

To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: <http://www.renesas.com>

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (<http://www.renesas.com>)

Send any inquiries to <http://www.renesas.com/inquiry>.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
 - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
 - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
 - "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

16

CE300H

H8/3052 Series Compact Emulator

Microcomputer Development Environment System

User's Manual

CE300H - Compact Emulator for H8/3052 Series Microcomputer

User's Manual

Published by : Renesas System Solutions Asia Pte. Ltd.

Date : December 16th , 2003, Version 2.0

Copyright(C) Renesas System Solutions Asia Pte. Ltd. All rights reserved.

Trademarks

a) General

All brand or product names used in this manual are trademarks or registered trademarks of their respective companies or organizations.

b) Specific

Microsoft Windows is registered trademarks of Microsoft Corporation.

Pentium is a registered trademark of Intel.

IMPORTANT INFORMATION

- **READ this user's manual before using this emulator product.**
- **KEEP the user's manual handy for future reference.**

Do not attempt to use the product until you fully understand its mechanism.

ALExxxx & CExxxx Emulator:

Throughout this document, the term "ALExxxx emulator" & "CExxxx emulator" shall be defined as the ALExxxx or CExxxx emulator, user system interface cable, PC interface board, and optional SIMM memory module produced only by Renesas System Solutions Asia Pte. Ltd. excludes all subsidiary products.

The user system or a host computer is not included in this definition.

Purpose of the Emulator Product:

This emulator product is a software and hardware development tool for systems employing the H8 series microcomputer. This emulator product must only be used for the above purpose.

Improvement Policy:

Renesas System Solutions Asia Pte. Ltd. (hereafter collectively referred to as Renesas) pursues a policy of continuing improvement in design, performance, and safety of the emulator product. Renesas reserves the right to change, wholly or partially, the specifications, design, user's manual, and other documentation at any time without notice.

Target User of the Product:

This product should only be used by those who have carefully read and thoroughly understood the information as well as restrictions contained in the user's manual. Do not attempt to use the product until you fully understand its mechanism. It is highly recommended that first-time users. Be instructed by users that are well versed in the operation of emulator product.

LIMITED WARRANTY

Renesas warrants its products to be manufactured in accordance with published specifications and free from defects in material and/or workmanship. The foregoing warranty does not cover damage caused by fair wear and tear, abnormal store condition, incorrect use, accidental misuse, abuse, neglect, corruption, misapplication, addition or modification or by the use with other hardware or software, as the case may be, with which the product is incompatible. No warranty of fitness for a particular purpose is offered. The user assumes the entire risk of using the product. Any liability of Renesas is limited exclusively to the replacement of defective materials or workmanship.

DISCLAIMER

RENESAS MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, ORAL OR WRITTEN, EXCEPT AS PROVIDED HEREIN, INCLUDING WITHOUT LIMITATION THEREOF, WARRANTIES AS TO MARKETABILITY, MECHERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE OR USE, OR AGAINST INFRINGEMENT OF ANY PATENT. IN NO EVENT SHALL RENESAS BE LIABLE FOR ANY DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY NATURE, OR LOSSES OR EXPENSES RESULTING FROM ANY DEFECTIVE EMULATOR PRODUCT, THE USE OF ANY EMULATOR PRODUCT OR ITS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCEPT AS EXPRESSLY STATED OTHERWISE IN THIS WARRANTY, THIS EMULATOR PRODUCT IS SOLD "AS IS". AND YOU MUST ASSUME ALL RISK FOR THE USE AND RESULTS OBTAINED FROM THE EMULATOR PRODUCT.

State Law:

Some states do not allow the exclusion or limitation of implied warranty or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may have other rights which may vary from state to state.

The Warranty is Void in the Following Cases:

Renesas shall have no liability or legal responsibility for any problems caused by misuse, abuse, misapplication, neglect, improper handling, installation, repair or modifications of the emulator product without Renesas's prior written consent or any problems caused by the user system.

Restrictions:

1. Earthing (applies only to manual for Renesas hardware products)
This hardware is designed for use with equipment that is fully earthed.
Ensure that all equipments used are appropriately earthed.
Failure to do so could lead to danger for the operator or damaged to equipments.
2. Electrostatic Discharge Precautions (applies only to manuals for Renesas hardware products)
This hardware contains devices that are sensitive to electrostatic discharge.
Ensure appropriate precautions are observed during handling and accessing connections.
Failure to do so could result in damage to the equipment.

All Right Reserved:

This user's manual and emulator product are copyrighted and all rights are reserved by Renesas. No part of this user's manual, all or part, any be reproduced or duplicated in any form, in hardcopy or machine-readable form, by any means available without Renesas's prior written consent.

Other Important Things to Keep in Mind:

1. Circuitry and other examples described herein are meant merely to indicate the characteristics and performance of Renesas Technology's semiconductor products. Renesas assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples described herein.
2. No license is granted by implication or otherwise under any patents or other rights of any third party or Renesas.
3. MEDICAL APPLICATIONS: Renesas Technology's products are not authorized for use in MEDICAL APPLICATIONS without the written consent of the appropriate officer of Renesas Technology (Asia Sales company). Such use includes, but is not limited to, use in life support systems. Buyers of Renesas Technology's products are requested to notify the relevant Renesas Technology (Asia Sales offices) when planning to use the products in MEDICAL APPLICATIONS.

Figures:

Some figures in this user's manual may show items different from your actual system.

Limited Anticipation of Danger:

Renesas cannot anticipate every possible circumstance that might involve a potential hazard. The warnings in this user's manual and on the emulator product are therefore not all inclusive. Therefore, you must use the emulator product safely at your own risk.

PREFACE

About this manual

This manual explains how to setup the Compact Emulator for usage of the H8/3052 series microcomputers. Operation using Renesas High-performance Embedded Workshop [pure debugger] software is also detailed in the context.

- | | |
|-------------------|---|
| Section 1 | Introduction
Gives an introduction to the system package and specification. It also highlights the precautionary measures when using the emulator. |
| Section 2 | Installation
Explains the initial installation and configuration of the PC, in order to operate the emulator. |
| Section 3 | Emulation System Setup
Usage Note 1 – Describes the setup steps before embarking on a new project development. |
| Section 4 | Emulation Functions
Usage Note 2 – Describes the various functions used in the CE300H-H8/3052 emulator. |
| Section 5 | H8/3052 Function Support
Usage Note 3 – Covers the emulation of the peripherals and features for the H8/3052 microcomputer in the CE300H-H8/3052 emulator. |
| Section 6 | Differences between the H8/3052 Microcomputer and the Emulator
Usage Note 4 – Highlights the differences between the usage of the emulator and the actual MCU. |
| Section 7 | User System Interface
Usage Note 5 – Details information about the user interface circuitry and memory access timing |
| Section 8 | Tutorial
Demonstrate step-by step guide on using HEW features |
| Section 9 | Diagnostic
Performs diagnostic test either in standalone mode or with HEW |
| Section 10 | Emulator Upgrade
Perform upgrade to the Emulator Logic & OS |
| Section 11 | Trouble-Shooting
Advises on some basic fault locating methods and commonly made mistakes. |

Assumptions

This manual assumes that the user has a working knowledge of

- High-Performance Embedded Workshop (Compiler, Assembler and Linker)
- H8/3052 Microcomputer Architecture
- General Hardware Interface Circuitry
- General Personal Computer Operation
- Microsoft Windows

Related Manuals:

- H8, H8/300 series C/C++ Compiler, Assembler, Optimizing Linkage Editor User's Manual
- High-Performance Embedded Workshop
- H8/3052 Series Hardware Manual
- SODIMM User's Manual
- CE300H User Cable User's Manual

Table of Contents

SECTION 1. INTRODUCTION.....	1
1.1 OVERVIEW	1
1.2 PACKAGE CONTENTS	2
1.2.1 Hardware Components.....	2
1.2.2 Software Components	2
1.2.3 Optional Components	2
1.3 SYSTEM REQUIREMENT.....	3
1.4 SUPPORTED MCU SERIES BY CE300H-H8/3052 EMULATOR.....	3
1.5 SUMMARY OF CE300H-H8/3052 EMULATOR FUNCTIONS.....	4
1.6 PRECAUTIONARY MEASURES	7
SECTION 2. INSTALLATION.....	8
2.1 EXPRESS SETUP STEPS	8
2.2 INSTALLING HEW DEBUGGER SOFTWARE.....	9
2.3 INSTALLATION DETAILS.....	12
2.4 POWER UP THE EMULATOR.....	12
2.5 CHECKING THE SYSTEM (STANDALONE MODE).....	13
2.5.1 LED indication.....	13
2.6 INSTALLING THE USB DRIVER.....	14
2.7 PC USB LINKAGE	18
2.8 HEW TOOLS ADMINISTRATION	19
2.9 SETUP COMPLETION.....	20
SECTION 3. EMULATION SYSTEM SETUP FOR DEVELOPMENT	21
3.1 CREATING THE WORKSPACE.....	21
3.1.1 Without Toolchain.....	22
3.1.2 With Toolchain (debug setting).....	24
3.2 CONFIGURE THE PLATFORM.....	27
3.2.1 Device and Package Selection	27
3.2.2 Operating Mode Selection	27
3.2.3 Clock Selection.....	28
3.2.4 User Signal Masking Control (RESET, NMI & STBY).....	28
3.2.5 Illegal Access Break Control	28
3.2.6 Downloading of Emulation Function (Programmable Function Generator).....	28
3.3 MEMORY MAPPING.....	29
3.4 CONNECTION TO TARGET SYSTEM	31
3.4.1 Target Power Supply.....	31
3.4.2 Types of User Interface Cable	32
SECTION 4. PERFORMING EMULATION.....	33
4.1 OVERVIEW	33
4.1.1 Emulation.....	33
4.1.2 High-performance Embedded Workshop.....	34
4.2 COMPILER CONFIGURATION & DEBUGGER SESSION.....	36
4.3 DEBUG SETTING.....	38
4.4 CONNECTING & DISCONNECTING WITH THE EMULATOR	39
4.5 EMULATOR SETTING	40
4.5.1 Configure Platform	40
4.5.2 Memory Mapping.....	42

4.6	VIEWING OF PROGRAM	46
4.6.1	Source Code level.....	46
4.6.2	Dis-assembly level.....	47
4.7	MCU RELATED INFORMATION	48
4.7.1	Registers.....	48
4.7.2	Memory	49
4.7.3	Status.....	50
4.7.4	PinView	52
4.7.5	Monitor (continuous POTF)	53
4.7.6	Symbol.....	54
4.7.7	I/O	58
4.7.8	Break Functions	59
4.7.9	Trace - PFG Function – Trace (Trace & Events Break).....	63
4.7.10	Stack Trace.....	64
4.7.11	Image.....	65
4.7.12	Waveform	67
4.8	MCU MEMORY MANIPULATION	68
4.9	EXECUTION OF MCU CODE	69
4.9.1	Reset CPU	69
4.9.2	Go, Reset Go, Goto Cursor, Set PC to Cursor, Run... ..	70
4.9.3	Single-Step	71
4.10	C-SOURCE LEVEL DEBUGGING.....	72
SECTION 5. H8/300H FUNCTION SUPPORTED.....		73
5.1	MCU OPERATING MODE SETTING.....	73
5.2	MEMORY AREA.....	74
5.2.1	Internal ROM Area	74
5.2.2	Internal RAM Area.....	74
5.2.3	Internal I/O Area.....	74
5.2.4	External Area	74
5.3	LOW POWER MODE.....	75
5.3.1	Hardware Standby Mode	75
5.3.2	Sleep and Software Standby Modes	75
5.4	INTERRUPTS	75
5.5	CONTROL INPUT SIGNALS (RES, NMI, STBY)	75
5.6	WATCH DOG TIMER (WDT)	75
5.7	INTEGRATED TIMER PULSE UNIT (ITU) AND TIMING PATTERN CONTROLLER (TPC)	76
5.8	SERIAL COMMUNICATIONS INTERFACE (SCI)	76
5.9	DMA CONTROLLER (DMAC)	76
5.10	WAIT STATE CONTROLLER	76
5.11	I/O PORTS.....	76
5.12	A/D AND D/A CONVERTER.....	76
5.13	REFRESH CONTROLLER.....	77
SECTION 6. DIFFERENCES BETWEEN H8/300H MCU AND EMULATOR.....		78
6.1	POWER UP AND RESET	78
6.2	USER INTERFACE	78
SECTION 7. USER SYSTEM INTERFACE.....		79
SECTION 8. TUTORIAL.....		83
8.1	OVERVIEW.....	83

8.2	TUTORIAL PROGRAM LISTING	84
8.3	TUTORIAL SETUP.....	86
8.4	DISPLAYING THE PROGRAM LISTING.....	87
8.5	SETTING A PC BREAKPOINTS	88
8.5.1	Setting a Program Count (PC) Breakpoint	88
8.6	EXECUTING THE PROGRAM	89
8.7	REVIEWING THE BREAKPOINTS	90
8.8	EXAMINING MCU REGISTERS	91
8.9	EXAMINING MEMORY AND VARIABLES	92
8.9.1	Viewing Memory	92
8.9.2	Watching Variables.....	93
8.10	STEPPING THROUGH A PROGRAM.....	95
8.10.1	Step In (F11)	95
8.10.2	Step Out (Shift F11)	95
8.10.3	Step Over (F10).....	96
8.11	USING THE HARDWARE BREAKPOINT.....	97
8.11.1	Defining a PFG Breakpoint	97
8.12	WATCHING LOCAL VARIABLES.....	99
8.13	USING THE TRACE BUFFER	100
8.14	SAVE THE SESSION	100
SECTION 9. DIAGNOSTIC.....		101
9.1	STANDALONE SELF TEST	101
9.2	HEW DEBUGGER DIAGNOSTIC TEST	102
SECTION 10. EMULATOR UPGRADE		103
SECTION 11. TROUBLE-SHOOTING.....		105
APPENDIX A : USER CONNECTOR PIN ASSIGNMENT		A-1
APPENDIX B : USER CONNECTOR SPECIFICATION		B-1
APPENDIX C : USER CONNECTOR PIN LAYOUT		C-1
APPENDIX D : CASING ASSEMBLY		D-1
APPENDIX E : SUMMARY OF HEW DEBUGGER		E-1
APPENDIX F : TECHNICAL SPECIFICATION		F-1
APPENDIX G : FREQUENTLY ASKED QUESTIONS.....		G-1

RENESAS TECHNOLOGY (ASIA SALES OFFICES)

Figures & Tables

FIGURE 1	CE300H-H8/3052 EMULATOR	1
FIGURE 2	CE300H-H8/3052 EMULATOR PACKAGE	2
FIGURE 3	BASIC SETUP OF CE300H-H8/3052 EMULATOR.....	8
FIGURE 4	COMPACT EMULATOR FRONT PANEL.....	8
FIGURE 5	RUN DIALOGUE BOX.....	9
FIGURE 6	HEW DEBUGGER INSTALLER WELCOME! SCREEN.....	9
FIGURE 7	SELECT DESTINATION DIRECTORY SCREEN	10
FIGURE 8	SELECT COMPONENTS.....	10
FIGURE 9	SETUP DIRECTORY.....	11
FIGURE 10	INSTALLATION COMPLETION.....	11
FIGURE 11	HEW DEBUGGER ICONS.....	12
FIGURE 12	POWER-SUPPLY PLUG.....	12
FIGURE 13	LED DESCRIPTIONS	13
FIGURE 14	FOUND NEW DEVICE.....	14
FIGURE 15	LOCATE DRIVER FILES.....	14
FIGURE 16	SELECTING THE USB DRIVER LOCATION.....	15
FIGURE 17	WIN2K DRIVER LOCATION.....	15
FIGURE 18	WIN 9X DRIVER LOCATION.....	16
FIGURE 19	SELECTED DRIVER FILE WINDOW.....	16
FIGURE 20	COMPACT EMULATOR USB DRIVER INSTALLED	17
FIGURE 21	DEVICE MANAGER.....	18
FIGURE 22	HIGH-PERFORMANCE EMBEDDED WORKSHOP WINDOW.....	19
FIGURE 23	EXECUTE HEW DEBUGGER FROM START MENU	21
FIGURE 24	SELECT PLATFORM DIALOGUE BOX.....	22
FIGURE 25	HEW DEBUGGER (WITHOUT TOOLCHAIN) START-UP MESSAGES	22
FIGURE 26	SELECT TARGET.....	23
FIGURE 27	DEBUGGER CONFIGURATION.....	23
FIGURE 28	DEBUGGER SETTING SUMMARY.....	23
FIGURE 29	DEBUGGER SETTING SUMMARY.....	24
FIGURE 30	STEP 7 SELECTION OF TARGET.....	24
FIGURE 31	DEBUG SETTING WINDOW [ABSOLUTE PATH]	25
FIGURE 32	DEBUG SETTING WINDOW [RELATIVE PATH]	26
FIGURE 33	CONFIGURATION WINDOW.....	27
FIGURE 34	MEMORY MAPPING WINDOW	29
FIGURE 35	EDITING THE MEMORY MAPPING.....	30
FIGURE 36	TARGET SUPPLY IN STATUS WINDOW.....	31
FIGURE 37	USER INTERFACE CABLE – DIRECT CONNECTION	32
FIGURE 38	USER INTERFACE CABLE – ACTUAL FOOTPRINT.....	32
FIGURE 39	HIGH-PERFORMANCE EMBEDDED WORKSHOP WINDOW	33
FIGURE 40	HIGH-PERFORMANCE EMBEDDED WORKSHOP WINDOW	34
FIGURE 41	HIGH-PERFORMANCE EMBEDDED WORKSHOP WINDOW	36
FIGURE 42	OPTION - EMULATOR	40
FIGURE 43	CONFIGURE PLATFORM.....	40
FIGURE 44	SELECT PFG FUNCTION	41
FIGURE 45	UVCC IN STATUS WINDOW.....	42
FIGURE 46	MAPPING OF OPTIONAL MEMORY SHOWN IN SYSTEM STATUS WINDOW	43
FIGURE 47	MEMORY MAPPING.....	44
FIGURE 48	EDIT MEMORY MAPPING	44

FIGURE 49	MEMORY MAPPING.....	44
FIGURE 50	MAPPING RESOLUTION & RANGE	45
FIGURE 51	SOURCE LEVEL	46
FIGURE 52	DISASSEMBLY WINDOW (MIXED DISPLAY).....	47
FIGURE 53	VIEW CPU	48
FIGURE 54	REGISTER.....	48
FIGURE 55	SET MEMORY	49
FIGURE 56	STATUS –MEMORY WINDOW	50
FIGURE 57	STATUS – PLATFORM WINDOW	51
FIGURE 58	STATUS – EVENTS WINDOW	51
FIGURE 59	PINVIEW (CHIP AND TREE VIEW)	52
FIGURE 60	MONITOR WINDOW	53
FIGURE 61	VIEW SYMBOL	54
FIGURE 62	LABEL.....	55
FIGURE 63	WATCH	56
FIGURE 64	LOCALS.....	57
FIGURE 65	TOOLTIP	57
FIGURE 66	INPUT AND OUTPUT REGISTER	58
FIGURE 67	VIEW CODE.....	59
FIGURE 68	BREAKPOINT SETTING.....	61
FIGURE 69	PFG 1 - BREAKPOINT SETTING.....	61
FIGURE 70	PFG 2 – BREAKPOINT SETTING	62
FIGURE 71	TRACE.....	63
FIGURE 72	STACK TRACE	64
FIGURE 73	VIEW GRAPHIC	65
FIGURE 74	BITMAP	65
FIGURE 75	WAVEFORM	67
FIGURE 76	MEMORY FUNCTIONS.....	68
FIGURE 77	DEBUG FUNCTIONS	69
FIGURE 78	RUN PROGRAM	70
FIGURE 79	STEP PROGRAM	71
FIGURE 80	BASIC BUS CYCLE TIMING IN EXPANDED MODE	80
FIGURE 81	BASIC BUS CYCLE TIMING IN EXPANDED MODE	80
FIGURE 82	INTERFACING CIRCUITRY	81
FIGURE 83	RESISTOR PACK POSITION.....	82
FIGURE 84	RESETPRG.C FILE AFTER RESET CPU	87
FIGURE 85	SETTING A BREAKPOINT	88
FIGURE 86	PROGRAM BREAK	89
FIGURE 87	PLATFORM PAGE OF STATUS WINDOW.....	89
FIGURE 88	EVENTPOINTS WINDOW	90
FIGURE 89	POPUP MENUS IN EVENTPOINTS WINDOW	90
FIGURE 90	CPU REGISTERS WINDOW	91
FIGURE 91	CHANGING REGISTER VALUE.....	91
FIGURE 92	OPEN MEMORY WINDOW	92
FIGURE 93	MEMORY WINDOW	92
FIGURE 94	INSTANT WATCH DIALOG.....	93
FIGURE 95	WATCH WINDOW.....	93
FIGURE 96	ADD WATCH DIALOG	94
FIGURE 97	WATCH WINDOW.....	94
FIGURE 98	STEP IN	95
FIGURE 99	STEP OVER.....	95

FIGURE 100	STEP OVER.....	96
FIGURE 101	BREAKPOINT SETTING DIALOG	97
FIGURE 102	EVENTPOINTS WINDOW	97
FIGURE 103	PFG BREAK	98
FIGURE 104	LOCALS WINDOW	99
FIGURE 105	TRACE WINDOW	100
FIGURE 106	TOOLKIT SELECTION	102
FIGURE 107	DIAGNOSTIC WINDOW	102
FIGURE 108	CE PROGRAMMER WINDOW	103
FIGURE 109	CE OS AND LOGIC PROGRAMMING	104
TABLE 1	CE300H-H8/3052 EMULATOR FUNCTIONS	4
TABLE 2	TYPES OF BREAKS ENCOUNTERED DURING EMULATION.	60
TABLE 3	MCU OPERATING MODES	73

Section 1. Introduction

1.1 Overview

The CE300H-H8/3052 *Compact Emulator* is one of the Renesas's Development Tool series. It is produced as a cost-effective, easy-to-use support tools.

The CE300H-H8/3052 Compact Emulator has an easy to setup USB 2.0 link and a common user-friendly Windows-based interface, *High-performance Embedded Workshop* (HEW). Its flexibility is evident in its *Programmable Function Generator* (PFG) which allows user to select and download different emulation features. It has a built-in *Self Test* module that will inform user of its working condition through the LED indicator. Moreover, it has integrated a *PinView* module, that allow user to have an instant graphical view of the microcomputer pins' status.

The CE300H-H8/3052 Compact Emulator can be used as a standalone unit for software development and debugging. It can also be connected to a target system via a user interface cable for troubleshooting user's hardware. It is an indispensable tool that caters to the needs of an embedded designer.



Figure 1 CE300H-H8/3052 Emulator

1.2 Package Contents

The CE300H-H8/3052 emulator is supplied in a package containing the following components:

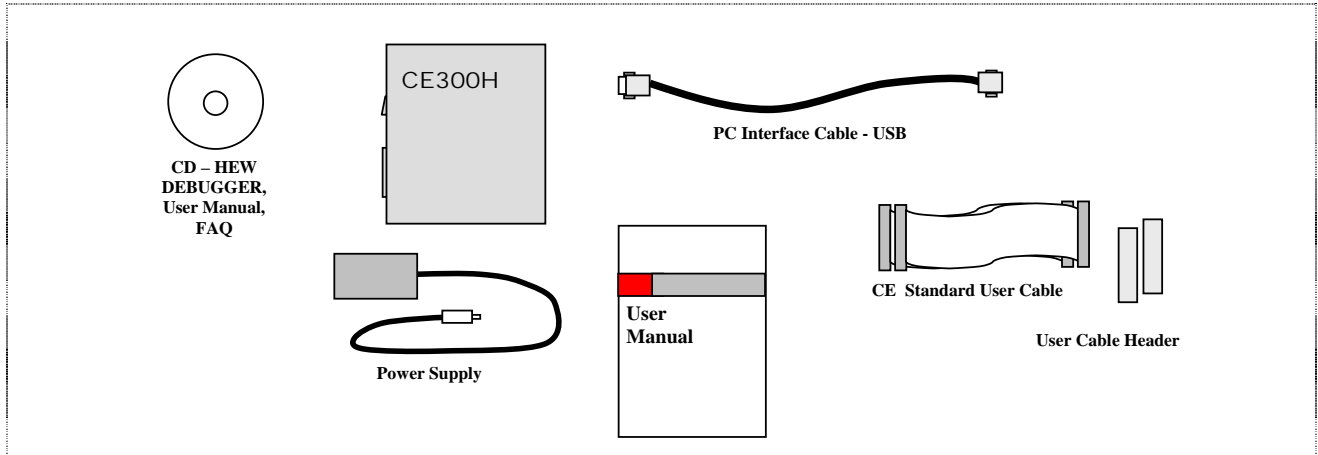


Figure 2 CE300H-H8/3052 Emulator Package

1.2.1 Hardware Components

The hardware components included in the package are listed below.

- 1 x CE300H-H8/3052 emulator
- 1 x PC interface cable - USB 2.0 cable
- 2 x 300mm KEL user cable to connect to target system (Part No.: 8822E-080-171-30-AC)
- 2 x KEL connector plugs for the target system (Part No.: 8830E-080-170S)
- 1 x 5V-2.8A power supply adaptor (Input Ratings: 110-240V/50-60Hz)

1.2.2 Software Components

The software components included in the package are listed below.

- 1 x CD containing HEW Debugger Installation, user's manual and FAQ

Before proceeding, user has to check for all the items listed in the packing list. Please contact the relevant Renesas Sales Office if any item is missing.

1.2.3 Optional Components

The following items can be purchased to further enhance the emulation capability:

- Actual footprint user cable
- 2Mbytes of SODIMM memory

1.3 System Requirement

The following items are not supplied but they are required to be used with the CE300H-H8/3052 emulator.

- A minimum Pentium™ or equivalent based processor personal computer with USB version 1.1.
- Microsoft Windows 98 (2nd Edition), Microsoft Windows 2000, Microsoft Windows XP, or Microsoft Windows ME
- Approximately 60Mbytes of hard disk space for the installation

1.4 Supported MCU Series by CE300H-H8/3052 Emulator

- H8/3003 - H8/3001, 3002, 3003*
- H8/3005 - H8/3004, 3005
- H8/3007 - H8/3006, 3007, 3008
- H8/3022 - H8/3022F, 3022, 3021, 3020
- H8/3024 - H8/3024F, 3026, 3026F
- H8/3032 - H8/3032, 3031, 3030
- H8/3035 - H8/3035, 3034, 3033
- H8/3039 - H8/3039F, 3039, 3038, 3037, 3036
- H8/3042 - H8/3042, 3041, 3040
- H8/3048 - H8/3048F, 3048F-ONE, 3048, 3047, 3045, 3044
- H8/3052 - H8/3052F, 3052BF, 3052 (Hi-speed)
- H8/3062 - H8/3062(R)F, 3062AF, 3062BF, 3062BF(Hi-speed),
3062, 3061, 3060, 3064F, 3064BF, 3064AF (Hi-speed)
- H8/3067 - H8/3067(R)F, 3067, 3066, 3065
- H8/3068 - H8/3068F
- H8/3069 - H8/3069F
- H8/3078 - H8/3078, 3078 (Hi-speed)*

* Limitation

Limited support for H8/3003:

- Does not support the following Pins and functions:

1. Port 5[7..4] multiplexed with A[23..20]
2. Port C[0..7] multiplexed with CS [4-7] and TEND2, DREQ2, TEND3, DREQ3, IRQ6,7.

Limited support for H8/3078:

1. Does not support the RAM overlap function

1.5 Summary of CE300H-H8/3052 Emulator Functions

Table 1 CE300H-H8/3052 Emulator Functions

Items	Specifications
Supported Microcomputers	<ul style="list-style-type: none"> • H8/3003 – H8/3001, 3002, 3003* (limitation –Page 3) • H8/3005 – H8/3004, 3005 • H8/3007 – H8/3006, 3007, 3008 • H8/3022 – H8/3022F, 3022, 3021, 3020 • H8/3024 – H8/3024, 3024F, 3026, 3026F • H8/3032 – H8/3032, 3031, 3030 • H8/3035 – H8/3035, 3034, 3033 • H8/3039 – H8/3039F, 3039, 3038, 3037, 3036 • H8/3042 – H8/3042, 3041, 3040 • H8/3048 – H8/3048F, 3048F-ONE, 3048, 3047, 3045, 3044 • H8/3052 – H8/3052F, 3052BF, 3052 (Hi-speed) • H8/3062 – H8/3062(R)F, 3062AF, 3062BF, 3062BF(Hi-speed), 3062, 3061, 3060, 3064F, 3064BF, 3064AF (Hi-speed) • H8/3067 – H8/3067(R)F, 3067, 3066, 3065 • H8/3068 – H8/3068F • H8/3069 – H8/3069F • H8/3078 – H8/3078, 3078 (Hi-speed)* (limitation –Page 3)
Operating Frequency	<ul style="list-style-type: none"> • 2 MHz (Min) • 25 MHz (Max) (# Device dependent)
Operating Modes	<ul style="list-style-type: none"> • Mode 1-7 (# Device dependent) • Target
Supported Operating Voltage Range	<ul style="list-style-type: none"> • 2.7 Volts - 5 Volts (# Device dependent)
Host Machine	<ul style="list-style-type: none"> • Minimum Pentium™ or equivalent processor PC • Microsoft Windows 98SE, 2000, XP, or ME
Host Interface	<ul style="list-style-type: none"> • USB 2.0 (480 Mbps) or • USB Ver 1.1 (12Mbps)
Supported File Formats	<ul style="list-style-type: none"> • Motorola S-type, ELF/Dwarf2
Interface Software	<ul style="list-style-type: none"> • High-performance Embedded Workshop (HEW)
Emulation Functions	<ul style="list-style-type: none"> • Perform real-time emulation of a target program • C-source level debugging (e.g. C-level step execution, instant watch, view labels...) • Display MCU operating status (e.g. Run, Sleep and Standby) during User run mode. • Display accessed address during execution

Items	Specifications
	<ul style="list-style-type: none"> • Modify and display MCU registers • Assemble instruction mnemonics • Dis-assemble memory contents • Radix (Bin, Octal, Dec, Hex, ASCII) input • Loading and saving of session • Reset MCU • Go at current PC/ Reset Go/ Go to Cursor
Emulation Memory	<ul style="list-style-type: none"> • 2014 Kbytes internal ROM (max) (# Device dependent) • 32 Kbytes internal RAM (max) (# Device dependent) • Provision of 4 banks of selectable optional memory block – 2Mbytes SODIMM. (Not in Package)
Memory Functions	<ul style="list-style-type: none"> • Copy, Search, Fill, Load and Save memory functions. • Memory can be edited and viewed in ASCII/ Byte/ Word/ Long/ Single Float/ Double Float format
Parallel On the Fly	<ul style="list-style-type: none"> • Memory viewing and modification during user program execution
Single Step Functions	<ul style="list-style-type: none"> • Step In /Step Out /Step Over
Breaks	<ul style="list-style-type: none"> • PC breakpoints (max. 255) • Reserved Access Break • Write-Protected Break • User break by ESC key • Two Events breaks (address/address mask /access)
Programmable Function Generator	<ul style="list-style-type: none"> • User configurable emulator functions • Current function includes: <ul style="list-style-type: none"> <u>PFG 1.</u> 256 cycles of Trace Buffer & 2 channels of Address/Data/Mask Breakpoints <u>PFG 2.</u> 2 channels of sequence event breaks
<u>PFG 1.</u> - Trace	<ul style="list-style-type: none"> • Trace memory size: 64-bit x 256 bus cycle • Signals displayed of each bus cycle: <ul style="list-style-type: none"> ○ 24-bit address bus ○ 16-bit data bus ○ Displays mnemonics of instructions executed during emulation • Filter and search for specified trace information (address or data). • Save Trace data into ASCII format
- Event Breaks	<ul style="list-style-type: none"> • Two event breaks are provided to trigger on the following conditions; <ul style="list-style-type: none"> ○ Address range ○ Data with Masking ○ Event counter

Items	Specifications
<u>PFG 2.</u> - Sequence Event Breaks	<ul style="list-style-type: none"> Two event breaks are provided to trigger on the following conditions; <ul style="list-style-type: none"> Address range Data with Masking Break occurs on <ul style="list-style-type: none"> Break Event 1 THEN Break Event 2 Break Event 1 OR Break Event 2 Break Event 1 AND Break Event 2 (no predefined sequence)
PinView	<ul style="list-style-type: none"> Provides instant graphical package view of all the pins of selected microcomputer. Provides an alternative view in text form.
Clock selection	<ul style="list-style-type: none"> Software selection of 2 types of clocks: <ul style="list-style-type: none"> User system clock (via user cable) Emulator internal clock 2–25 MHz at 100KHz resolution
Execution time measurement	<ul style="list-style-type: none"> Measure the start (run) till end (break) of an execution. Resolution: 100ns No upper limit
User Cable interface	<ul style="list-style-type: none"> Two fine-pitch user cable assemblies (KEL-8822E-080-171-040-AC) are used to interface to two 1.27mm pitch plug (KEL-8830E-080-170S) on both sides of the emulator and user target. (Provided in package) Selected actual footprint user cables for each microcomputer series are also available. (Not in package)
Voltage Follower	<ul style="list-style-type: none"> Automatic tracking of the target system supply voltage to ensure that the emulator draws no power.
Power Supply	<ul style="list-style-type: none"> Power Adaptor Input: 100-240 Vac, 50-60 Hz Emulator Voltage : 5V Regulated Current consumption : 2.8A (max)
Environmental	<ul style="list-style-type: none"> Operating Temperature : 10°C to 35°C Humidity : 30% to 85% RH No condensation No corrosive gas
Field Upgrade	<ul style="list-style-type: none"> Re-programming of emulator logic and flash OS via USB interface.

1.6 Precautionary Measures

The emulator must be handled with care. Otherwise, it may not work as intended.

Before Power On

- Check all components by referring to the packing list
- Never place heavy objects on the casing

Observe the following conditions in which the emulator is to be used:

- Keep out of direct sunlight or heat
- Use in an environment with constant room temperature and humidity
- Protect the emulator from dust
- Avoid subjecting the emulator to excessive vibration
- Protect the emulator from excessive impact and stresses
- Check the emulator's specifications such as power output, voltage, and frequency before connecting the power supply,
- When moving the emulator, pack with a good protective box or otherwise damage it.
- Never allow exposed power supply to come into contact with the emulator casing which is grounded.

Section 2. Installation

2.1 Express Setup Steps

The following are the basic steps involved in setting up the emulator.

- Unpack and verify parts against packing list
- Install HEW Debugger by running setup file
- Power up the CE300H-H8/3052 emulator (Performed after installation of HEW software)
- Connect the CE300H-H8/3052 emulator to the PC USB port
 - PC will prompt to install the USB Driver
- Run HEW

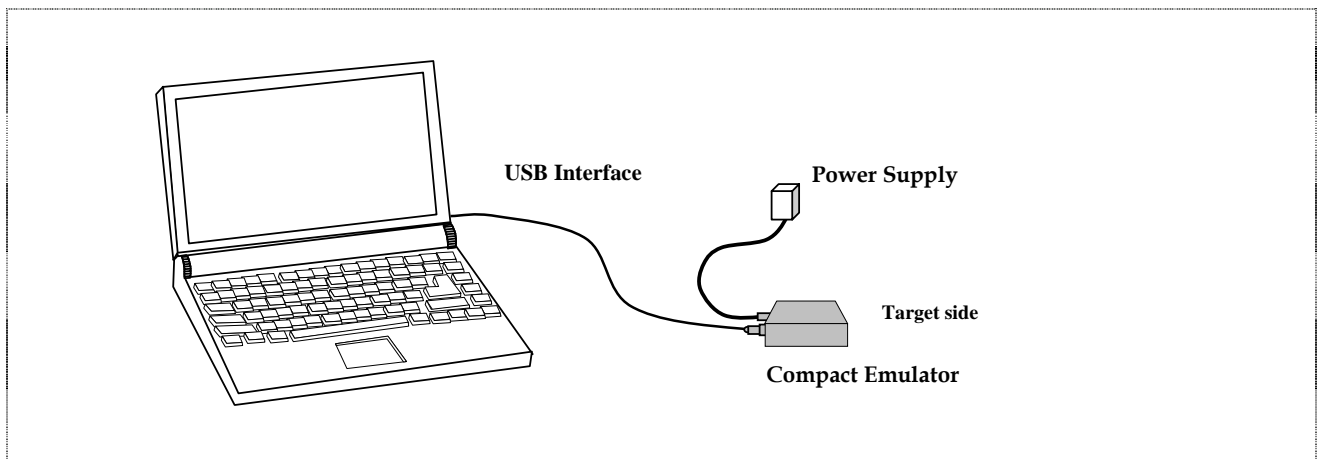


Figure 3 Basic Setup of CE300H-H8/3052 Emulator

The following topics detail the essential steps before proper emulation can be started.

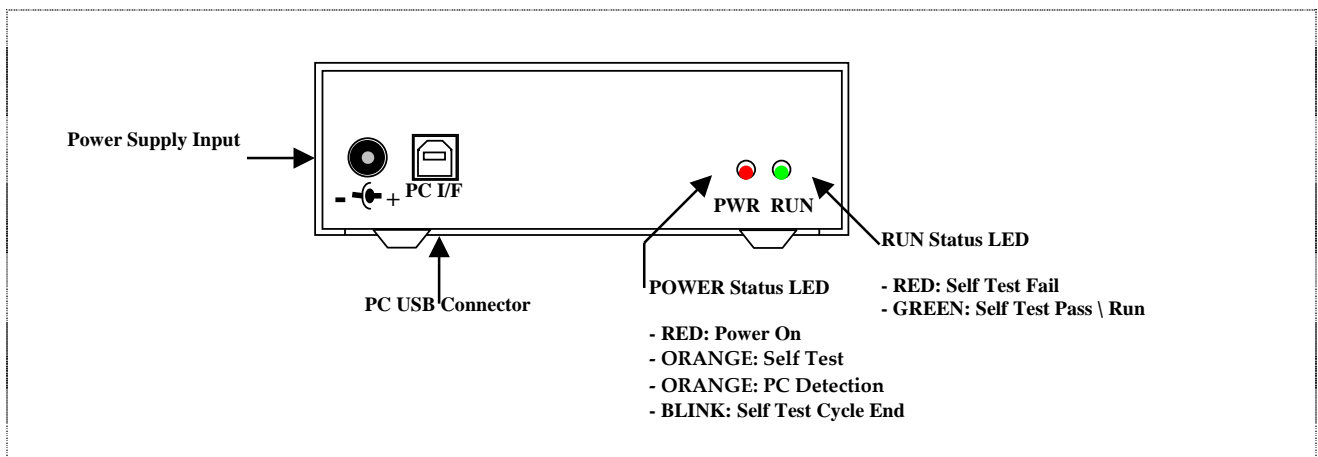


Figure 4 Compact Emulator Front Panel

2.2 Installing HEW debugger software

First install the HEW software from the installation disk, proceed as follows:

- Insert the HEW debugger installation CD.
- Click [CE300H.exe] to install CE300H HEW Debugger
- Or
- Choose [Run...] from the Program Manager File menu.
Type [CE300H] and click [OK]

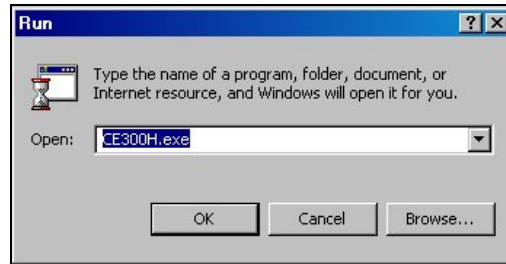


Figure 5 Run Dialogue Box

This will runs the HEW debugger installer, and the following Welcome! Screen will be displayed:

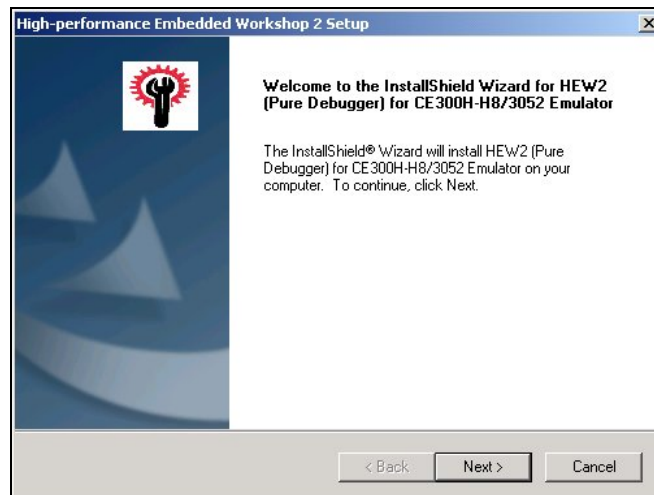


Figure 6 HEW DEBUGGER Installer Welcome! Screen

- Click *Next* to proceed with the installation and follow the on screen instruction.
- Click *Next* to install into the default directory [C:\HEW3], or specify an alternative directory by clicking on Browse button.

Note:

1. User may install this HEW debugger in the same directory as the previously setup HEW toolchain (Make sure both are in the same version).
2. User may install the debugger into another directory, and register this component into the other HEW tool administration menu.
3. Do not install a HEW toolchain over (in the same directory) the HEW debugger
4. A new Toolchain can be installed if it is installed to another directory (different from the toolchain directory) and register either components to the respective HEW tool administration menu

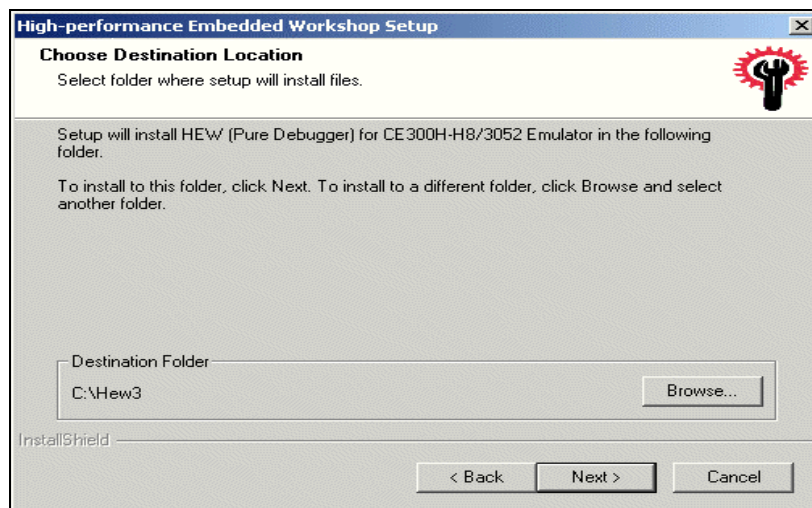


Figure 7 Select Destination Directory Screen

- Select the required components to be installed. And click [Next] to proceed.

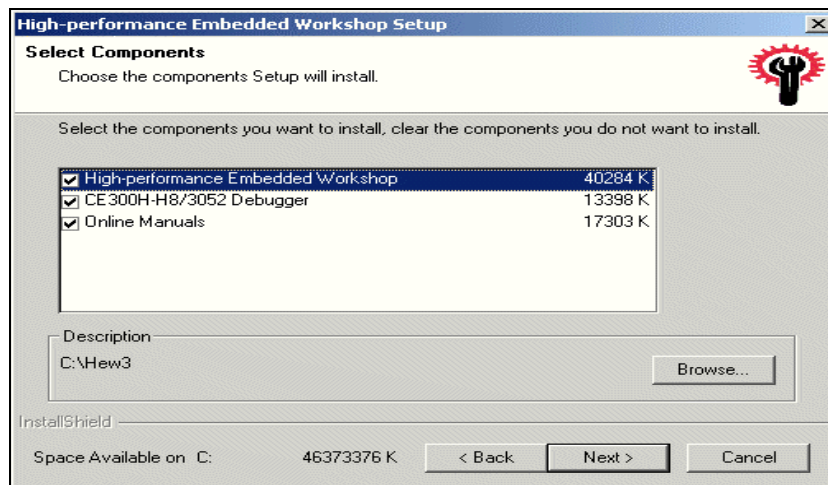


Figure 8 Select Components

- Click [Next] to install the selected components into the default directory.

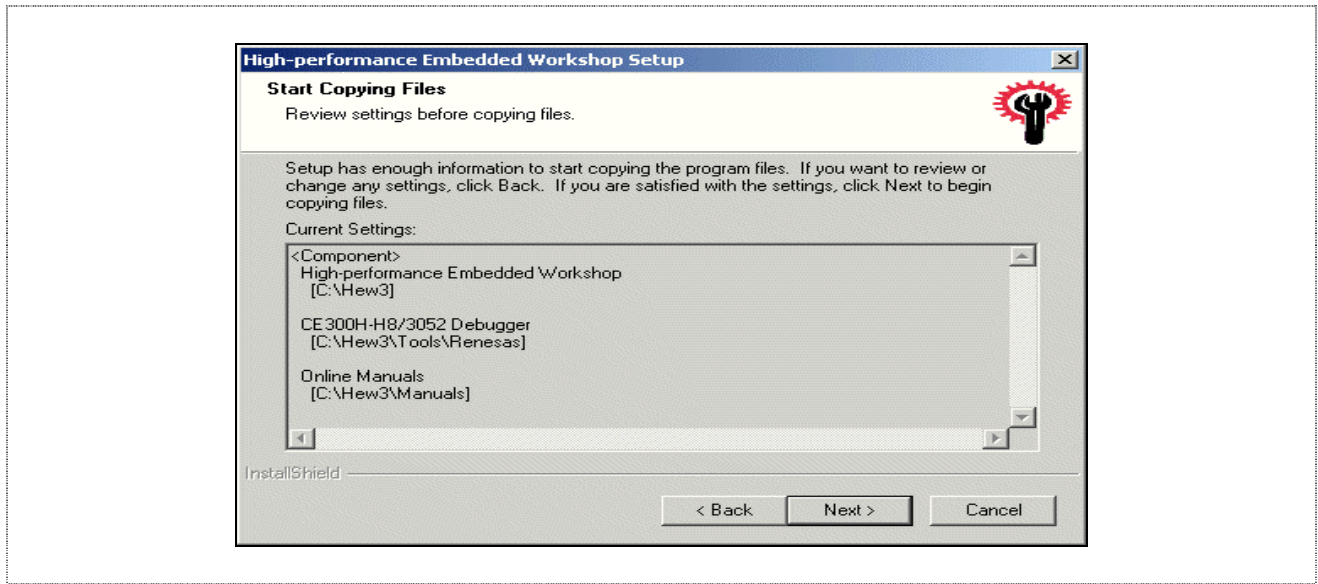


Figure 9 Setup directory

- Click [Finish] to complete the installer process.

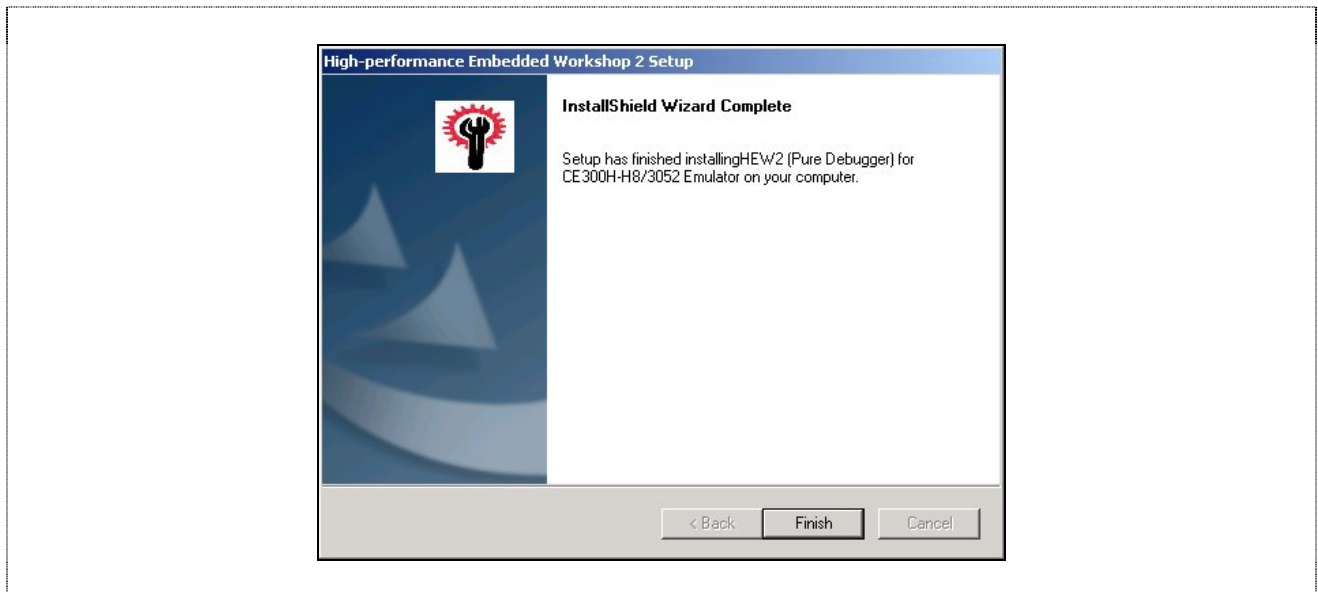


Figure 10 Installation Completion

2.3 Installation Details

The installer creates the following:

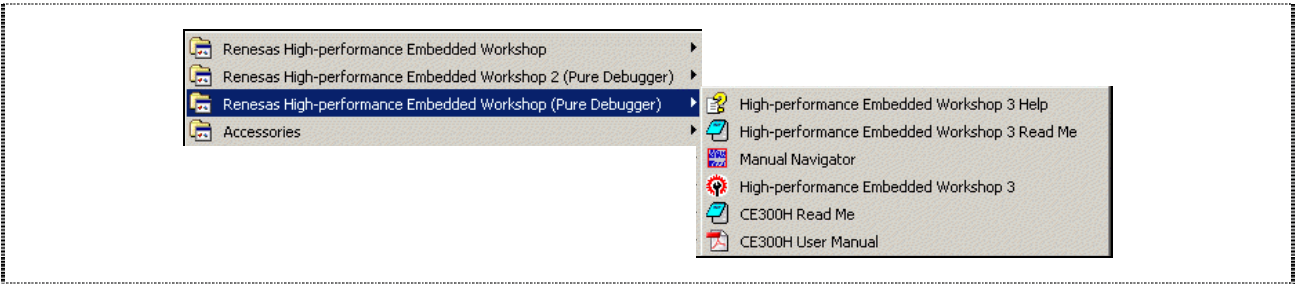


Figure 11 HEW DEBUGGER Icons

The installer has created the HEW interface and the debugger component.

A directory is generated in the installed HEW directory

- [c:\Hew3\Tools\Renesas\DebugComp\Platform\Emulator\CE3052]

2.4 Power Up the Emulator

A power supply is included in the CE300H-H8/3052 emulator package. It can accommodate 110-240V 50-60Hz AC supply. The unit is capable of a regulated 5V, 2.8A output.

The following diagram shows the polarity of the power-supply plug:

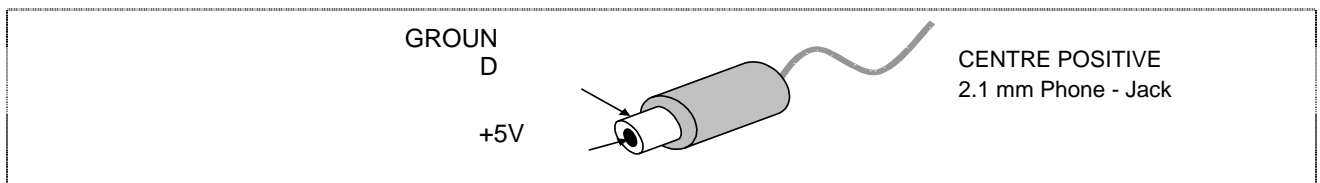


Figure 12 Power-supply Plug

Connect the plug to the power input of the emulator. The POWER LED will lights up (red colour).

2.5 Checking the System (Standalone mode)

If the emulator is powered up and USB link to PC is not established within 3 minutes, it will enter into self test mode.

The POWER LED changes its colour from RED to ORANGE when it has entered the self test mode. This test takes about 3 minutes. At the end of the test, both the POWER & RUN LED start to blink. If any of the tests failed, the adjacent RUN LED lights up in RED. For more details about this test, please refer to section 8.

The self test will continue to run until the emulator is powered down. The self test will confirm the working condition of the emulator.

NOTE: The same tests can be activated by HEW diagnostic window. Refer to section 9.

2.5.1 LED indication

Indication	POWER LED	RUN LED
Power On	Red	-
Enumeration (Connected)	Orange	-
Enumeration (Run)	Orange	Green
Self Test (Pass)	Orange	-
Self Test (Fail)	Orange	Red
Self Test End (Pass)	Blinking Orange	Blinking Green
Self Test End (Fail)	Blinking Orange	Blinking Red

POWER Status LED

RUN Status LED

- RED: Power On
- ORANGE: Self Test
- ORANGE: PC Detection

- RED: Self Test Failed

Figure 13 LED descriptions

2.6 Installing the USB Driver

- Power up the emulator & check the emulator as in step 2.4 & 2.5.
- Connect the emulator to the PC through the USB cable.
 - This will activate the Windows auto-detect feature, and prompt the user to install the USB driver for the emulator



Figure 14 Found New Device

- Click [Next] to search for a suitable USB driver



Figure 15 Locate Driver Files

- Click [Next] to specify the Driver location

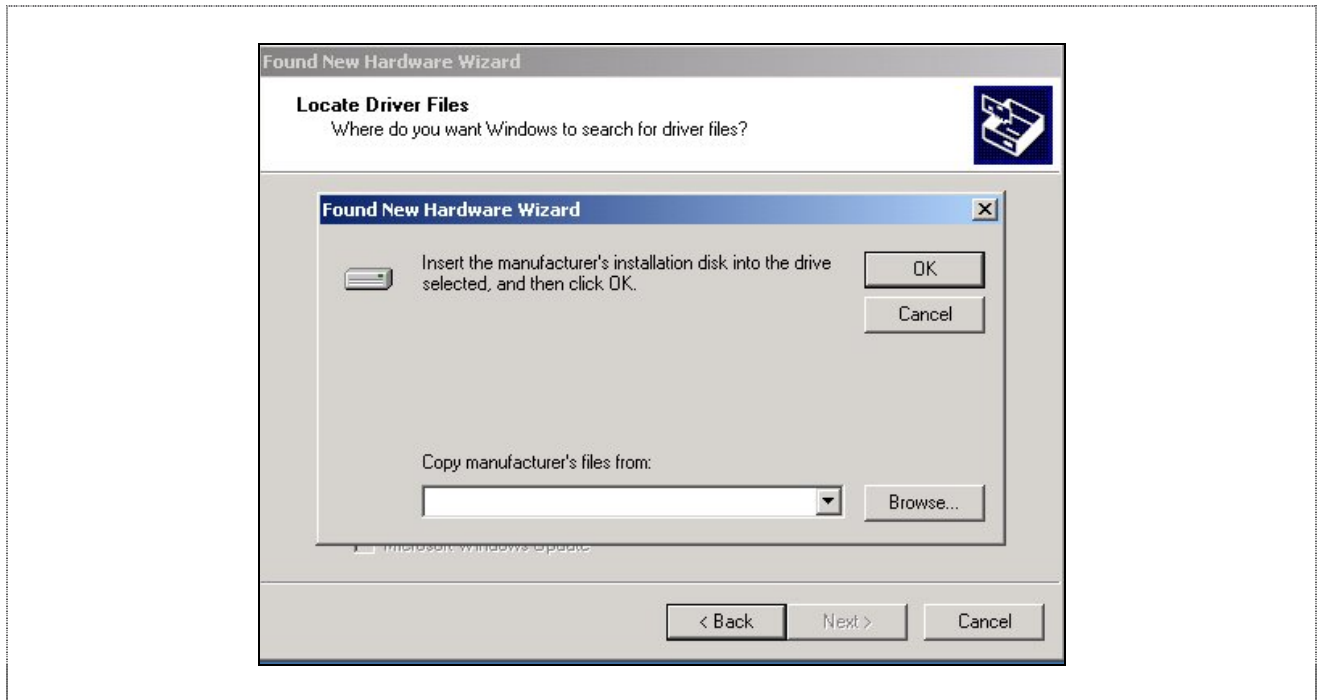


Figure 16 Selecting the USB Driver Location

- Click on [Browse...] to
[C:\Hew3\Tools\Renesas\DebugComp\Platform\Emulator\CE3052\Driver]

and select either

- Win2K directory (for Microsoft Window 2000 or XP operating system) or

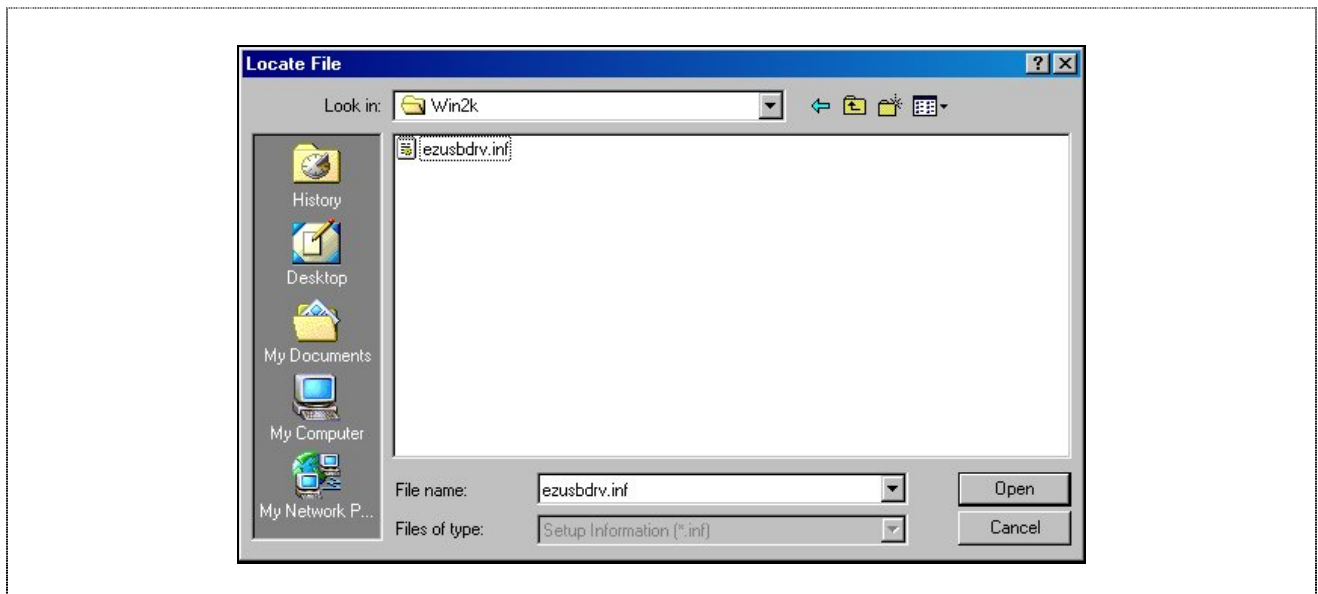


Figure 17 Win2K Driver Location

- Win9x directory (for Microsoft Window 98 2nd edition or ME operating system)

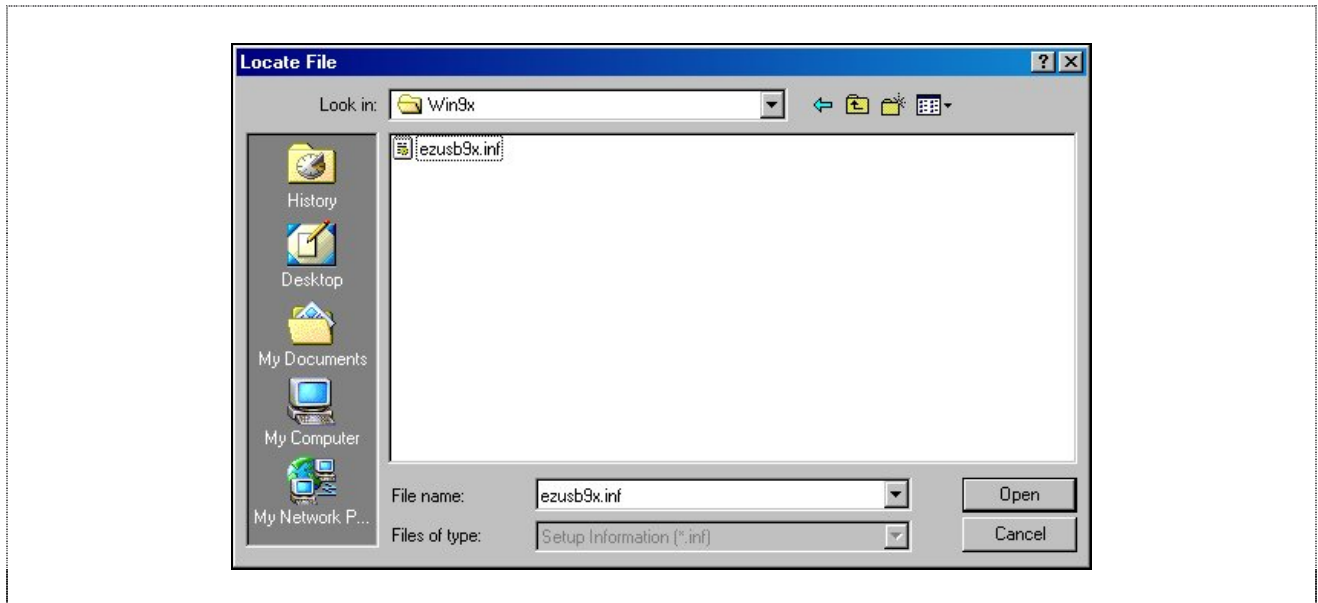


Figure 18 Win 9x Driver Location

- Select the file available

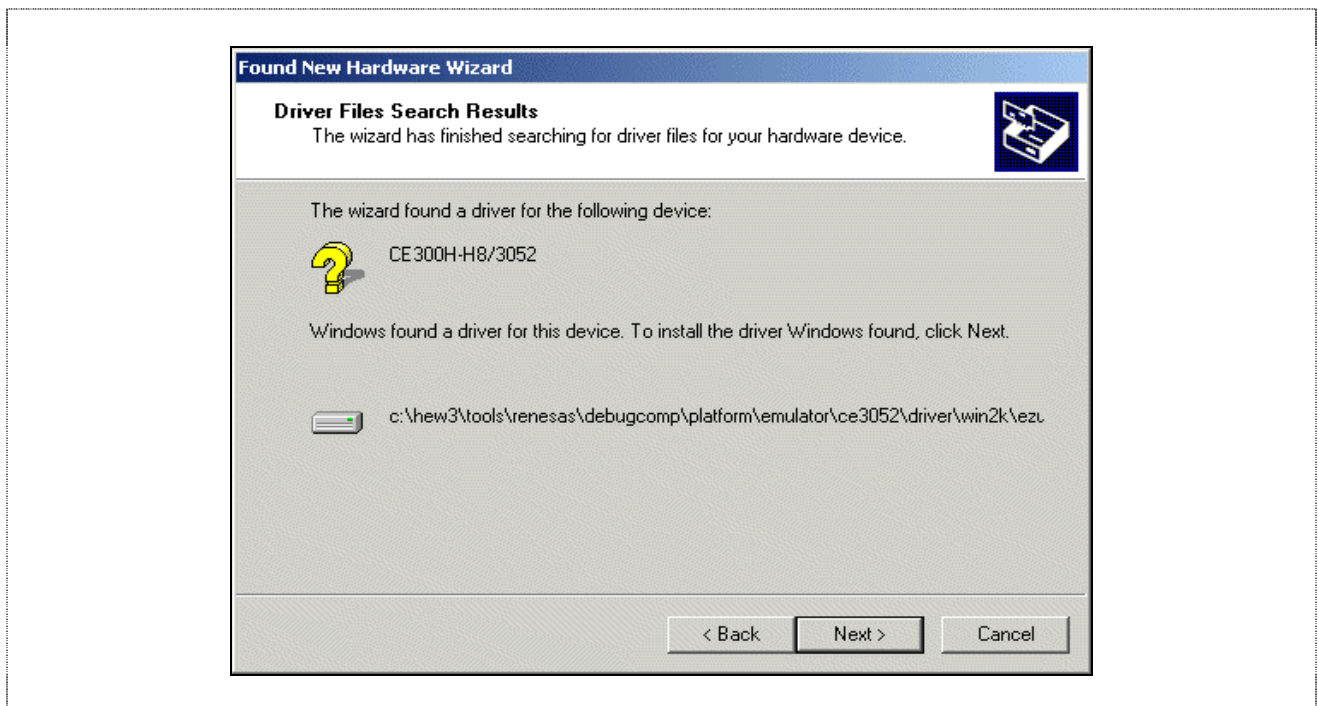


Figure 19 Selected Driver File Window

- Click on [Next] to install the driver



Figure 20 Compact Emulator USB Driver Installed

- Click on [Finish] to complete the installation

2.7 PC USB Linkage

Steps 2.5 (Standalone Test) tests the emulator functionality, whereas the current step (2.7) confirmed the PC & emulator linkage.

Step 2.7 is a additional step to confirm the successful installation of USB driver in Step 2.6.

Power up the emulator and link the emulator to the PC window via the USB cable. Within a second, the window will recognize (In USB term - Enumeration) the emulator, and load the correct driver to communicate with the emulator.

Observation:

- i. The emulator red Power LED will become orange in colour.
- ii. Window will detect the device "Compact Emulator(CE300H) USB Interface"

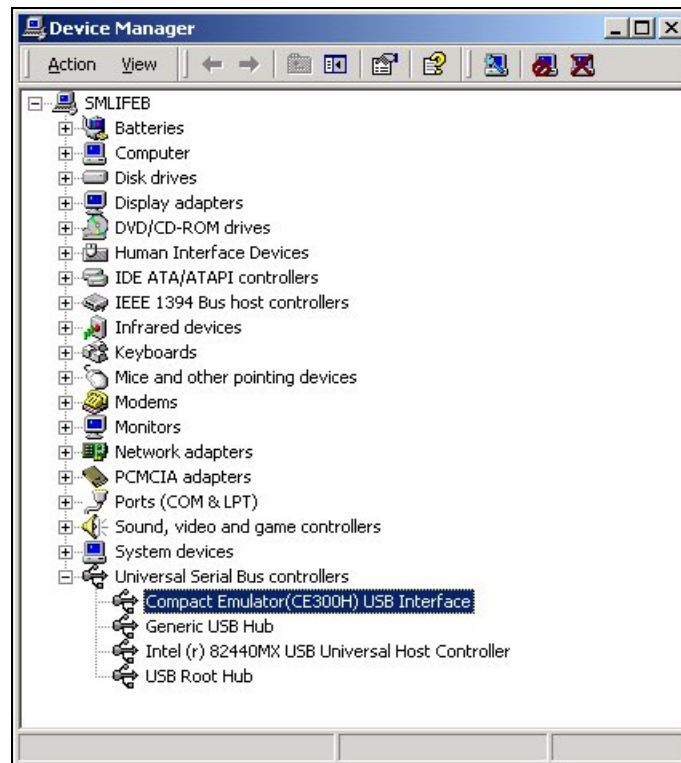


Figure 21 Device Manager

The above step will confirm the correct linkage of the emulator and the PC window.

2.8 HEW Tools Administration

HEW has numerous components. In this session, HEW CE300H debugger component is installed onto the basic HEW interface, which may already reside in users' PC. To verify that every components are installed correctly to the same HEW system, user can activate the HEW tools administration window for verification.

- Activate HEW
- Click on the [Administration...] button when the [Welcome] window pops up, after HEW is activated, or, click on the pull down button [Tools\Administration...]
- Click on the (+) to expand the tree to view the detail components

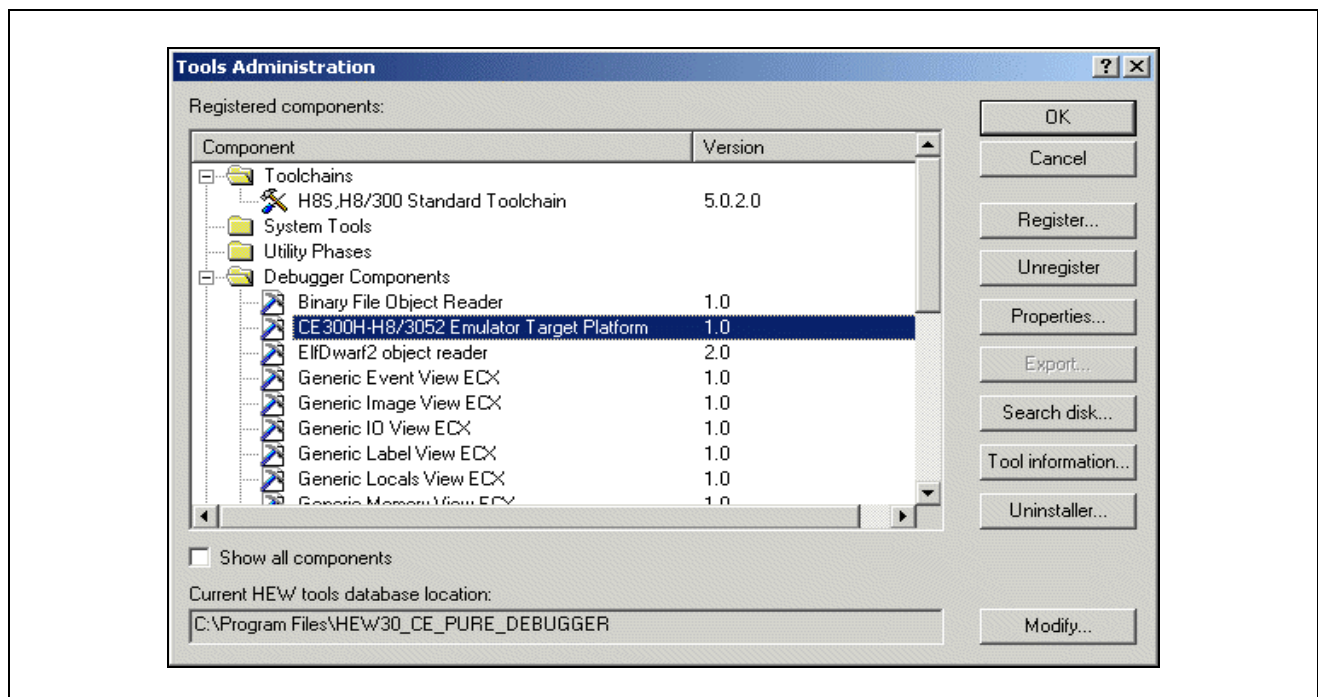


Figure 22 High-performance Embedded Workshop Window

In this example , the following components are installed

- CE300H-H8/3052 Emulator Target System
- H8S, H8/300 Standard Toolchain
- ..

This is just an example showing the co-existence of a toolchain and a debugger.

The CE300H Debugger installer will not install the H8 Toolchain.

If user has installed the component (e.g. CE300H Debugger) in a separated directory from the current activated HEW directory, user may "register " the component to the current setup, so as to use all components together. The component information is stored in the HEW registration file (HRF). On the other hand, user may also remove (uninstaller) the component from the current HEW.

2.9 Setup Completion

This will complete the initial setup of the emulator;

- HEW debugger installation
- USB Driver installation
- Emulator working condition

Section 3. Emulation System Setup for Development

In this section, the focus is to highlight the basic steps for any initial setup for a project. On subsequent HEW activation, user will just be required to select the desired workspace/session, and the setup will be done automatically.

To activate the emulation system, user has to:

- Ensure that the CE300H-H8/3052 emulator is powered up i.e., check that the POWER LED is illuminated and the colour is RED.
- Ensure that the emulator is enumerated i.e the USB cable is linked between the emulator and PC, and the PC has recognised the attached USB device.
- Click on the *High-performance Embedded Workshop 3* icon.

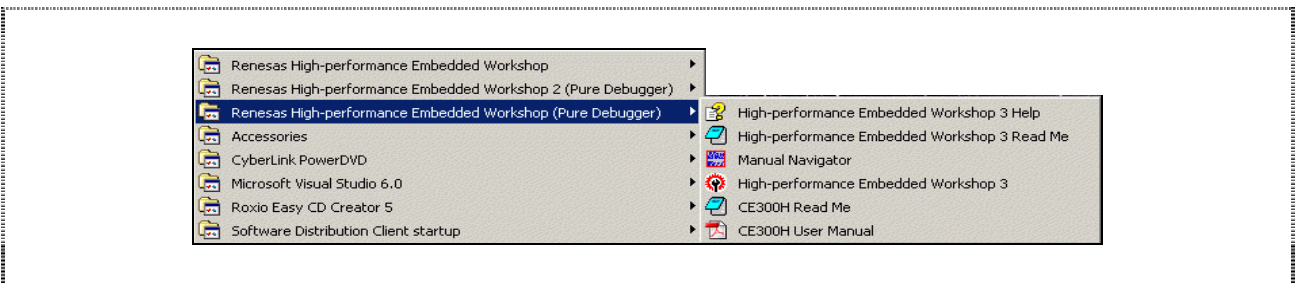


Figure 23 Execute HEW DEBUGGER from Start Menu

3.1 Creating the workspace

Unlike the past GUI, whereby the activation of the GUI will link up the GUI with the emulator instantly, user need to initialize the HEW debugger before the linkage can be achieved. This step is to create a workspace, to inform the HEW environment, what type of tool is to be used. This will enable user to have the same setup(workspace) at the following activation of the tool.

Since it is possible that user do not have any installed toolchain, there will be two different possibilities when the workspace is being set up.

NOTE: Toolchain is a HEW component that enable user to create, compile and link a project.

3.1.1 Without Toolchain

If no toolchain is installed, the linkage between the emulator and the HEW debugger is still possible.

- Click on [Create a new project workspace]

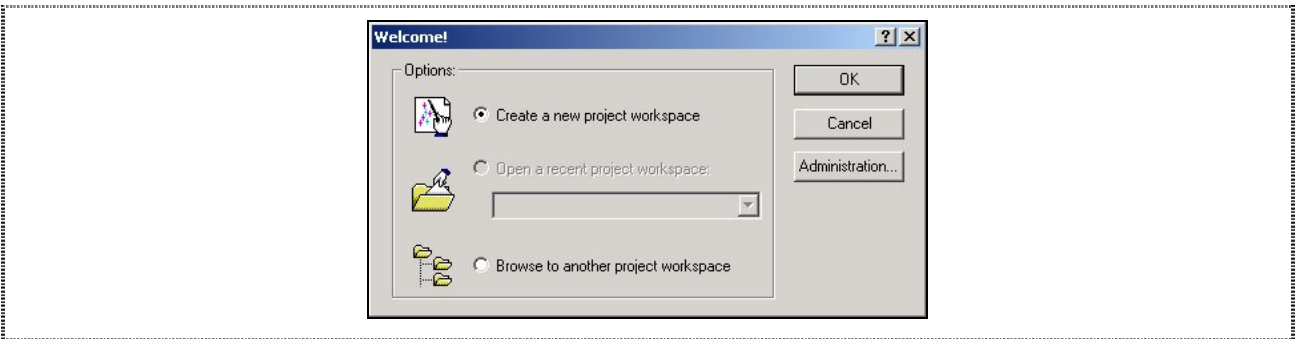


Figure 24 Select Platform Dialogue Box

- Select a directory and key the workspace name as required.

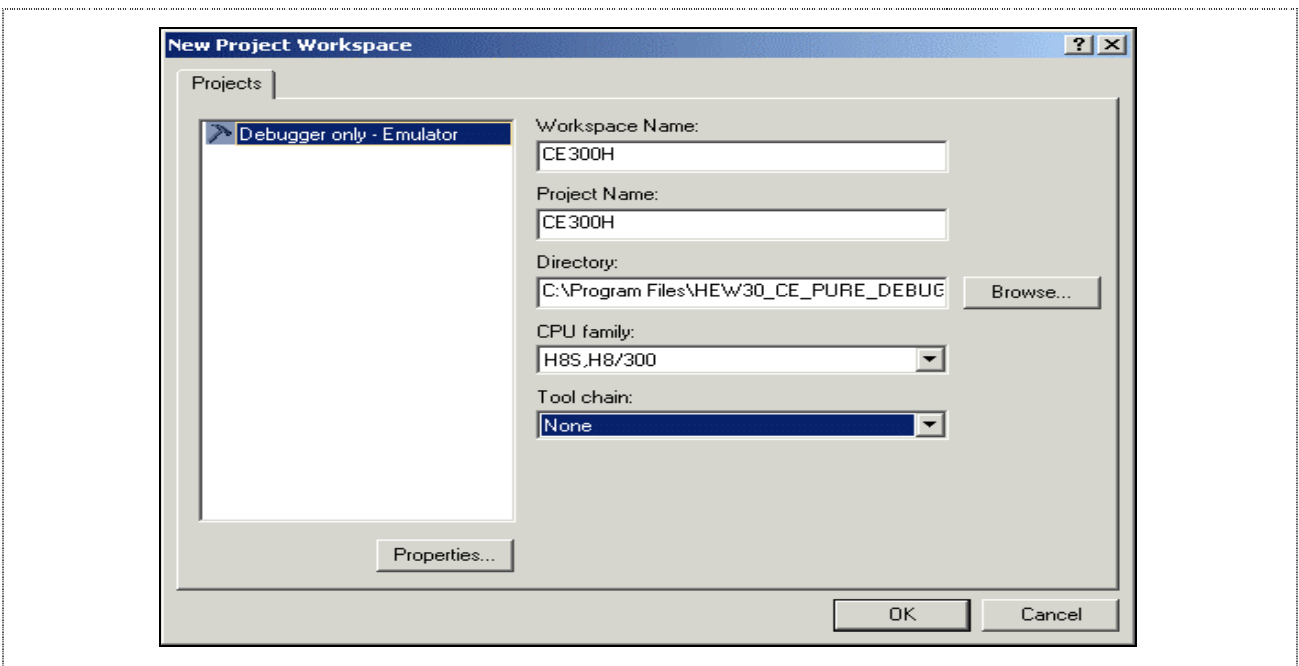


Figure 25 HEW DEBUGGER (without toolchain) Start-Up Messages

- Select [CE300H-H8/3052 Emulator] as the target

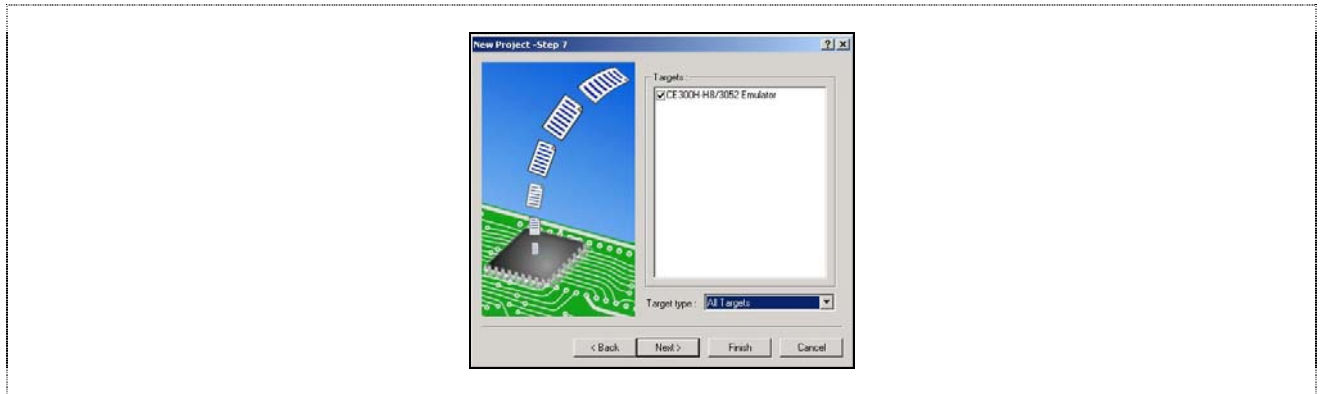


Figure 26 Select Target

- Click [Finish]

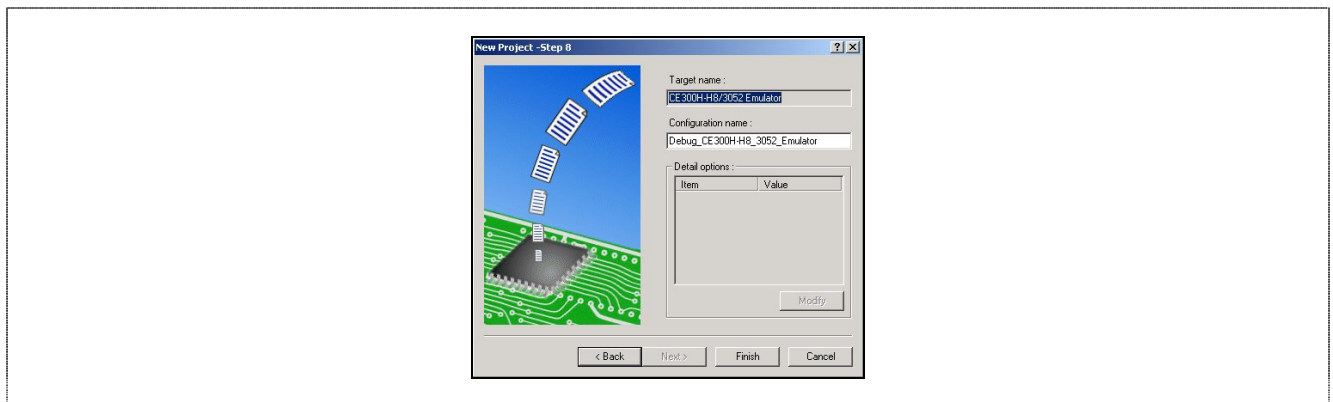


Figure 27 Debugger configuration

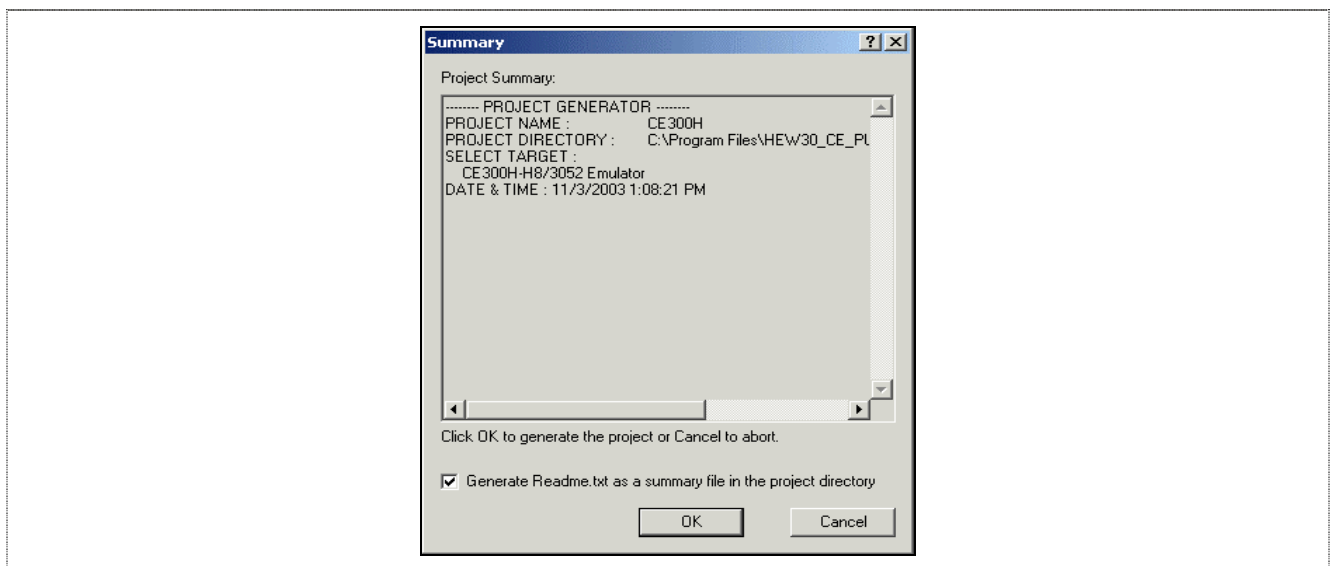


Figure 28 Debugger setting summary

This will invoke the configure platform window as explain in 3.2

3.1.2 With Toolchain (debug setting)

The steps of generating a workspace is detailed in the HEW user manual. User may open the workspace directly, or create a new workspace. This section will focus on the steps required to instruct HEW to use CE300H as the target platform. This is done in [Options\Debug Setting].

- Click on [Create a new project workspace]

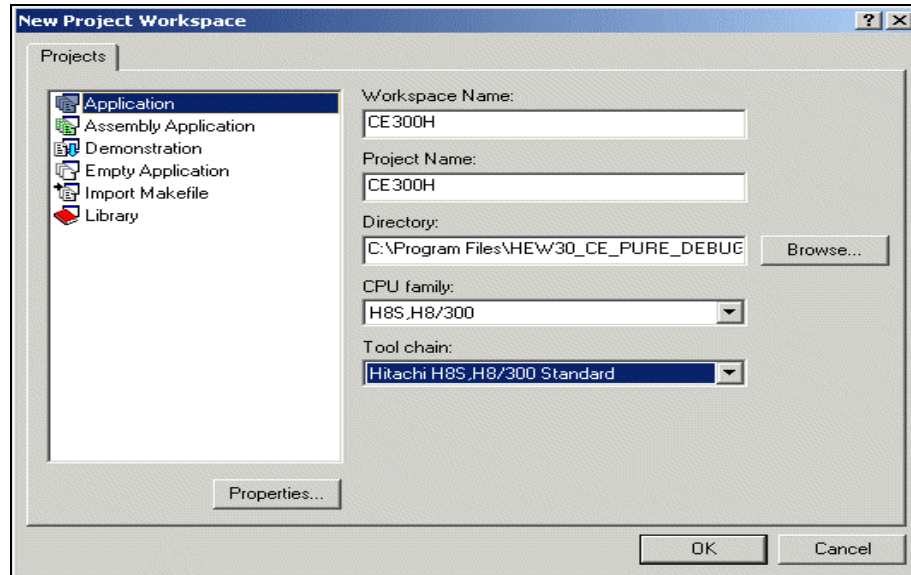


Figure 29 Debugger setting summary

- The subsequent steps are not illustrated until step 7
- Select [CE300H-H8/3052 Emulator] as the target.

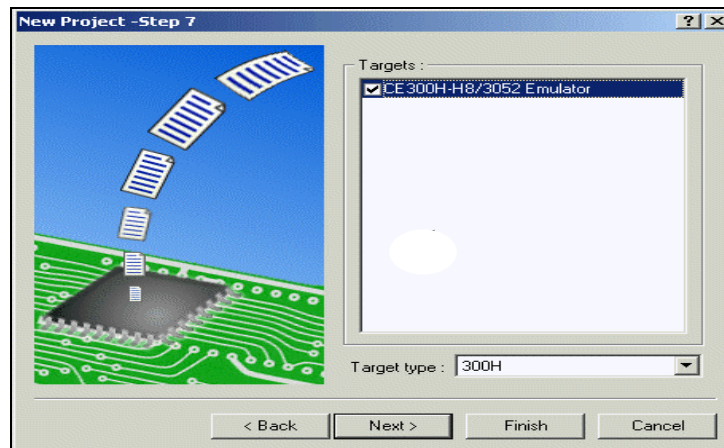


Figure 30 Step 7 Selection of Target

- Complete the workspace setup and click on the [Finish] button

Unlike the HEW (without Toolchain), this will not activate the Configure Platform window. This will lead the user to the HEW interface, whereby user can edit the generated code.

Once the project is compiled correctly, user will be able to debug the code in the emulator.

- Goto [Options\Debug Setting...]
 - Select Target as [CE300H-H8/3052 Emulator]
 - Select Default Debug Format as [Elf/Dwarf2]
 - Add Download Modules

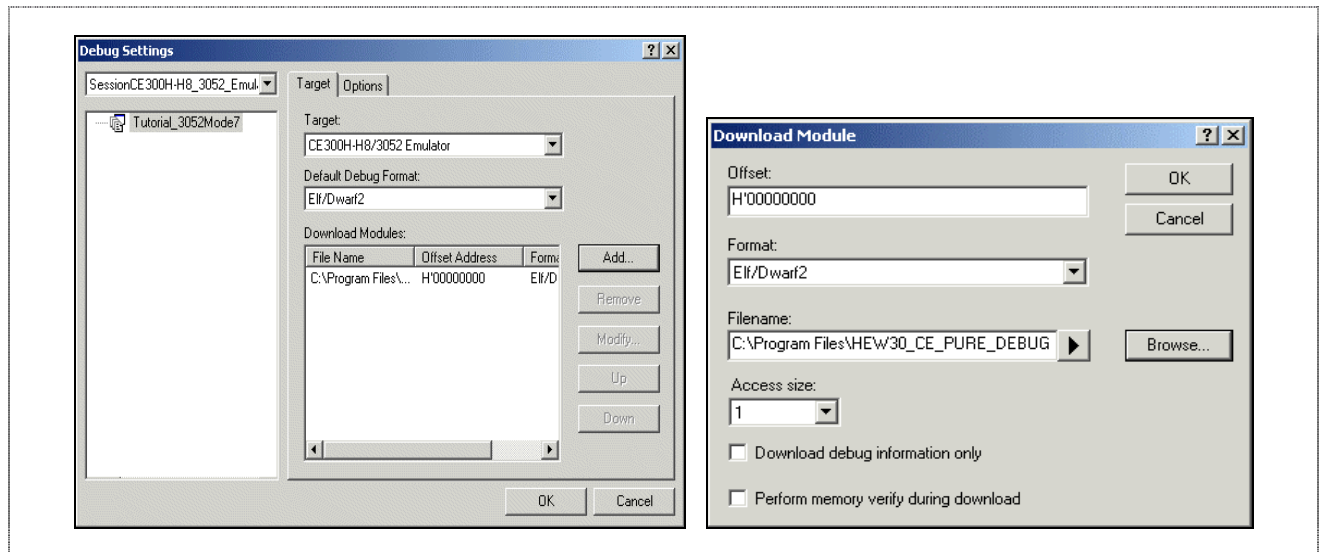


Figure 31 Debug Setting window [Absolute path]

NOTE:

1. User can select a number of files be be download to the emulator. The (abs) file can be from the debug or release sub directory.
2. In the example, an absolute path is defined for the filename. In a integrated enviroment, a relative path name [\$(CONFIGDIR)\\$(PROJECTNAME).abs]is recommended. Please refer to section 4 for more detail explanation related to debug setting.

- To choose the relative directory: Click on the 'Place Holder', , and choose
 - Configuration directory
 - Project name

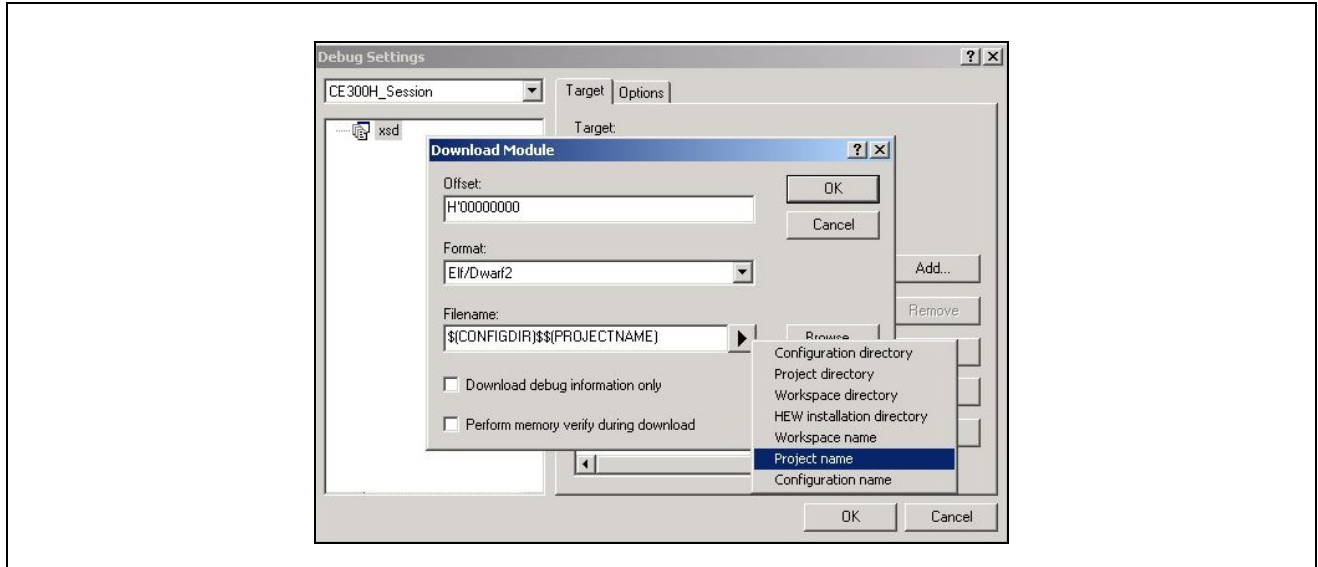


Figure 32 Debug Setting window [Relative path]

This will invoke the configure platform window as explain in 3.2

3.2 Configure the Platform

The configure window will pop up before any emulation can proceed. User has to configure the platform for the desired application. This will ensure a proper control over the targeted application.

The following setting is fixed for each workspace created. [Device, mode, & package]. However the other components of the configure window can be changed in the later part of the emulation. This can be found in [Options/ Emulator/ System ...].

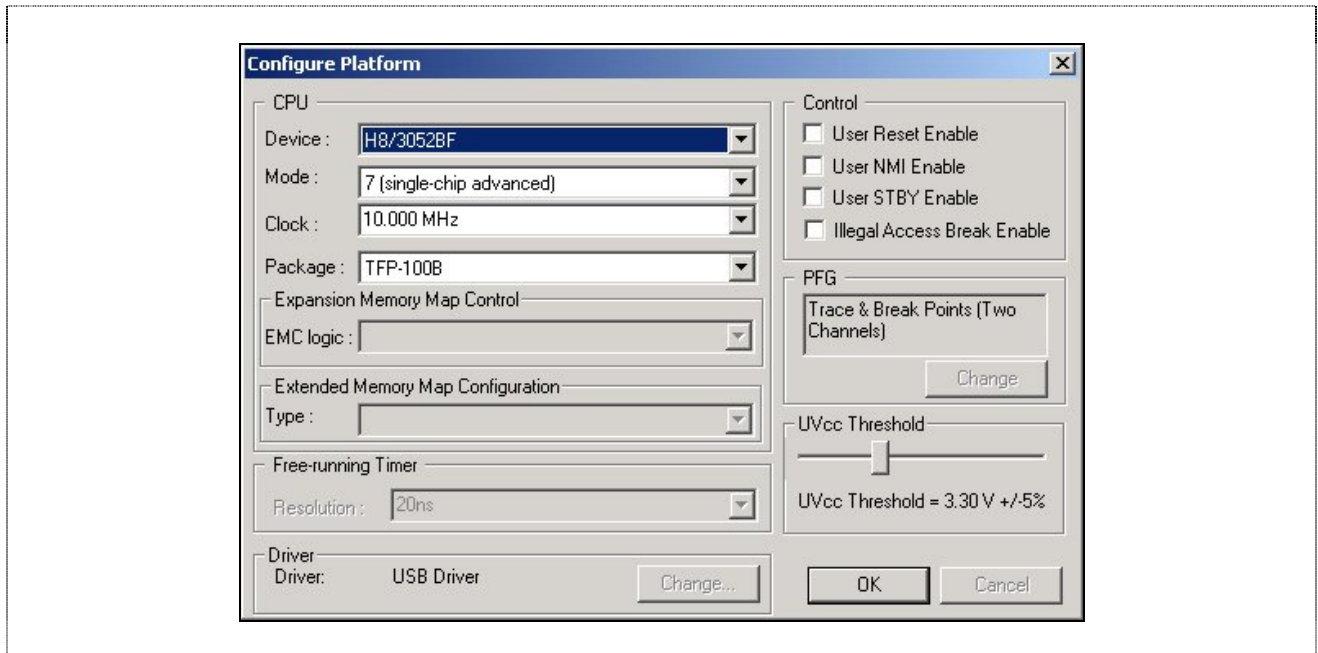


Figure 33 Configuration Window

3.2.1 Device and Package Selection

User has to select the desired device and package. The selection will determine the mapping window setting. The package selection will also determine the type of graphical display in the pinview window.

3.2.2 Operating Mode Selection

Depending on the device selected, the modes available for the device will be selectable. Target Mode will be selectable, if target system (CABLE_IN_N is detected Low) is connected.

NOTE:

For ROMless mode operation, user has to make sure that external memory is connected before any emulation can occur. If user's target system is not ready, user may map the area to the optional SODIMM memory for temporary usage.

3.2.3 Clock Selection

User can choose from two different sources:

- Internal or External Target Clock.

For internal clock, user can key in any frequency from 1MHz to 25MHz, in the step of 100KHz. The emulator will generate the requested clock for the running processor.

For external target clock, user can either input an oscillating clock into the EXTAL pin, or place a crystal at the actual footprint user cable (EXTAL and XTAL pins).

3.2.4 User Signal Masking Control (RESET, NMI & STBY)

These signals can be masked by the emulator when user executes the programs. At startup, the RESET and NMI signals are not masked, whereas STBY signal is masked

NOTE:

If STBY signal is asserted during user run mode, the emulator will enter standby mode, and the emulator control registers are initialised.

3.2.5 Illegal Access Break Control

Enabling the 'Illegal Access' Break will cause user program to stop execution, when 'write protected' area is being written or 'guarded area' is being accessed.

User may disable the control, so as to use the internal ROM area as read/write area temporarily.

3.2.6 Downloading of Emulation Function (Programmable Function Generator)

At power up, a default Function is set in the PFG. If another function is required, User will have to download the selected function to the Programmable Function Generator before it can be used. This has to be done again if the emulator is switched off.

Please refer to section 4 for more details.

3.3 Memory Mapping

After the selection of Device, Package and Mode, the default mapping will be generated. This can be viewed under the [Option /System/ Memory resources].

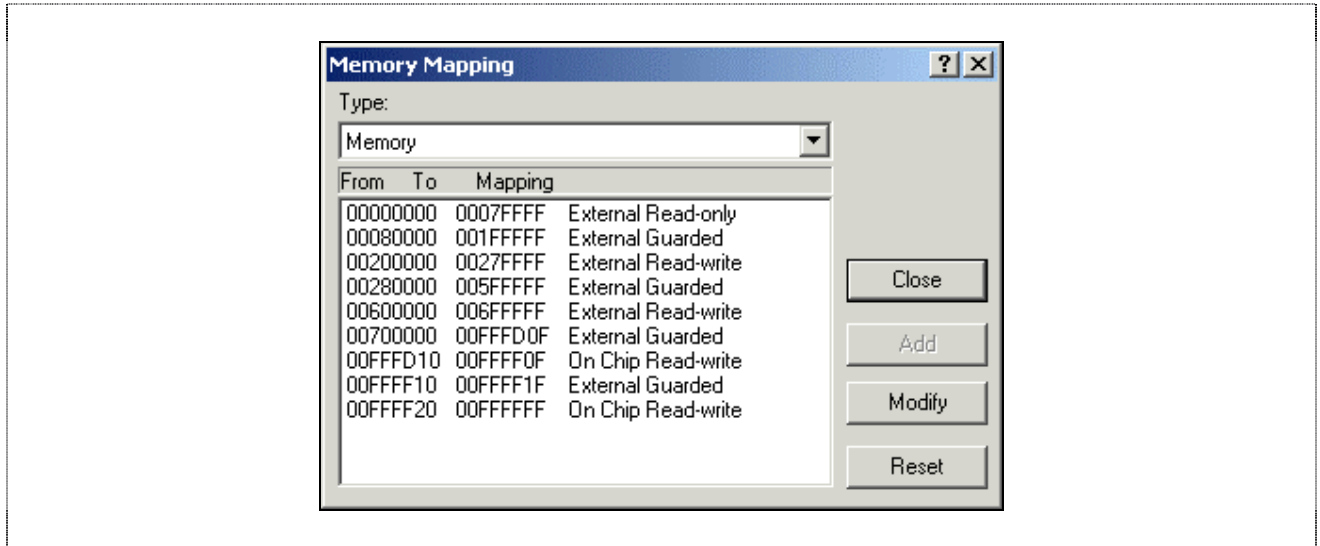


Figure 34 Memory Mapping Window

Usually, user does not require to change this setting. However, it may be changed for the following reasons:

- Addition of target system with memory
- Addition of optional SODIMM memory

The nine available attributes are:

- On Chip
 - Read Write
 - Read Only
 - Guarded
- Emulator
 - Read Write
 - Read Only
 - Guarded
- External
 - Read Write
 - Read Only
 - Guarded

To change the setting, user has to:

- Click on the [Modify] button in the memory mapping window.
- Key in the desired address at [From] and [To].
- Select the attribute [Setting].

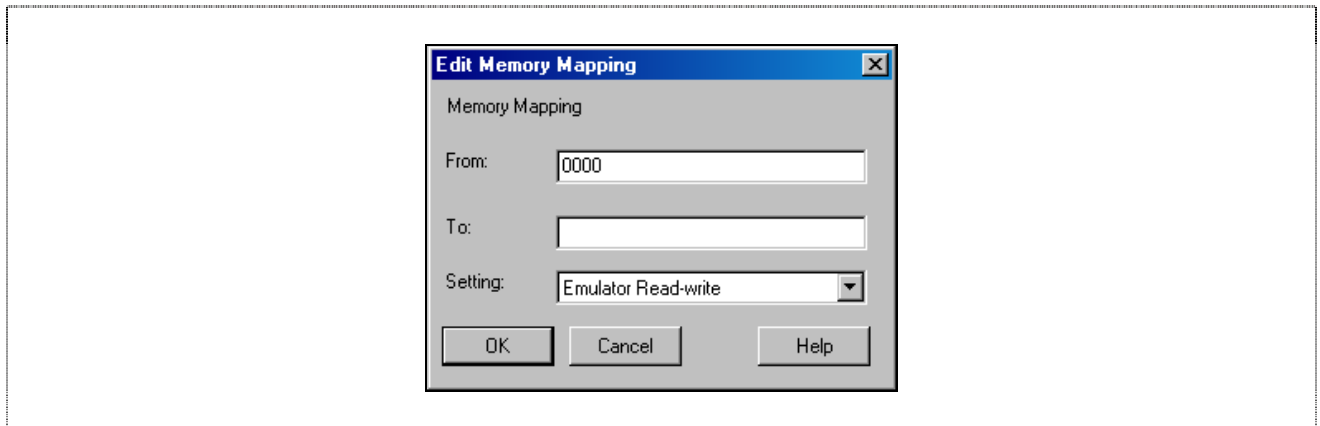


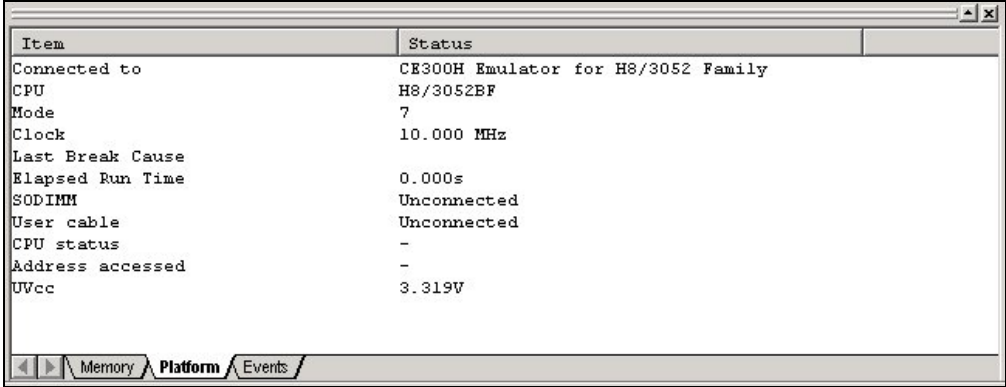
Figure 35 Editing the Memory Mapping

For more details of memory mapping, please refer to section 4.

3.4 Connection to Target System

3.4.1 Target Power Supply

The CE300H-H8/3052 emulator has an automatic voltage follower. Once target is connected (target connector's signal: CABLE_IN_N = Lo, please refer to the Appendix A), the CE300H-H8/3052 emulator will operate at the user supply (2.7V–5V). Otherwise, it will operate at 3.3V. When target is connected, HEW will indicate [User Cable : connected] in the [View/ CPU / Status] window.



Item	Status
Connected to	CE300H Emulator for H8/3052 Family
CPU	H8/3052BF
Mode	7
Clock	10.000 MHz
Last Break Cause	
Elapsed Run Time	0.000s
SODIMM	Unconnected
User cable	Unconnected
CPU status	-
Address accessed	-
UVcc	3.319V

Navigation tabs: Memory, Platform, Events

Figure 36 Target Supply in Status window

The above example show that there is no “user cable” connection & UVcc is at 3.3V.

NOTE:

- Do not connect the user system interface cable (with header board) to the emulator without any user system connection (i.e. without target user supply).
- Power up the user system followed by the emulator.

3.4.2 Types of User Interface Cable

There are two ways to connect the target system to the CE300H i.e.,

- via an actual footprint user cable (purchase separately), or
- direct connection using the KEL connectors (supplied in the package)

For the actual footprint user cable, user is advised to refer to the Microcomputer Hardware Manual for the footprint information.

For the direct connection method, the connector information such as pin definitions, layout, dimensions and part number are detailed in the Appendices A, B and C.

For Direct Connection,

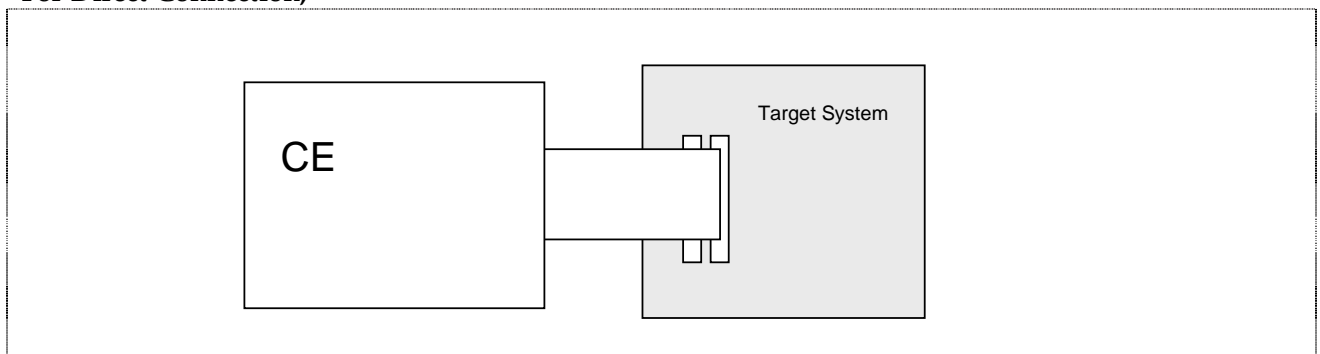


Figure 37 User Interface Cable – Direct Connection

NOTE:

- User has to connect the signal CABLE_IN_N to ground.
- User has to take care of XTAL & EXTAL. Direct connection do not provide roscillation circuitry for crystal.

For Actual Footprint.

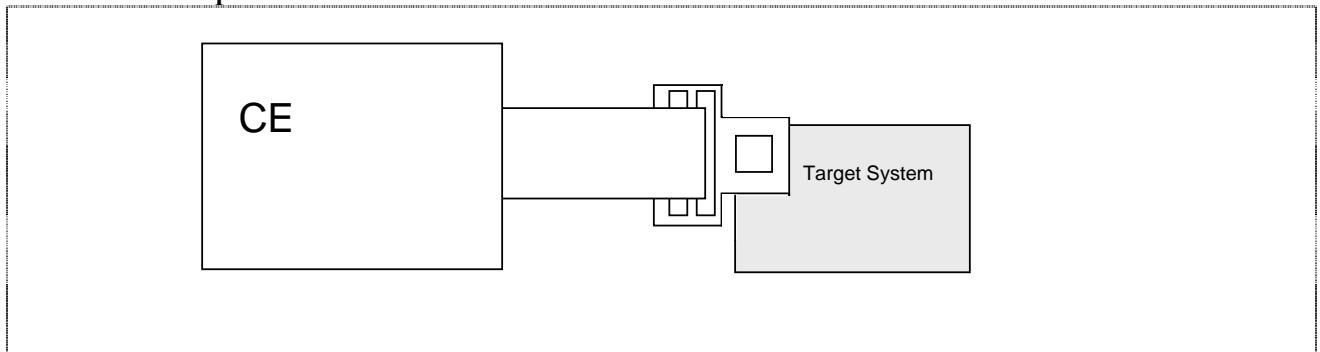


Figure 38 User Interface Cable – Actual Footprint

Section 4. Performing Emulation

4.1 Overview

4.1.1 Emulation

The CE300H-H8/3052 emulator operates in two modes: Break and User modes.

To execute the user program, user can either *Single-Step*, *Run at current program counter* or *Reset Go*. This will cause it to operate in the *User mode*. To terminate the User Run state, a break condition has to be asserted to bring the emulator to the *Break mode*. This can either be a preset condition(eg. PC Break, Event Break) or a force break condition (Hit ESC key).

During **Break mode**, user can manipulate their target system and memory by accessing the HEW DEBUGGER I/O, Memory, Configuration ... window.

During **Run mode**, information such as accessed address, CPU states (e.g. instruction fetch cycle, sleep mode...) and run time can be observed. User can also view and edit the memory contents in the internal ROM/RAM, SODIMM, or external memory area. This process is referred as Parallel On the Fly (POTF).

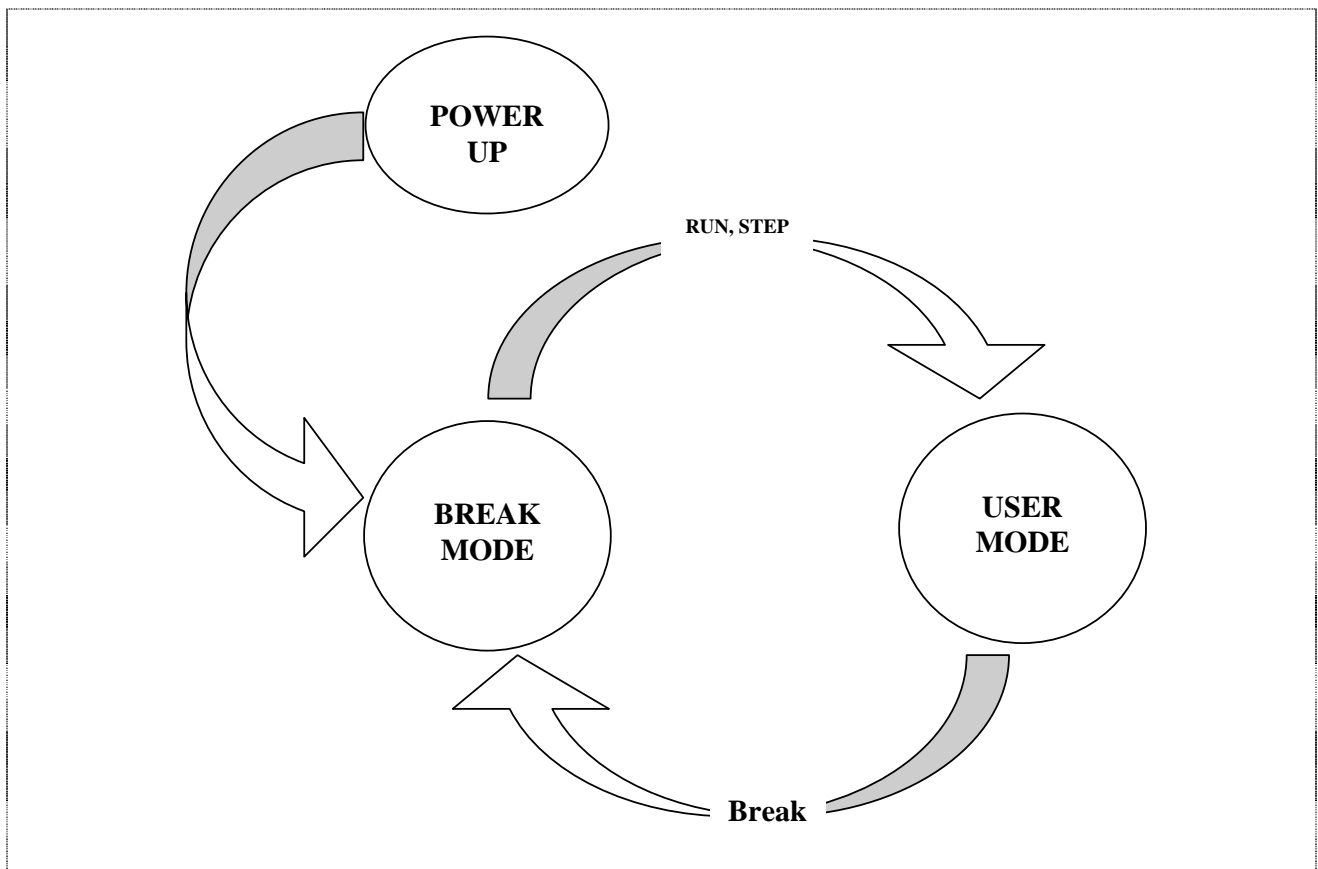


Figure 39 High-performance Embedded Workshop Window

4.1.2 High-performance Embedded Workshop

The following figure is a snap shot of the HEW DEBUGGER desktop window.

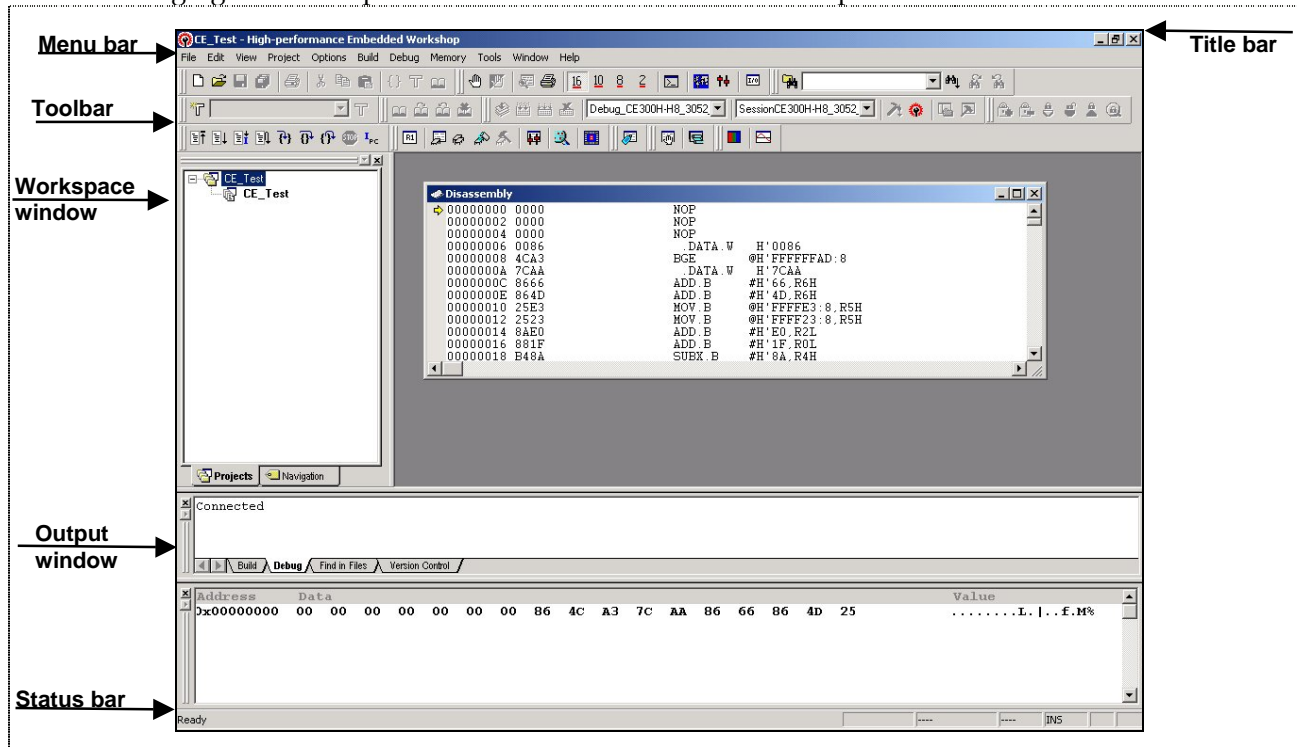


Figure 40 High-performance Embedded Workshop Window

The key features of HEW DEBUGGER are described in the following sections:

- Title Bar** : Displays the name of the currently open workspace, project and file.
- Menus Bar** : Contains nine standard menus (File, Edit, View...) Debug Menu will be activated /appeared when a debugging platform is connected.
- Toolbar** : Provides convenient buttons as shortcuts for the most frequently used menu commands. The tool bar can be docked or floated. It can be created, modified and removed.
- Status Bar** : Displays the status of the CE300H Emulator. For example, progress information about downloads.
- Workspace** : Display the detail of current workspace, and provide a quick & easy mean of navigation.
- Output Window** : The output window has four tabs. The Debug tab shows the output from the debugger process

The Emulation Function can be described and classified into 4 main topics:

The major topics are highlighted.

	Menu	General Description	Sub Menu	Usage
1	Option	Emulation Setting	Emulator	Clock, PFG select, User signal mask
			Memory Resource	Memory mapping, protection
2	View	MCU related information	Disassembly	
			CPU	Register, memory, Status, Pinview, Monitor
			Symbol	Label
			IO	
			Code	Breakpoints, Trace
			Graphic	Image, waveform
3	Memory	MCU memeory manipulation	Search	
			Copy	
			Fill	
			Refresh	
4	Debug	Execution of MCU Code	Reset CPU	
			Go/Reset Go /Go to Cursor/Run	
			Step In/ Over/ Out/ .../ Step mode	
			Initialize	

Please refer to the Appendix for the summary of commonly used short-cut and icon for the HEW debugger.

4.2 Compiler Configuration & Debugger Session

In HEW compiler, every setting is stored in a configuration, The default configuration are [Debug] & [Release]. When a HEW debugger is installed, a new **Configuration**, [Debug_CE300H-H8_3052_Emulator] is created.

At the HEW debugger enviroment, a default debugger **Session**, [SessionCE300H-H8_3052_Emulator] is created to store information of

- Target plaform
- Downloadable program
- Mode settings
- Window positioning
- Clock settings
- Registers value settings
- PFG function loading



Figure 41 High-performance Embedded Workshop Window

Session is not directly related to a configuration. This means that multiple sessions can share the same download module and avoid unnecessary program rebuilds.

Users can create new configuration & session under the [Options\Build Configuration...] and [Options\Debug Session...] pull down menu respectively

Generally , HEW will organise the workspace's configuration & session in the following manner.

<u>Root Directory</u>	<u>Workspace directory Files</u>	<u>Configuration directory Files</u>
(xxx.hws)		
	Debug (DIR)	Configuration Information & Output (abs,lst...)
	Release (DIR)	Configuration Information & Output (abs,lst...)
	Debug_CE300H-H8_3052_Emulator (DIR)	Configuration Information & Output (abs,lst...)
	SessionCE300H-H8_3052_Emulator(hsf)	
	DefaultSession (hsf)	
	C & header files	

Example of usage:

User may use [SessionCE300H-H8_3052_Emulator] to link to CE300H emulator, & [Debug] configuration setting to debug on the project output file (xxx.abs) store in the Debug sub directory. User may switch the configuration to [Release] and debug on the new setting (e.g optimization on...).

On the other hand, user may switch the session to [DefaultSession], which may be set to link to a simulator. At this session, user may switch the configuration from [Release] to [Debug], so as to debug on the generated output (xxx.abs) in the simulator environment.

NOTE:

The path name defined in the [Options\Debug Setting..] must be relative [\$(CONFIGDIR)\\$(PROJECTNAME).abs]. If otherwise, when the session is switched, the download module will not be able to be selected correctly.

4.3 Debug Setting

The Debug Setting in [Options\Debug Setting...] is to set the environment for a session.

In HEW debugger, user has been provided with two sessions

- SessionCE300H-H8_3052_Emulator
- DefaultSession

In each session, user is to set

- Target (CE300H, Simulator...)
- Default Debug Format (Elf\Dwaf2, S-record, IntelHex...)
- Download module (\$(CONFIGDIR)\\$(PROJECTNAME).abs)

User is advised to use [SessionCE300H-H8_3052_Emulator] session to link to the emulator. In each session users can set a list of command chain to be executed at the [option] tab.

- At connecting the emulator
- Immediately before downloading
- Immediately after downloading

4.4 Connecting & Disconnecting with the Emulator

The open (activation) or close (exit) of the HEW and/or workspace will determine the emulator and HEW connectivity.

The alternative method is to use the “session” control

In HEW debugger, user is provided with two sessions

- SessionCE300H-H8_3052_Emulator (linking with emulator)
- DefaultSession (no target)

Thus by switching between the session, the emulator can be connected & disconnected from the HEW.

4.5 Emulator Setting

The emulator setting which consists of the system configuration & memory mapping has to be done before any emulation.

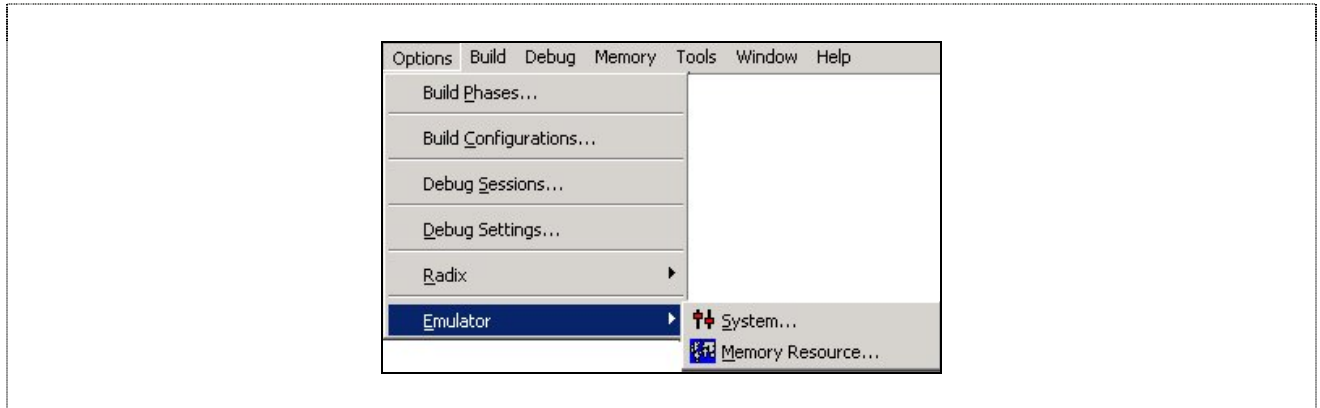


Figure 42 Option - Emulator

4.5.1 Configure Platform

The configure platform enable the user to set their target Device, mode, package at startup. Thereafter user can have the following control.

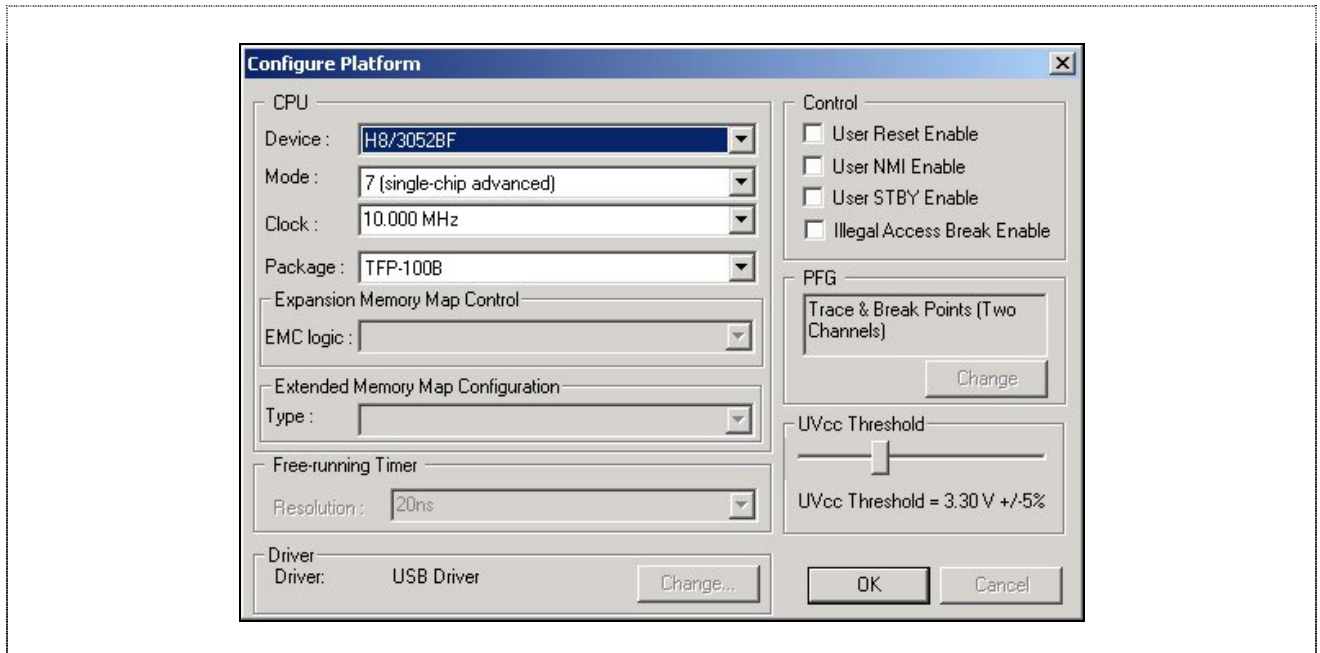


Figure 43 Configure Platform

4.5.1.1 User Reset Enable

When Disabled, user reset will be masked (ignored). [Normally Enable]

4.5.1.2 User NMI Enable

When Disabled, user NMI will be masked (ignored). [Normally Enable]

4.5.1.3 User Standby Enable

This is to mask the standby signal.[Normally Disable]. If this signal, which has the highest priority, is enabled and asserted, the emulator will not be able to overtake the control of the MCU, unless the signal is de-activated.

4.5.1.4 Illegal Access Break Enable

This enables user to ignore the access error and continue emulation. An usage is for temporary use of the internal ROM space for read/write access.

4.5.1.5 PFG selection

This allows user to download the desired emulation function during debugging [Options/Emulator/System...]. At the power-up state, A default function is loaded in the PFG. i.e PFG Break and Trace function. User can select another function to be download. The PFG function can only be download during break mode. To access the function after downloading, user has to open up the break or trace window. The programmed function will remain in the emulator as long as it is not powered down.

Currently, the available functions are

1. Trace & Break Points [Two Channels] - "Integrated 256 Cycles of Bus Trace and Two Event Breaks"
2. Sequence Break [Two Channels] - " Two Sequence Event Breaks"

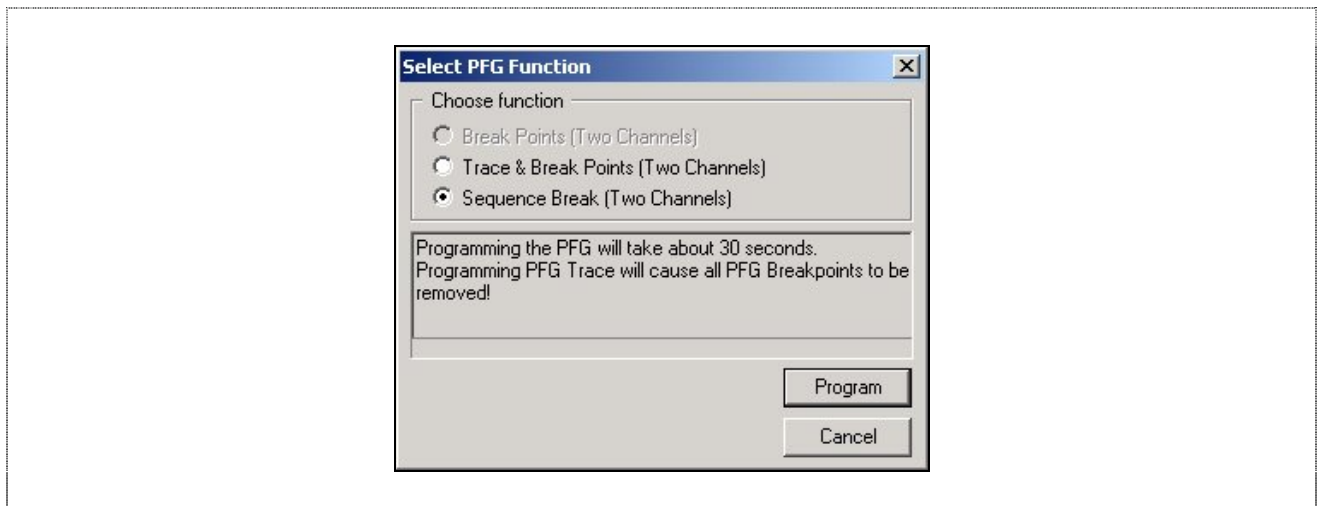


Figure 44 Select PFG function

For the details of each Break capability, refer to the following "Break" discussion.

More functions will be generated at a later date. Please approach the relevant Renesas Technology (Asia Sales Office) for further information.

4.5.1.6 User Vcc Threshold setting

This is set to monitor the voltage level of User VCC (Target Power Supply). If User Vcc fall below the preset threshold level during execution, the emulator will halt the running code. At startup, the preset value is 3.3Volts. The status window will display the read value of User Vcc.

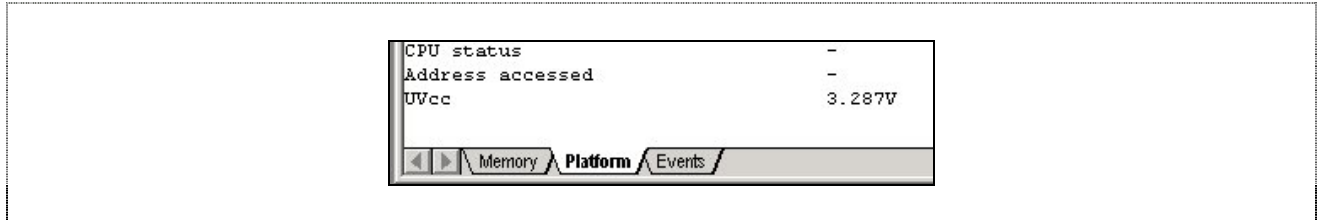


Figure 45 UVCC in status window

4.5.2 Memory Mapping

4.5.2.1 Usage

Once the device and operating mode are selected, the default memory mapping will be set.

The main objective of memory mapping is to instruct/control the emulator to have correct memory access, namely:

- Internal memory (Internal ROM, RAM, IO)
- Emulator memory (SODIMM) <normally used when target external target memory is not available>
- External memory (target memory)

The setting also allow proper control of the emulation. Example

- When program access to guarded or reserved area, the emulator will stop/break the running code.
- When program write to write-protected area (ROM), the emulator will will stop/break the running code.

There are nine available attributes;

- On Chip
 - Read Write
 - Read Only
 - Guarded
- Emulator
 - Read Write
 - Read Only
 - Guarded
- External
 - Read Write
 - Read Only
 - Guarded

4.5.2.2 Changes to memory mapping

User will change the memory setting at the following two conditions;

1. To access external target memory
 - Set [External Guarded] to [External Read Write] or [External Read] Only
2. To access optional SODIMM memory
 - Set [External Guarded] to [Emulator Read Write] or [Emulator Read] Only

NOTE:

1. For the *On Chip* attributes, user cannot change it to other setting. However for the *External* attributes, user can set it to any combination.

2. Emulator memory/Optional memory is available if only if it is installed. For the 2Mbytes SODIMM, It has 4 banks of 512Kbyte of memory selection. If the [External Target memory] fall within the same address range as the emulator [optional memory], the emulator optional memory will be accessed instead.

The availability of the Emulator Optional memory can be viewed under view status/ memeory.

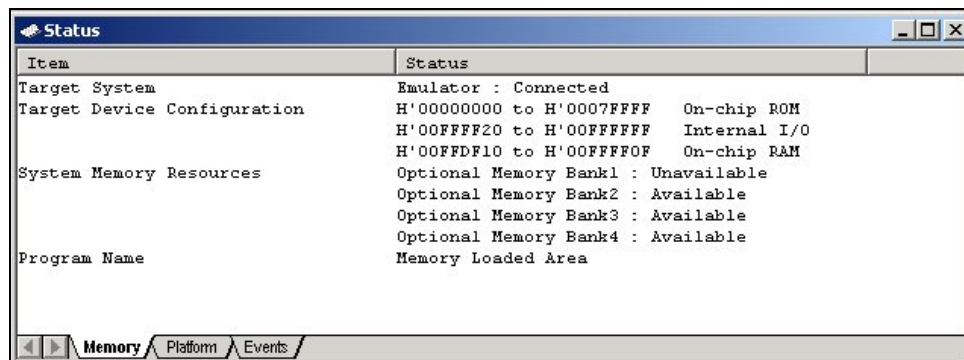


Figure 46 Mapping of Optional Memory Shown in System Status Window

To changed the setting:

- At [Option/Memory resource...] memory mapping window, click on [Modify].

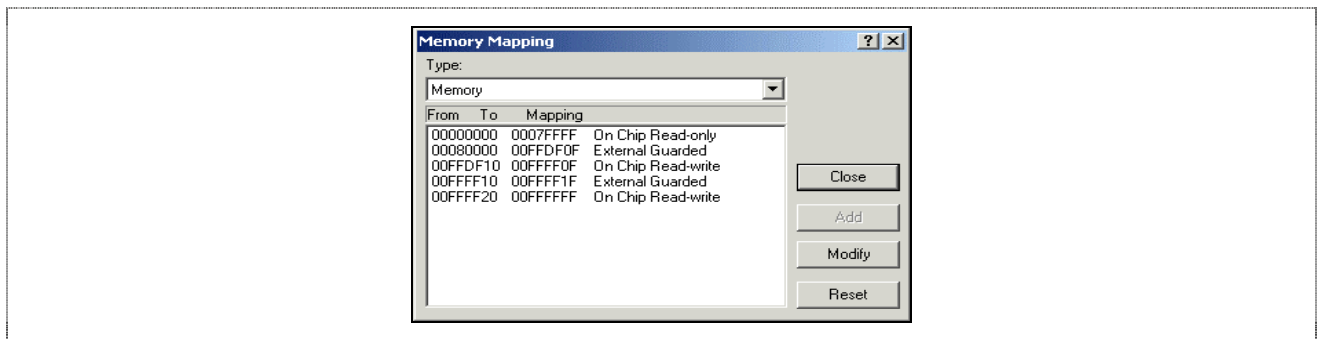


Figure 47 Memory Mapping

- Key in the desired address at [From] and [To].

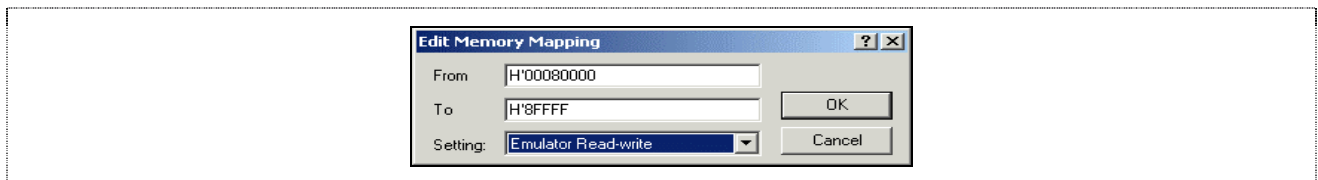


Figure 48 Edit memory Mapping

- Select the attribute at the [Setting]

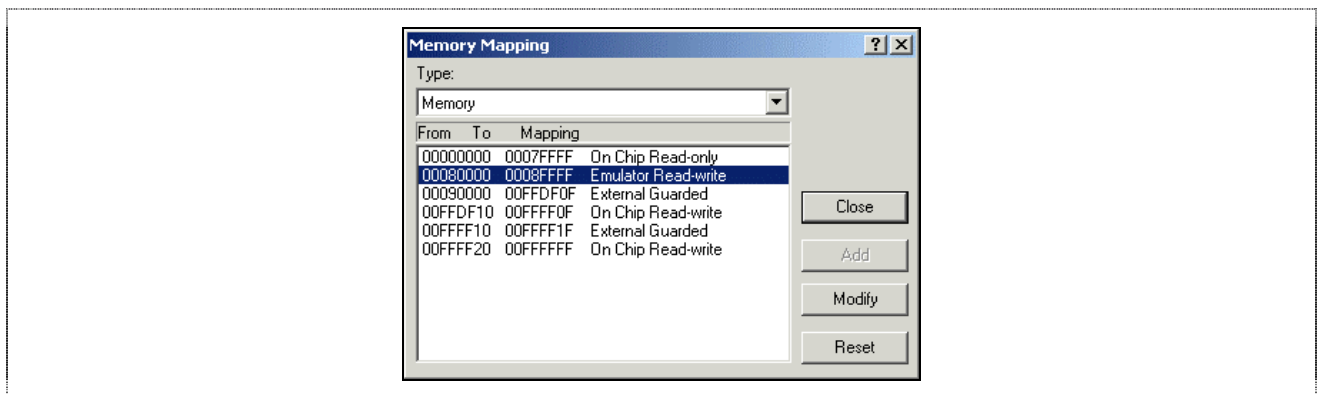


Figure 49 Memory Mapping

NOTE: Users are advised to check for successful mapping in the [Memory Mapping] window.

NOTE:

For CE300H-H8/3052, the memory mapping resolution allowed is limited to the following

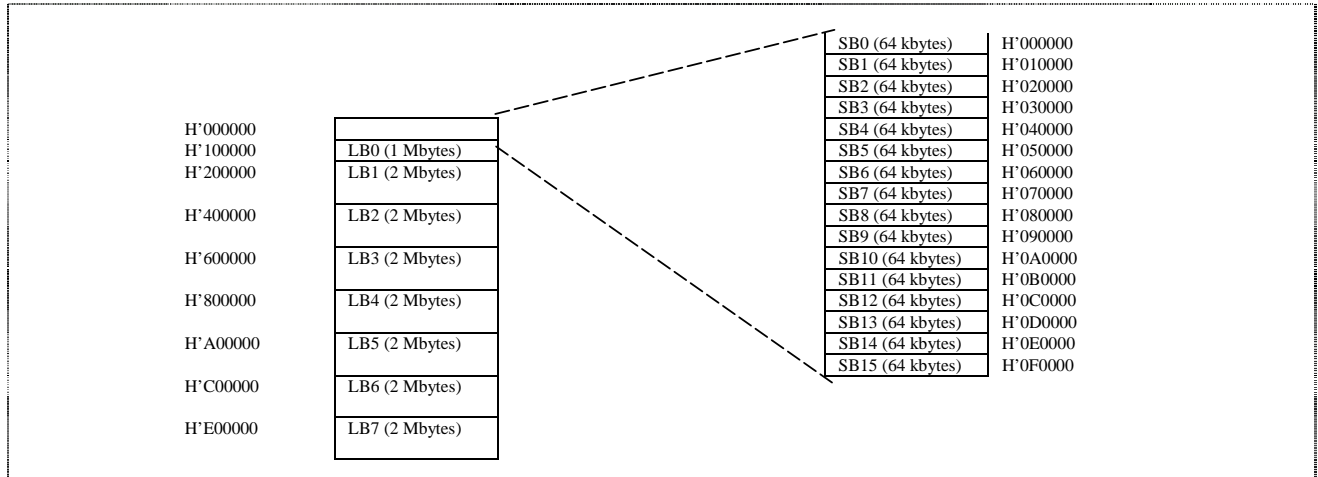


Figure 50 Mapping resolution & range

4.6 Viewing of Program

Programs can be viewed as

- Source Code level (C or assembly-language)
- Disassembly level (assembly-language)
- Mixed level (C and assembly-language)

4.6.1 Source Code level

Users may click on the file located in the workspace window to view the source code. However this is merely in “editor” point of view. Users have to download the code to the emulator. Once the code is downloaded, user can observe that “address values” have appeared in the source address column of the source file.

NOTE:

When a break condition occurred during a running program, HEW will open up the source code or disassembly window.

1. If the source code information is not available, the disassembly window will be opened.
2. If the downloaded project is a Elf/Dwarf2-based file, and the project has been moved from its original path, the source file may not be automatically found. In this case, HEW will open a source file browser dialog box, requesting user to manually locate the file.

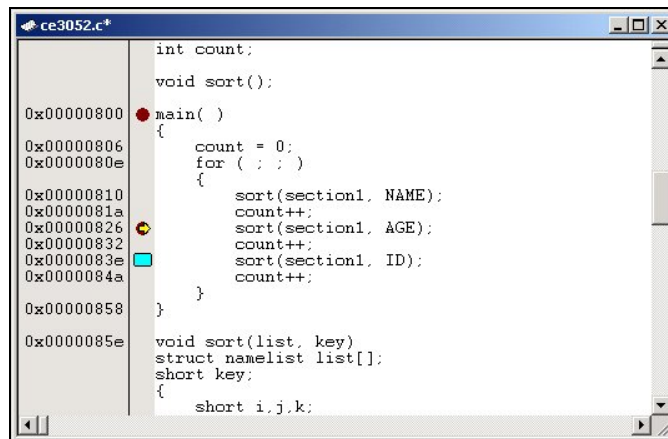


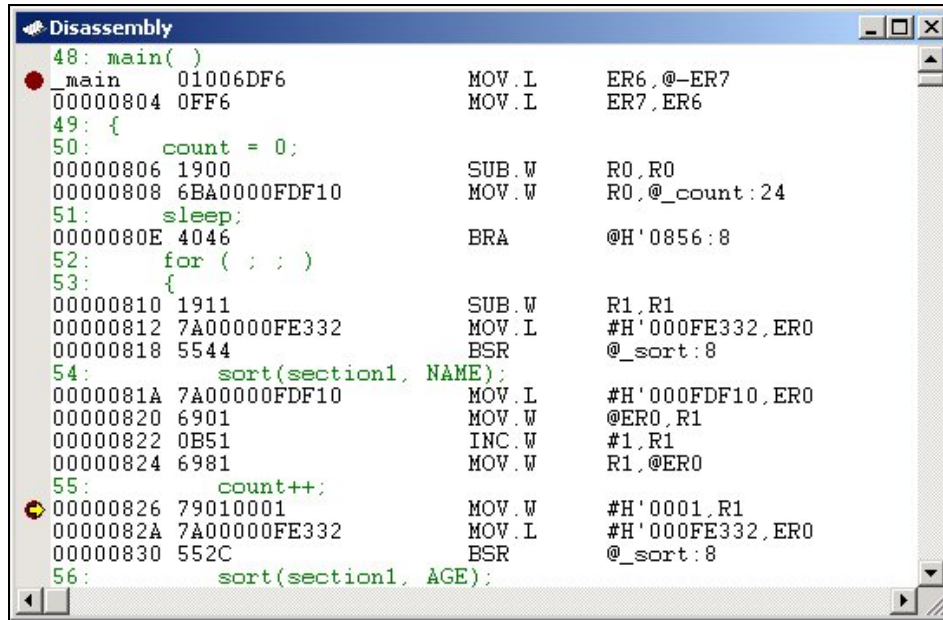
Figure 51 Source Level

Information available:

- Corresponding address for source file
- ➡ PC location
- Bookmark
- Breakpoint

4.6.2 Dis-assembly level

User can open the disassembly window via [View\Disassembly] pull down menu, or Right click on the source window, and select [Goto Disassembly]



```

Disassembly
48: main( )
  _main 01006DF6      MOV.L    ER6,@-ER7
00000804 0FF6             MOV.L    ER7,ER6
49: {
50:     count = 0;
00000806 1900             SUB.W    R0,R0
00000808 6BA0000FDF10    MOV.W    R0,@_count:24
51:     sleep;
0000080E 4046             BRA      @H'0856:8
52:     for ( ; ; )
53:     {
00000810 1911             SUB.W    R1,R1
00000812 7A00000FE332    MOV.L    #H'000FE332,ER0
00000818 5544             BSR      @_sort:8
54:         sort(section1, NAME);
0000081A 7A00000FDF10    MOV.L    #H'000FDF10,ER0
00000820 6901             MOV.W    @ER0,R1
00000822 0B51             INC.W    #1,R1
00000824 6981             MOV.W    R1,@ER0
55:         count++;
00000826 79010001        MOV.W    #H'0001,R1
0000082A 7A00000FE332    MOV.L    #H'000FE332,ER0
00000830 552C             BSR      @_sort:8
56:         sort(section1, AGE);
  
```

Figure 52 Disassembly Window (mixed display)

4.7 MCU related information

The MCU information can be monitor & control under the view menu.

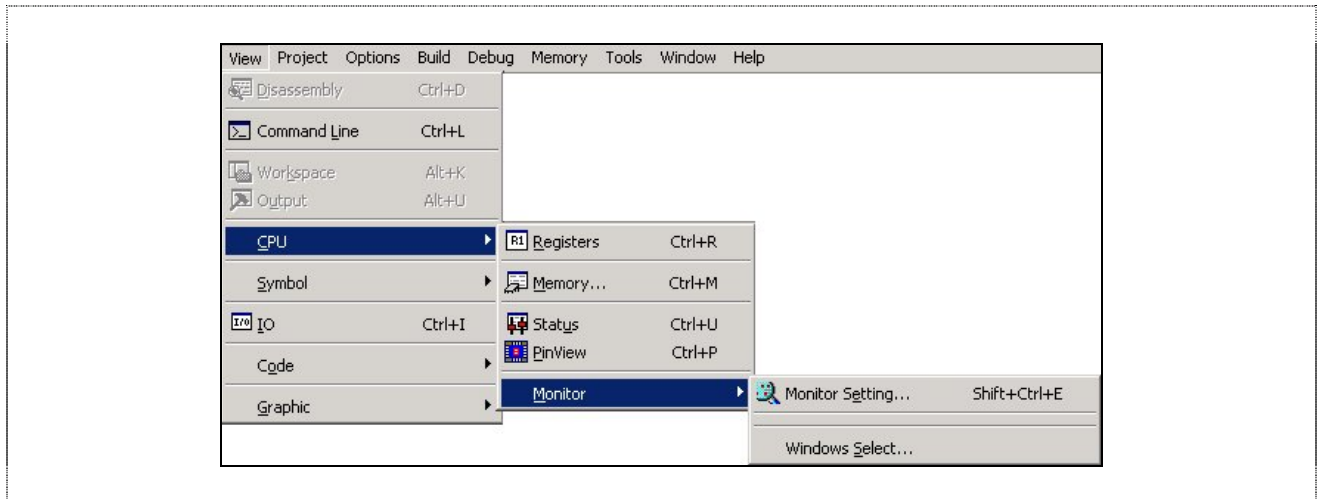


Figure 53 View CPU

4.7.1 Registers

User can access these registers directly through the Register windows during break mode only.

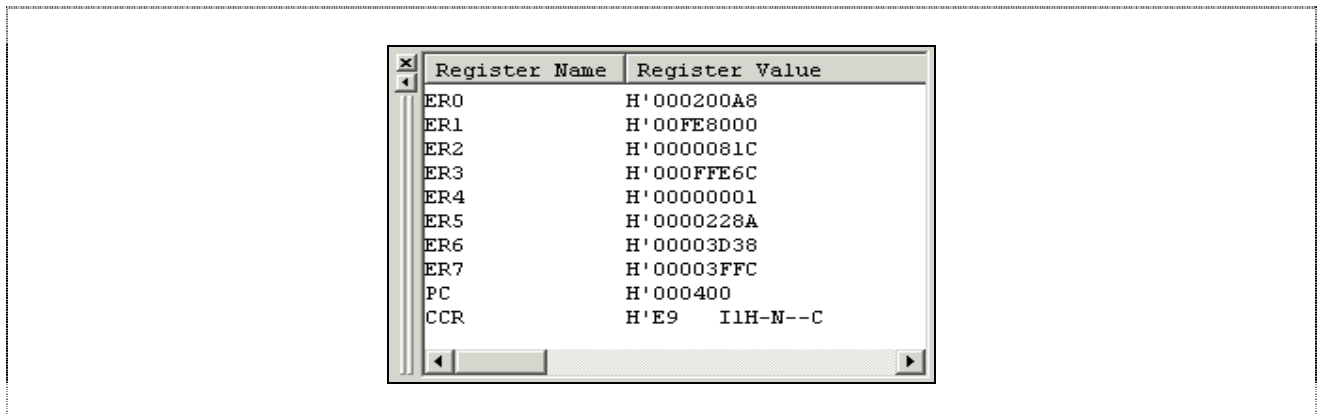


Figure 54 Register

4.7.2 Memory

User will have to set a pre-define address range to be monitor, before user can access the memory through the memory windows. The memory window will not refresh constantly by itself. The access methodology is different when emulation is in different mode (Run or Break). More memory functions are explained in Memory manipulation.

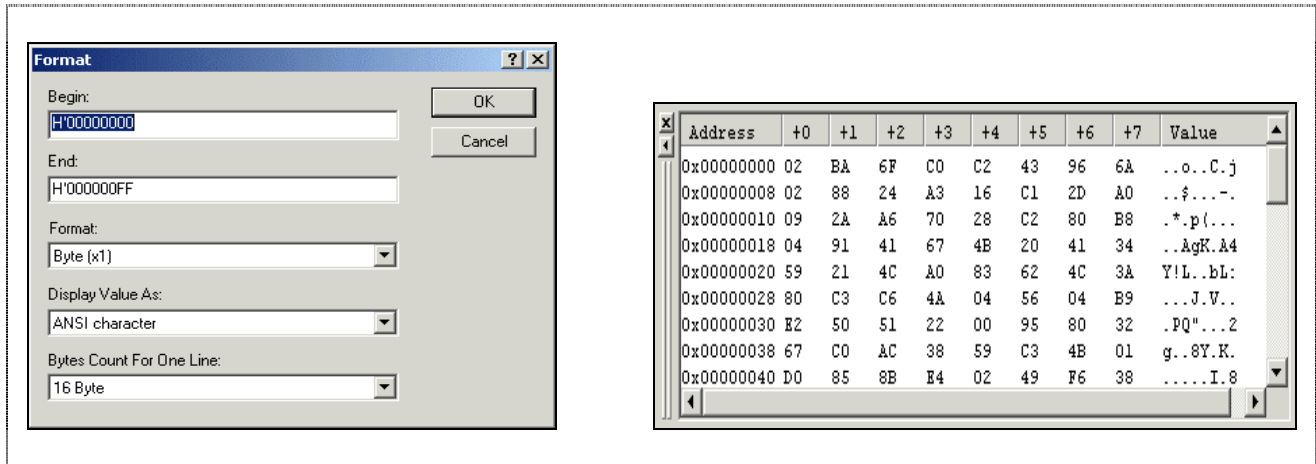


Figure 55 Set Memory

4.7.2.1 Parallel-On-The-Fly (POTF)

The function allows user to access memory during run mode i.e., user can read or write to the internal ROM/RAM, optional SODIMM and external target memory when the program is running. User has to refresh the memory window [Memory/Refresh] in order to view the latest changes to the memory.

NOTE:

1. This process will steal a cycle from the user running program in order to access the memory. If the refresh command is not issued, no cycle stealing will happen.
2. For internal ROM, the interruption will take about 400 ns.
3. For other area (such as target memory) , the interruption will depend on the operating clock, approximately 20xperiod.

Therefore, user is advised to open the necessary memory window at minimal size while executing POTF during program running.

Another accessing option is to use the Monitor window, whereby POTF is activated in continuous mode.

4.7.3 Status

The status window uses three different tabs to monitor the emulator setting.

4.7.3.1 Status - Memory

The memory tab displays

- the available memory setting for the selected device & mode.
- the Optional memory(SODIMM) resource usage is also displayed.
- the address range where the project is loaded

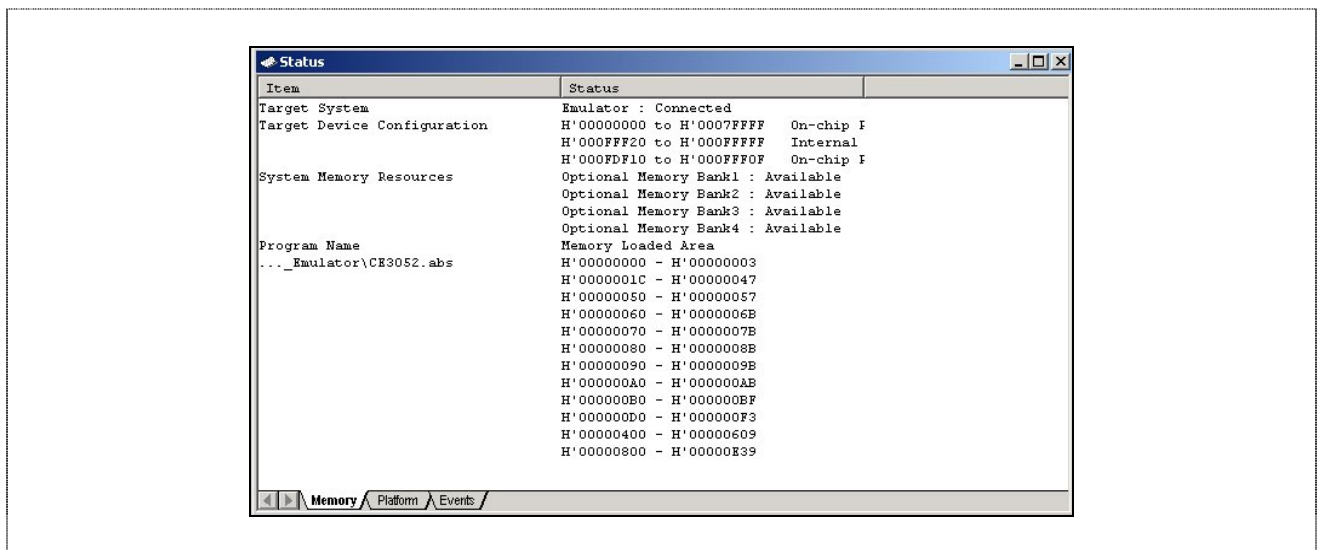


Figure 56 Status –memory window

4.7.3.2 Status - Platform

This platform tab shows the current emulation condition

- CPU, mode and clock setting
- Run time (at 100 ns resolution, no upper limit)
- Detect optional memory (Sodimm) Present
- Detect User Cable (Target) connection
- Display snapped address & CPU status during run
- Display read UVCC (Target power supply) voltage level

For H8/3052, possible display of MCU status are:

- DMAC
- Refresh
- High/Medium (Instruction Fetch)
- Standby
- Sleep
- Bus Released State
- Data access cycle
- Other

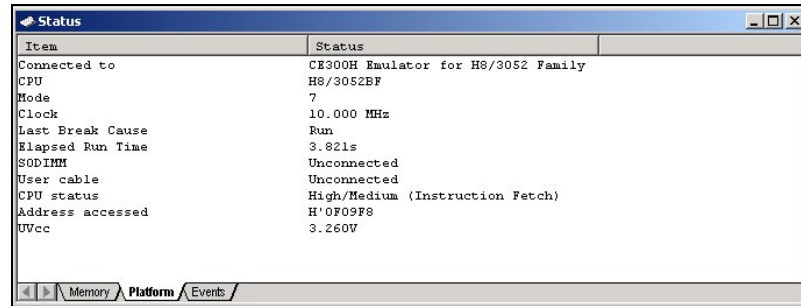


Figure 57 Status – Platform window

4.7.3.3 Status - Events

The events tab show the usage of

- PC Breakpoints
- PFG Breakpoints
- EV Breakpoints

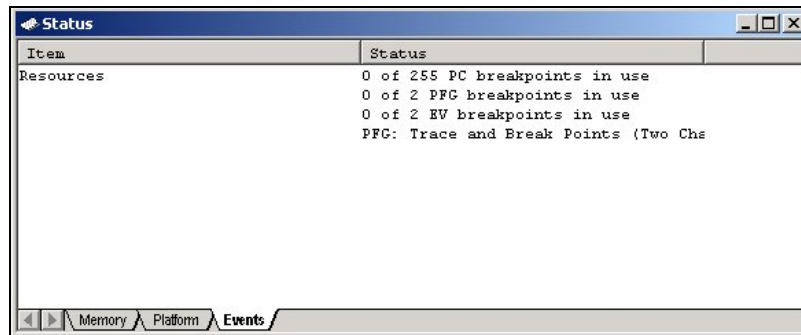


Figure 58 Status – Events window

4.7.4 PinView

The pinview function allows user to have a graphical view of the microcomputer chip' pins. It can be activated in the break and user mode [View/CPU/Pinview]. User can select different microcomputer packages to be displayed [Options/Emulator/System...] *in the initial setup stage.*

PinView can only display a snap shot of the MCU Pin status at a regular interval. Its performance is incomparable to the real-time characteristic of an oscilloscope. For slower PC, user is advised to view the pin status in Tree form (text form).

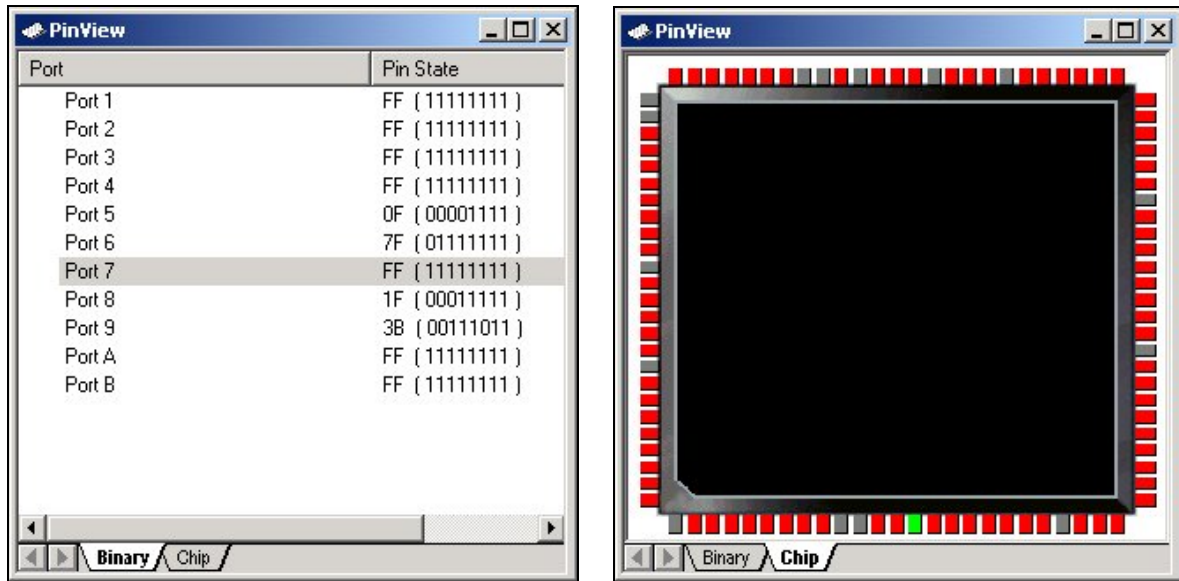


Figure 59 PinView (Chip and Tree View)

This function will help user

- to debug the program when the target system is not ready.
- to debug the target system when it is connected to the emulator.
- to debug the target system connection when it is connected to the emulator.

4.7.5 Monitor (continuous POTF)

The monitor window make use of the POTF mechanism to access the MCU memory during RUN. The window provide a mean to have update of memory data at constant preset interval. Moreover the changes in data value will be highlighted with a change in colour.

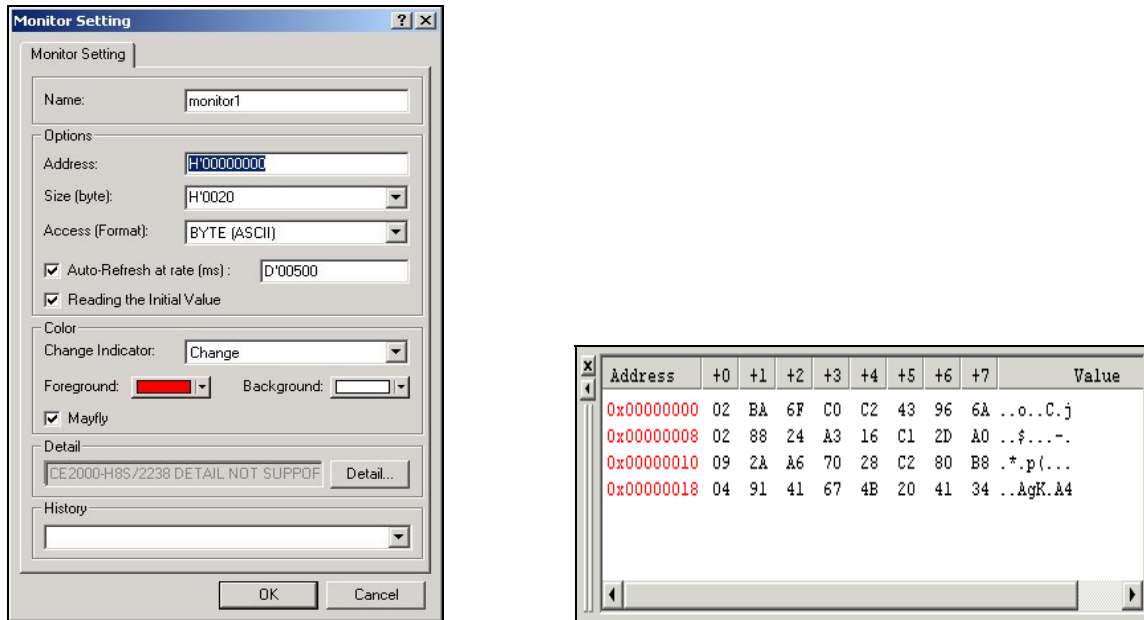
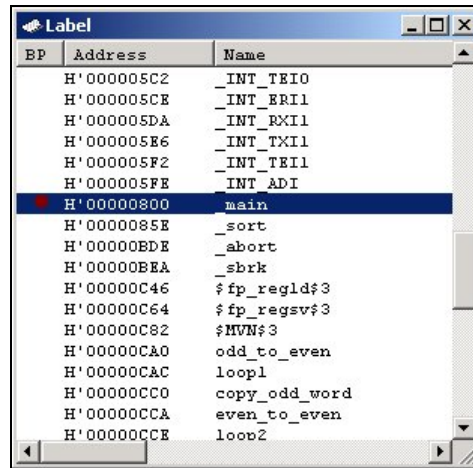


Figure 60 Monitor window

4.7.6.1 Label

When debug information is included, detail of all labels will be displayed in the Label window. User can add new label into the window for simple reference too.



BP	Address	Name
	H'000005C2	_INT_TEI0
	H'000005CE	_INT_ERI1
	H'000005DA	_INT_RXI1
	H'000005E6	_INT_TXI1
	H'000005F2	_INT_TEI1
	H'000005FE	_INT_ADI
	H'00000800	main
	H'0000085E	_sort
	H'00000BDE	_abort
	H'00000BEA	_sbrk
	H'00000C46	\$fp_regld\$3
	H'00000C64	\$fp_regsv\$3
	H'00000C82	\$MVN\$3
	H'00000CA0	odd_to_even
	H'00000CAC	loop1
	H'00000CC0	copy_odd_word
	H'00000CCA	even_to_even
	H'00000CCE	loop2

Figure 62 Label

NOTE:

When a label values match an operand, the corresponding instruction's operand is replaced by the label. If two or more labels have the same value, the earlier label (alphabetical order) will be displayed.

4.7.6.2 Watch

User will has to add the variable into the watch window.

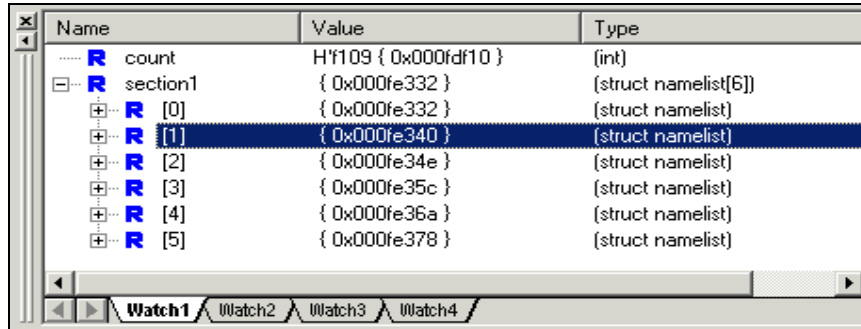


Figure 63 Watch

[R] indicates the realtime status of the watch variables. This is enable in be selecting [Auto Update] when right click on the window.

NOTE:

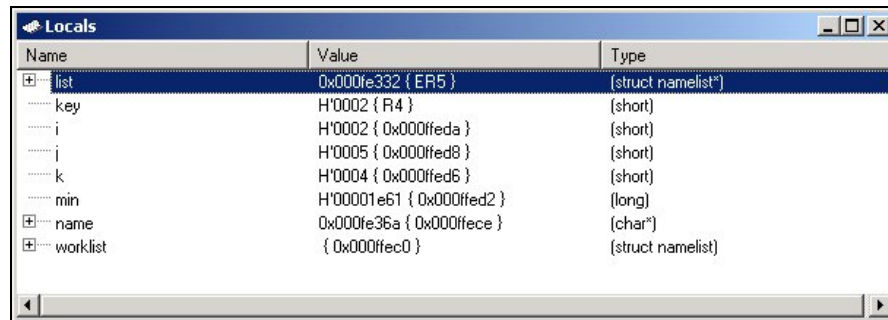
- The variables can be displayed only if debug information is included in the absolute file(abs)
 - The variables have not been excluded after the compiler optimization
 - The variables are not cleared as macro
- When realtime [R] monitoring is activated, realtime emulation is not possible as execution is interrupted temporary at constant interval.

4.7.6.3 Local

The Local variables will appear in the Locals window when user code has break/stop at a sub-routine.

NOTE:

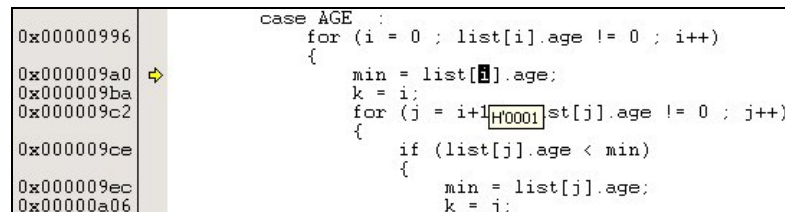
Local variables are temporary data stored in stack. Therefore it can only be viewed when execution stops within a routine.



Name	Value	Type
list	0x000e332 { ER5 }	(struct namelist*)
key	H'0002 { R4 }	(short)
i	H'0002 { 0x000feda }	(short)
j	H'0005 { 0x000fed8 }	(short)
k	H'0004 { 0x000fed6 }	(short)
min	H'00001e61 { 0x000fed2 }	(long)
name	0x000fe36a { 0x000fece }	(char*)
worklist	{ 0x000fec0 }	(struct namelist)

Figure 64 Locals

Tooltip watch- place the cursor at the variable and the general information of the variable will appear.

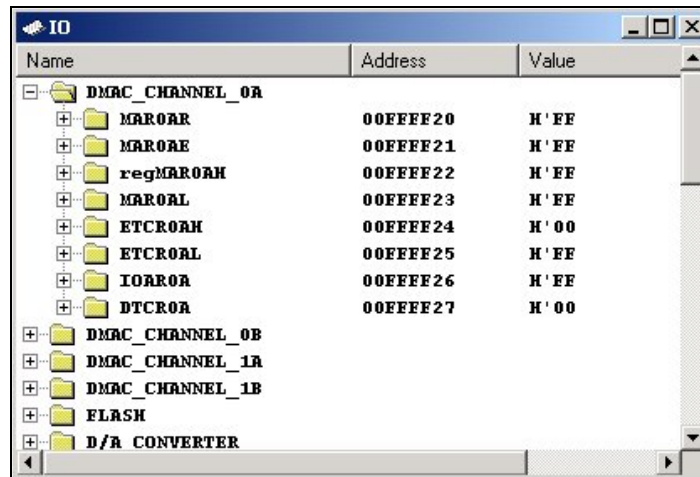


0x00000996	case AGE :
0x000009a0	for (i = 0 ; list[i].age != 0 ; i++)
0x000009ba	{
0x000009c2	min = list[i].age;
0x000009ce	k = i;
0x000009ec	for (j = i+1; list[j].age != 0 ; j++)
0x00000a06	{
	if (list[j].age < min)
	{
	min = list[j].age;
	k = j;
	}
	}

Figure 65 Tooltip

4.7.7 I/O

The IO window provide a easy access to MCU IO registers. The Address & Data value of respective peripherals & controllers registers are displayed in the IO window.



Name	Address	Value
DMAC_CHANNEL_0A		
MAR0AR	00FFFF20	H'FF
MAR0AE	00FFFF21	H'FF
regMAR0AH	00FFFF22	H'FF
MAR0AL	00FFFF23	H'FF
ETCR0AH	00FFFF24	H'00
ETCR0AL	00FFFF25	H'FF
IOAR0A	00FFFF26	H'FF
DTCR0A	00FFFF27	H'00
DMAC_CHANNEL_0B		
DMAC_CHANNEL_1A		
DMAC_CHANNEL_1B		
FLASH		
D/A CONVERTER		

Figure 66 Input and Output Register

4.7.8 Break Functions

Various breakpoints setting is discussed as follow.

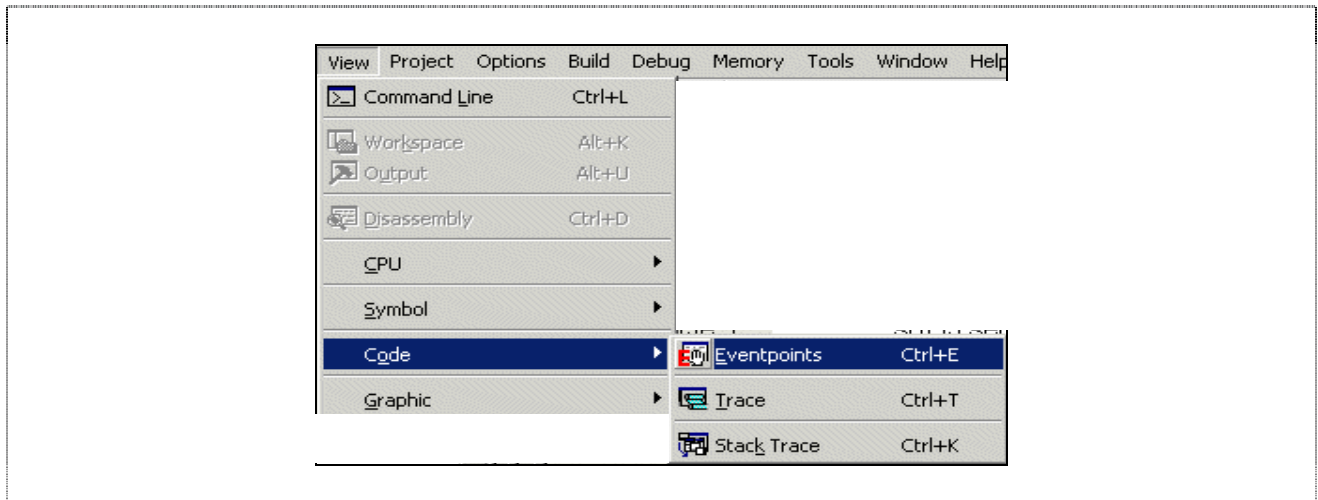


Figure 67 View Code

Breaks are events used to intercept the normal program execution when a specific condition is matched. There are six types of break in CE300H.

In general, these break functions can be classified into two main classes, namely :

- Hardware Event
- Software PC break.

For Hardware Event break, the preset break condition will cause the break event to occur after an instruction is executed.

For Software PC break, the break condition causes the break event to occur before the break condition.

Table 2 Types of Breaks Encountered During Emulation.

	Types of Break	Description
1	Event Break (Hardware Break)	A break occurs when the CPU matches with a condition specified in the Eventpoints window, or when the pre-fetch cycle of the CPU agrees with the specified states.
2	PFG Break (Hardware Break)	A break occurs when the CPU matches with a condition specified in the Eventpoints window, or when the pre-fetch cycle of the CPU agrees with the specified states. The configuration of PFG break is dependent on the type of PFG breaks being selected.
3	PC Break (Software Break)	A break occurs at the program address specified by Eventpoints window. The instruction at this address is replaced with a system instruction before the execution of code. If a PC breakpoint is detected, the emulation stops at the specified address before executing the subsequent instruction.
4	User Break (Hardware Break)	Pressing the ESC key of the host PC generates a break.
5	Reserved Area Break (Hardware Break)	A reserved area break occurs when user code accesses (reads from or writes to) prohibited area of the MCU memory map. This can be disabled in the Option/Emulator/System... Configure platform window.
6	Write Protect Break (Hardware Break)	A write protect break occurs when user codes attempts to write to the ROM area. This can be disabled in the Option/Emulator/System... Configure platform window.

NOTE:

Software PC break cannot be set in External ROM and Flash area.

User is advised to use the Hardware Break such as Event Break and PFG Break in this condition

4.7.8.1 Event (EV) Breakpoint

For the CE300H-H8/3052 emulator, two event breakpoints are supplied permanently. The conditions to determine the breaks are:

- Address
- CPU Access (Read/Write)

The break condition occurs in a “AND” condition. If any of the factors defined are not fulfilled, the particular break condition will be ignored. A “0” in address mask is to ignore the particular address bit.

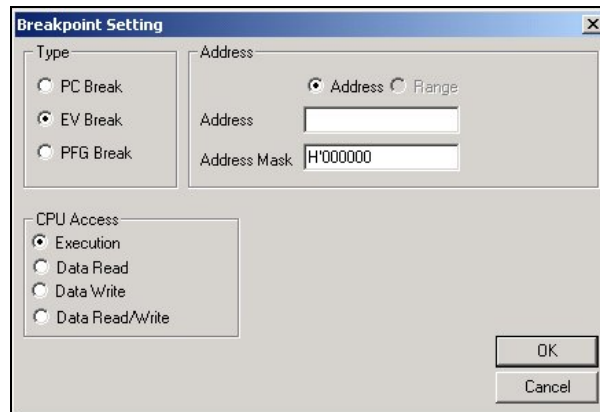


Figure 68 Breakpoint Setting

4.7.8.2 PFG Function – Event Breakpoint (Trace & Event Break)

Two event breakpoints can be downloaded into the PFG. The PFG Event Breaks have the following conditions:

- Address
- Data
- Access (Read/Write)

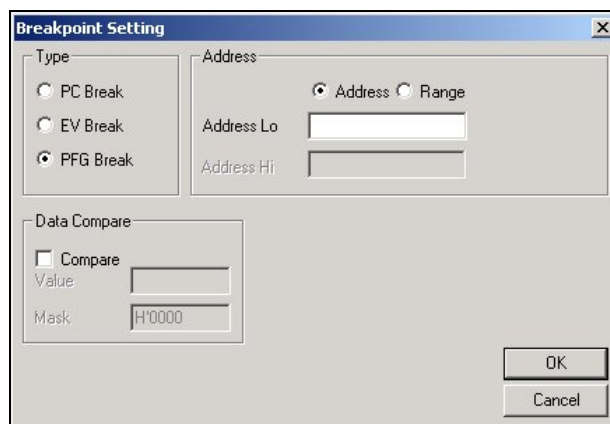


Figure 69 PFG 1 - Breakpoint Setting

4.7.8.3 PFG Function – Event Breakpoint (Sequence Event Break)

Two event breakpoints can be downloaded into the PFG. The PFG Event Breaks have the following conditions:

- Address (In-Range or Out of Range)
- Data compare
- Access (Read/Write)
- Event counts
- Sequence (Then, AND)

Figure 70 PFG 2 – Breakpoint Setting

Note:

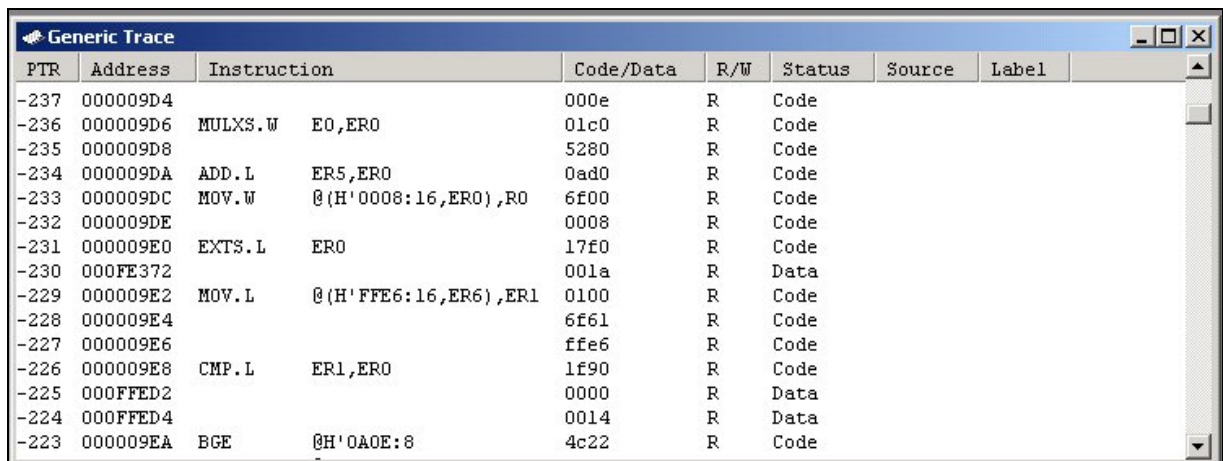
To activate the IF-THEN or AND condition, the first event must be set before setting the sequence condition for the second event.

4.7.9 Trace - PFG Function – Trace (Trace & Events Break)

There is no permanent trace provided. However, user can download the Trace Function into the PFG to keep track of the program. The Trace provided is a 256-cycles trace. The Trace function will display the last 256 cycles of information upon encountering a break condition.

In each trace cycle, the available displayed data are

- 24-bit Address
- 16-bit Data
- Read / Write
- MCU status
- The Executed Code (C or assembly)

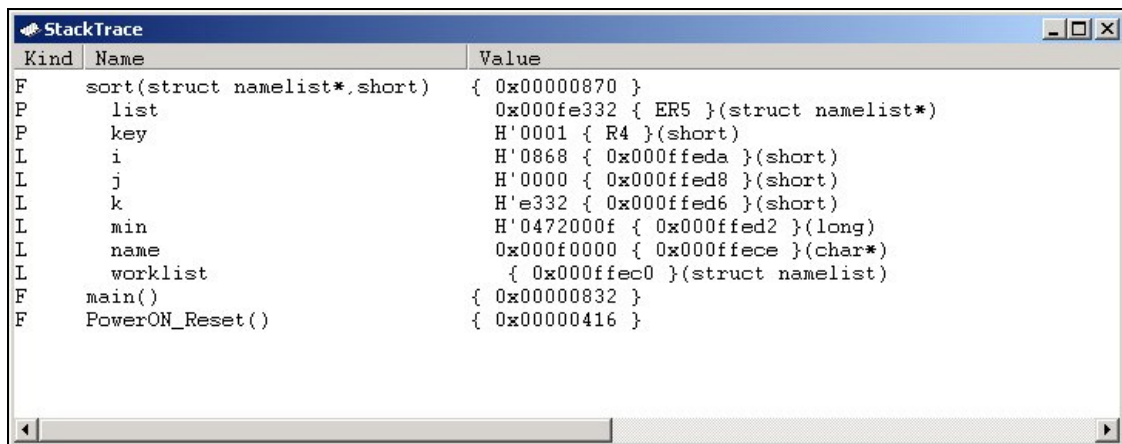


PTR	Address	Instruction	Code/Data	R/W	Status	Source	Label
-237	000009D4		000e	R	Code		
-236	000009D6	MULXS.W E0,ERO	01c0	R	Code		
-235	000009D8		5280	R	Code		
-234	000009DA	ADD.L ER5,ERO	0ad0	R	Code		
-233	000009DC	MOV.W @(H'0008:16,ERO),R0	6f00	R	Code		
-232	000009DE		0008	R	Code		
-231	000009E0	EXTS.L ERO	17f0	R	Code		
-230	000FE372		001a	R	Data		
-229	000009E2	MOV.L @(H'FFE6:16,ER6),ER1	0100	R	Code		
-228	000009E4		6f61	R	Code		
-227	000009E6		ffe6	R	Code		
-226	000009E8	CMP.L ER1,ERO	1f90	R	Code		
-225	000FFED2		0000	R	Data		
-224	000FFED4		0014	R	Data		
-223	000009EA	BGE @H'0A0E:8	4c22	R	Code		

Figure 71 Trace

4.7.10 Stack Trace

The Stack Trace window can be selected if only debug information has been supplied. Stack Trace window shows the function call history.



Kind	Name	Value
F	sort(struct namelist*,short)	{ 0x00000870 }
P	list	0x000fe332 { ER5 }(struct namelist*)
P	key	H'0001 { R4 }(short)
L	i	H'0868 { 0x000ffeda }(short)
L	j	H'0000 { 0x000ffed8 }(short)
L	k	H'e332 { 0x000ffed6 }(short)
L	min	H'0472000f { 0x000ffed2 }(long)
L	name	0x000f0000 { 0x000ffece }(char*)
L	worklist	{ 0x000ffec0 }(struct namelist)
F	main()	{ 0x00000832 }
F	PowerON_Reset()	{ 0x00000416 }

Figure 72 Stack Trace

The following items can be displayed:

Kind	Indicate the symbol type
F:	Function
P:	Function parameter
L:	Local variable
Name	Indicate the symbol name
Value	Indicate the value, address and symbol type

At default, the function parameter and local variable are not displayed.

To enable all the items, right click in the [Stack Trace] window and select [View Setting...].

4.7.11 Image

The image window displays an image based on the MCU memory data. The image attribute is determined by the following setting.

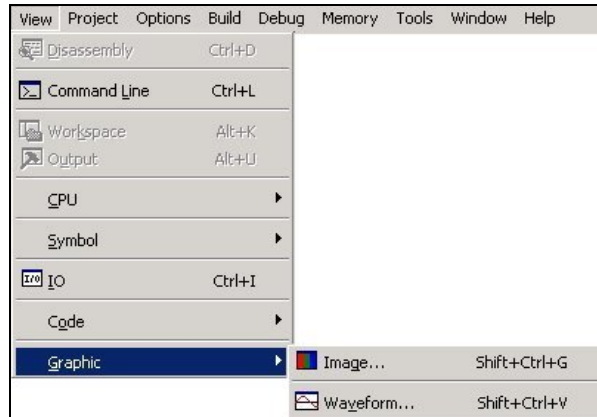


Figure 73 View Graphic

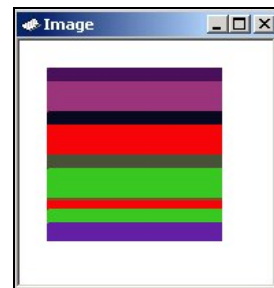
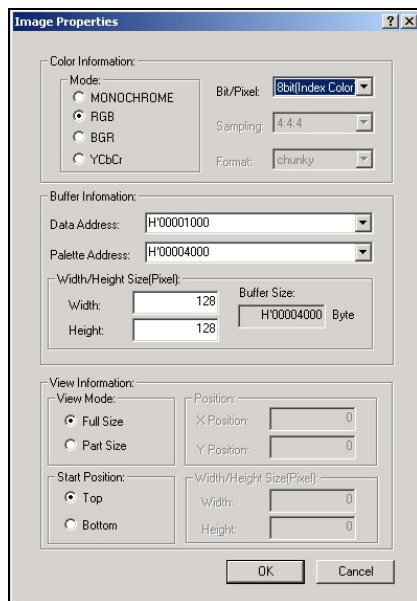


Figure 74 Bitmap

The [Image Properties] dialog box is used to specify the display method of the [Image] window.

The following items are to be specified:

- [Colour Information]:** Specifies the colour information of the image to be displayed
- [Mode]: Specifies the format
 - [MONOCHROME]: Display in black & white
 - [RGB]: Display in Red, Green and Blue
 - [BGR]: Display in Blue, Green and Red
 - [YcbCr]: Display by Y (brightness), Cb (colour difference in blue) and Cr (colour difference in red)
 - [Bit/Pixel]: Specifies Bit/Pixel according to the selected [Mode].
(valid when RGB or BGR is selected)
 - [Sampling]: Specifies the format of sampling
(valid when YcbCr is selected)
- [Buffer Information]:** Specifies the area to store data, size, and the address of the palette.
- [Data address]: Specifies the start address of the memory when image data is to be displayed (Display in Hexadecimal)
 - [Palette Address]: Specifies the start address of the memory of the colour palette data.
(Diaply in Hexadecimal)(Valid when 8bit is selected for RGB or BGR)
 - [Wifith/Height size]: Specifies the width & height of the image
 - [Width (Pixel)]: Specifies the width of the image
(When a prefix is omitted, the values are input and displayed in decimal)
 - [Height (Pixel)]: Specifies the width of the image
(When a prefix is omitted, the values are input and displayed in decimal)
 - [Buffer Size]: Displays the buffer size of the image from the width and height
(Display in Hexadecimal)
- [View Information]:** Specifies the location, size and data start location of the part to be displayed among the entire image
- [View module]: Specifies the entire/part to be display in the image
 - [Start Position]: Displays data from upper left or lower left
 - [Position]: Specifies the start position (X,Y) of the image when part of the image is to be displayed.
(When a prefix is omitted, the values are input and displayed indecimals)
 - [Width/Height Size]: Specifies the width and height of the image to be displayed partly.
(When a prefix is omitted, the values are input and displayed indecimals)

4.7.12 Waveform

The waveform window displays an waveform based on the MCU memory data. The waveform attribute is determined by the following setting.

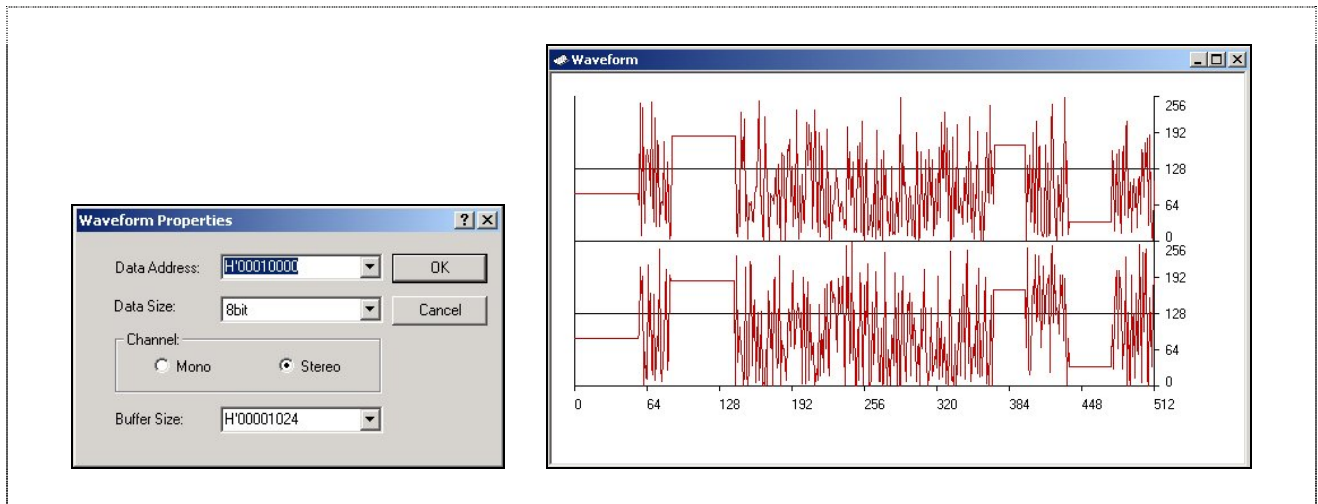


Figure 75 Waveform

NOTE:

The [Auto Refresh] refers to the automatic refresh of memory data to the graphical display after a break condition occurred.

4.8 MCU memory manipulation

General supported functions are

- search
- copy
- fill
- refresh
- save (in File menu)
- verify (in File Menu)

Memory Data display format can be in

- Byte (x1)
- Word (x2)
- Long (x4)
- Double (x8)

Memory value display format can be in

- ANSI character
- unsigned char
- signed char
- unicode character
- float
- double
- 16 bit fixed
- 32 bit fixed

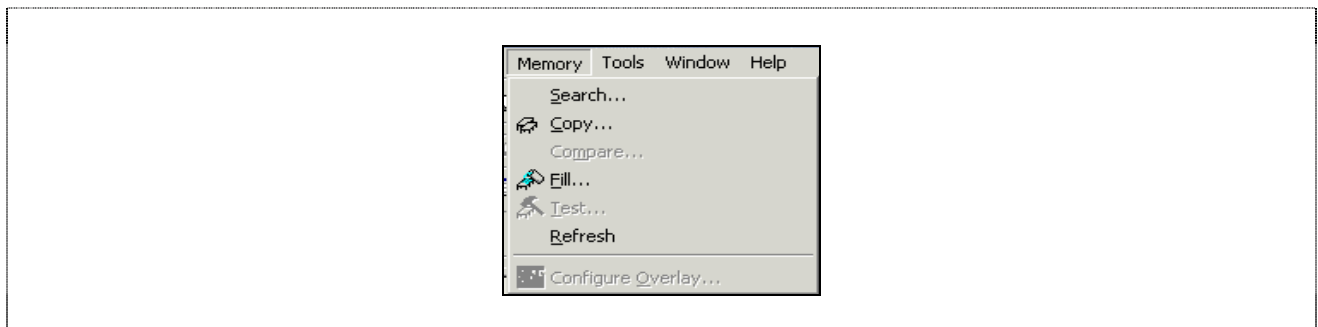


Figure 76 Memory Functions

NOTE: User has to set the bus state controller correctly before the external memory can be accessed.

These functions can be applied on all the three memory type of the emulator, namely:

- On-Chip Memory (internal ROM/RAM)
- Optional SODIMM Memory (2Mbytes SRAM)
- User Target Memory

4.9 Execution of MCU Code

The MCU executes the user code either in “RUN” or “STEP” modes.

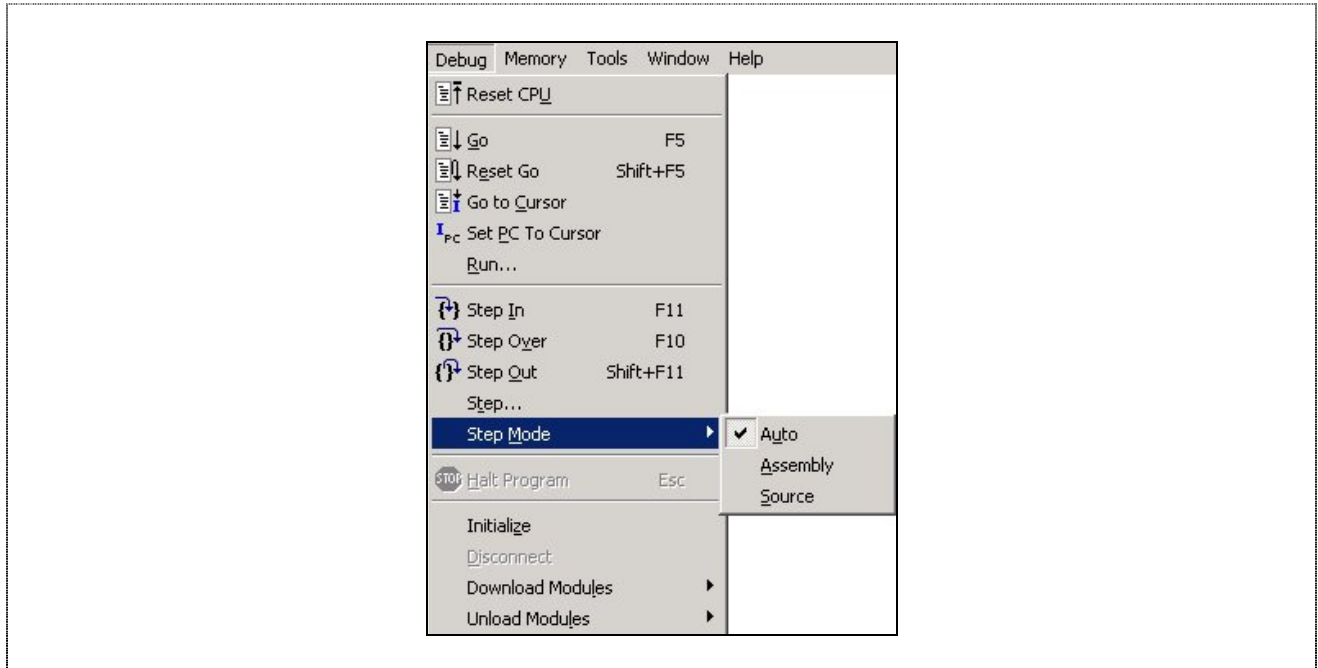


Figure 77 Debug Functions

4.9.1 Reset CPU

When [RESET CPU] command is activated, the following actions will take place,

PC	=	Power on Reset vector value
ER7	=	no change
ER0-6	=	no change
CCR	=	no change

The microcomputer is reset.

i.e all internal peripherals registers will be at default state.

4.9.2 Go, Reset Go, Goto Cursor, Set PC to Cursor, Run...

Real-time execution [Debug] by the H8 chip based on the user setting. There is no “cycle steal” during the execution mode, unless user activates the “refresh memory” command [Memory/ Refresh], Auto Update the watch window, activate the monitor window. The command will cause the HEW Ddebugger to steal a cycle from the running chip, in order to access the memory (Parallel on the Fly - POTF).

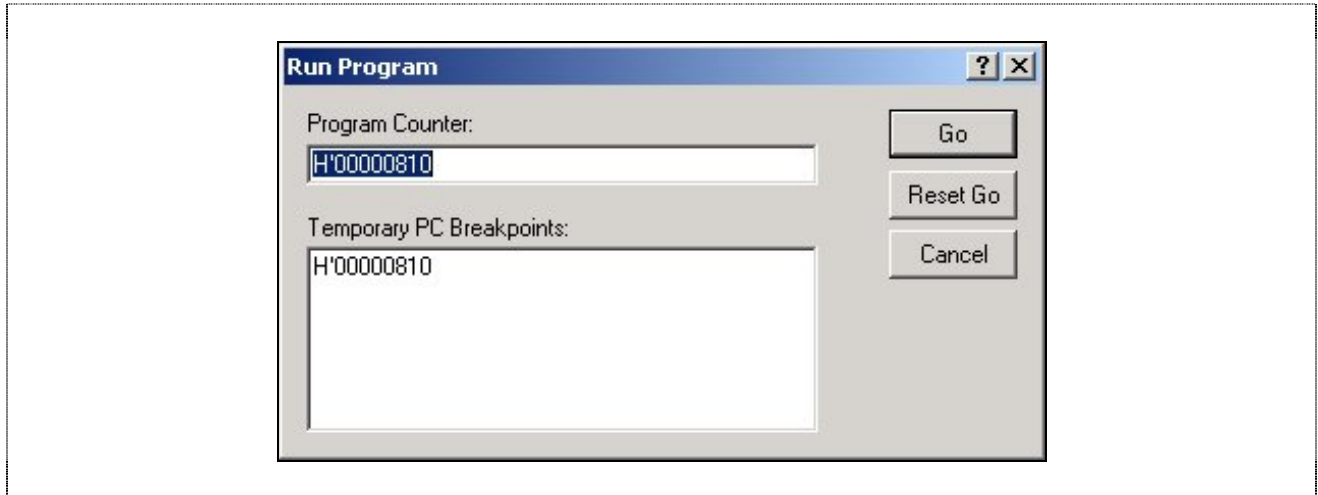


Figure 78 Run Program

NOTE

1. [Goto Cursor] will not halt if the running program never executed the code at the cursor. Stopping of the execution is possible via [ESC] key.
2. [Run...] the temporary PC breakpoint will be cleared after the completion of the current execution.

4.9.3 Single-Step

There are four types of Single Step:

- Step-In,
- Step-Out &
- Step-Over.
- Step...
- Step Mode (Auto, Assembly and Source)

Single Step executes the instruction at the current program counter. If an interrupt is asserted, the interrupt service routine will not be serviced unless a “Go” command is issued.

Step-In will execute a single instruction only. For C source file, a single step will execute a “single C source code”. Whereas for an assembly file, a single step will execute a single assembly instruction code.

Step-Out executes till it has branched out of the current routine.

Step-Over executes a function call (and any function call called by the function) and halt at the next instruction.

Step... will execute multiple Step-in as specified by the user. The delay enable a visual view of the code running sequence.

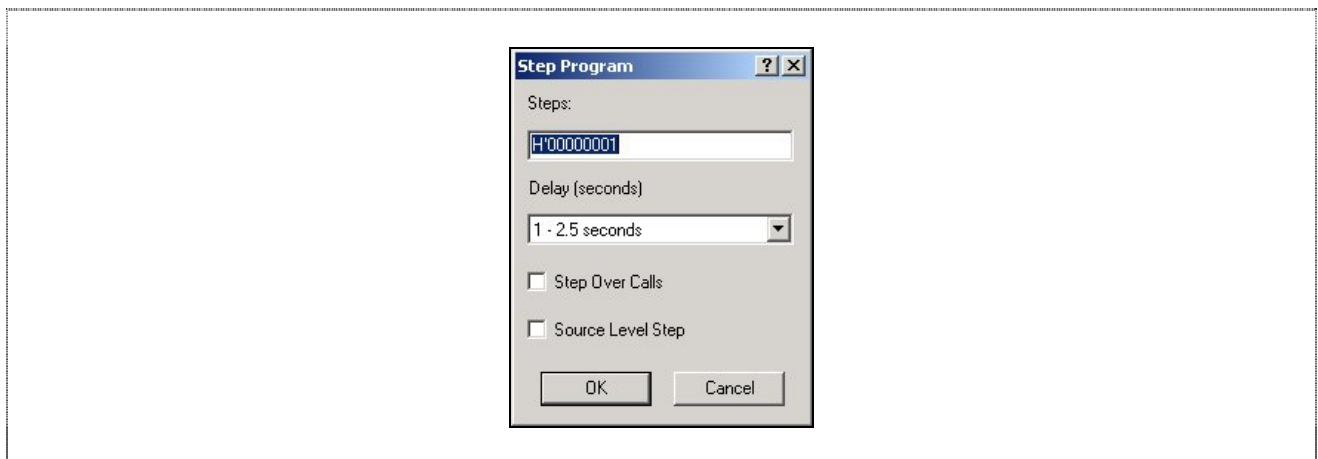


Figure 79 Step program

Step Mode setting decide on how the step instruction operates.

- *Auto:* The execution mode will depend on the active window. i.e. when step instruction is activated in a C Source window, a C-source level step will be invoked.
- *Source:* When Step instruction is executed, user will see a C-source level step. i.e. a series of assembly code is run in the background.
- *Assembly :* When step is executed, the current assembly code located at current PC will be executed. The disassembly window will pop up if the current window is a C source window.

4.10 C-source Level Debugging

If user compiles and links the code with the Debug option enabled, the ELF/DWARF2 (.abs) file with the debugging information is generated. This enables user to debug the code in C-source level i.e. ,

- Display code in C source level,
- Step in,out & over code in C source level,
- View label,
- Goto label (address),
- View local
- Instant/add watches (local and user defined)
- Stack Trace

Section 5. H8/300H Function Supported

The CE300H-H8/3052 emulator can support the H8 series of microcomputer. The various functions support for the H8 are detailed below.

5.1 MCU Operating Mode Setting

The CE300H-H8/3052 emulator supports the four operating modes of the H8. User can select the MCU operating mode via the [Option/Emulator/System] window. The following table shows the MCU settings.

				H8/3048, 3022, 3035				H8/3078			
Operating mode	MD2	MD1	MD0	MCU Mode	CPU mode	Address Space	Bus Width	MCU Mode	CPU mode	Address Space	Bus Width
Mode 0	0	0	0	-	-	-	-	-	-	-	-
Mode 1	0	0	1	ROM invalid expansion	Adv	1M	8	ROM invalid expansion	Adv	1M	8
Mode 2	0	1	0			1M	16			1M	16
Mode 3	0	1	1			16M	8			16M	8
Mode 4	1	0	0			16M	16			16M	16
Mode 5	1	0	1			1M	8			1M	8
Mode 6	1	1	0			16M	8			1M	16
Mode 7	1	1	1	Single-chip		1M	-	-	-	-	-

				H8/3032,3042,3039, 3001/2/3/4/5,				H8/3067,3024,3062,3068,3069			
Operating mode	MD2	MD1	MD0	MCU Mode	CPU mode	Address Space	Bus Width	MCU Mode	CPU mode	Address Space	Bus Width
Mode 0	0	0	0	-	-	-	-	-	-	-	-
Mode 1	0	0	1	ROM invalid expansion	Adv	1M	8	ROM invalid expansion	Adv	1M	8
Mode 2	0	1	0			1M	16			1M	16
Mode 3	0	1	1			16M	8			16M	8
Mode 4	1	0	0			16M	16			16M	16
Mode 5	1	0	1			1M	8			16M	8
Mode 6	1	1	0	Single-chip	Normal	64K	-	Single-chip	Normal	64K	-
Mode 7	1	1	1	Single-chip	Adv	1M	-	Single-chip	Adv	1M	-

				H8/3008, 3006,3007			
Operating mode	MD2	MD1	MD0	MCU Mode	CPU mode	Address Space	Bus Width
Mode 0	0	0	0	-	-	-	-
Mode 1	0	0	1	ROM invalid expansion	Adv	1M	8
Mode 2	0	1	0			1M	16
Mode 3	0	1	1			16M	8
Mode 4	1	0	0			16M	16
Mode 5	1	0	1			16M	8
Mode 6	1	1	0			16M	8
Mode 7	1	1	1	-	-	-	-

Table 3 MCU Operating Modes

5.2 Memory Area

The H8 has a maximum memory area of 16 Mbytes. The four classification of memory are:

5.2.1 Internal ROM Area

The emulator has a substitute RAM for the H8 internal ROM. Access to this substitute RAM is as follows:

- Access arising from user program execution : Read only, write disabled
- HEW DEBUGGER : Read and write enabled

Therefore, the user can modify the internal ROM area memory and load the object program.

5.2.2 Internal RAM Area

The emulator has a substitute RAM for the H8 internal RAM. When user tries to access the internal RAM, this substitute RAM is always accessed instead. User can access the internal RAM area from the user program or with an HEW DEBUGGER.

5.2.3 Internal I/O Area

When the internal I/O area is accessed, the emulator accesses the internal I/O. User can read and write to the internal I/O area from the user program or with HEW DEBUGGER.

5.2.4 External Area

The external target area will be accessed as long as the area does not belong to:

- Internal ROM
- Internal RAM
- Internal I/O
- Mapped Emulator Optional Memory (SODIMM)

User has to set the area to be read/write or write protected, in order to access the area. Otherwise, the area will be treated as Access inhibited or Guarded.

The number of accessing states and type of access (e.g. DRAM access) will be determined by the H8 registers (eg. BSC, WCR1...).

5.3 Low Power Mode

For reduced power consumption, the H8 has sleep, hardware standby and software standby mode.

5.3.1 Hardware Standby Mode

The hardware standby mode is switched by the STBY signal input. In default, the STBY signal is masked in the HEW DEBUGGER setup windows.

If activated, this signal will initialise the emulator registers, and force the emulator to enter power-on reset state after the signal is released

5.3.2 Sleep and Software Standby Modes

The sleep and software standby modes are switched using the SLEEP instruction. These modes can be cleared with either the normal clearing function or with the break condition fulfilled (including 'ESC' key input). The program will then be put to a stop.

Trace information is not acquired in sleep and software standby modes.

NOTE: When restarting after a break, the code after the SLEEP instruction will be executed.

5.4 Interrupts

During emulation, the user can interrupt the H8. If an interrupt occurs while the emulator is in the break mode, the interrupt is not processed. However, if an edge-sensitive interrupt occurs while the emulator is in the break mode, the emulator latches the interrupt and executes the interrupt processing routine when the Run command is instructed again.

Interrupt request is masked during single step.

5.5 Control Input Signals (RES, NMI, STBY)

The H8 input signals (RES, NMI and STBY) are controlled by the emulator. These signals can be masked by the HEW DEBUGGER. In default, the STBY signal is masked. The RES signal is valid only when emulator enters run mode.

5.6 Watch Dog Timer (WDT)

The WDT operates during run mode emulation, and does not operate when the emulator is in the break mode. The timer is disabled at a break and enabled when emulation resumes.

5.7 Integrated Timer Pulse Unit (ITU) and Timing Pattern Controller (TPC)

The ITU and TPC operate during the command input wait state as well as during emulation. i.e., the timer pins are valid even when the user program has stopped. The user can rewrite the timer registers with the HEW DEBUGGER I/O window or *Memory* window.

5.8 Serial Communications Interface (SCI)

The serial communications interface signals are connected to the user system directly from the H8 MCU in the emulator. Therefore, the interface is valid both in the break and run mode. For example, when writing data to the TDR (transmit data register) in the HEW DEBUGGER I/O or *Memory* Window, the serial communications interface output will be initialized, and data will be output to the TXD line.

5.9 DMA Controller (DMAC)

The DMAC operates during emulation execution and in the break mode. When a transfer request occurs, the emulator carries out DMA transfer.

5.10 Wait State Controller

The H8/300H wait state controller has a programmable wait mode and a WAIT pin input. The programmable wait mode is valid when the emulation memory or user (target or external) memory is accessed, but input to the WAIT pin is only valid when the user memory is accessed. Input to the user WAIT pin is always valid during refresh cycles.

5.11 I/O Ports

The H8/300H I/O ports can be used as peripheral module I/O pins or as an address/data bus, depending on the operation mode or internal register settings. The I/O port pins are valid during execution & break state. User can access or modify the I/O port pins (multiplexed with peripheral module I/O pins) with the HEW DEBUGGER I/O and MEMORY window.

5.12 A/D and D/A Converter

The analogue I/O pins are connected to the user system directly from the H8/300H MCU. Thus the reading are valid during the emulation and break state. Connect the AVCC pin (the VCC pin for the A/D) and the VREF (reference voltage pin) to the A/D and D/A conversion power supply and the reference power supply on the user's system.

NOTE:

AVCC and VREF pins must be connected to VCC pins when the A/D and D/A converters are not used. The conversion precision on the emulator is lower than that of the actual MCU. This is mainly due to the extra resistance and capacitance introduced in the cabling and printed circuit boards.

5.13 Refresh Controller

The refresh controller operates continuously, even during the command input wait state. However, the emulator does not support battery back up mode. When entering battery, back up mode, the data in DRAM may thus be lost.

Section 6. Differences between H8/300H MCU and Emulator

6.1 Power up and reset

When the emulator initialises the system or resets the Microcomputer as a result of a command such as switching of the clock, or when the reset command is used, note that the general-purpose registers and part of the control registers are initialized.

Table 5-1 Differences Between H8 MCU and Emulator

Status	Register	Emulator	H8 MCU
Emulator Initialization (Power on)	PC	Power on reset vector value	Power on reset vector value
	ER0 to ER6	H'00000000	Undefined
	ER7 (SP)	H'0FFF10	Undefined
	CCR	The I mask is set to 1 and the other bits are undefined (B'1 xxx xxxx)	The I mask is set to 1 and the other bits are undefined (B'1xxx xxxx)
Emulator Initialization (Reset CPU command)	PC	Power on reset vector value	Power on reset vector value
	ER0 to ER6	Undefined	Undefined
	ER7 (SP)	Undefined	Undefined
	CCR	The I mask is set to 1 and the other bits are undefined (B'1 xxx xxxx)	The I mask is set to 1 and the other bits are undefined (B'1 xxx xxxx)

6.2 User Interface

User may have to adjust the target system as follows:

- The emulator's user system interface is provided with pull-up resistors and/or a buffer, causing the signals to be delayed slightly.
- The pull-up resistors will change high-impedance signals to high level signals.
- The Analog to Digital conversion's resolution will have a slight degradation.
- Load capacitance of the emulator as compared to the actual chip will be larger.
- Crystal oscillator can only used if the actual footprint user cable is used (there is an oscillating circuitry built on-board). If user uses the direct method of connecting to the target system, the actual clock signal has to be connected to the EXTAL pin (**NOTE:** there is no XTAL pin).
- If user uses the direct connection method, the signal (CABLE_IN_N) has to be connected to ground.

Please refer to section 6 for the details of user interface circuitry

Section 7. User System Interface

The user target system is connected to the emulator via the interface cable. Interface circuitry is inserted in between to remove noise and protect the emulator. When connecting the user target system to the emulator, user has to consider the adjustment of the user system hardware. Compensation for FAN-IN, FAN-OUT and propagation delays are necessary. In general, the one-way propagation delay of the cabling is about 4ns. However, the user clock signal is delayed for approximately 25 ns.

All signals are connected to the MCU with no buffering with the exception of

- NMI
- RESET
- STBY
- MD[2..0]
- EXTAL

The emulator does not detect the following signals

- FWE
- OSC1
- OSC2
- CVCC
- XTAL

All port signals are pulled up by a 47K ohm resistor with the exception of all analogue pins (Ports 7).

All signals satisfy the MCU AC timing specification.

Except

Parameter	H8/300H	Emulator
Tr _{DS}	20ns (min)	30 ns (min)

The illustration of the user interface circuitry are as follows.

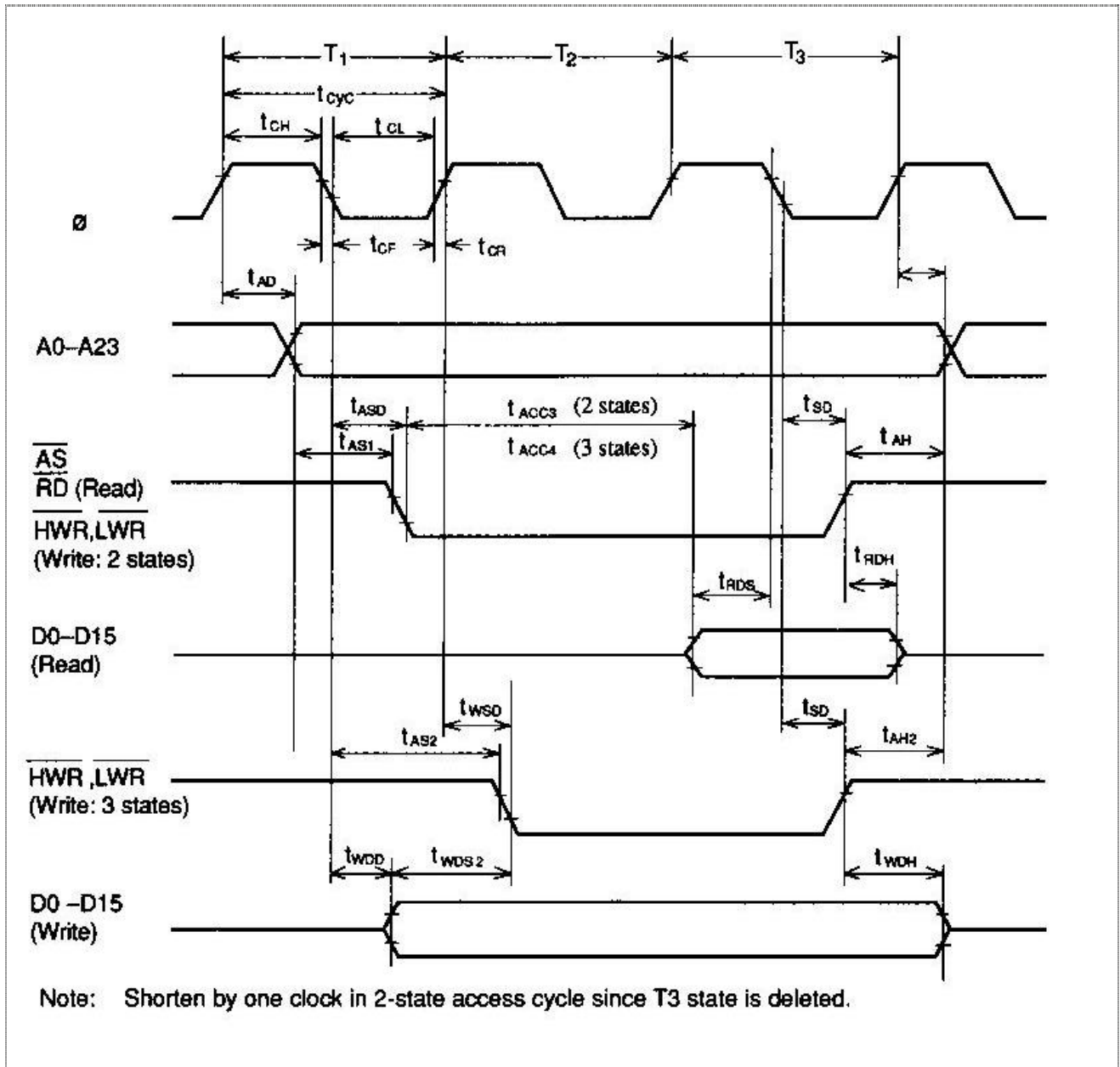


Figure 80 Basic Bus Cycle Timing in Expanded mode

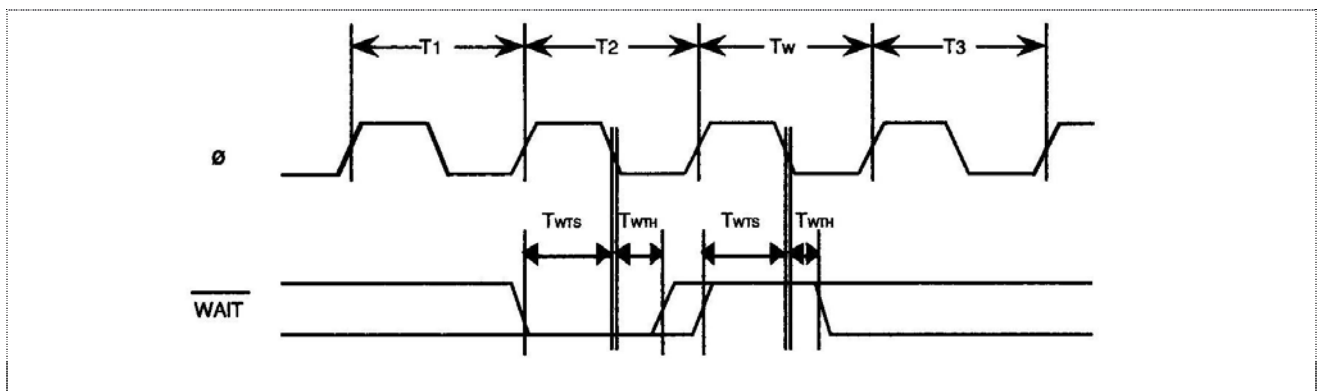


Figure 81 Basic Bus Cycle Timing in Expanded mode

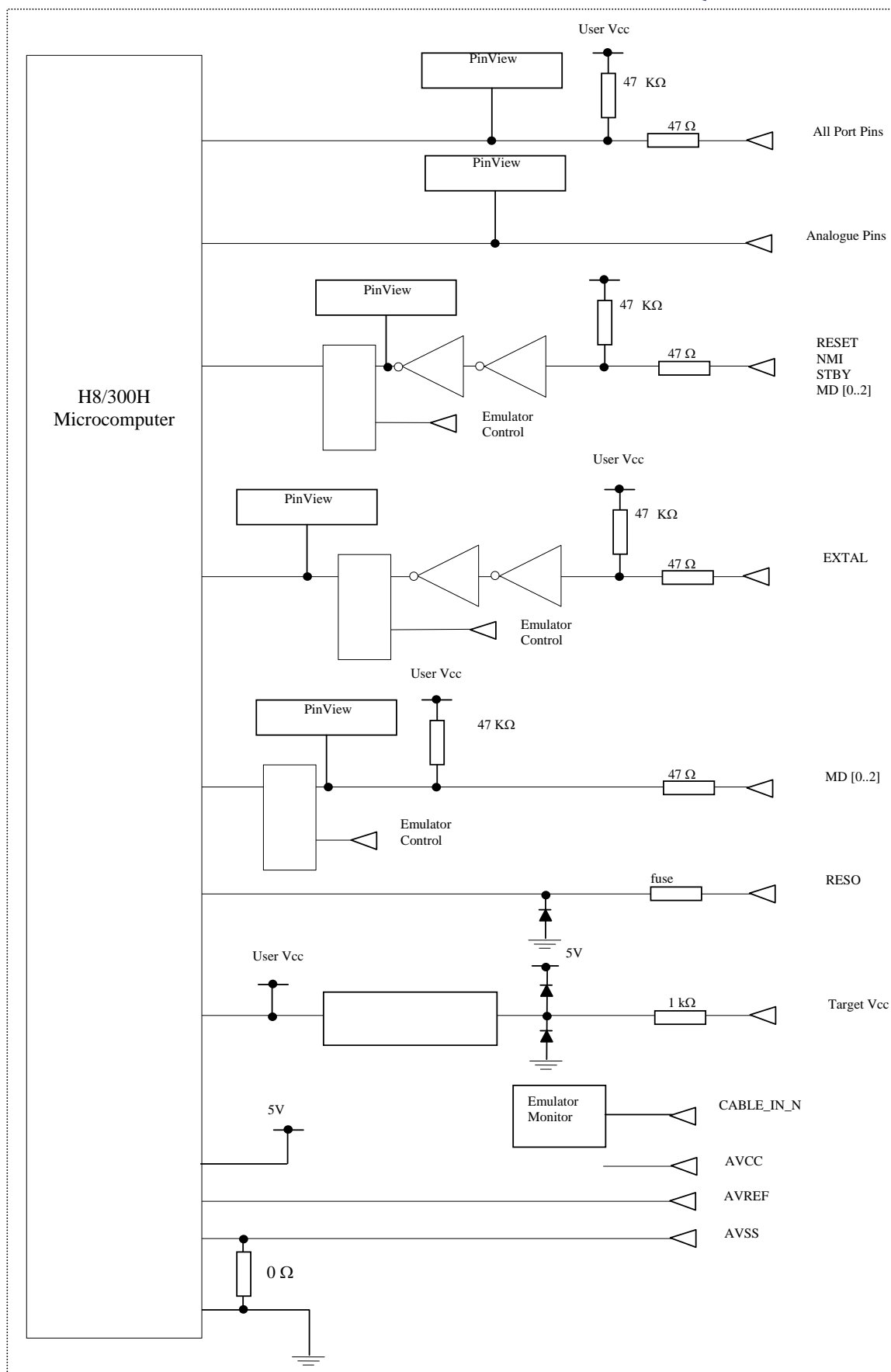


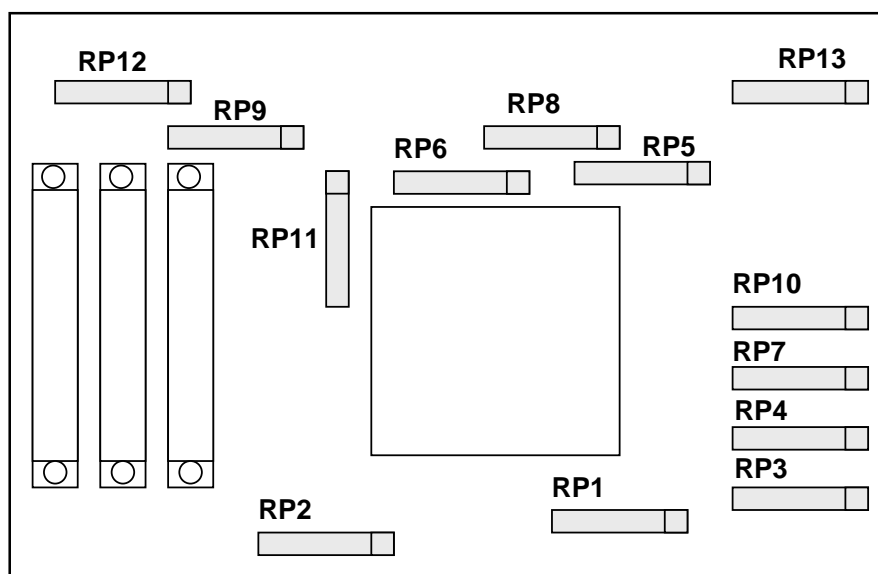
Figure 82 Interfacing Circuitry

The pull up resistor location is specified as follow:

Port	Resistor Pack
1	3
2	4
3	7
4	10
5	1
6	5
8	8
9	11
A	2
B	6
C	9

The resistor packs connection:

Pin	Port Signal Numbering
1	User Vcc
2	Px0
3	Px 1
4	Px 2
5	Px 3
6	Px 4
7	Px 5
8	Px 6
9	Px 7



Bottom View of Emulator

Figure 83 Resistor Pack Position

Section 8. Tutorial

The following describes a sample debugging session, designed to introduce the main features of the CE300H-H8/300H emulator used in conjunction with the High-performance Embedded Workshop(HEW) software.

The tutorial is designed to run in the CE300H emulator's resident memory so that it can be used without connecting the CE300H emulator to any external user system.

User has to setup the tool as stated in section 1 and 2 before the tutorial can be begun.

8.1 Overview

The tutorial is written in C source code.

This program is an infinite loop that sort elements based on NAME in the alphabetical order, and AGE and ID in the numeral ascending order.

The tutorial is provided on the installation disk as the project file - [Tutorial_3052Mode7].
The generated output, in

[install directory\Tools\Renesas\DebugComp\Platform\Emulator\CE3052\Tutorial_3052Mode7]

are: - Motorola S-Record - CE3052.mot and
 - Elf/DWARF2 - CE3052.abs

8.2 Tutorial Program Listing

The Tutorial can be classified in four main parts;

i. Header

```
#include "machine.h"
#include "string.h"
```

ii. Definition and declaration for the constants, structures, and functions:

```
#define NAME (short)0
#define AGE (short)1
#define ID (short)2
#define LENGTH 8

struct namelist {
    char name[LENGTH];
    short age;
    long idcode;
};

struct namelist section1[] = {
    "Naoko", 17, 1234,
    "Midori", 22, 8888,
    "Rie", 19, 7777,
    "Eri", 20, 9999,
    "Kyoko", 26, 3333,
    "", 0, 0
};

int count;

void sort();
```

iii. Main routine

```
main( )
{
    count = 0;
    for ( ; ; ){
        sort(section1, NAME);
        count++;
        sort(section1, AGE);
        count++;
        sort(section1, ID);
        count++;
    }
}
```

iv. Subroutine

```

void sort(list, key)
struct namelist list[];
short key;
{
    short i,j,k;
    long min;
    char *name;
    struct namelist worklist;

    switch(key){
        case NAME :
            for (i = 0 ; *list[i].name != 0 ; i++){
                name = list[i].name;
                k = i;
                for (j = i+1 ; *list[j].name != 0 ; j++){
                    if (strcmp(list[j].name , name) < 0){
                        name = list[j].name;
                        k = j;
                    }
                }
                worklist = list[i];
                list[i] = list[k];
                list[k] = worklist;
            }
            break;

        case AGE :
            for (i = 0 ; list[i].age != 0 ; i++){
                min = list[i].age;
                k = i;
                for (j = i+1 ; list[j].age != 0 ; j++){
                    if (list[j].age < min){
                        min = list[j].age;
                        k = j;
                    }
                }
                worklist = list[i];
                list[i] = list[k];
                list[k] = worklist;
            }
            break;

        case ID :
            for (i = 0 ; list[i].idcode != 0 ; i++){
                min = list[i].idcode;
                k = i;
                for (j = i+1 ; list[j].idcode != 0 ; j++){
                    if (list[j].idcode < min){
                        min = list[j].idcode;
                        k = j;
                    }
                }
                worklist = list[i];
                list[i] = list[k];
                list[k] = worklist;
            }
            break;
    }
}

```

8.3 Tutorial Setup

The execution and setup of HEW is detailed in section 3.
Thus these steps will not be fully illustrated in this section.

1. Open tutorial workspace in
[install directory\Tools\Renesas\DebugComp\Platform\Emulator\CE3052\Tutorial_3052Mode7.hws].

NOTE :

On a first loading of the tutorial, a dialog box prompting the move of workspace from previous installed directory is displayed. Please click [YES] and the workspace would be configured to the current installed directory permanently

2. Activate [Option/Debug setting...],
 - a. Select [CE300H-H8/3052 Emulator] as target platform
 - b. Select [Elf/Dwarf2] as debug format
 - c. Select [install directory\Tutorial_3052Mode7\Tutorial_3052Mode7\Debug_CE300H-H8_3052_Emulator\CE3052.abs] as the download module.

NOTE:

Since the output module name is not equivalent to the Project name or workspace name. To use the relative path, user can click on [Configuration directoiry] and key in [\\CE3052.abs].

3. Configure Platform
 - H8/3052BF, Mode 7, 10MHz (all at default setting)
4. Download the CE3052.abs via [Debug/Download Modules/...]

Now the platform is ready for the tutorial.

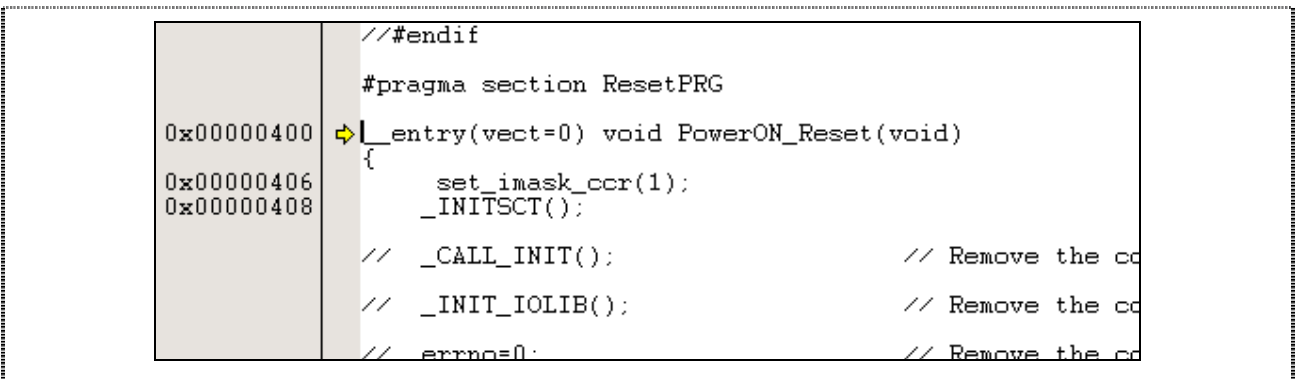
8.4 Displaying the Program Listing

HEW allows user to debug a program at C-Source and Assembly level.

There are three methods to view the main source file.

- i. File/open...
- ii. Double click on file listed in the workspace window
- iii. [Debug/ Reset CPU], “resetprg.c” file will be opened.

NOTE: HEW will prompt for the resetprg.c file if this is not in the current directory.



```

//#endif
#pragma section ResetPRG
0x00000400  → _entry(vect=0) void PowerON_Reset(void)
0x00000406  {
0x00000408      set_imask_ccr(1);
                _INIT_SCT();
                // _CALL_INIT();                // Remove the cc
                // _INIT_IOLIB();                // Remove the cc
                // errno=0;                // Remove the cc

```

Figure 84 Resetprg.c file after RESET CPU

8.5 Setting a PC Breakpoints

The simplest debugging aid is the program breakpoint, it causes execution to stop when a particular point in the program is reached. You can then examine the state of the MCU and memory at that point in the program.

8.5.1 Setting a Program Count (PC) Breakpoint

The program window provides a very simple way of setting a program breakpoint. For example, set a breakpoint at main () function entry point as follows:

- Right-click in the [Editor Window] and select [Toggle Breakpoint] at the popup menu at the desired line, or
- Double click on the event column.

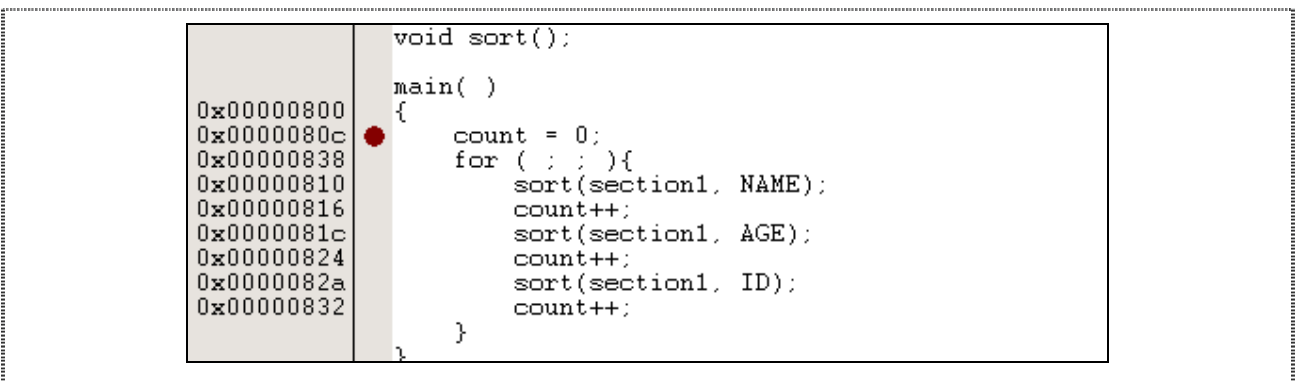


Figure 85 Setting a Breakpoint

The dot will be displayed there to indicate that a program breakpoint is set at that address.

8.6 Executing the Program

To run the program from reset:

- Click on [Debug/Reset Go], or
- [Debug/ Go] (Since reset has been executed)

The program will execute up to the breakpoint that has been set, and a yellow arrow will indicate the new PC location.

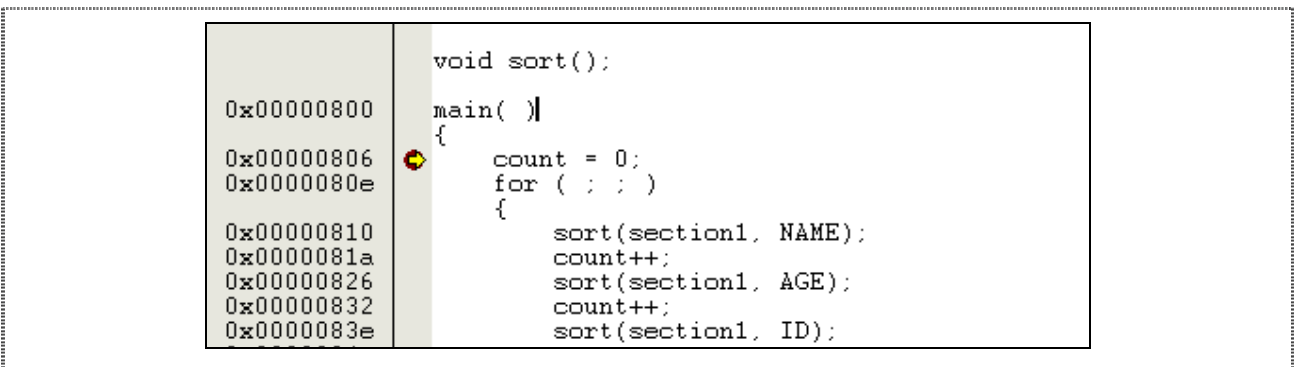


Figure 86 Program Break

To view the status of the emulation

- Click on [View/ CPU/ Status] – [Platform]

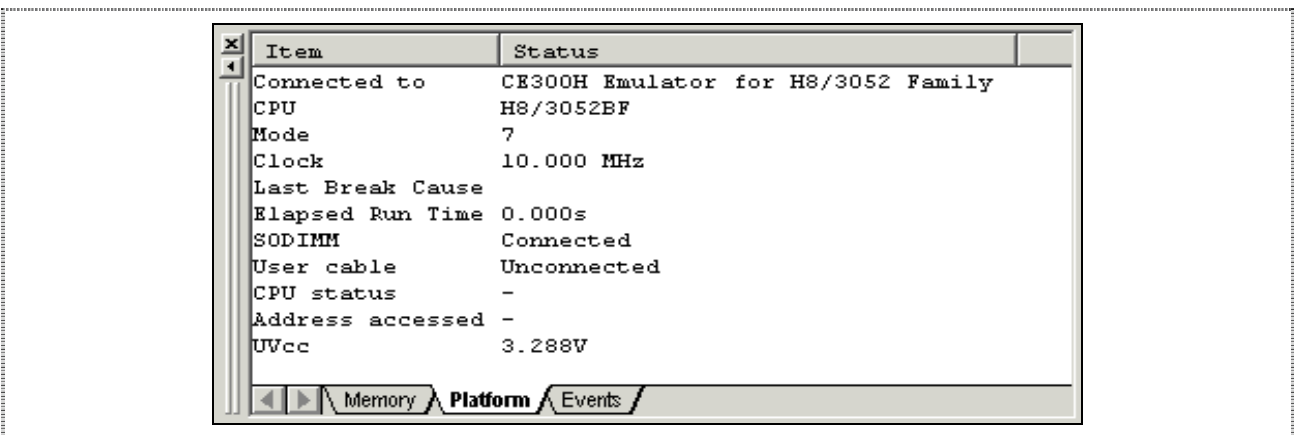


Figure 87 Platform page of Status Window

The status show

- Last break cause – PC break
- Elapsed Run Time...

8.7 Reviewing the Breakpoints

The list of all the breakpoints set in the program can be viewed in the Eventpoints window.

- Choose [View/Code/ Eventpoints]

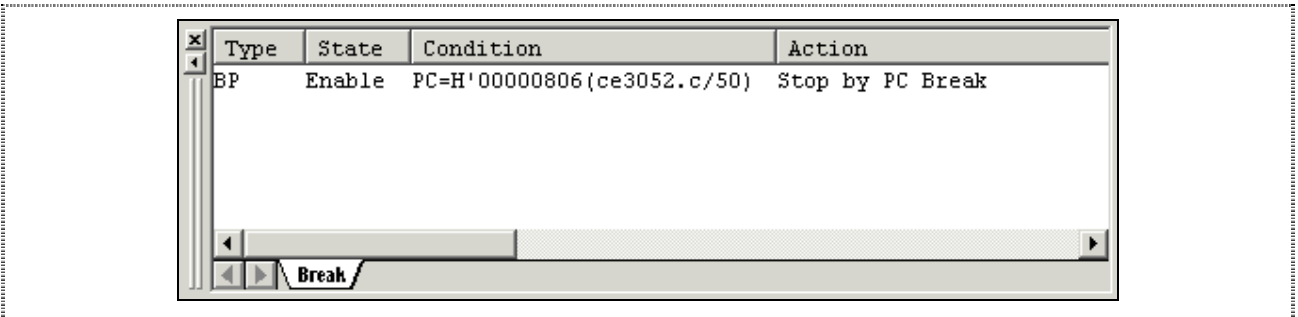


Figure 88 Eventpoints Window

The Eventpoints window also allows user to perform the following:

- Define new breakpoints
- Delete existing breakpoints
- Disable existing breakpoints

Perform a right-mouse click within the Eventpoints-window to show the available breakpoint functions:

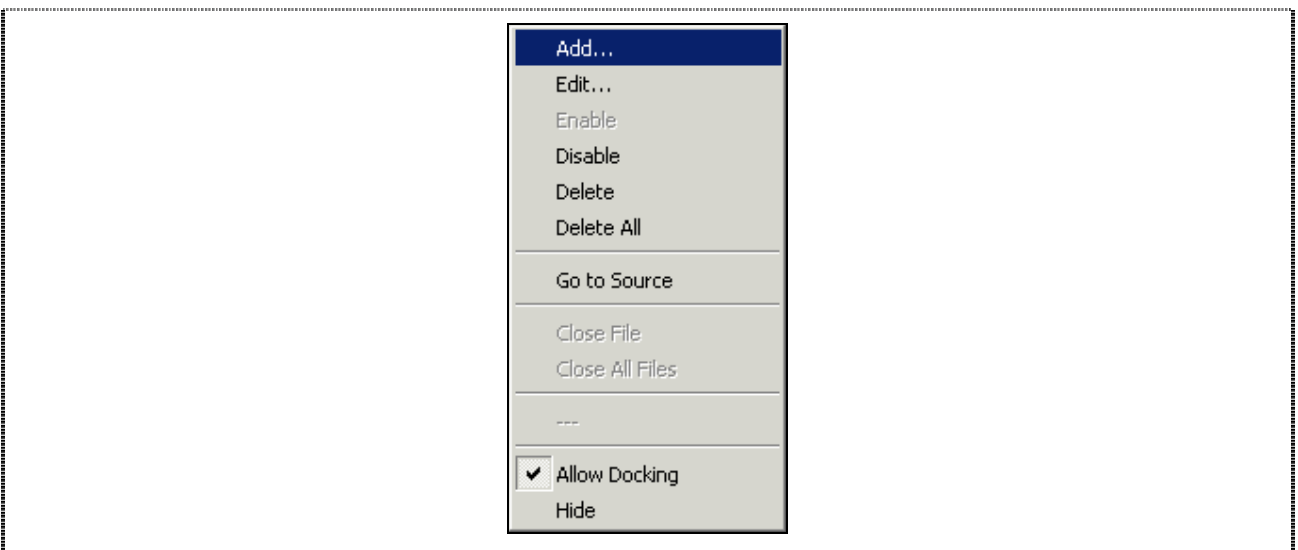


Figure 89 Popup menus in Eventpoints Window

8.8 Examining MCU Registers

While the program is halted, you can examine the contents of the H8/300H Series MCU registers. These are displayed in the Registers Window.

- Click on [View/CPU/Register]

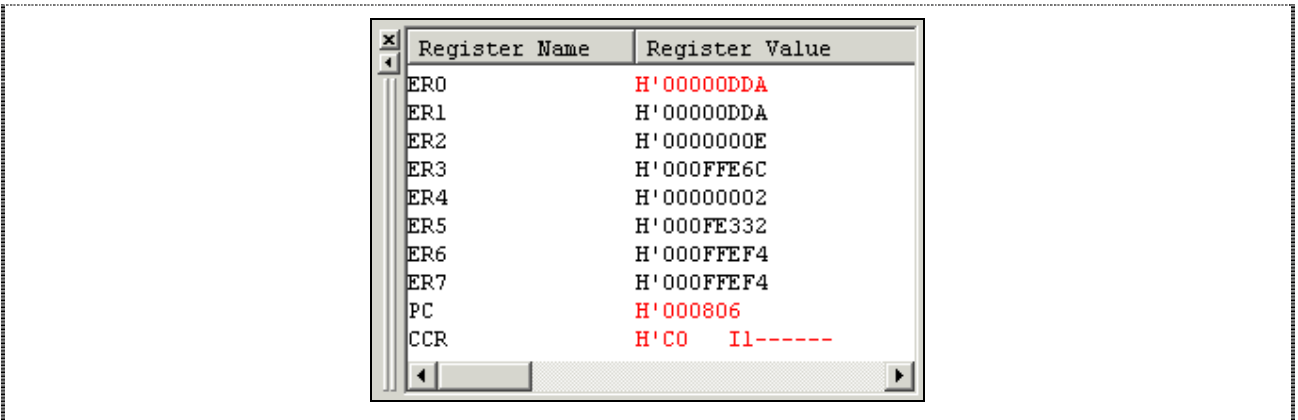


Figure 90 CPU Registers Window

The registers' values can be changed from the Registers window. For example, change the value of the PC:

- Double-click on PC in the Registers window

The Register-PC dialog allows you to edit the value.

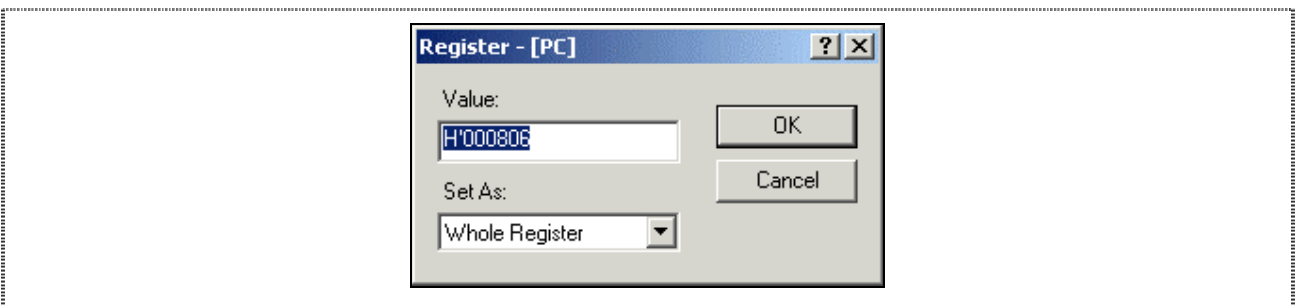


Figure 91 Changing Register Value

8.9 Examining Memory and Variables

The behavior of a program can be monitored by examining the contents of an area of memory, or by displaying the values of variables used in the program.

8.9.1 Viewing Memory

The contents of a block of memory can be viewed in the Memory Window.

For example, to view memory:

- Click on [View/CPU/Memory]
- Enter the start and end address, or symbol name and relative range

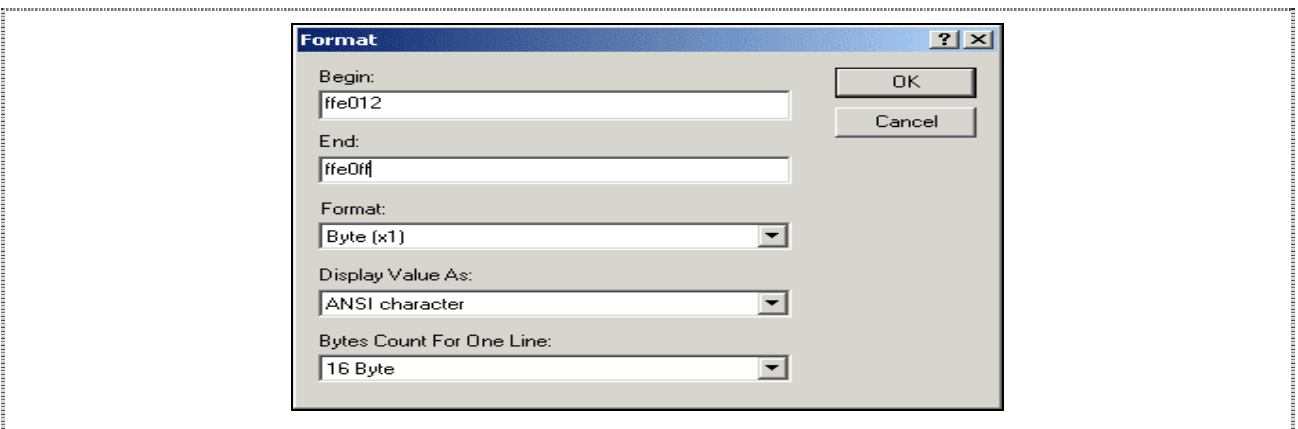


Figure 92 Open Memory Window

- Click OK to open the Memory window showing the specified memory area.

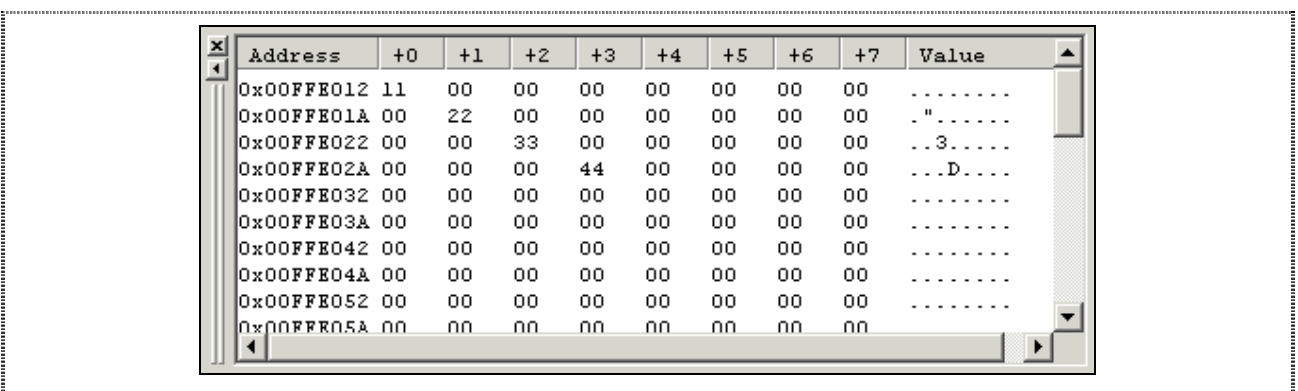


Figure 93 Memory Window

8.9.2 Watching Variables

It is useful to be able to watch the values of variables as the program is being stepped.

- Right click on the variables (section1 in main source file) and select [Instant Watch]

The Instant Watch dialog will be displayed:

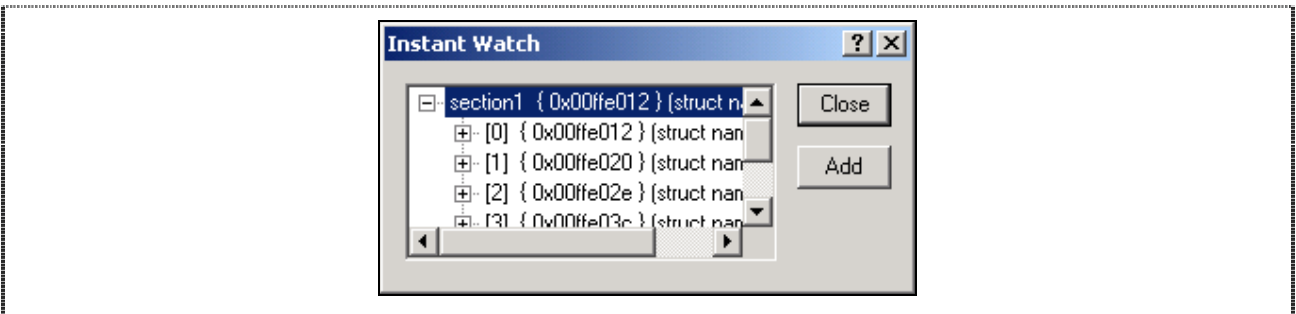


Figure 94 Instant Watch dialog

- Click [Add] button to add the variable to the Watch Window.

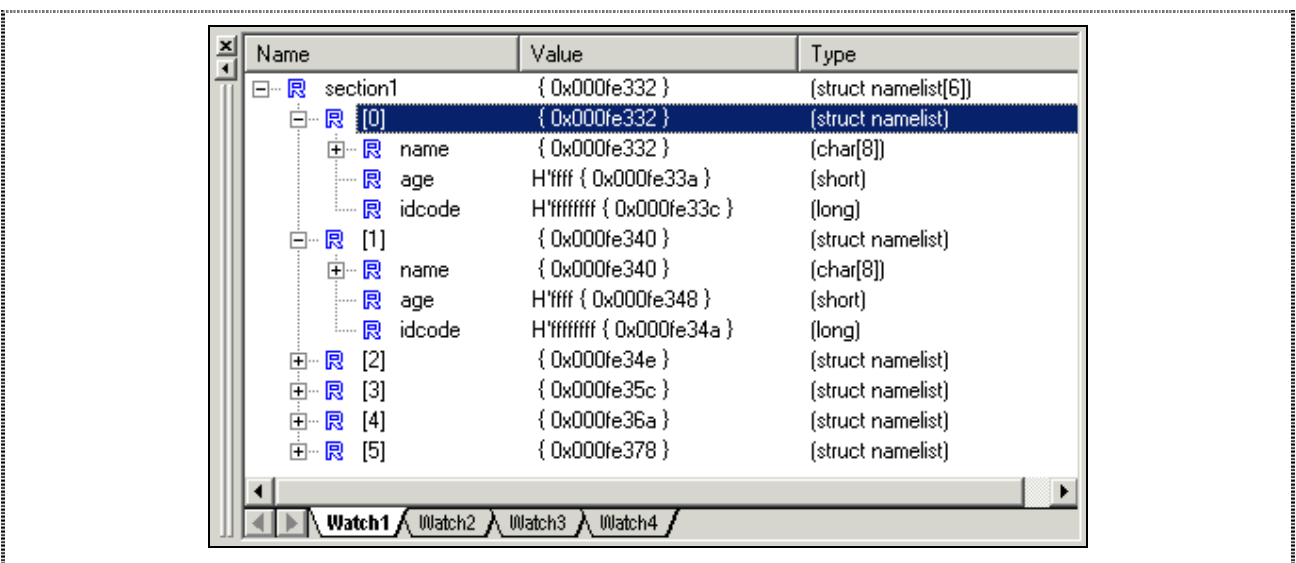


Figure 95 Watch Window

The + symbol allow the expanded view of the variables.

A variable can be added to the Watch Window by specifying its name. Use this method to add a Watch on the variable [count] as follows:

- Right click within the Watch window and choose [Add Watch...] from the pop-up menu.

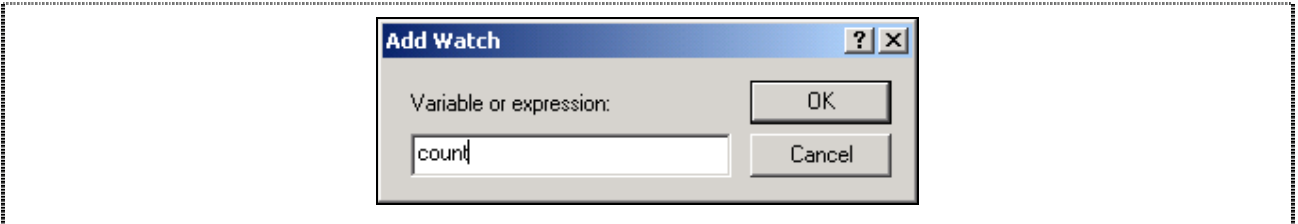


Figure 96 Add Watch Dialog

- Type the variable [count] and click OK.

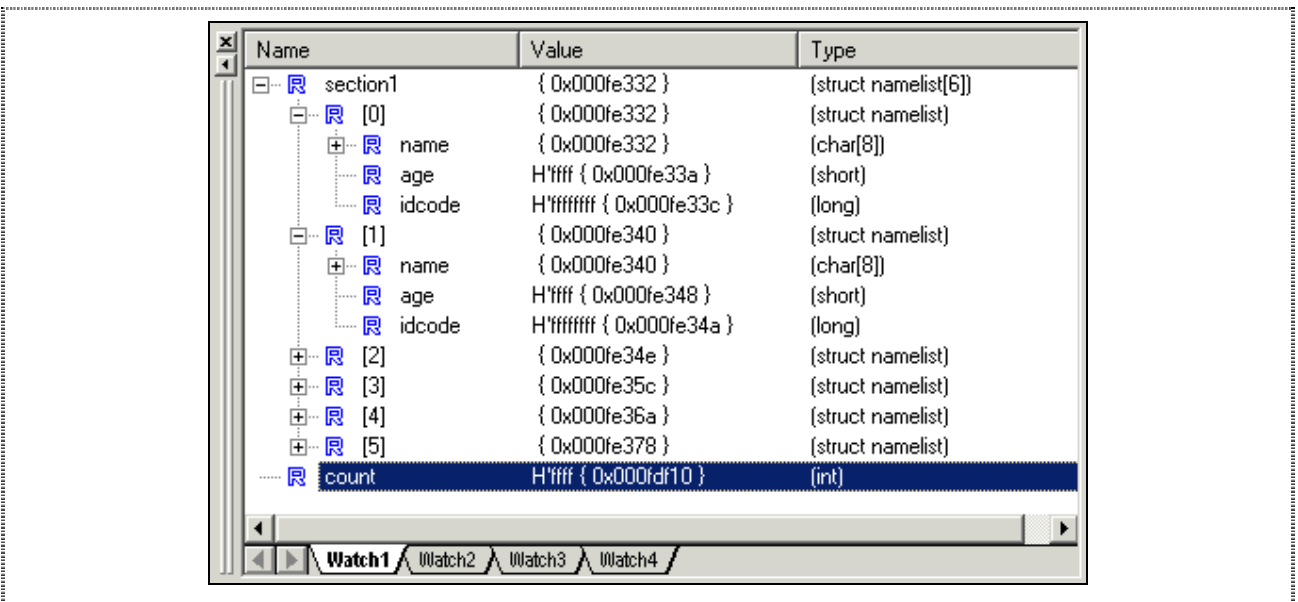


Figure 97 Watch Window

8.10 Stepping Through a Program

There is basically three types of step functions;

8.10.1 Step In (F11)

Activate [Debug/Step In] in C source level until the PC execute(step) into the sort routine (5 steps))

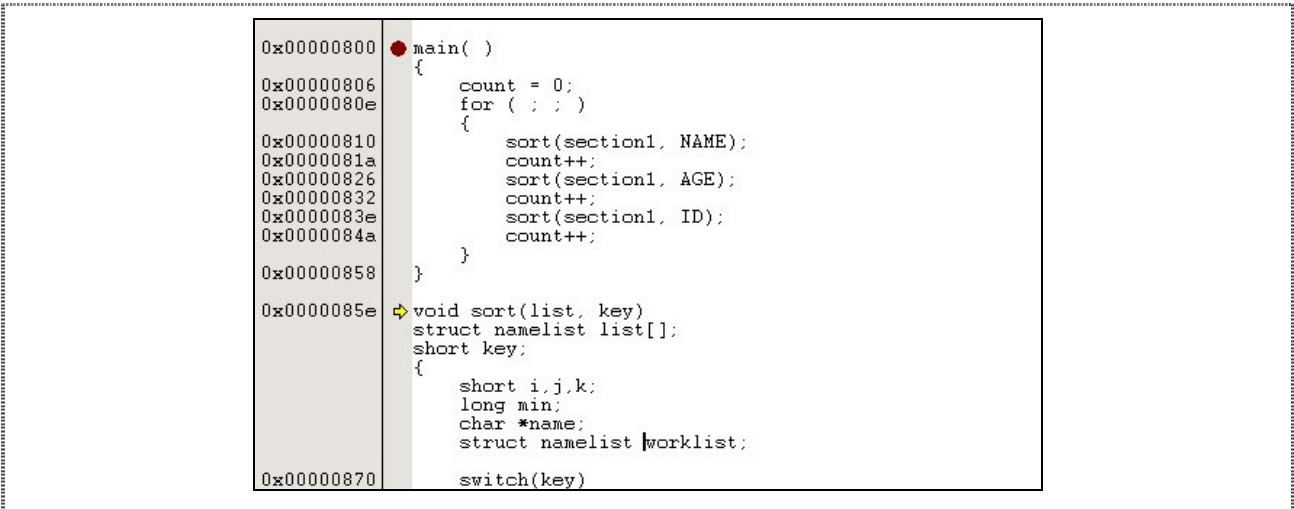


Figure 98 Step In

The program stops at the first instruction of the sort routine.

8.10.2 Step Out (Shift F11)

Activate [Debug/Step Out] to execute the code until it is out of the current routine (1 step).

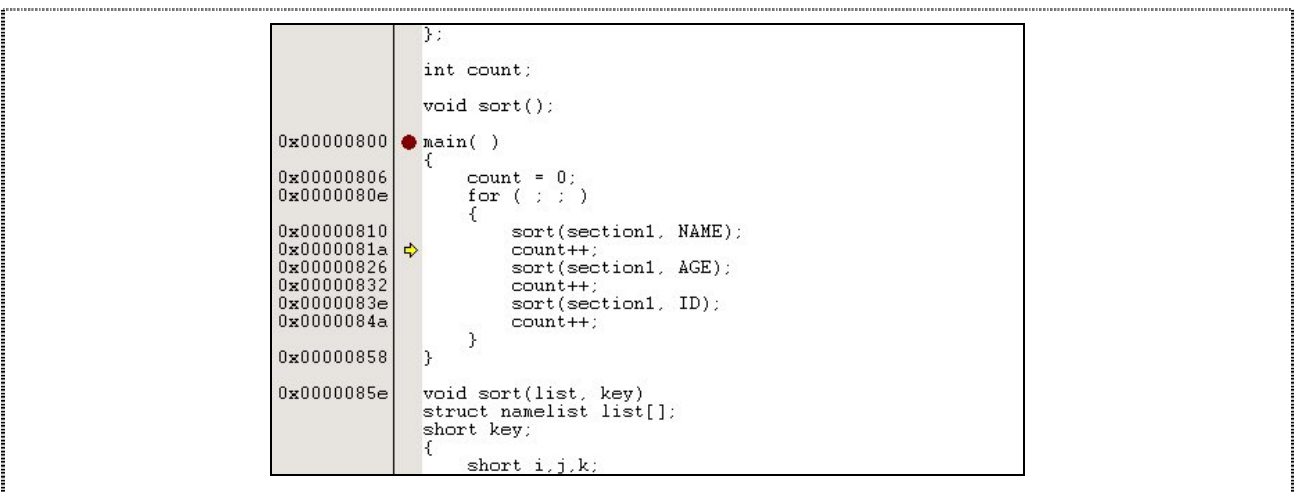


Figure 99 Step Over

The program stops in the main routine, a instruction just after the first sort routine

8.10.3 Step Over (F10)

Activate [Debug/Step Over] to execute the code (2 steps).

- It can be observed that the sort routine is being executed (step) over, without entering into the routine

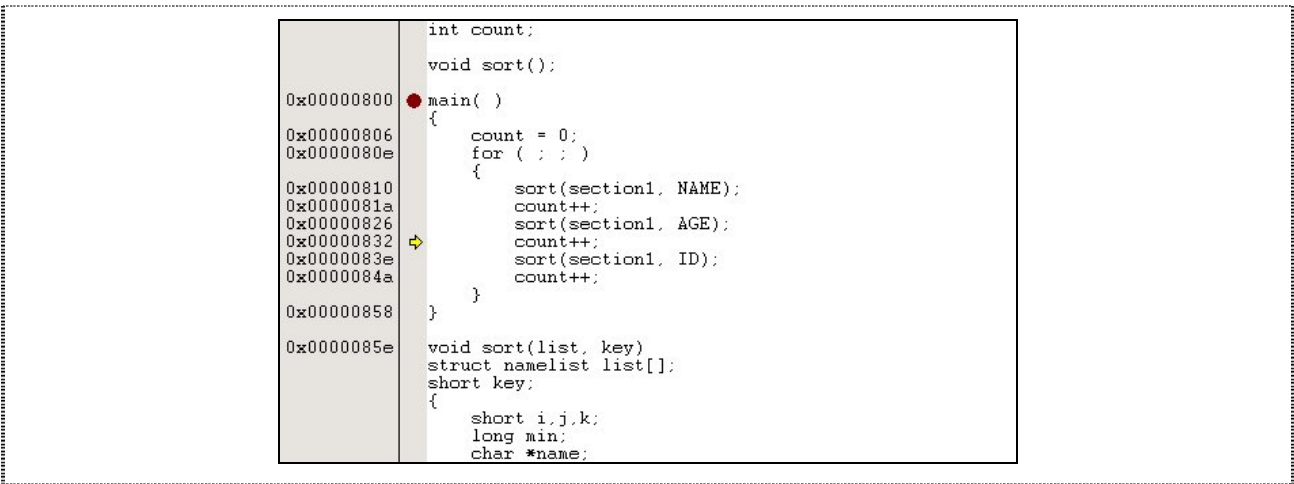


Figure 100 Step Over

The program stops at the instruction after the secondsort routine.

8.11 Using the Hardware Breakpoint

The PFG Breakpoint allows user to halt the program based on several conditions matching at the same time (such as address, data, ...).

8.11.1 Defining a PFG Breakpoint

- Click [View/ Code/ Breakpoint Setting...]
- Right Click at the breakpoint window and select [Add] to define a new breakpoint.
- Add the PFG break at address = H'870 (within the sort routine)

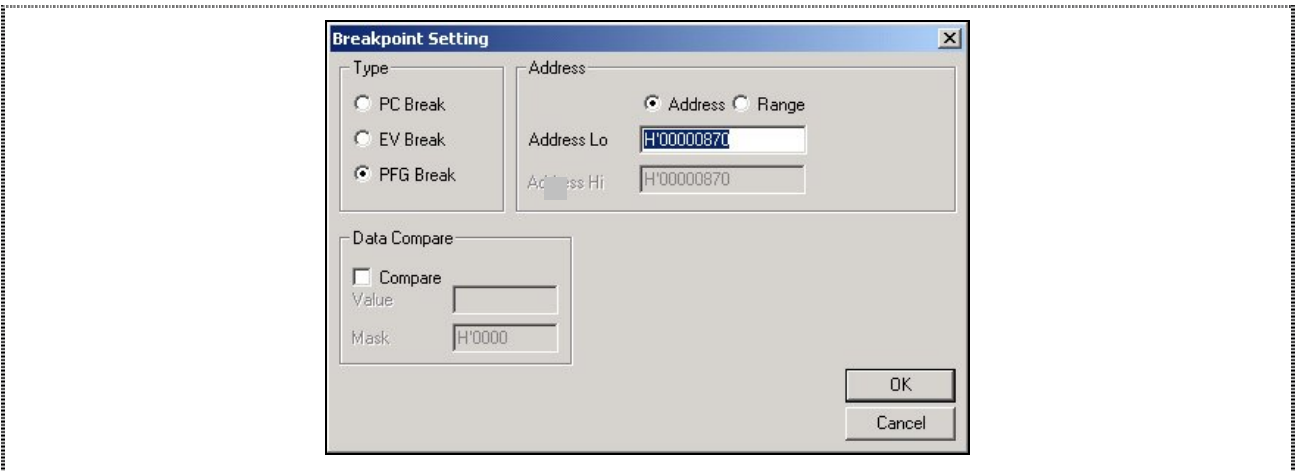


Figure 101 Breakpoint Setting Dialog

- The Eventpoints window show the new breakpoint that have been defined.

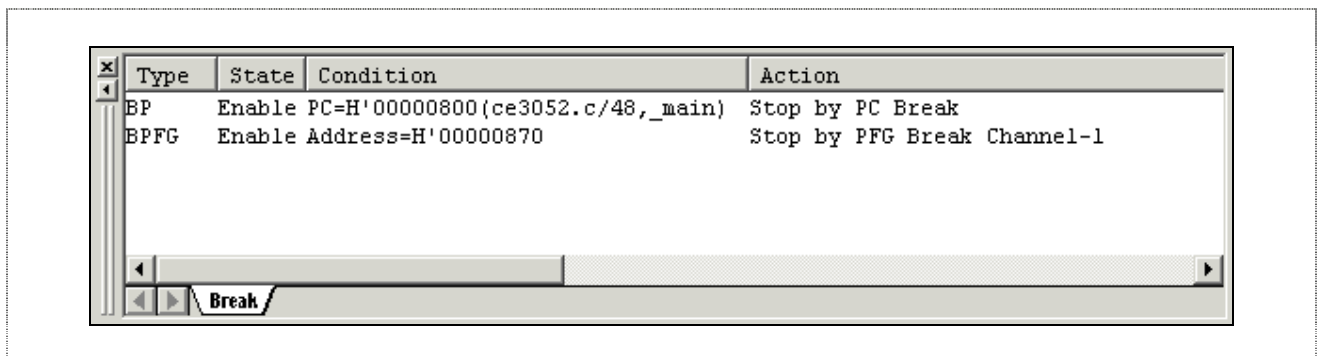


Figure 102 Eventpoints window

- Run the program from the current position [Debug/ Go]

Execution will stop as below:

```

Disassembly
62: void sort(list, key)
    _sort 01006DF6      MOV.L    ER6, @-ER7
00000862 0FF6      MOV.L    ER7, ER6
00000864 5E000C64 JSR      @$fp_regsv$3:24
00000868 7937001C SUB.W    #H'001C, R7
0000086C 0F85      MOV.L    ER0, ER5
0000086E 0D14      MOV.W    R1, R4
63: struct namelist list[];
64: short key;
65: {
66:     short i, j, k;
67:     long min;
68:     char *name;
69:     struct namelist worklist;
70:
71:     switch(key)
00000870 0D40      MOV.W    R4, R0
00000872 0C00      MOV.B    R0H, R0H
00000874 58600358 BNE      @H'0BD0:16
00000878 A800      CMP.B    #H'00, R0L
0000087A 4710      BEQ      @H'088C:8
0000087C A801      CMP.B    #H'01, R0L
0000087E 58700114 BEQ      @H'0996:16
00000882 A802      CMP.B    #H'02, R0L

```

Figure 103 PFG Break

The program will halt at the next assembly code location.

The system status window will display [Last Break Cause = PFG Break];

8.12 Watching Local Variables

The localised variables within a function can be viewed using the Local Variables Window.

- Open the Local window by choosing [View/Local]

Note:

There will be no display of local variables in the Local Window if there is no local variable declared or local functions have not yet been entered. In another words, user target program execution should halt within the specific function (with the local variables) in order to examine its local variables.

In this tutorial, the local variables of function sort() will be shown in Locals Window, when the execution halts within the function sort(),

- Click on the [+] symbol to display the individual elements of the arrays.

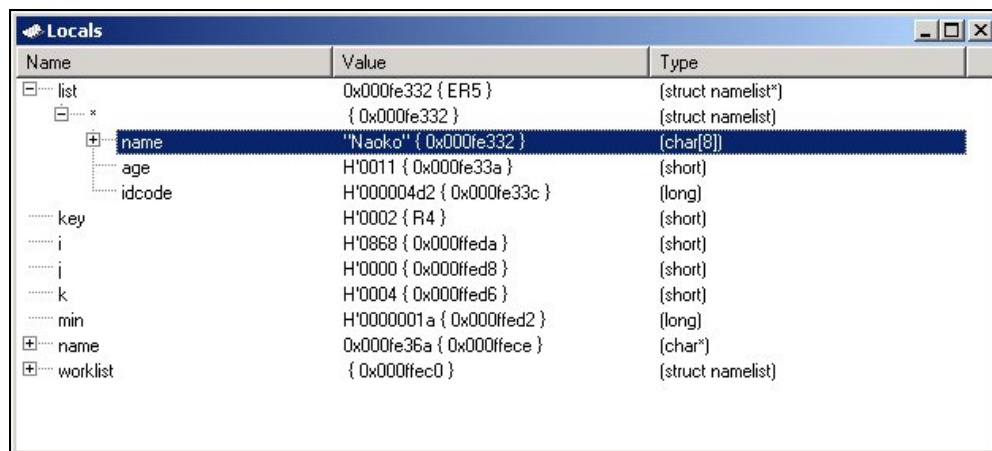


Figure 104 Locals Window

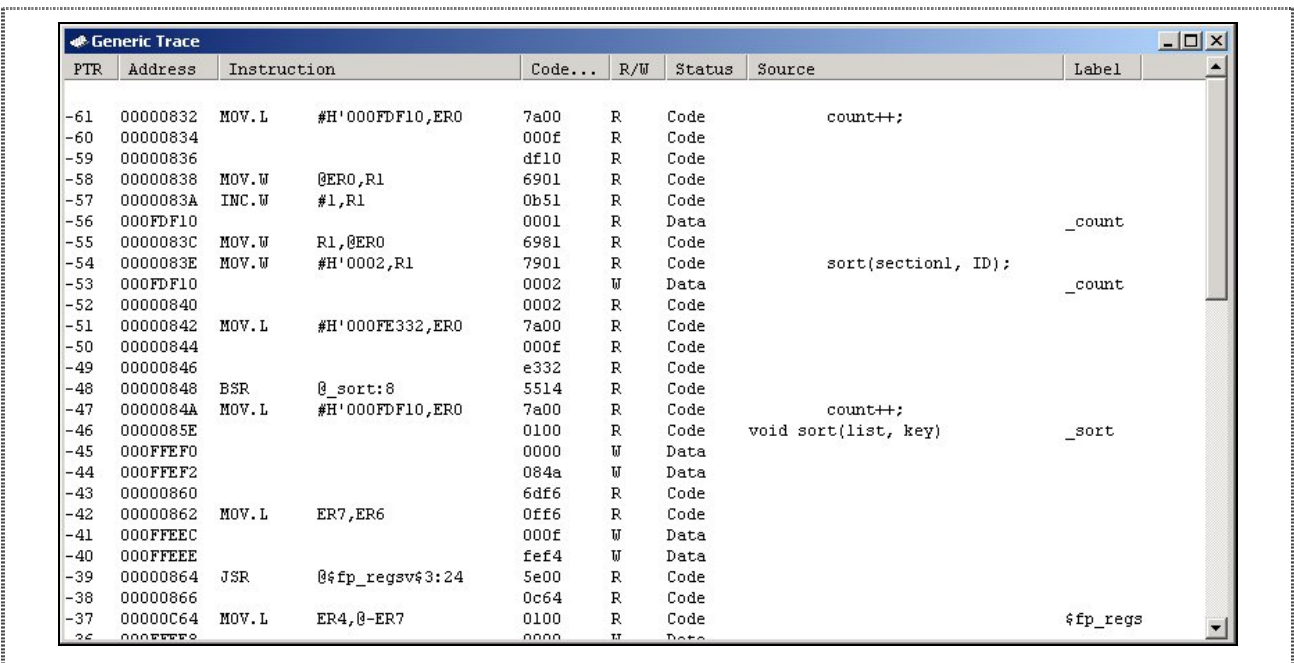
8.13 Using the Trace Buffer

The trace buffer allows the review of the executed code till abreak condition is matched.

The trace buffer will display the captured bus cycle and decode the data in the trace window.

- Open the Trace Window by [View/ Code/ Trace]

The Trace window is displayed, as shown in the following figure.



PTR	Address	Instruction	Code...	R/W	Status	Source	Label
-61	00000832	MOV.L #H'000FDF10,ER0	7a00	R	Code	count++;	
-60	00000834		000f	R	Code		
-59	00000836		df10	R	Code		
-58	00000838	MOV.W @ER0,R1	6901	R	Code		
-57	0000083A	INC.W #1,R1	0b51	R	Code		
-56	000FDF10		0001	R	Data		_count
-55	0000083C	MOV.W R1,@ER0	6981	R	Code		
-54	0000083E	MOV.W #H'0002,R1	7901	R	Code	sort(section1, ID);	
-53	000FDF10		0002	W	Data		_count
-52	00000840		0002	R	Code		
-51	00000842	MOV.L #H'000FE332,ER0	7a00	R	Code		
-50	00000844		000f	R	Code		
-49	00000846		e332	R	Code		
-48	00000848	BSR @_sort:8	5514	R	Code		
-47	0000084A	MOV.L #H'000FDF10,ER0	7a00	R	Code	count++;	
-46	0000085E		0100	R	Code	void sort(list, key)	_sort
-45	000FFEFO		0000	W	Data		
-44	000FFEF2		084a	W	Data		
-43	00000860		6df6	R	Code		
-42	00000862	MOV.L ER7,ER6	0ff6	R	Code		
-41	000FFEEC		000f	W	Data		
-40	000FFEEE		fef4	W	Data		
-39	00000864	JSR @fp_regsv\$3:24	5e00	R	Code		
-38	00000866		0c64	R	Code		
-37	00000C64	MOV.L ER4,@-ER7	0100	R	Code		\$fp_regs
-36	000FFEE8		0000	W	Data		

Figure 105 Trace Window

Adjust the width of each column by dragging the column dividers on either side of the labels just below the title bar.

8.14 Save the Session

Before exiting, it is good practice to save the session so that debugging work can be resumed instantly with the same configuration at the next debugging session.

Section 9. Diagnostic

The CE300H-H8/3052 emulator is designed to have all possible protective measures, but it is still subjected to damage by user system or other unforeseen means. The CE300H-H8/3052 emulator has two built-in test modes: Standalone Test and HEW DEBUGGER Diagnostic. Both tests have the same test procedures, in which all the internal functions and pins are tested thoroughly.

9.1 Standalone Self Test

The test is implemented to allow user to have a quick check of the hardware, before linking it to the PC.

This will help user to isolate the cause of the PC communication problem. The testing steps are:

- Power on the CE300H-H8/3052 emulator (the POWER LED lights up in RED).
- Ensure the USB cable is not connected between the PC and CE300H-H8/3052 emulator.
- Wait for 3 minutes.
- The emulator will enter Self Test mode (POWER LED changes from RED to ORANGE colour). After the tests are completed (about 2 minutes), the POWER LED will start to blink, whereas the adjacent RUN LED will light up in RED if the self test fails. However, if the self test passes, the RUN LED will not light up.

NOTE:

During the self test, the RUN LED will be turned on to GREEN occasionally. This merely indicates that the emulator is in RUN mode.

9.2 HEW DEBUGGER Diagnostic Test

An alternative is to use the built-in test in the HEW DEBUGGER with the following steps :

1. Open any workspace which can link to CE300H emulator, select View\ TCL Toolkit

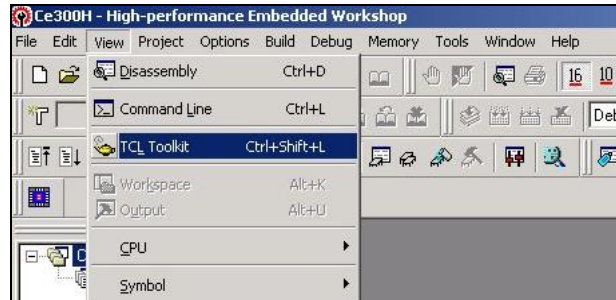


Figure 106 Toolkit selection

2. In the Console window, select File \ Source...
3. Select [c:\installed_dir\Tool\Renesas\DebugComp\Platform\Emulator\CE3052\Diagnostic_CE300H.tcl]
4. Click on Diagnose button, and the test will carry out.
5. After the completion of tests, user must re-initialize the platform.

In this window, users will be able to know which tests have been passed.

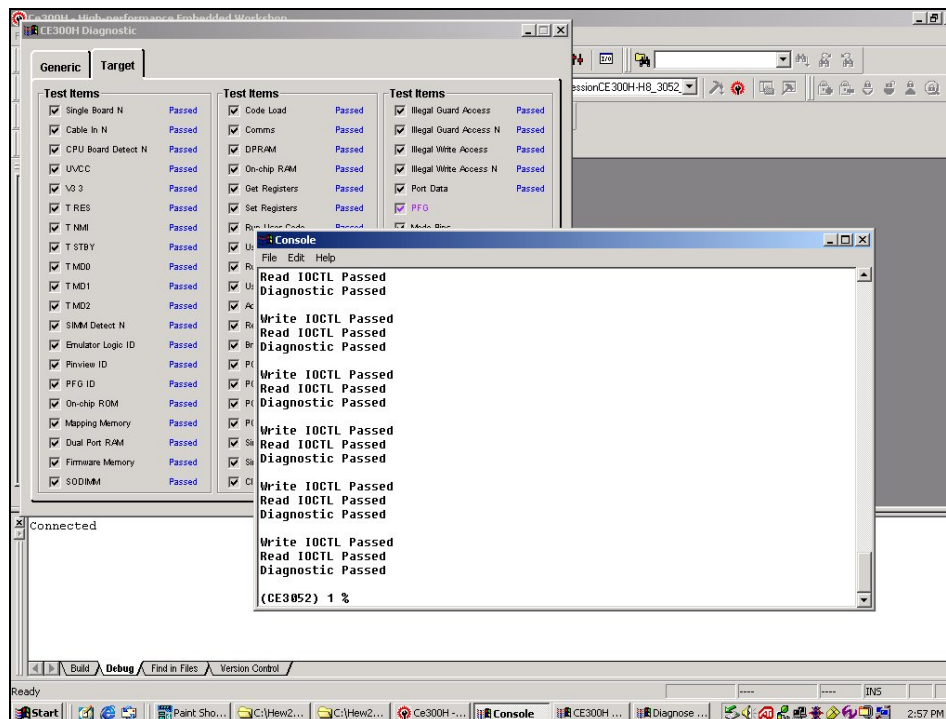


Figure 107 Diagnostic Window

NOTE: The diagnostic test will clear all memory contents.

Section 10. Emulator Upgrade

The emulator can be upgraded to support new features or devices, without sending the tool back to the service centre.

At HEW startup, the Debugger will check on the OS & Logic revision of the emulator. If it is an older version, the debugger will request the user to upgrade the emulator.

User are required to activate the CE Programmer to upgrade the system. The CE Programmer is located in the [CE Programmer] sub-directory of the directory where HEW DEBUGGER is installed.

[Installed dir\Tools\Renesas\DebugComp\Platform\Emulator\CE3052\CE Programmer]

1. Simply click on the [CE Programmer.exe] to activate the programmer (exit HEW)
2. (Re)Power up the emulator and connect the USB cable (This is to ensure that the emulator is in the correct state before the upgrading)

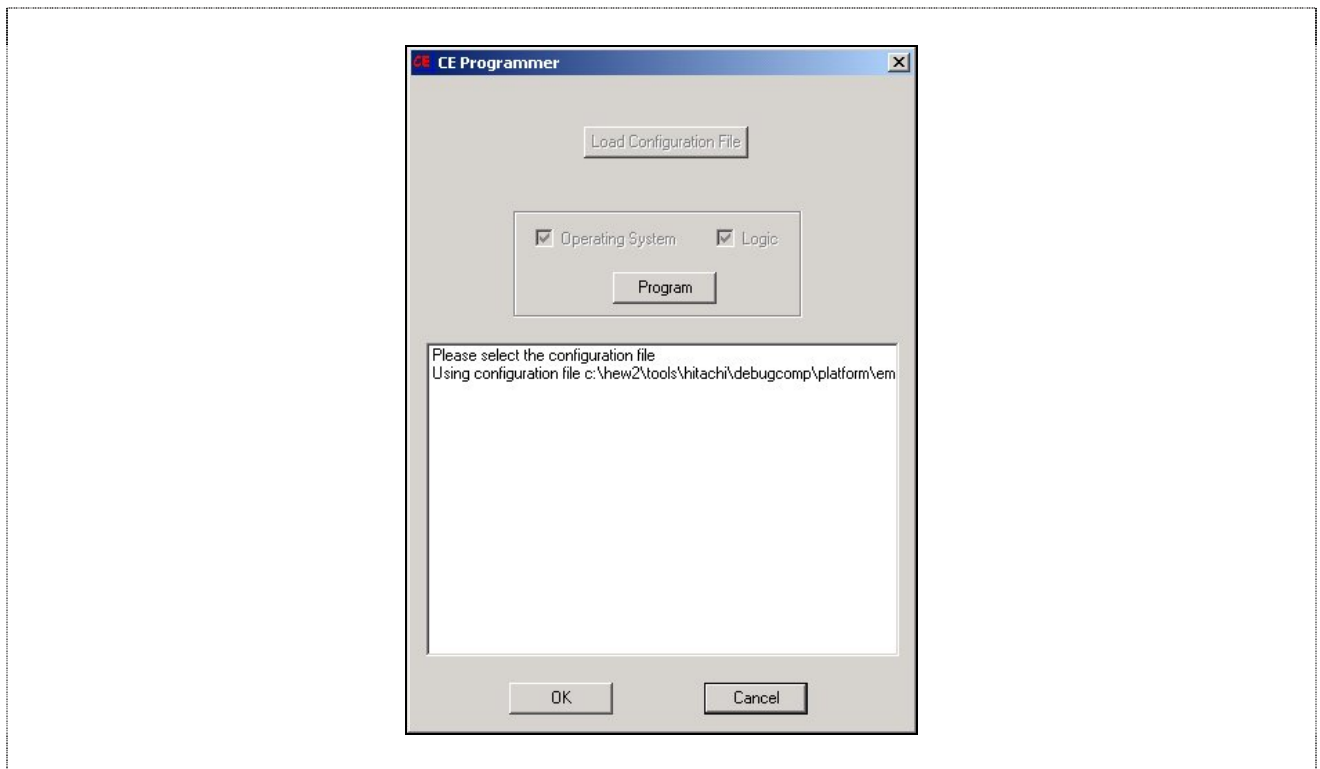


Figure 108 CE Programmer Window

3. Click on the [Program] button to program the OS & Logic (This will take about 2 minutes).

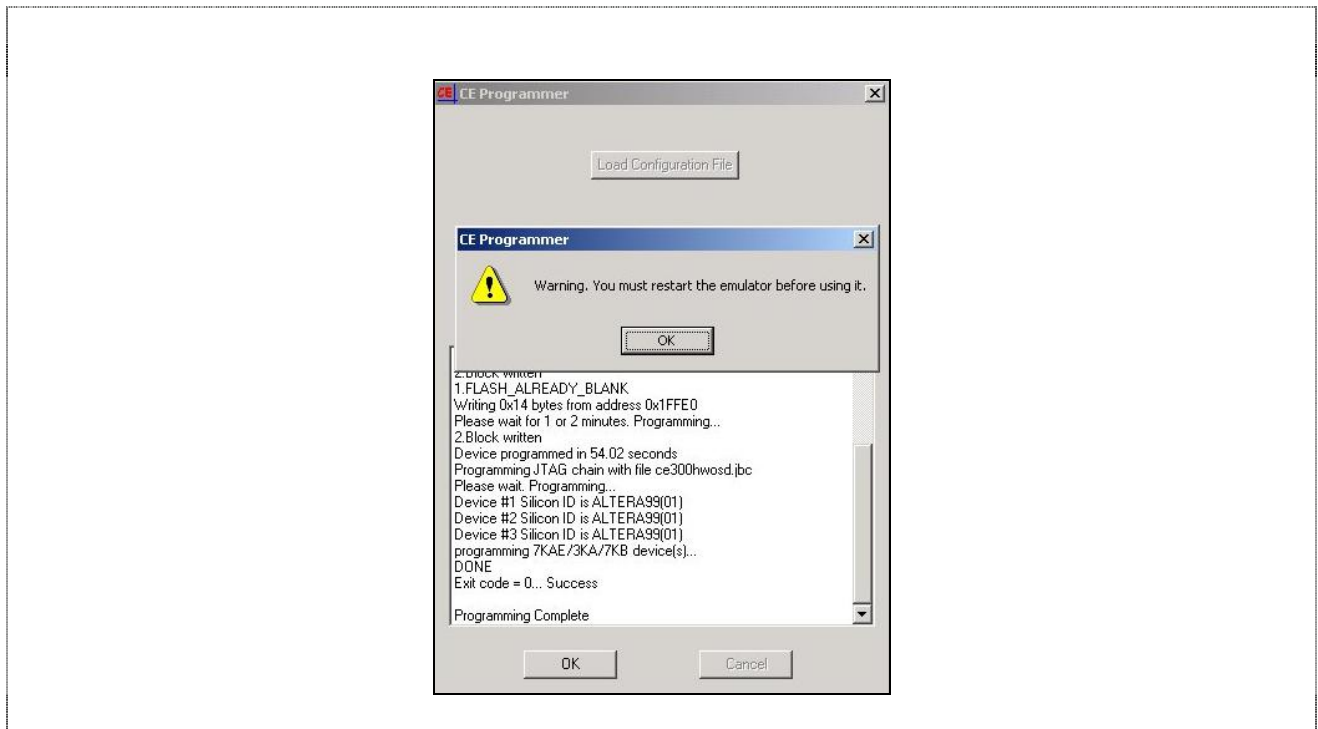


Figure 109 CE OS and Logic Programming

4. Click on the *OK* button, once the programming has completed.

5. Repower up the emulator when HEW is activated.

NOTE:

The on-site upgrade have been designed to withstand interruption, such as power supply trip, PC crashes...

Re-activation of the Programmer to upgrade the system is still possible after the interruption.

Section 11. Trouble-Shooting

The following are a few suggestions on how to perform a basic troubleshooting, if the emulator does not respond to the user's commands.

Symptoms	Checks	Expected Observation & Action
Cannot link to PC	Power Supply	POWER LED lights up in Red.
	PC Bios setup	USB must be enabled
	Window Device Manager	USB do not have any contention. Compact Emulator must be detected instead of a USB device. –Observe in system hardware configuration
	Standalone Self Test	POWER LED lights up in Orange denoting entering self test mode
		Adjacent RUN LED do not light up after the test (Blinking POWER LED)
	USB cable and connector contacts	Clean the contact. Change the USB cable.
	Target System	Is user supply provided to the target? Removed the target and user cable adaptor.
	HEW DEBUGGER Setup	Has HEW DEBUGGER been setup in the PC?
Cannot run code on target	Clock setting	Is clock set to “target”? Is there any clock signal input to the emulator? Unlike a actual footprint user cable, when a direct user cable is used, user has to make sure that the clock is a oscillating signal (e.g. TTL)
	Mode setting	Is mode set to “target”? or preset to some other setting?
	Reset setting	Is reset masked in the HEW DEBUGGER configuration platform window? A constant reset or other watchdog reset may cause user program to run abnormally. User is advised to mask the signal for troubleshooting purposed

STBY setting	Is STBY masked in the HEW DEBUGGER configuration platform window? The signal is masked in default. The activation of the signal will reset the emulator internal registers!!!
NMI	Is NMI masked in the HEW DEBUGGER configuration platform window? A constant NMI may cause user program to run abnormally. User is advised to mask the signal for troubleshooting purposed
Power Supply	Is Target supplying power to the emulator? Observe HEW status window UVcc value.
Cable detection	Has HEW DEBUGGER detected the Cable? Is CABLE_IN_N tied to ground in the target system? Observe HEW status window UVcc value.
Target connection	Is the target connection tight and secure?
Signal level	Are the emulator pull-up resistors driving the target system? Are the internal pull-up resistor (47Kohm) causing a unknown state to the target when a pull down resistor is also connected to the target? User may like to remove the internal pull- up resistor.

- Blank Page -

Appendix A : User Connector Pin Assignment

User Connector 1 - Pin Assignment

Column	Pin	Description	Column	Pin	Description
A	40	AVSS	A	20	TxD0/P90
B	40	AVCC	B	20	TxD1/P91
A	39	AVCC	A	19	RxD0/P92
B	39	AVSS	B	19	RxD1/P93
A	38	VREF	A	18	IRQ4*/SCK0/P94
B	38	AVSS	B	18	IRQ5*/SCK1/P95
A	37	VSS	A	17	
B	37	WDTOVF	B	17	
A	36	* CABLE_IN_N	A	16	VSS
B	36	NMI	B	16	PA0/TP0/TEND0*/TCLKA
A	35	STBY_N	A	15	PA1/TP1/TEND1*/TCLKB
B	35	RES_N	B	15	PA2/TP2/TIOCA0/TCLKC
A	34	MD0	A	14	PA3/TP3/TIOCB0/TCLKD
B	34	MD1	B	14	PA4/TP4/TIOCA1
A	33	MD2	A	13	PA5/TP5/TIOCB1
B	33	VSS	B	13	PA6/TP6/TIOCA2
A	32	T_EXTAL_CLK/User CLK	A	12	PA7/TP7/TIOCB2
B	32	VSS	B	12	VSS
A	31	VCC	A	11	TIOCA3/TP8/PB0
B	31	VCC	B	11	TIOCB3/TP9/PB1
A	30	RESERVED <VCC_generated>	A	10	TIOCA4/TP10/PB2
B	30	RESERVED <VCC_generated>	B	10	TIOCB4/TP11/PB3
A	29	RESERVED <3V3>	A	9	TOCXA4/TP12/PB4
B	29	RESERVED <5V>	B	9	TOCXB4/TP13/PB5
A	28	VSS	A	8	DREQ0*/TP14/PB6
B	28	P70/AN0	B	8	ADTRG*/DREQ1*/TP15/PB7
A	27	P71/AN1	A	7	VSS
B	27	P72/AN2	B	7	PC0
A	26	P73/AN3	A	6	PC1
B	26	P74/AN4	B	6	CS4*/TEND2*/PC2
A	25	VSS	A	5	CS5*/DREQ2*/PC3
B	25	P75/AN5	B	5	CS6*/TEND3*/PC4
A	24	P76/AN6	A	4	CS7*/DREQ3*/PC5
B	24	P77/AN7	B	4	IRQ6*/PC6
A	23	P80/RFSH*/IRQ0*	A	3	IRQ7*/PC7
B	23	P81/CS3*/IRQ1*	B	3	VSS
A	22	P82/CS2*/IRQ2*	A	2	
B	22	P83/CS1*/IRQ3*	B	2	
A	21	P84/CS0*	A	1	
B	21	VSS	B	1	

NOTE :

1. * When target is connected, user has to "Ground" the pin [CABLE_IN_N].
2. EXTAL Pin can only accept Oscillating Clock.
3. User Pins Not available are: FWE, XTAL, CVCC, OSC1 and OSC2..

User Connector 2 - Pin Assignment

Column	Pin	Description	Column	Pin	Description
A	40	VSS	A	20	Reset_out
B	40	P10/A0	B	20	P53/A19
A	39	P11/A1	A	19	P54/A20
B	39	P12/A2	B	19	P55/A21
A	38	P13/A3	A	18	P56/A22
B	38	P14/A4	B	18	P57/A23
A	37	P15/A5	A	17	P60/WAIT
B	37	P16/A6	B	17	P61/BREQ
A	36	P17/A7	A	16	P62/BACK
B	36	VSS	B	16	VSS
A	35	P20/A8	A	15	P63/AS
B	35	P21/A9	B	15	P64/RD
A	34	P22/A10	A	14	P65/HWR
B	34	P23/A11	B	14	P66/LWR
A	33	P24/A12	A	13	P67
B	33	P25/A13	B	13	
A	32	P26/A14	A	12	
B	32	P27/A15	B	12	
A	31	VSS	A	11	VSS
B	31	P30/D8	B	11	
A	30	P31/D9	A	10	
B	30	P32/D10	B	10	
A	29	P33/D11	A	9	
B	29	P34/D12	B	9	
A	28	P35/D13	A	8	
B	28	P36/D14	B	8	
A	27	P37/D15	A	7	VSS
B	27	VSS	B	7	P80/IRQ0
A	26	P40/D0	A	6	P81/IRQ1
B	26	P41/D1	B	6	P82/IRQ2
A	25	P42/D2	A	5	P83/IRQ3
B	25	P43/D3	B	5	P94/IRQ4
A	24	P44/D4	A	4	P95/IRQ5
B	24	P45/D5	B	4	IRQ6
A	23	P46/D6	A	3	
B	23	P47/D7	B	3	VSS
A	22	VSS	A	2	
B	22	P50/A16	B	2	
A	21	P51/A17	A	1	
B	21	P52/A18	B	1	

NOTE :

1. * When target is connected, user has to "Ground" the pin [CABLE_IN_N].
2. EXTAL Pin can only accept Oscillating Clock.
3. User Pins Not available are: FWE, XTAL, CVCC, OSC1 and OSC2..

Appendix B : User Connector Specification

PART NUMBER

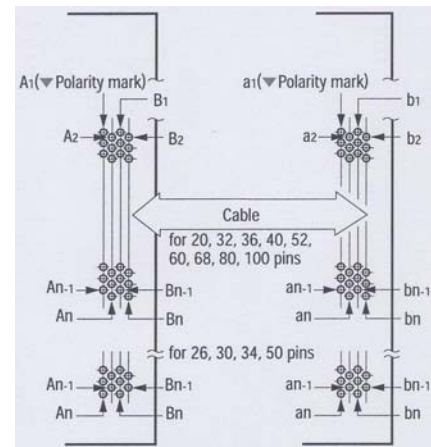
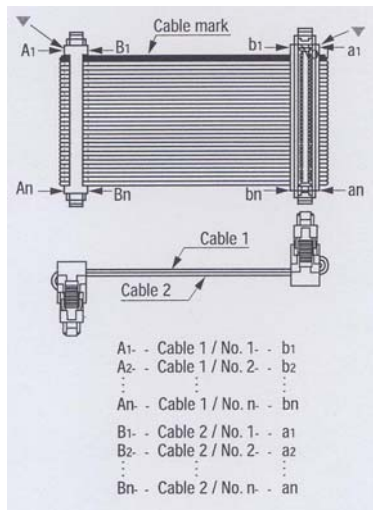
KEL 1.27mm(50mil) PITCH Plug	= 8830E-080-170S
KEL Cable Assembly	= 8822E-080-171-030-AA

LABELLING

NOTE:

The pin assignement of the user connector in Appendix A refers to the defination of the pins at the target side (Capital letter A-B), but not the Compact emulator side (Small Letter a-b). User is advised to refer to the cable assembly illustration to avoid confusion.

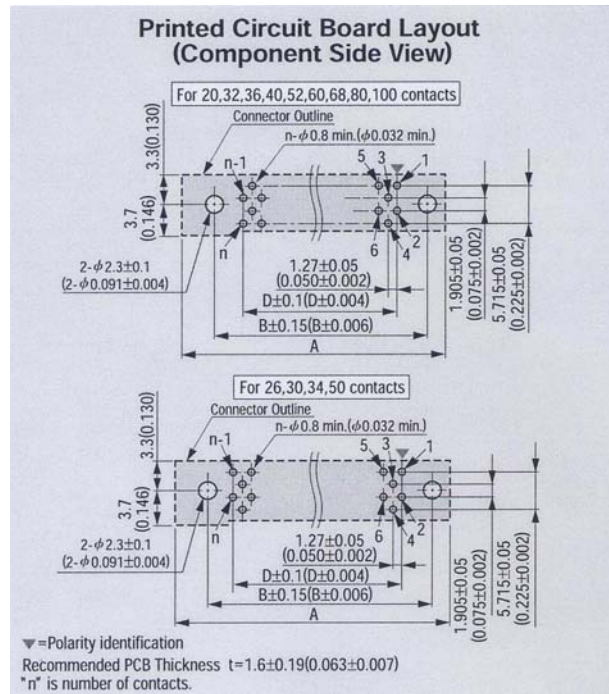
[b40 (Compact Emulator)	=	A40 (Target)]
[b1 (Compact Emulator)	=	A1 (Target)]
[a40 (Compact Emulator)	=	B40 (Target)]
[a1 (Compact Emulator)	=	B1 (Target)]



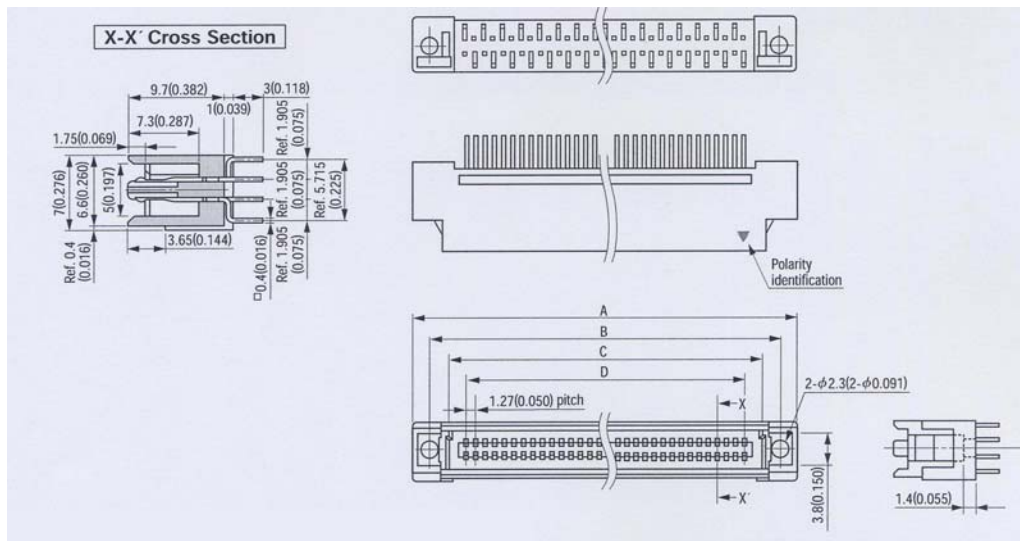
DIMENSION

Part Number: 8830E-080-170S

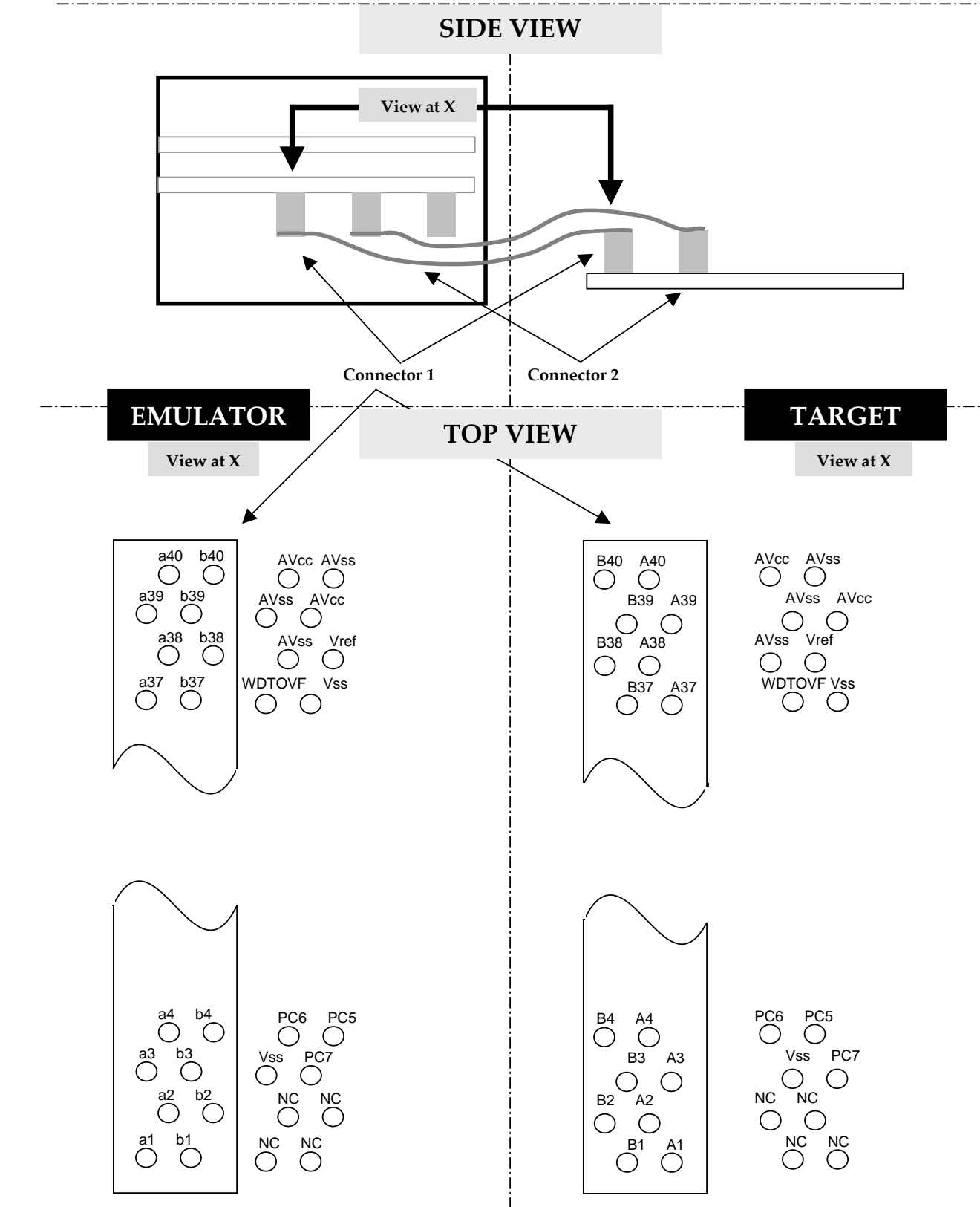
Unit :mm (inch)



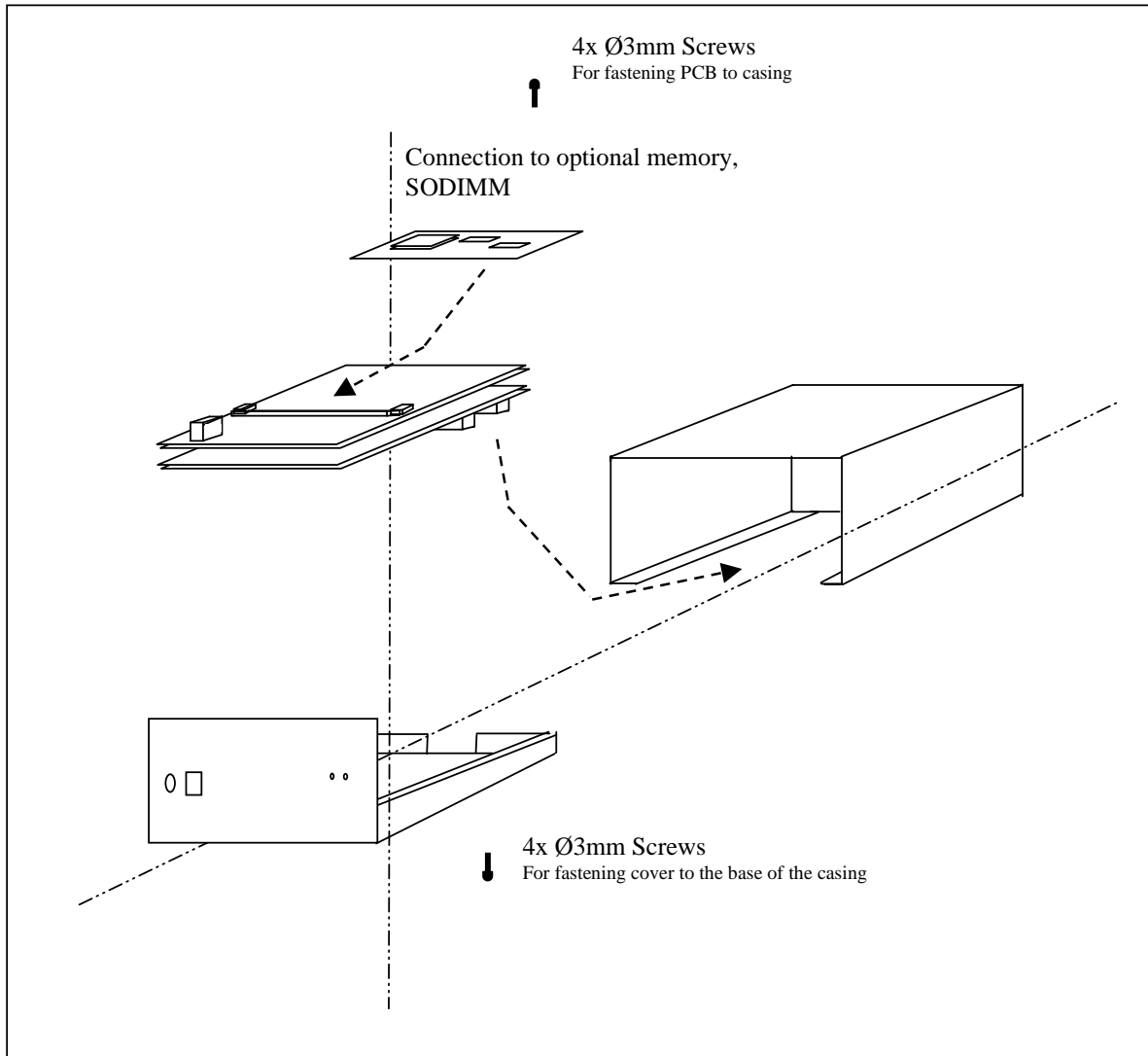
No. of contacts	Part Number	A	B	C	D
80	8830E-080-170S	65.93 (2.596)	60.96 (2.400)	53.93 (2.123)	49.53 (1.950)



Appendix C : User Connector Pin Layout















Appendix D : Casing Assembly



Appendix E : Summary of HEW Debugger

Menu	Option	Shortcut	Toolbar Button	Remark
View	Disassembly	Ctrl + D		
	TCL Toolkit			
	Command Line	Ctrl + L		
	Workspace	Alt + K		
	Output	Alt + U		
	CPU	Registers	Ctrl + R	
		Memory...	Ctrl + M	
		IO	Ctrl + I	
		Status	Ctrl + U	
	Symbol	Labels	Shift + Ctrl + A	
		Watch	Ctrl + W	
		Locals	Shift + Ctrl + W	
	Code	Eventpoints	Ctrl + E	
		Trace	Ctrl + T	
		Stack Trace	Ctrl + K	
	Graphic	Image...	Shift + Ctrl + G	
		Waveform...	Shift + Ctrl + V	
Option	Debug Session...			To list, add, or remove debug session
	Debug Setting			To set the debugging conditions or download modules
	Radix	Hexadecimal		
		Decimal		
		Octal		
		Binary		
	Emulator	System...		To open [configure platform] window, for setting of the debugging environment
		Memory Resources...		To open [memory mapping] window, for setting of the emulation memory resources

Menu	Option	Shortcut	Toolbar Button	Remark
Debug	Reset CPU			Reset the MCU and set the PC to the reset vector address
	Go	F5		Start executing user code from the current PC
	Reset Go	Shift F5		Reset the MCU and execute user code from the reset vector
	Go To Cursor			Start executing the user program at the current PC until the PC reaches the address indicated by the current text cursor position
	Set PC to Cursor			Set the PC to the address at the row of the text cursor
	Run...			Launches the [Run Program] dialog box allowing the user to enter the PC or PC breakpoint during executing the user program.
	Step In	F11		Execute a block of user code before breaking
	Step Over	F10		Execute a block of user code before breaking. If a subroutine call is reached, then the subroutine will not be entered
	Step Out	Shift + F11		Execute the user code to reach the end of the current function
	Step...			Launches the [Step Program] dialog box allowing the user to modify the setting for stepping
	Step Mode	Auto		Step only one source line when the [Source] window is active. When the [Disassembly] window is active, stepping is executed in a unit of assembly instruction
		Assembly		Execute stepping in the unit of assembly instruction
		Source		Step only one source line
	Halt Program	ESC		Stop the execution of user code
	Initialize			Disconnects the debugging platform and connects it again
	Disconnect			Disconnect the debugging platform. This option cannot be used in some product
	Download Module			Download the selected object program
	Unload Modules			Unload the object program information
Memory	Search...			
	Copy...			
	Compare...			
	Fill...			
	Refresh			Force a manual update of the content of all the opened [Memory] window
	Configure Overlay...			Select the target section group when the overlay function is used

Appendix F : Technical Specification

Physical Characteristics

Item		Specification
CE300H Main Unit	Dimensions	145 x 120 x 60 mm (l x w x h)
	Weight	1Kg
Power Supply Adaptor	Dimensions	85 x 45 x 35 mm (l x w x h)
	Weight	0.2 Kg
Power Supply Adaptor Cable	Length	1.8 m
PC Interface Cable	Length	1.0 m
User Interface Cable	Length	0.3 m

Environmental Conditions

Item		Specification
Temperature	Operating	10°C to 35°C
Relative Humidity (non-condensing)	Operating	30% to 85%
Corrosive Gas		None

Electrical Characteristics

CE300H

Item		Specification
Maximum Operating Condition		
Supply voltage		-0.3 to 7.0V
Supply current		2.6A maximum
Target Supply voltage		-0.3 to 7.0V

Power Supply Adaptor

Item		Specification
Input		110 / 230V 47-63 Hz
		4.75 - 5.25 Volt
Output		2.6A 5% overload

Appendix G : Frequently Asked Questions

1. WHY DIDN'T THE CLOCK INPUT TOGGLE IN THE PINVIEW WINDOW?
 2. CAN CE WORK WITH OTHER USB DEVICES?
 3. WHAT IS POTF? WHAT CAN IT DO?
 4. HOW TO DISABLE THE POTF FUNCTION?
 5. HOW TO MINIMIZE THE INTRUSION OF THE POTF?
 6. HOW MUCH CURRENT WILL THE EMULATOR DRAW FROM THE TARGET SYSTEM?
 7. WHY IS THE PROGRAM HALTED AT AN ADDRESS THAT IS NOT SPECIFIED IN THE PRESET COMBINATION BREAKPOINT?
 8. WHAT IS THE CABLE_IN_N SIGNAL USED FOR?
 9. HOW ABOUT MODE, CLOCK , RES, NMI & STBY SIGNALS? HOW DOES THE EMULATOR CONTROL THESE SIGNALS?
 10. WHY IS STBY PIN MASKED IN DEFAULT?
 11. WHY IS C-LEVEL STEPPING NOT POSSIBLE?
 12. WHY MUST THE PROGRAMMABLE FUNCTION GENERATOR (PFG) BE PROGRAMMED? WHAT IS THE PFG USED FOR ?
 13. WHY IS THE STATE CHANGES OF PORT NOT REFLECTED IN THE PINVIEW WINDOW?
 14. WHY CAN'T THE TWO MBYTES OF OPTIONAL MEMORY BE UTILIZED FULLY?
 15. WHY IS THE HEW DEBUGGER RESPONSE SLOWED DOWN WHEN I/O WINDOW IS OPENED?
 16. WHY CAN'T THE OTP WORK AS ACCORDINGLY?
 17. WHY IS THE SELECTION OF "TARGET CLOCK " NOT AVAILABLE IN THE SYSTEM SETUP WINDOW?
 18. WHY CAN'T THE PC BREAK STOP THE RUNNING PROGRAM?
 19. WHY IS THE DOWNLOAD SPEED OF CE BECOME SO SLOW SUDDENLY?
-

1. Why didn't the clock input toggle in the PinView window?

The PinView window reads the pins' status from the emulator at a constant interval. Thus, there is a possibility that it always snaps the same level from the chip.

2. Can CE work with other USB devices?

CE can work with other USB devices. If another hub is used, CE must be the last device to be plugged-in.

3. What is POTF? What can it do?

POTF stands for Parallel On The Fly. It can read or write memory while the user program is running. In this way, user can have the instant view of the outcome of the executing program. In order to access these memory, the emulator will “stop” the running program for a short interval, approximately 150 ns, for a word access.

4. How to disable the POTF function?

As long as the user do not modify the memory contents or perform a refresh memory window command, POTF function will not be activated to intrude the real time operation of the user program.

5. How to minimize the intrusion of the POTF?

POTF will be activated based on the HEW DEBUGGER commands, such as memory edit or refresh. If several memory windows are opened, HEW DEBUGGER will read back all the windows content. Thus, user is advised to open the memory window at minimal size, in order to avoid unnecessary intrusion.

6. How much current will the emulator draw from the target system?

The emulator will not consume the target system supply. It will generate an user VCC for internal usage, which has an identical level as the target system power supply.

7. Why is the program halted at an address that is not specified in the preset combination breakpoint?

The emulator will break out of the user code execution once it has detected the preset condition. However the emulator will not stop execution immediately as it has to complete its current tasks, thus the code will not break at its preset address.

Another point to note is the prefetch condition, if the preset condition matches the prefetch instruction, the emulator will also enter the break mode.

The worst case scenario is: if a break condition is set at the beginning of subroutine B, which codes are stacking behind subroutine A. A break condition will happen when subroutine A is called. This is due to the prefetching of subroutine B code when subroutine A is returning to the main routine.

8. What is the CABLE_IN_N signal used for?

This signal is used by the emulator to identify target connection. When CABLE_IN_N is lo, the emulator will know that target is connected, and the power supply will be switched to follow the target power supply.

9. How about Mode, Clock , RES, NMI & STBY signals? How does the emulator control these signals?

Generally, the emulator will control all the above signals. The only uncontrollable item will be the target power supply. The emulator will track and follow the target supply level once the cable is detected. User has to set the above signals in the HEW DEBUGGER/setup/configure platform window. User can set the Mode & Clock signal to follow the target system or any options that is available in the selection. The RES & NMI pins are set to follow the target in default. User can choose to mask these signals in the configure window. STBY is masked in default.

10. Why is STBY pin masked in default?

STBY pin will initialise some of the emulator registers. Thus, user is advised not to use the signal unless necessary.

11. Why is C-Level Stepping not possible?

If the disassembly window is opened, the step-in instruction will command a single step execution of the assembly code. User has to close the disassembly windows. This will inform the system to perform C-Level stepping of the loaded C-code.

12. Why must the Programmable Function Generator (PFG) be programmed? What is the PFG used for ?

The PFG is implemented using a RAM-based FPGA. Thus it must be programmed when required functions is not the default function. The PFG is implemented for its flexibilities. A single chip FPGA is used to implement the PFG which can be programmed to any functions that can help user to debug their target system. Please feedback your needs to the design group so that more effective functions can be developed. For CE300H, a default function is preloaded at startup.

13. Why is the state changes of port not reflected in the Pinview window?

The pinview module gives an snapshot of the microcomputer pins at constant interval. It is not an oscilloscope that can snap signal as fast as 2ns. It is a tool used to help user to have a feel of the status of the microcomputer pins. It will be very helpful when it is at static state. User can track any discontinuity in the emulator to target connection.

14. Why can't the two Mbytes of optional memory be utilized fully?

The two Mbytes of optional memory consists of 4 x 512Kbytes of memory. Each memory is selected based on the upper address. Thus, if user intends to use the whole memory at a starting address that is not multiple of 512Kbytes, user will not be able to utilize the full memory capacity (e.g. if the external address begin at H'20000). However, if external address begin at H'0, as in the ROMless mode, user will be able to map the whole 2Mbytes space (H'0 to H'1F FFFF).

15. Why is the HEW DEBUGGER response slowed down when I/O window is opened?

Unlike other windows, I/O window requires much more communication between the PC and the emulator, in order to obtain the required I/O data. Thus, user is advised to close the I/O window if it is not used.

16. Why can't the OTP work as accordingly?

User has to note the differences as stated in the user manual, such as the initialization of registers (SP...), the delay of signals (about 4ns of cabling delay...), the unused pins (OSC1, OSC2, CVCC...), the regenerated power supply for emulator, the smoothed main clock signal, ...

17. Why is the selection of "target clock " not available in the system setup window?

HEW DEBUGGER will disallow the target selection if the target is not available (The Signal CABLE_IN_N is Hi).

18. Why can't the PC Break stop the running program?

This is likely that the PC break is not set. This can happen if the set location is a external write-only location, such as Flash, EPROM... . The PC break is implemented using a software mean to replace the user instruction with a special system instruction. Thus the operation can be completed with a write only location.

19. Why is the download speed of CE become so slow suddenly?

If user download their code to the internal ROM area, the speed will be very fast. However when the download area is the external target area, the download speed will be much slower. This is because of the transfer mechanism, the data will need to transfer from the PC to the emulator, and transfer to the external target area. This will also depend on the emulation speed at that time.

Renesas Technology (Asia Sales Offices)

URL: <http://www.renesas.com>

URL: <http://www.sg.renesas.com/sales>

ASIA HEADQUARTERS & TECHNICAL SUPPORT :	
South Asia Headquarters : Singapore Renesas Technology Singapore Pte. Ltd. 1, HarbourFront Avenue, #06-10, Keppel Bay Tower, Singapore 098632. Tel : (65)-6213-0200 Fax : (65)-6278-8001 Email : contact.singapore@renesas.com	Technical Support Renesas System Solutions Asia Pte. Ltd. 1, Harbourfront Avenue, #06-06, Keppel Bay Tower, Singapore 098632. Tel : (65)-6213-0333 and 6387-2839 Fax : (65)-6278-1226
North Asia Headquarters : Hong Kong Renesas Technology Hong Kong Ltd. 7/F., North Tower, World Finance Centre, Harbour City, Canton Road, Hong Kong. Tel: (852) 2265-6688 Fax: (852) 2375-6836 Email : contact.hongkong@renesas.com	
ASIA SALES OFFICES :	
China	
Renesas Technology Hong Kong Ltd Shenzhen Representative Office Unit 1511-12, Shun Hing Square Di Wang Commercial Centre, 5002 Shennan Road East, Shenzhen City 518008, China Tel : (86) (755) 8246-1711 Fax : (86) (755) 8246-1728 Email : contact.china@renesas.com URL : http://www.cn.renesas.com	
Renesas Technology (Shanghai) Co., Ltd. 26/F., Ruijin Building, No. 205 Maoming Road (S), Shanghai 200020, China Tel : (86) (21) 6472-1001 Fax : (86) (21) 6415-2952 Email : contact.china@renesas.com URL : http://www.cn.renesas.com	
Renesas Technology (Shanghai) Co., Ltd. Beijing Office Room 1654, Office Building, New Century Hotel, No. 6 Southern Rd. Capital GYM., Beijing 100044, China Telex : 210509 HTCBJ CN Tel : (86) (10) 6849-2430 Fax : (86) (10) 6849-2819 Email : contact.china@renesas.com URL : http://www.cn.renesas.com	
Taipei	
Renesas Technology Taiwan Co., Ltd. (effective July 1, 2003) FL. 10, #99, Fu-Hsing N. Rd., Taipei, Taiwan Tel : (886)(2) 2715-2888 Fax : (886)(2) 2713-2999 Email : contact.taiwan@renesas.com URL : http://www.tw.renesas.com	

CE300H

