button [ ].

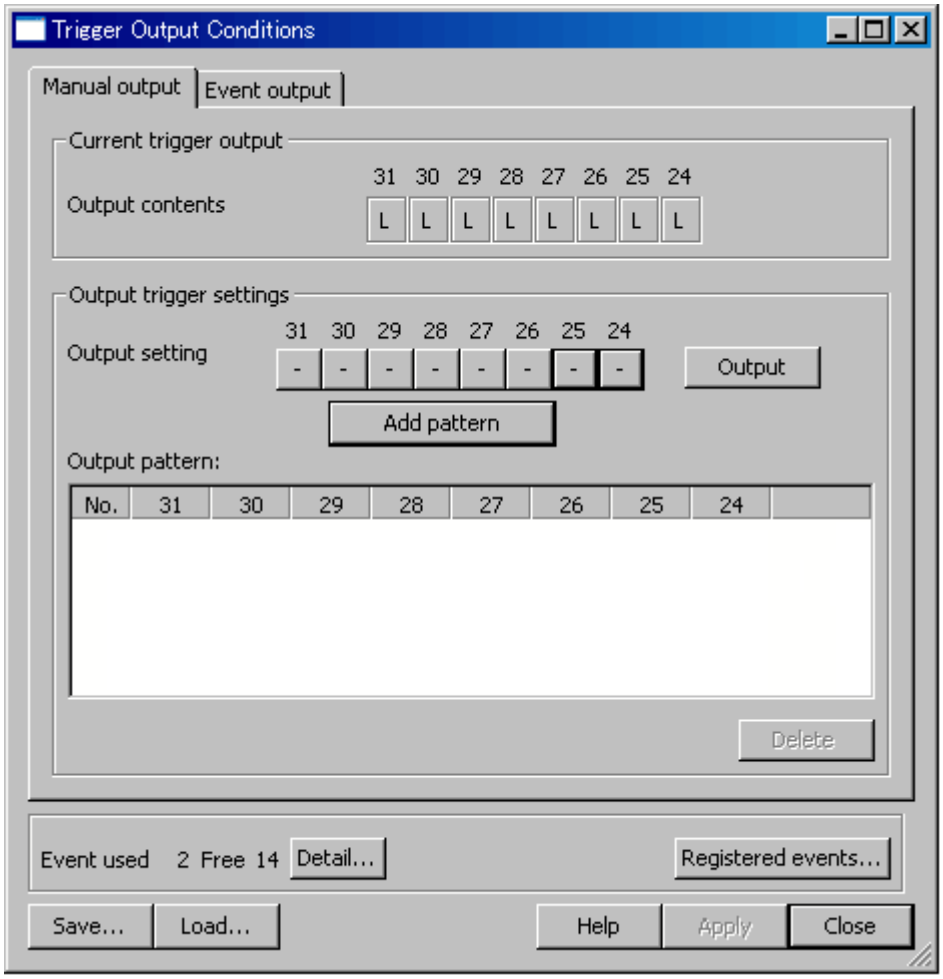


Figure 5.155 Trigger Output Conditions dialog box

Note that you cannot open the Trigger Output Conditions dialog box in either of the following cases.

- 'EXT 0-31 INPUT' has been selected on the System page of the Configuration properties dialog box.
- An external trigger cable is not connected.

Downloaded from Arrow.com.

5.16.3 Manual Setting for Output through Trigger Pins 31 to 24

Make the manual settings for output through trigger pins 31 to 24 on the Manual output page.

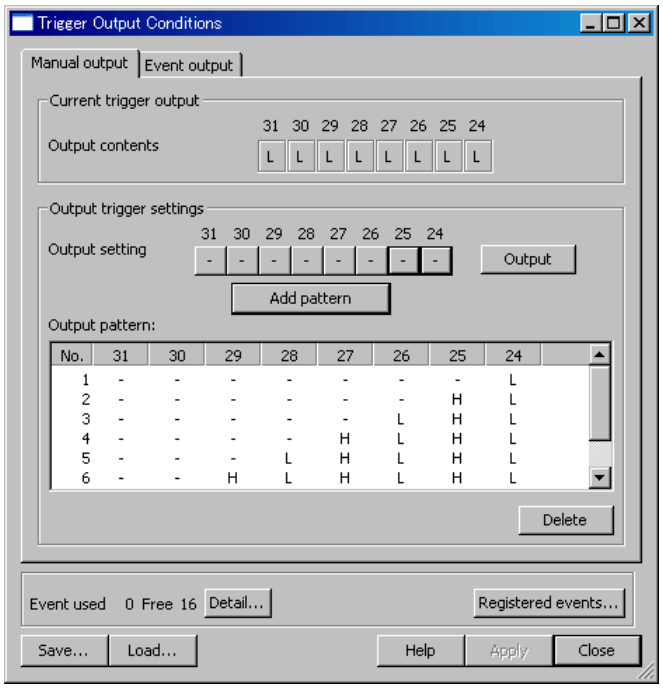


Figure 5.156 Trigger Output Conditions dialog box (Manual output page)

(1) Display of output states: 'Output contents'

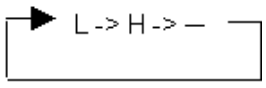
'Output contents' indicates the current signal levels on trigger pins 31 to 24.

H: High

L: Low

(2) 'Output setting'

'Output setting' indicates the levels of signals to be output through trigger pins 31 to 24. Clicking on one of these buttons changes the state of the corresponding pin in the following order.



L: Low

H: High

–: The previous setting is retained.

When the Trigger Output Conditions dialog box is opened, the states of all signals in the 'Output setting' section are always indicated as '–', whether the previous setting was L or H.

(3) Starting output of signals

Click on the 'Output' button to validate the settings and start output of signals.

Downloaded from Arrow.com.

(4) Saving output patterns

- You can save the settings on trigger pins 31 to 24 and reflect a saved setting as the 'Output setting'. This simplifies operations.
- After making settings for an 'Output setting', click on the 'Add pattern' button. The new setting will be added as the last line in the 'Output pattern' list.
- Up to 256 patterns can be added.
- Double-clicking on a line in the 'Output pattern' list reflects the information on the line as the 'Output setting'.
- The order of the lines (patterns) can be changed by dragging and dropping.
- To delete a pattern, select the line and click on the 'Delete' button.

5.16.4 Setting for Output through Trigger Pins 20 to 16

The Event output page allows manual setting for output through trigger pins 20 to 16.

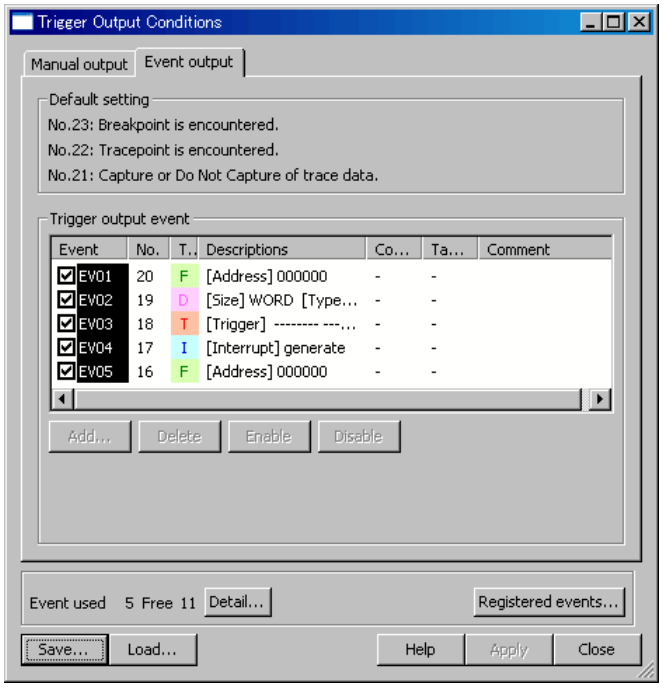


Figure 5.157 Trigger Output Conditions dialog box (Event output page)

(1) Default setting

'Default setting' indicates the trigger output conditions on pins 23 to 21. These pins are always enabled. Signals are output through these pins when the respective conditions are satisfied. Table 5.47 gives details on how the conditions control output.

Table 5.47 Trigger Output Conditions and Output

No.	Condition	Output
23	A breakpoint is encountered	Continued output of a high-level signal is started.
22	A trace point is encountered	A high-level signal is output only during cycles in which the trace-point condition is satisfied.
21	Specific trace data is extracted or deleted	A high-level signal is output only in cycles where trace data is being extracted or discarded.

Downloaded from Arrow.com.

(2) Trigger output event

You can specify an event for trigger pins 20 to 16. A high-level signal will only be output while the event is occurring.

CAUTION
The actual trigger output follows event detection after some delay. The number of cycles of delay varies with the product. The delay for trigger output in the R0E530650MCU00 is 8 cycles.

5.16.5 Events

For details on the setting of events, see section "5.7 Using Events" (page 110).

Downloaded from Arrow.com.

6. Troubleshooting (Action in Case of an Error)

6.1 Flowchart for Remediation of Trouble

Figure 6.1 shows the flowchart for remediation of trouble arising between activation of the power supply to the emulator system and the emulator debugger starting up. Go through the checks with the user system disconnected. For the latest FAQs, visit the Renesas Tools Homepage.

https://www.renesas.com/tools

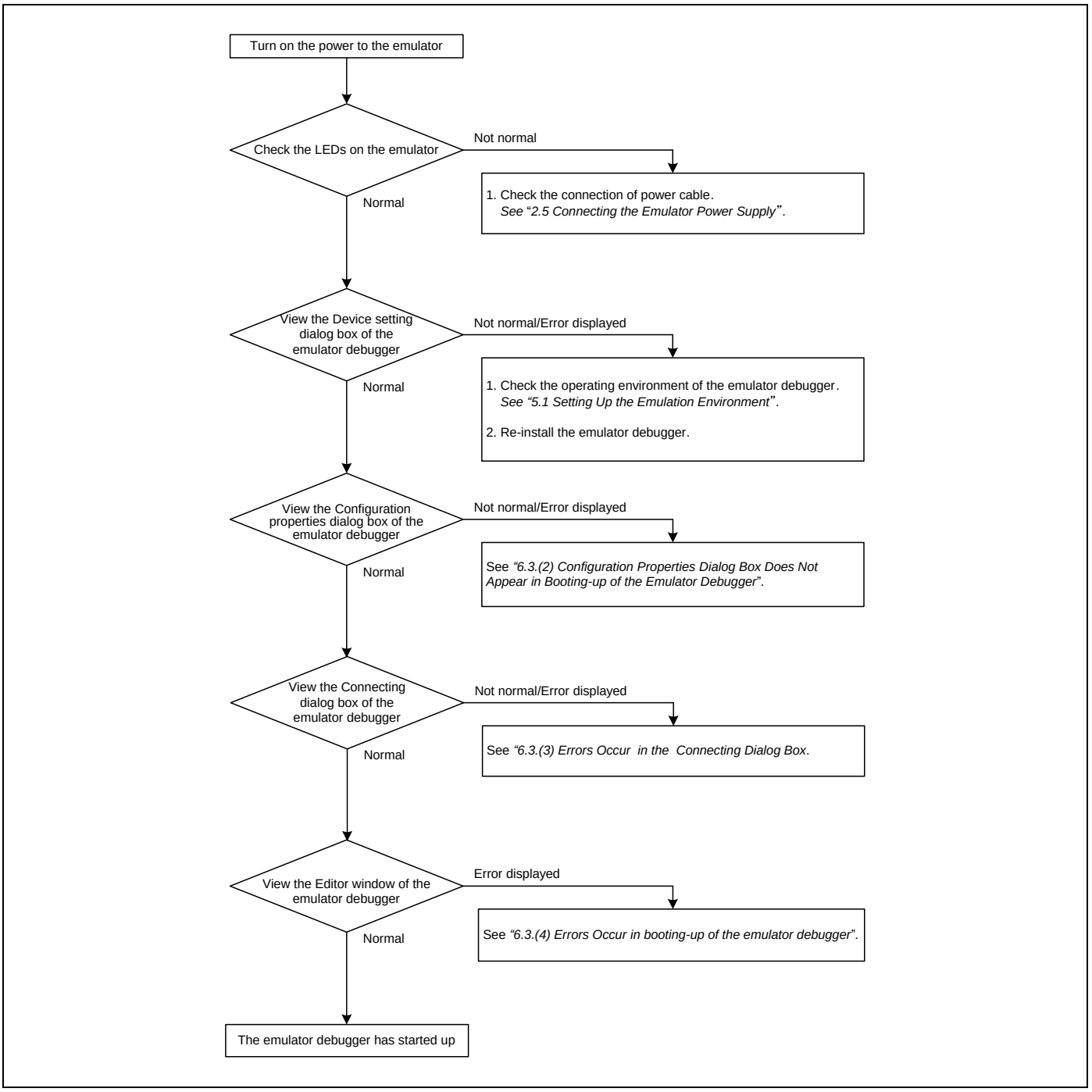


Figure 6.1 Flowchart for remediation of trouble

Downloaded from Arrow.com.

6.2 Error in Self-checking

When an error occurs in self-checking, check the following items.

(1) Re-check the connection between the E100 emulator main unit and the MCU unit.
(2) Download the proper firmware again.
(3) Check the error log from self-checking by the debugger software, and refer to the instructions given therein (see Figure 6.2).

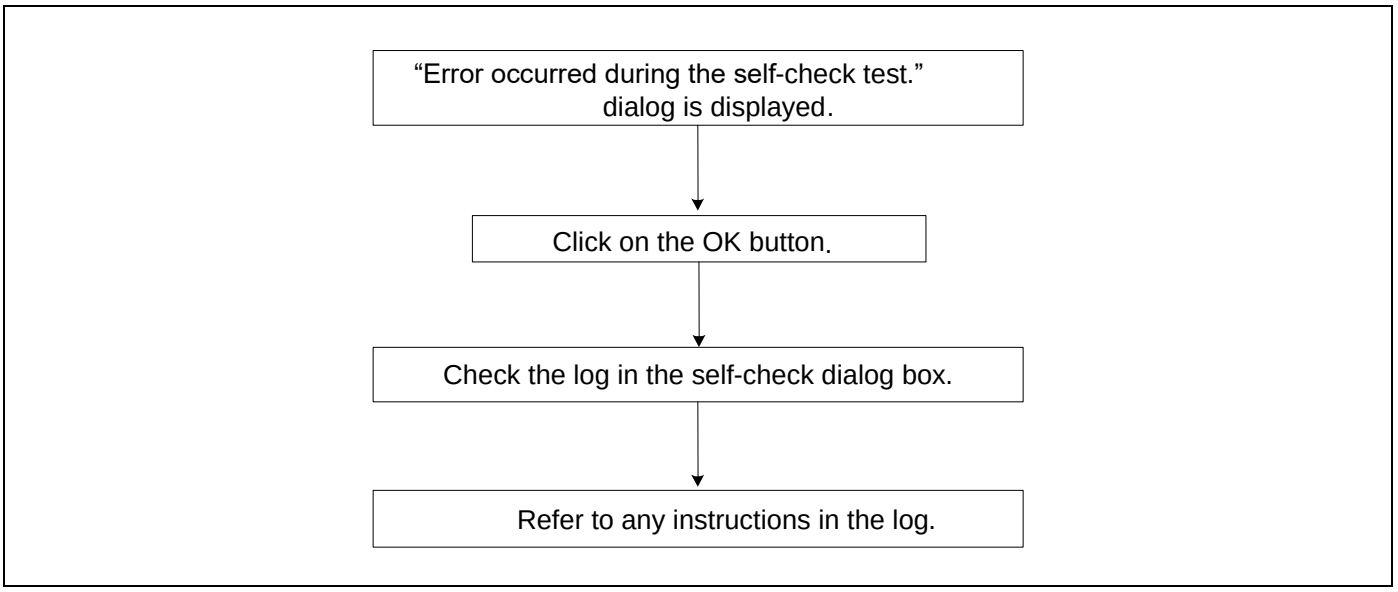


Figure 6.2 Flowchart for checking of an error in self-checking

Notes on the Self-checking:

- Disconnect the MCU unit from a converter board and the user system before you start self-checking.
- If the results of self-checking are not normal (excluding status errors of the target system), the product may have been damaged. Contact your local distributor.

Downloaded from Arrow.com.

6.3 Errors Reported in Booting-up of the Emulator

(1) States of the LEDs on the E100 are incorrect

Table 6.1 Points to check for errors indicated by incorrect states of the LEDs on the E100

Error	Connection to the user system	Point to check
SAFE LED remains lit.	-	Check that the USB cable is connected. *See "2.4 Connecting the Host Machine" (page 29).*
SAFE LED does not light up.	-	Re-check the connection between the E100 and the MCU unit. *See "2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit" (page 28).*
Target Status POWER LED does not light up.	Connected	Check that power (Vcc) is being correctly supplied to the user system and that the user system is properly grounded (GND).
Target Status RESET LED does not go out.	Connected	(1) Check that the reset pin of the user system is being pulled up. (2) When using the emulator without the user system, check to see if the converter board is disconnected from the emulator.

Downloaded from Arrow.com.

(2) Configuration Properties Dialog Box Does Not Appear in Booting-up of the Emulator Debugger

Table 6.2 Points to check for errors in booting-up of the emulator debugger (1)

Error	Point to check
Communication initialize error	Check all emulator debugger settings and the connection of the interface cable. *See "4. Preparing to Debug" (page 72).*
A communication error.	

(3) Error Occurs in the Connecting Dialog Box

Table 6.3 Points to check for errors in booting-up of the emulator debugger (2)

Error	Point to check
MCU unit is not connected.	Re-check the connection between the E100 and the MCU unit. *See "2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit" (page 28).*
The system configuration of the E100 emulator is not corresponding to the content of the E100.ENV file.	The combination between the emulator software and the MCU unit is not correct. Refer to the release notes of the emulator software, and confirm the combination between the emulator software and the MCU unit.
A timeout error. The MCU is in the reset state. Is system reset issued?	Check the oscillation of the oscillator module mounted on the MCU unit, and confirm that the oscillator module is properly mounted.
A timeout error. The MCU's internal clock is halted. Is system reset issued?	
A timeout error. No clock signal is supplied to the MCU. Is system reset issued?	
A timeout error. The power supply to the MCU is off. Is system reset issued?	Check that power is being correctly supplied to the user system and that the user system is properly grounded.

(4) Errors Occur in booting-up of the emulator debugger

Table 6.4 Points to check for errors in booting-up of the emulator debugger (3)

Error	Point to check
A timeout error.	(1) Check that the NQPACK etc. mounted on the user system is soldered properly. (2) Check that the connector is installed properly to the user system.

Downloaded from Arrow.com.

6.4 How to Request Support

After checking the items in "6. Troubleshooting (Action in Case of an Error)", contact us from the following URL.

https://www.renesas.com/contact

For a prompt response, please fill in the following information:

(1) Operating environment
- Operating voltage: _____ [V]
- Operating frequency: _____ [MHz]
- User-system: Connected/Disconnected
- Clock supply to the MCU: Internal oscillator/External oscillator

(2) Condition
- The emulator debugger starts up/does not start up
- The error is detected/not detected in the self-checking
- Frequency of errors: always/frequency ( )

(3) Details of request for support

Downloaded from Arrow.com.

7. Hardware Specifications

This chapter describes specifications of the MCU unit.

7.1 Target MCU Specifications

Table 7.1 lists the specifications of target MCUs which can be debugged with the MCU unit.

Table 7.1 Specifications of target MCUs for the R0E530650MCU00

Item	Description
Applicable MCU	M16C/60 Series M16C/65 and M16C/64A Group MCUs with 792KB ROM or less
Evaluation MCU	R5F3650TNFG-EVA ROM size : 8KB+16KB+256KB+512KB, RAM size: 47KB
Applicable MCU mode	Single-chip mode, memory expansion mode, microprocessor mode
Maximum ROM/RAM capacity	1. Internal flash ROM: 792 KB 0E000h--0FFFFh: Data flash 10000h--13FFFh: Program 2 ROM 40000h--FFFFFh: Program 1 ROM 2. Internal RAM: 47 KB 00400h--0BFFFh
Power supply voltage	Vcc1=Vcc2 : 2.7 to 5.5V
Operating voltage/frequency	Power supply voltage: 2.7 to 5.5V, 32MHz (with PLL)

Downloaded from Arrow.com.

7.2 Differences between the Actual MCU and Emulator

Differences between the actual MCU and emulator are shown below. When debugging the MCU using the MCU unit, be careful about the following precautions.

Note on Differences between the Actual MCU and Emulator:

- Operations of the emulator system differ from those of actual MCUs as listed below.
  (1) Reset condition
  Set the time for starting up (0.2 Vcc to 0.8 Vcc) 1 µs or less.
  (2) Initial values of internal resource data of an MCU at power-on
  (3) Interrupt stack pointer (ISP) after a reset is released
  (4) Capacities of the internal memories (ROM and RAM)
  The evaluation MCU of this product has RAM of 47 KB (00400h--0BFFFh) and flash ROM of 8 KB (0E000h--0FFFFh), 16 KB (10000h--13FFFh), 256 KB (40000h--7FFFFh) and 512 KB (80000h--FFFFFh).
  The ROM and RAM capacities may become larger than those of the target MCU specified on the Device page of the Device setting dialog box.
  (5) Oscillator circuit
  In the oscillator circuit where an oscillator is connected between pins X_{IN} and X_{OUT}, oscillation does not occur because a converter board is used between the evaluation MCU and the user system. It is the same for pins X_{CIN} and X_{COUT}.
  (6) A/D conversion
  The characteristics of the A/D converter differ from those of actual MCU because there are a converter board and other devices between the evaluation MCU and the user system.

Note on RESET# Input:

- A low input to pin RESET# from the user system is accepted only when a user program is being executed (only while the RUN status LED on the E100 upper panel is lit).

Note on Voltage Detection Circuit:

- This product differs from the actual MCU because there is a pitch converter board, etc. between the evaluation MCU and user system. Final evaluation of the voltage detection circuit (low voltage detection interrupt, low voltage detection reset, etc.) should be executed with the actual MCU.

Notes on Maskable Interrupts:

- Even if a user program is not being executed (including when run-time debugging is being performed), the evaluation MCU executes a debug control program. Therefore, timers and other components do not stop running. If a maskable interrupt is requested when the user program is not being executed (including when run-time debugging is being performed), the maskable interrupt request cannot be accepted, because the emulator disables interrupts. The interrupt request is accepted immediately after the user program execution is started.
- Take note that when the user program is not being executed (including when run-time debugging is being performed), a peripheral I/O interruption is not accepted.

Downloaded from Arrow.com.

Note on DMA Transfer:

- With this product, the user program is stopped with a loop program to a specific address. Therefore, if a DMA request is generated by a timer or other source while the user program is stopped, DMA transfer is executed. However, make note of the fact that DMA transfer while the program is stopped may not be performed correctly. Also note that the below registers have been changed to generate DMA transfer as explained here even when the user program is stopped.
  (1) DMA0 transfer count register TCR0 (2) DMA1 transfer count register TCR1
  (3) DMA2 transfer count register TCR2 (4) DMA3 transfer count register TCR3

Note on Final Evaluation:

- Be sure to evaluate your system with an evaluation MCU. Before starting mask production, evaluate your system and make final confirmation with a CS (Commercial Sample) version MCU.

Downloaded from Arrow.com.

7.3 Connection Diagram

7.3.1 Connection Diagram for the R0E530650MCU00

Figure 7.1 shows a partial circuit diagram of the connections of the R0E530650MCU00. This diagram mainly shows the circuitry to be connected to the user system. Other circuitry, such as that for the emulator's control system, has been omitted.

Table 7.2 shows IC electric characteristics of this product for reference purpose.

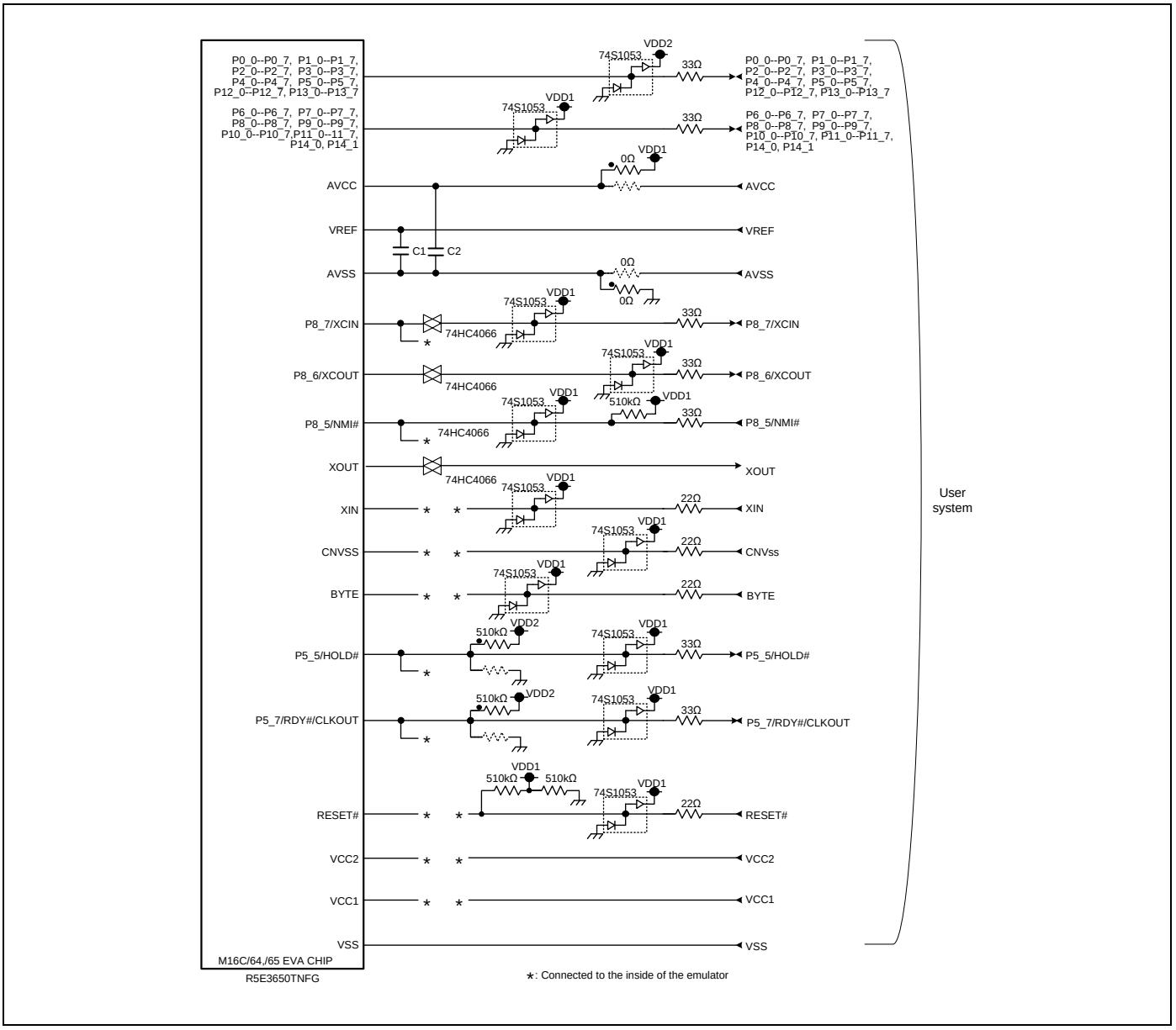


Figure 7.1 Connection diagram for R0E530650MCU00

Table 7.2 Electrical characteristics of the 74HC4066

Symbol	Item	Condition	Standard values			Unit
		Vcc	Min.	Standard	Max.	
RON	ON resistor	4.5V	-	96	170	Ω
ΔRON	ON resistor difference	4.5V	-	10	-	
IOFF	Leak current (Off)	12.0V	-	-	±100	nA
IIZ	Leak current (On, output: open)	12.0V	-	-	±100	

Downloaded from Arrow.com.

7.4 External Dimensions

7.4.1 External Dimensions of the E100 Emulator

Figure 7.2 shows external dimensions of the E100 emulator.

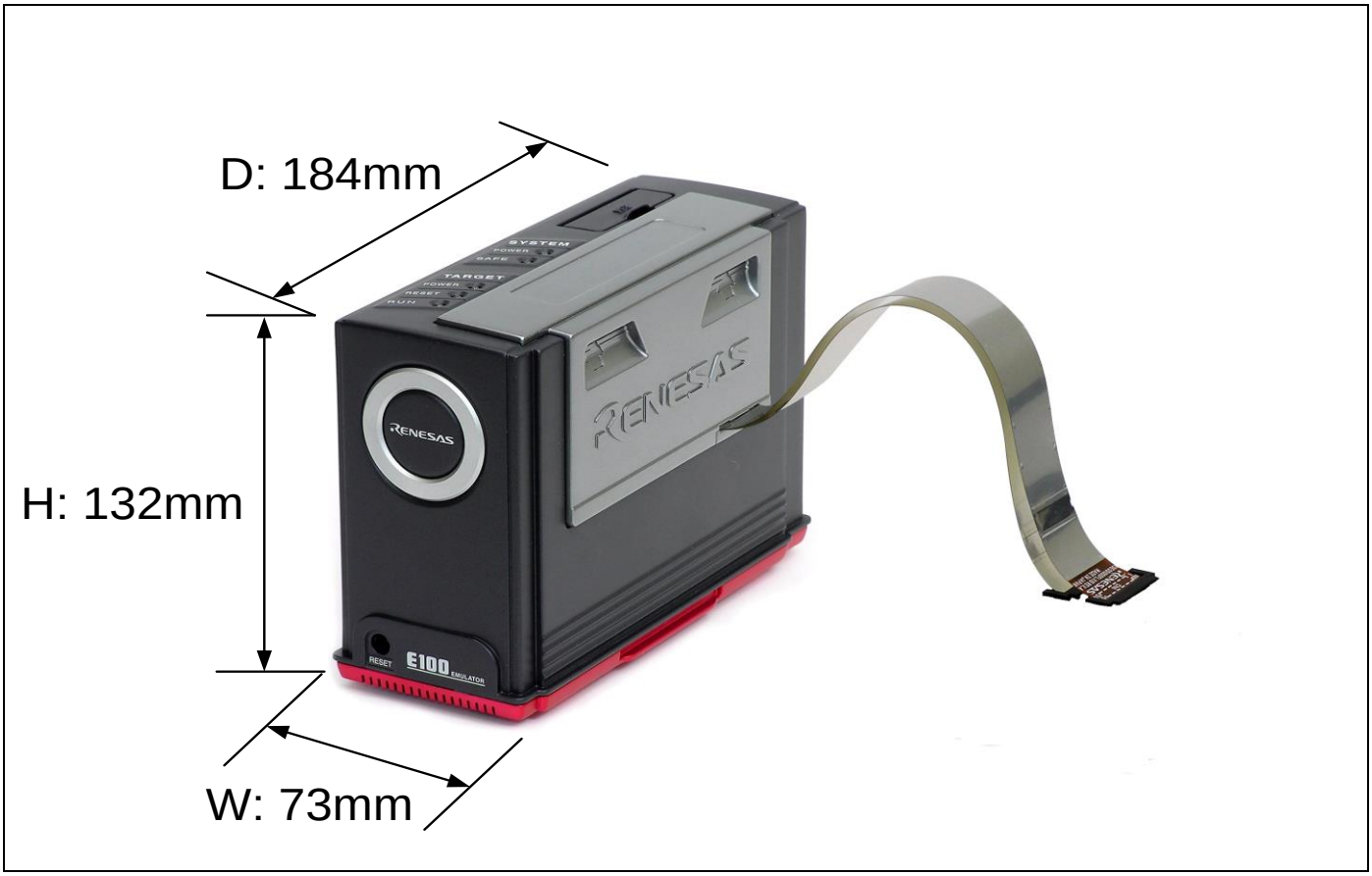


Figure 7.2 External dimensions of the E100 emulator

Downloaded from Arrow.com.

7.4.2 External Dimensions of the Converter Board R0E0100TNPFJ00

Figure 7.3 shows external dimensions and a sample foot pattern of the converter board R0E0100TNPFJ00 for a 100-pin 0.65mm pitch QFP.

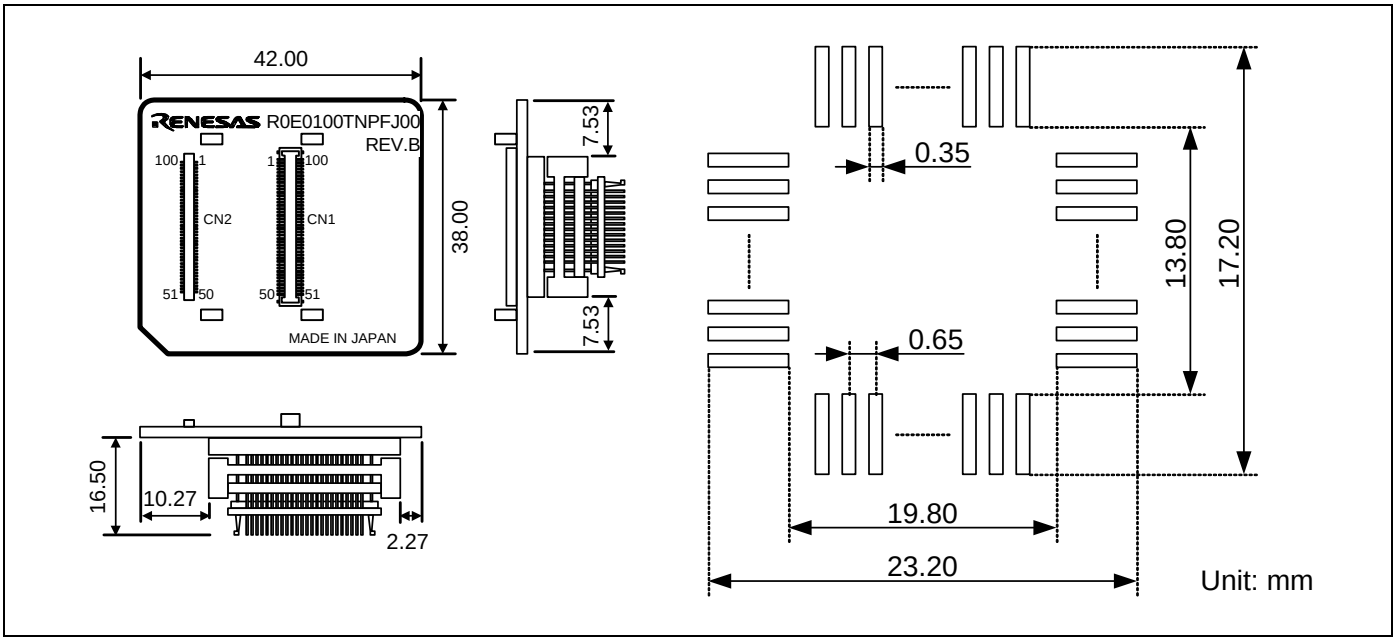


Figure 7.3 External dimensions and a sample foot pattern of the R0E0100TNPFJ00

Downloaded from Arrow.com.

7.4.3 External Dimensions of the Converter Board R0E0100TNPFK00

Figure 7.4 shows external dimensions and a sample foot pattern of the converter board R0E0100TNPFK00 for a 100-pin 0.5mm pitch LQFP.

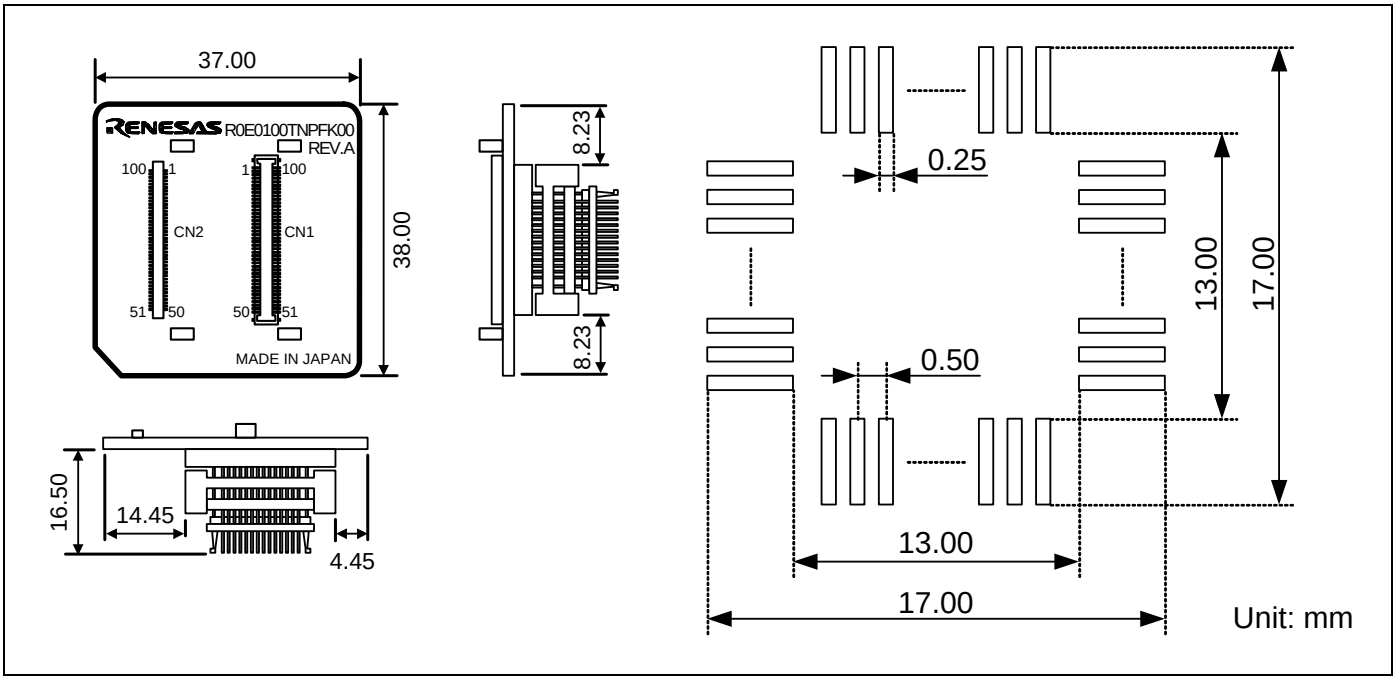


Figure 7.4 External dimensions and a sample foot pattern of the R0E0100TNPFK00

Downloaded from Arrow.com.

7.4.4 External Dimensions of the Converter Board R0E530650CFJ30

Figure 7.5 shows external dimensions and a sample foot pattern of the converter board R0E530650CFJ30 for an 80-pin 0.65mm pitch LQFP.

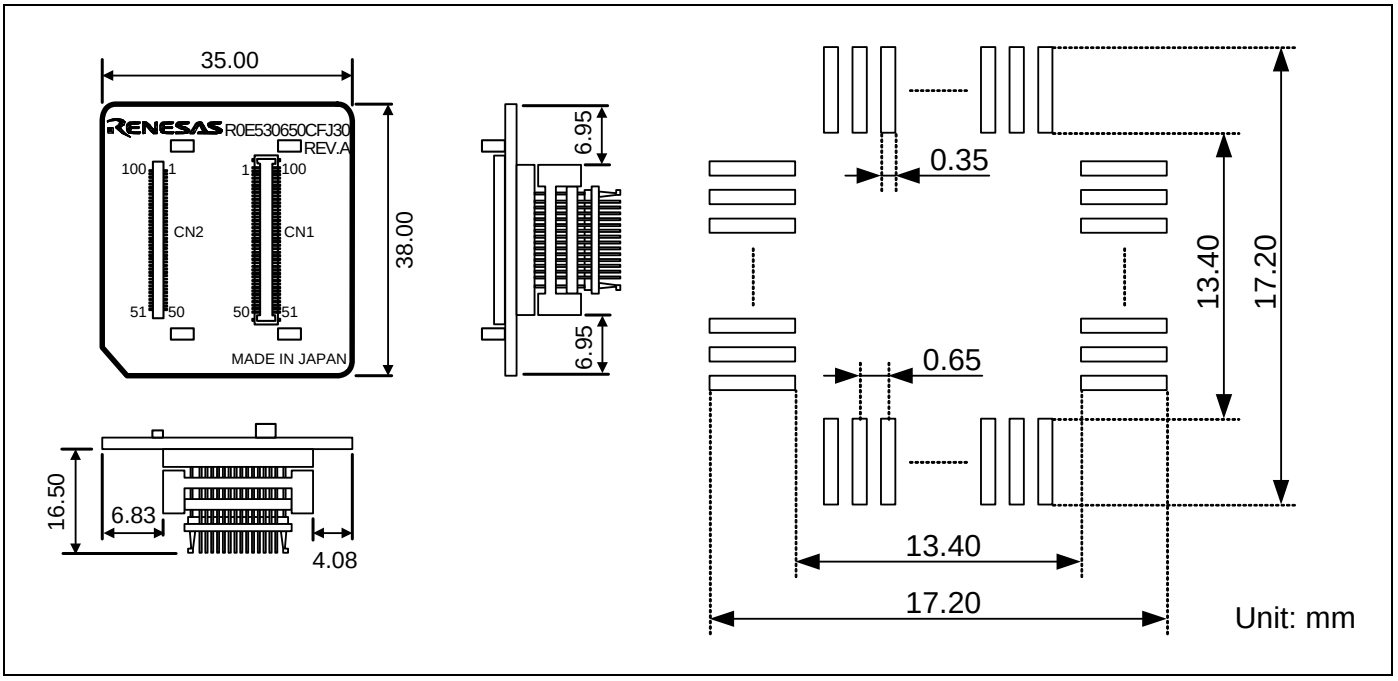


Figure 7.5 External dimensions and a sample foot pattern of the R0E530650CFJ30

Downloaded from Arrow.com.

7.4.5 External Dimensions of the Converter Board R0E530650CFK10

Figure 7.6 shows external dimensions and a sample foot pattern of the converter board R0E530650CFK10 for a 128-pin 0.5mm pitch LQFP.

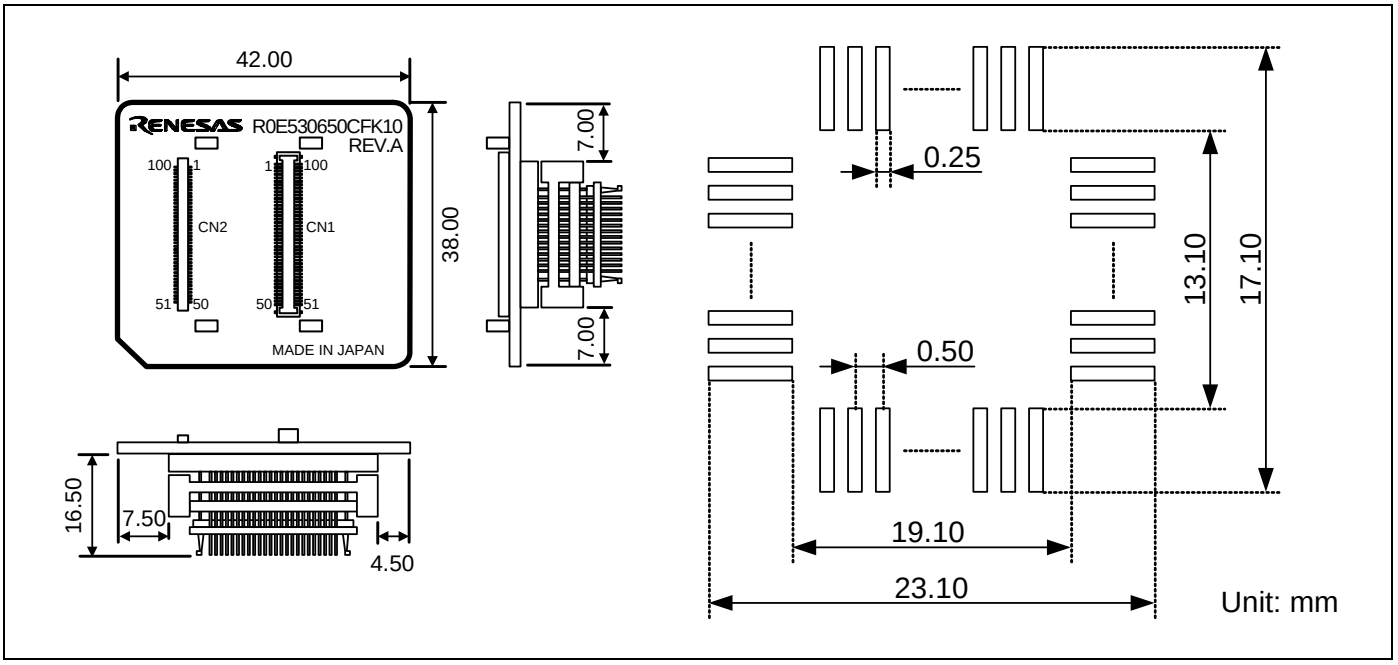


Figure 7.6 External dimensions and a sample foot pattern of the R0E530650CFK10

Downloaded from Arrow.com.

7.5 Notes on Using the MCU Unit

Notes on using the MCU unit are listed below. When you debug an MCU using the MCU unit, be careful about the following precautions.

Note on the Version of the Emulator Debugger:

- Be sure to use this product with the following emulator debugger.
  - M16C R8C E100 Emulator Software V.1.00 Release 00 or later

Notes on Downloading Firmware:

- Before using this product for the first time, it is necessary to download the dedicated firmware (emulator's control software installed in the flash memory in the E100). If you need to download at debugger startup, a message will appear. Download the firmware following the message.
- Do not shut off the power while downloading the firmware. If this happens, the product will not start up properly. If the power is shut off unexpectedly, re-download the firmware.
- Disconnect the MCU unit from the user system before you start downloading the firmware.

Notes on Self-checking:

- If self-checking does not result normally (excluding user system errors), the product may be damaged. Then contact your local distributor.
- Disconnect the MCU unit from the user system before you start self-checking.

Note on Quitting the Emulator Debugger:

- To restart the emulator debugger, always shut off the emulator power supply and then turn on it again.

Note on Display of MCU Status:

- "Status" you can view in the Connecting dialog box of the emulator debugger shows pin levels of the user system. Make sure that proper pin levels are selected according to the mode you use.

Note on the Register Used Exclusively by the Emulator:

- Do not access the registers D000h--D03Fh and D050h--D05Fh that are used exclusively by the emulator from the user program.

Note on Power-On Reset Function:

- The power-on reset function cannot be debugged.

Note on Low power consumption mode:

- When debugging in low power consumption mode, slow read mode, or the state that the flash memory is stopped, do not operate windows until the program stops at the breakpoint by setting the breakpoint at the line of the program which will be executed after each mode or state is cancelled.

Downloaded from Arrow.com.

Note on Clock Supply to the MCU:

- A clock supplied to the evaluation MCU is selected by the Emulator tab in the Init dialog box of the emulator debugger.
  (1) When "Emulator" is selected:
  A clock generated by the oscillator circuit board on the MCU unit is supplied. It is continually supplied regardless of the status of the user system clock and that of the user program execution.
  (2) When "User" is selected:
  A clock generated by the oscillator in the user system is supplied. It depends on the status of the oscillation (on/off) of the user system.
  (3) When "Generate" is selected:
  A clock generated by the dedicated circuit in the E100 is supplied. It is continually supplied regardless of the status of the user system clock and that of the user program execution.

Note on Stop and Wait Modes:

- Do not single step an instruction shifting to stop or wait mode. It may cause communication errors.

Note on the Watchdog Function:

- If the reset circuit of the user system has a watchdog timer, disable it when using the emulator.

Note on Protect Register:

- The protect is not canceled when bit 2 of protect register PRCR (PRC2), which enables writing into the port P9 direction register and the SI/Oi control register, is changed with the below procedure.
  (1) Step execution of an instruction setting PRC2 to "1"
  (2) Setting a break point between an instruction setting PRC2 to "1" and a point where the port P9 direction register or the SI/Oi control register is set
  (3) Setting PRC2 to "1" by the Memory window or Command Line window

Note on Access Prohibited Area:

- You cannot use internally reserved areas. Write signals to the areas will be ignored, and values read will be undefined.

Note on Breaks:

- The following break functions are available in the emulator debugger.
  (1) Software break
  This is a debugging function which generates a BRK interruption by changing an instruction at a specified address to a BRK instruction (00h) to break a program immediately before the system executes an instruction at a specified address. The instruction at the preset address will not be executed.
  (2) Hardware break
  This is a debugging function which breaks a program by setting the detection of an execution of an instruction at a specified address as a break event. The program will break after the instruction at the specified address is executed.
  (3) Exceptional event
  This is a debugging function which stops a program by an abnormal operation of the user program or overflow of each function's measurement counter, etc.

Downloaded from Arrow.com.

Notes on Software Breaks:

- The BRK instruction can be used for the emulator only. You cannot use it in a user program. As BRK instruction interrupt vector is used by the emulator system, the read data is different from expected value.
- You can neither set nor cancel a software breakpoint in the internal ROM area (A7000h--A7FFFh, E7000h--E7FFFh) of an MCU during user program execution, while you can set or cancel it in the internal RAM area of an MCU.
- When using the memory space expansion function (4-Mbyte mode), you can neither set nor cancel a software breakpoint in the internal ROM area (80000h--BFFFFh, E7000h--E7FFFh) of an MCU during user program execution, while you can set or cancel it in the internal RAM area of an MCU.

Notes on Power Supply to the User System:

- Pins Vcc1 and Vcc2 are connected to the user system to observe the voltage. Therefore, the power is not supplied to the user system from the emulator. Design your system so that the user system is powered separately.
- The voltage of the user system should be as follows.

  $2.7\ V \leq Vcc1 = Vcc2 \leq 5.5\ V$

Notes on Internal Flash ROM of the MCU:

- Because the number of write/erase cycles of the internal flash ROM of the MCU is limited, it must be replaced at the end of its service-life.
- If the following errors occur frequently when downloading a program, replace the MCU board.
  (1) An error has occurred in erasing of the flash ROM in the target MCU.
  (2) An error has occurred in programming of the flash ROM in the target MCU.
  (3) An error has occurred in verification of the flash ROM in the target MCU.

Note on Accessing Addresses 00000h and 00001h:

- With the M16C/60 Series MCUs, when a maskable interrupt is generated, the interrupt data (interrupt number and interrupt request level) stored in addresses 00000h and 00001h are read out. Also, the interrupt request bit is cleared when address 00000h or 00001h is read out. Consequently, when the address 00000h or 00001h readout instruction is executed or when address 00000h or 00001h is read out in the cause of a program runaway, a malfunction occurs in that the interrupt is not executed despite the interrupt request, because the request bit of the highest priority interrupt factor enabled is cleared.
  For this malfunction, when the reading out to address 00000h or 00001h is generated excluding the interrupt, an expansion monitor window will appear. At that time, check the user program. There is a possibility of wrong access.

Downloaded from Arrow.com.

Notes on Debugging in CPU Rewrite Mode:

- When you debug an M16C/60 Series MCU in CPU rewrite mode, do not change the block 0 area (FF000h--FFFFFh) of the flash memory. Otherwise, the emulator will be uncontrollable.
- If you check "Debug the program using CPU Rewrite Mode" in the System tab of the Configuration properties dialog box of the emulator debugger, you cannot use the following functions.
  - Other than 4-Mbyte Mode:
    Setting software breakpoints in the internal ROM area (A7000h--A7FFFh, E7000h--E7FFFh)
  - 4-Mbyte Mode:
    Setting software breakpoints in the internal ROM area (80000h--BFFFFh, E7000h--E7FFFh)
- In CPU rewrite mode and erase suspend mode, do not stop the program. And do not single step an instruction shifting to CPU rewrite mode or erase suspend mode. The emulator will be uncontrollable in CPU rewrite mode and erase suspend mode.
- To reference data after executing CPU rewrite, stop the program at other then a rewrite control program area and use the Memory window etc.
- As the following interrupt vectors are used by the emulator system, the read data is different from expected value.
  - Single-step (FFFECh--FFFEFh)
- As the user boot function cannot be debugged, do not enter the user boot mode.

Note on Memory Space Expansion Function (4-Mbyte mode):

- When using the memory space expansion function (4-Mbyte mode), a memory area that the evaluation MCU accesses is different depending on each setting. Refer to the tables below.

Access area of the evaluation MCU when using the memory space expansion function (4-Mbyte mode)

Processor mode	PM13[*1]	OFS[*2]	Access area of target MCU	Bank 0-5	Bank 6	Bank 7
Memory Expansion mode	1	0	40000h-7FFFFh	EXT[*3]	EXT	MAP[*4]
		1	40000h-7FFFFh	EXT	EXT	MAP
	0	0	40000h-7FFFFh	EXT	EXT	MAP
			80000h-BFFFFh	EXT	EXT	MAP
		1	40000h-7FFFFh	EXT	EXT	MAP
			80000h-BFFFFh	EXT	EXT	---
Microprocessor mode	---	0	40000h-7FFFFh	EXT	EXT	MAP
			80000h-BFFFFh	EXT	EXT	---
			C0000h-FFFFFh	---	---	MAP
		1	40000h-7FFFFh	EXT	EXT	MAP
			80000h-BFFFFh	EXT	EXT	---

*1: Shows bit 3 of address 00005h
*2: Shows bit 2 of address 0000Bh
*3: Shows a memory access on the user system
*4: MAP shows the area access according to the Emulation Memory Allocation setting on the Memory map page of the Configuration properties dialog box. When assigning a program, you cannot use the emulation memory. Use the memory on the user system (EXT) for it.

Downloaded from Arrow.com.

8. Maintenance and Warranty

This chapter covers basic maintenance, warranty information, provisions for repair and the procedures for requesting a repair.

8.1 User Registration

When you purchase our product, be sure register as a user. For user registration, refer to "User Registration" (page 15) of this user's manual.

8.2 Maintenance

(1) If dust or dirt collects on this product, wipe it off with a dry soft cloth.
Do not use thinner or other solvents because these chemicals can cause the surface coating to separate.

(2) When you do not use this product for a long period, disconnect it from the power supply, host machine and user system.

8.3 Warranty

(1) This product comes with a one-year warranty after purchase.
Should the product break down or be damaged while you're using it under normal condition based on its user's manual, it will be repaired or replaced free of cost.

(2) However, if the following failure or damage occurs to the product under warranty, the product will be repaired or replaced at cost.

a) Failure or damage attributable to the misuse or abuse of the product or its use under other abnormal conditions.
b) Failure or damage attributable to improper handling of the product after purchase, such as dropping of the product when it is transported or moved.
c) Failure or damage to the product caused by other pieces of equipment connected to it.
d) Failure or damage attributable to fire, earthquakes, thunderbolts, floods, or other natural disasters or abnormal voltages, etc.
e) Failure or damage attributable to modifications, repairs, adjustments, or other acts made to the product by other than Renesas Electronics Corporation.

(3) Consumables (e.g., sockets and adapters) are not covered by the aforementioned repair.

In the above cases, contact your local distributor. If your product is being leased, consult the leasing company or the owner.

Downloaded from Arrow.com.

8.4 Repair Provisions

(1) Repairs not covered by warranty

Problems arising in products for which more than one year has elapsed since purchase are not covered by warranty.

(2) Replacement not covered by warranty

If your product's fault falls into any of the following categories, the fault will be corrected by replacing the entire product instead of repairing it, or you will be advised to purchase a new product, depending on the severity of the fault.

- Faulty or broken mechanical portions
- Flaws, separation, or rust in coated or plated portions
- Flaws or cracks in plastic portions
- Faults or breakage caused by improper use or unauthorized repair or modification
- Heavily damaged electric circuits due to overvoltage, overcurrent or shorting of power supply
- Cracks in the printed circuit board or burnt-down patterns
- A wide range of faults that make replacement less expensive than repair
- Faults that are not locatable or identifiable

(3) Expiration of the repair period

When a period of one year has elapsed after production of a given model ceased, repairing products of that model may ecome impossible.

(4) Carriage fees for sending your product to be repaired

Carriage fees for sending your product to us for repair are at your own expense.

8.5 How to Make Request for Repair

If your product is found faulty, fill in a Repair Request Sheet downloadable from the following URL. And email the sheet and send the product to your local distributor.

https://www.renesas.com/repair

⚠ CAUTION

Note on Transporting the Product:

When sending your product for repair, use the packing box and cushion material supplied with this product when delivered to you and specify handling caution for it to be handled as precision equipment. If packing of your product is not complete, it may be damaged during transportation. When you pack your product in a bag, make sure to use conductive polyvinyl supplied with this product (usually a blue bag). When you use other bags, they may cause a trouble on your product because of static electricity.

Downloaded from Arrow.com.

Revision History

Rev.	Date	Description	
		Page	Summary
6.01	Dec.01.15	21	Regulatory Compliance Notices changed
6.02	Sep.01.21	—	The statement of inclusion of an AC adapter was removed.
		22	Regulatory Compliance Notices was changed.
		24	The specifications of the AC adapter were added.

Downloaded from Arrow.com.

E100 Emulator Main Unit for M16C/65 and M16C/64A Groups
User's Manual
R0E530650MCU00

Publication Date: Sep.01.21 Rev.6.02

Published by: Renesas Electronics Corporation

Downloaded from Arrow.com.

R0E530650MCU00
User's Manual

Renesas Electronics Corporation

R20UT0431EJ0602

Downloaded from Arrow.com.