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

CPUBD-2268F

H8S Super Low Power CPU Board

Microcomputer Development Environment System

User's Manual

2004.01

Renesas System Solutions Asia Pte. Ltd.
www.rss.renesas.com

CPUBD-2268F – CPU Board for H8S/2200 Super Low Power Series Microcomputer User's Manual

Published by : Renesas System Solutions Asia Pte. Ltd.

Date : January 7th, 2004, version 1.0

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, MS and MS-DOS is registered trademark.

Windows and Windows NT are 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.

CPUBD:

Throughout this document, the term "CPUBD" shall be defined as the H8S Super Low Power Series Low-cost CPU Board, CPUBD-2268F 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 Product:

This product is a development-supporting unit for use as training and evaluation tool. The 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 user 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 IMPLED, ORAL OR WRITTEN, EXCEPT AS PROVIDED HEREIN, INCLUDING WITHOUT LIMITATION THEREOF, WARRANTIES AS TO MARKETABILITY, MECRCHANTABILITY, 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 varies 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:

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 install and setup the H8S/2268F CPU board for evaluating the performance of the H8S/2268F microcomputer. Hereafter, the H8S/2268F CPU board shall term as 'CPUBD'.

Operation using the HEW pure debugger software is also detailed in the manual.

1. Introduction

Gives an introduction about the CPU board, package, specification and functions.

2. Installation

Explains how to install the hardware and accompanied software to a host computer.

3. Setup of HEW (Pure Debugger) for CPU Board

Describes the setup steps before embarking on a new project development.

4. Performing Emulation

Describes the various functions available in HEW

5. Usage Constraints

Highlights the various constraints that may encounter by user when operating the CPU board.

6. Hardware

Explains the various hardware blocks in the CPU board.

7. Monitor software

Explains the purpose of the monitor software, the implementation requirements and how to use the monitor software.

8. Flash Programming

Explains the difference between two programming modes and how CPU board operates in these modes.

9. Tutorial

Provides a step-by-step guide in using the CPU board to perform debugging.

10. Demonstration Program

Provides two demonstration programs for user to have hands-on experience with the CPU board.

11. Trouble-Shooting

Advises on some basic fault finding methods and commonly make mistakes.

Appendix A - CPUBD-2268F Board Layout

Appendix B – H8S/2268F Memory Map

Appendix C – Pin Assignment for JP1 ~ JP4

Appendix D - Pin assignment for CON1 & CON2

Appendix E – Schematic drawings

Appendix F – Bill of Materials

Technical Support

The CPUBD is a product for evaluation purposes only. We do NOT supply the same level of support as for the full functioned development tools, however, you may contact the sales offices for downloads and documents.

Related Manuals:

H8S, H8/300 series C/C++ Compiler, Assembler, Optimizing Linkage Editor User's Manual

H8S/2268 Series, H8S/2264 Series Hardware Manual

Table of Contents

Figures & Tables

Section 1. Introduction

H8S/2268F CPU board (CPUBD-2268F) is a low cost training and MCU performance evaluation tool for the H8S Super Low Power family series of microcomputers.

It is also implemented with flash programming feature for the H8S/2268 F-ZTAT microcomputer. It contains a FP-100B package H8S/2268F microcomputer on the board. This CPUBD is also able to support the H8S/2264 F-ZTAT microcomputer.

The H8S/2268F CPU board adopts the standard Renesas High-performance Embedded Workshop (HEW). HEW is a Window-based integrated development platform. In this package, a pure debugger is included in the HEW component.

The diagram below shows the H8S/2268F CPU Board:

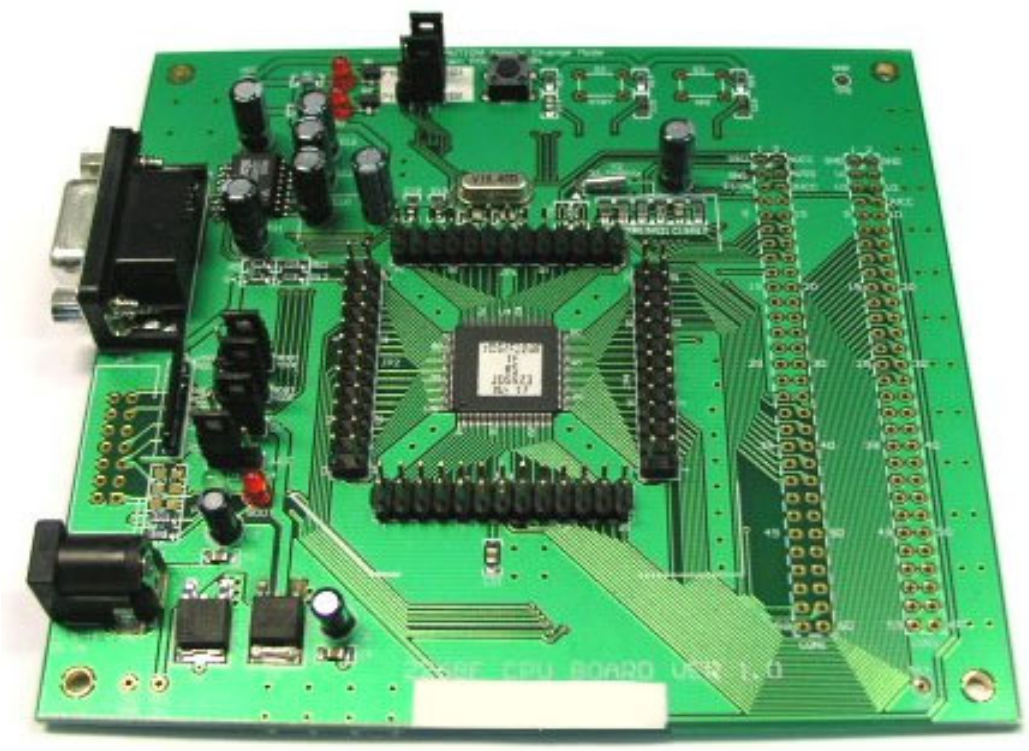


Figure 1.1 H8S/2268F CPU Board [CPUBD-2268F]

1.1. Specification

1.1.1. General

- H8S/2268F microcomputer (using HD64F2268FA20 FP-100B device)
- 256Kbytes of FLASH memory (Monitor software uses approx. 7Kbytes)
- 16Kbytes of on-chip RAM (Monitor software work area uses 4Kbytes)
- Two user LED indicators
- One push button for reset control
- One boot mode LED indicator
- One Power LED indicator
- All Input/Output signals are being pulled out for user connection via CON1 & CON2

1.1.2. Serial Communication

- Utilizes Serial Communication Interface 0 via RS-232 DB-9F socket and RS-232 transceiver chip.
- Supports communication at a baud rate of 115,200bps [non-configurable during debugging].

1.1.3. Power Input

- Accept dual DC power supply at +7.5 volt. ~ +9.0 volt only. [Ripple Rejection ratio more then 60dbm]

1.1.4. Memory Map

- If the CPUBD is to be used with debugger, a section in the memory area is reserved for monitor software. See Appendix *B* for memory map diagrams.

1.1.5. Interface with Application Board

- It is designed to interface with any application board via two 30x2pin connector sockets.

1.1.6. Interface with E7 emulator

- For debugging with E7 emulator, it is only supported with H8S/2264F microcomputer.

1.1.7. Monitor software

- A FLASH-resident debugging monitor software hosted on the CPUBD for performing debugging operations.

1.2. CPUBD Functional Blocks

The CPUBD comprises of a H8S/2268F or H8S/2264F microcomputer, serial port, and boot mode control and user interface.

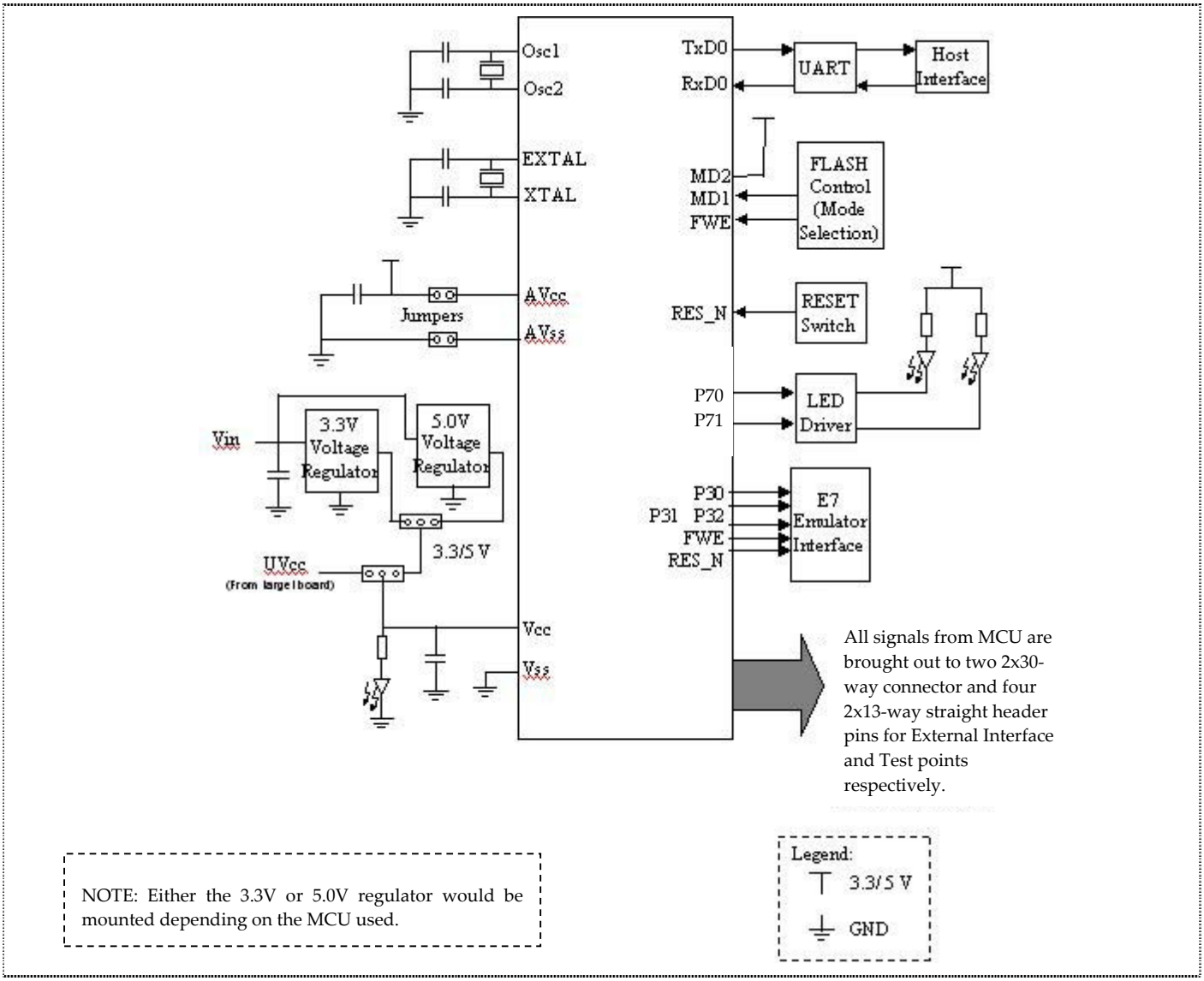


Figure 1.2 CPU Board Functional Blocks

The boot mode circuitry is necessary to place the CPUBD into Boot mode for programming the FLASH. To enter into Boot mode, respective jumper headers on the CPUBD must be shorted.

SCI0 is used to program the board's on-chip flash memory, using the flash programming software built-into the HEW with pure debugger. If the user is not using the serial port for flash programming the CPUBD or debugging, this serial port is available to user.

The HEW with pure debugger software combined with the monitor software programmed into the device provides high level debugging via SCI0.

When connecting external analogue signals, it is important that CPUBD is configured properly with respect to analogue voltage supply and reference. There are two user LEDs on board that can be used by user for their evaluation and are driven directly by the MCU.

All the I/O signals are being tracked out to four 2x13-way straight header connectors for user access as well as to two 2x30-way sockets to allow connection to a target board. These I/O signals are available to user if either flash programming or debugging is not used.

1.3. Package

The CPUBD is supplied in a package containing the following components:

Figure 1.3 CPUBD-2268F Package

1.3.1. Hardware Components

The hardware components included in the package are listed below.

- 1 x H8S/2268F CPU Board
- 1 x RS-232 Serial cable
- 1 x DC Power Input Jack free-end cable
- 2 x 30x2pin connectors [not assembled]
- 1 x 7x2pin connector [not assembled]

1.3.2. Software Components

The package includes a CD ROM containing:

- HEW installer
- User's Manual
- Tutorial program Source code
- Schematic drawings.

Before proceeding, user has to check that all the items listed in the packing list. Please contact the relevant Renesas Technology sales office in Asia if any item is missing.

1.4. Summary of CPUBD-2268F functions

Items	Specifications
Supported Microcomputers	▪ H8S/2268F and H8S/2264F
Operating Frequency	▪ 18.432MHz (System clock) ▪ 32.768KHz (Sub clock)
Supported Operating Voltage	▪ 3.3 and 5.0 Volts.*1
Host Machine	▪ Recommended Pentium™ III or equivalent processor PC ▪ Recommended 128Mbytes RAM and 100Mbytes hard disk space ▪ Microsoft Windows 98, Windows Me, Windows NT 4.0, Windows 2000 or Windows XP ▪ One Serial port
Host Interface	▪ RS-232 Serial Interface ▪ Baud rate @ 115,200 bps
Supported File Format	▪ Motorola S-type ▪ ELF/Dwarf2
Interface Software	▪ HEW with pure debugger
Emulation Functions	▪ C – source level debugging (e.g. instant watch….) ▪ Modify and display MCU registers ▪ Perform real-time emulation of a target program
Memory Functions	▪ Copy, Search, Fill, Load and Save memory functions ▪ Modifies and displays memory content
Breakpoint	▪ PC breakpoint (max. 256)
Step Functions	▪ Step In/ Step Out/ Step Over
On-board Programming	▪ Support on-board programming - Boot mode and User program mode
Flash Protection	▪ Flash program/ erase protection by hardware
User LEDs	▪ Supports two user's LEDs
Interface with E7 Emulator	▪ Supports E7 emulator*2
Interface with Target system	▪ Supports emulation on a target system
Power Supply for CPU board	▪ DC +7.5 Volt. to +9.0 Volt. supplied from external input
Environmental	▪ Operating Temperature: 10 °C to 35 °C ▪ Humidity: 30% to 85% RH ▪ No condensation and corrosive gas

NOTE: *1 Current microcomputer mounted operates at 5.0 Volts only.
***2 For H8S/2264F microcomputer only**

Section 2. Installation

2.1. Label of Parts on CPU Board

Figure 2.1 shows the name of each part of the CPUBD.

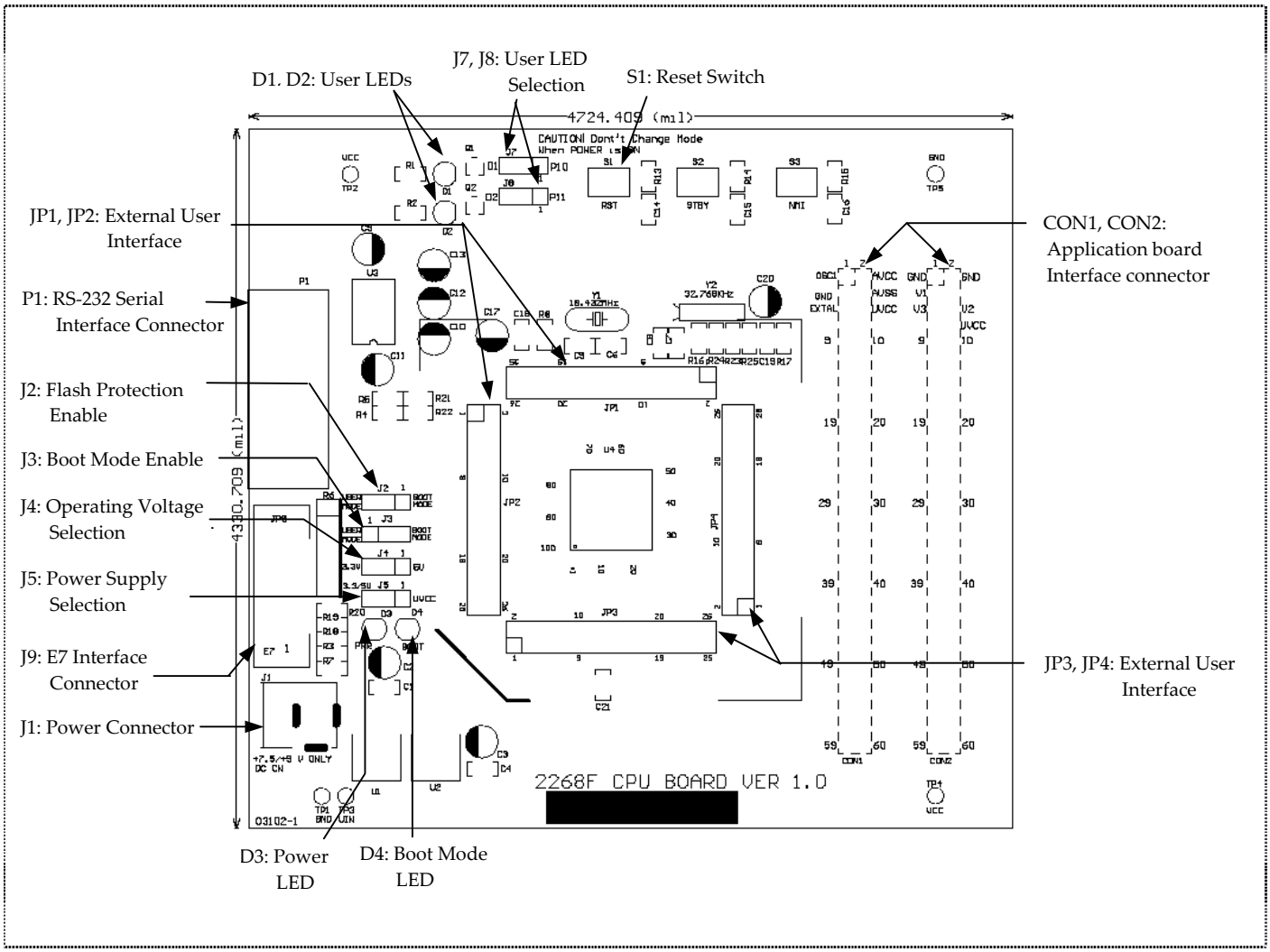


Figure 2.1 Names of Parts on CPU Board

2.2. Installing the CPU Board

Installing the CPUBD requires power and serial connection to a host computer. The serial communication cable for connecting the CPUBD to a host computer is supplied. The serial connection cable uses a 1:1 connectivity.

The diagram below shows how to connect the CPUBD to a host machine or notebook computer equipped with a DB-9P connector.

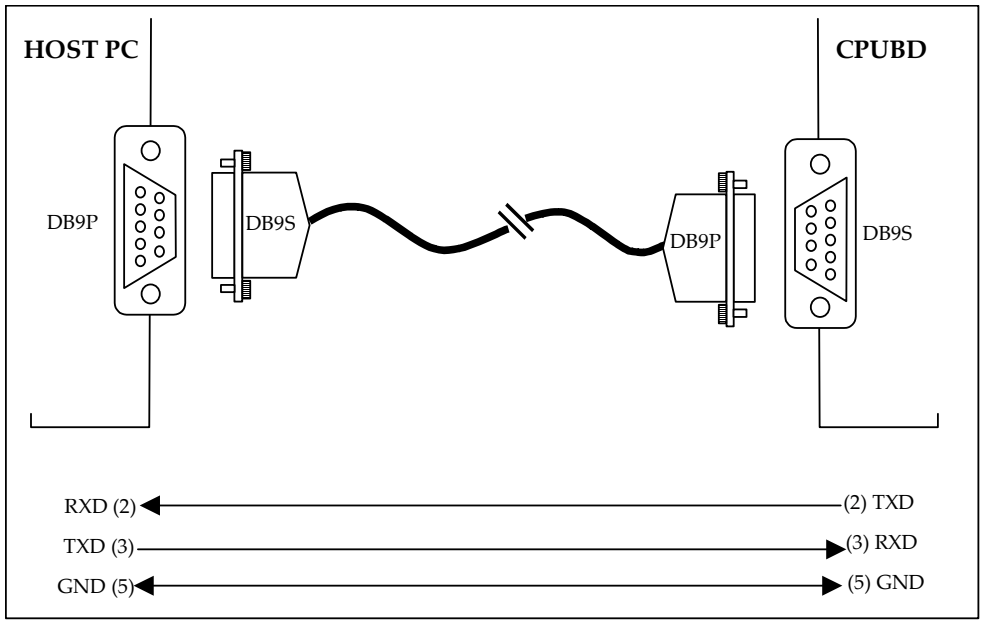


Figure 2.2 Serial Communication connections

2.3. Communication Port Baud Rate

The baud rate utilized by the CPUBD is FIXED at 115,200bps.

2.4. Power Supply for CPU Board

The CPUBD requires a D.C. power supply from +7.5 VDC ~ +9 VDC at approximately 100mA supplied to the J1 connector. Prepare the D.C. power supply separately. The power cable is included with this product. Since total power consumption can vary widely due to external connections, use a power supply capable of providing at least 250mA at +7.5VDC ± 5%.

Two regulators, 3.3 and 5.0 Volts, are used to step down the input voltage. This is to cater for 2 different operating voltages (3.3V and 5.0 V) of the microcomputer. The current CPU board uses the 5.0V MCU.

When power is supplied to the CPUBD, a PWR LED, D3 is lit; otherwise, check the power connection for polarity reversal.

Figure 2.3 shows the specification of the power connector and the DC plug respectively.

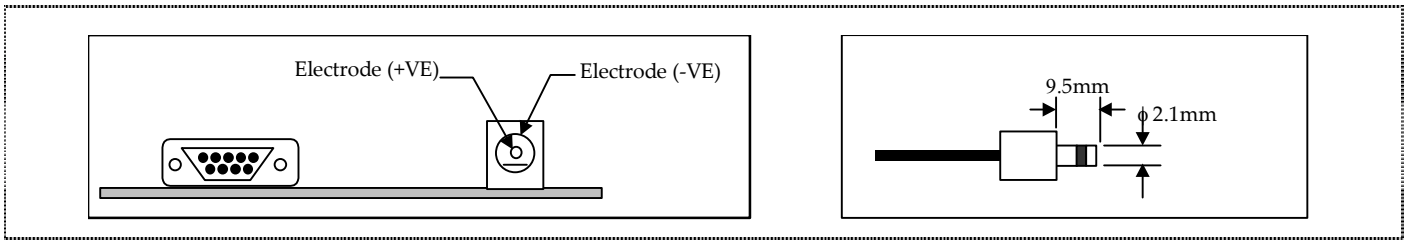


Figure 2.3 Power Connector & DC Plug

2.5. Jumpers Options

The CPUBD has several jumpers to allow various settings for the user:

Jumper Designator	Jumper Name	Jumper Descriptions
J2	FWE SEL	Select either to enable or disable FWE
J3	BOOT MODE SEL	Select either BOOT or USER mode
J4	3.3 / 5V SEL	Select either +3.3V or +5.0V depending on operating voltage of MCU
J5	VCC SEL	Select source of power supply
J7	LED SEL	Select either to use D1 or P10
J8		Select either to use D2 or P11

Table 2.1 List of Jumpers

Please refer to Figure 2.1 for the locations of the jumpers.

2.5.1. Voltage Regulator Selection

The CPUBD caters for operation of either 3.3V or 5.0V. The table below shows how to select the required operating voltage.

CPUBD Operating Voltage	Jumper Name	Jumper Designator	Jumper Selection	Descriptions
3.3 Volts.	3.3V	J4	Short Pin 2 to Pin 3	Power from output of 3.3 V voltage regulator
5.0 Volts.	5V		Short Pin 1 to Pin 2 [Default]	Power from output of 5.0 V voltage regulator

Table 2.2 Operating Voltage Selection Jumpers for MCU

2.5.2. Power Supply Selection Jumpers for MCU

This is the jumper switch to select the source of power supply to the MCU. As shown in Table 2.1 below, any setting not listed in Table 2.2 is not allowed.

Connect to Application Board	Jumper Name	Jumper Designator	Jumper Selection	Descriptions
Not Connected	3.3V/5V	J5	Short Pin 2 to Pin 3 [Default] [Do not short Pin 1 to Pin 2]	Power of MCU is supplied from the CPUBD. Verify the operating voltage selected in Table 2.1
Connected	UVCC		Short Pin 1 to Pin 2 [Do not short Pin 2 to Pin 3]	Power of MCU is supplied from application board

Table 2.3 Power Supply Selection Jumpers for MCU

2.5.3. User Program Mode [Standalone] Selection Jumpers [Default]

This is the jumper switch to place the CPUBD into the user program mode for standalone operation. This is necessary for flashing of the user software into the FLASH ROM of the MCU.

Jumper Designator	Jumper Selection	Descriptions
J2	Short Pin 1 to Pin 2	Enable FWE
J3	Short Pin 2 to Pin 3	To place CPUBD into User Program mode [Normal mode]
J5	Short Pin 2 to Pin 3	Power of MCU is supplied from the CPUBD
J7	Short Pin 1 to Pin 2	User LEDs D1 and D2 are used
J8	Short Pin 1 to Pin 2	

Table 2.4 User Program Mode [Standalone] Selection Jumpers [Default]

2.5.4. Boot Mode Selection Jumpers

This is the jumper switch to place the CPUBD into the boot mode. This is necessary for flashing the kernel software and monitor software into the FLASH ROM of the MCU.

Jumper Designator	Connection	Jumper Selection	Descriptions
J2	FWE	Short Pin 1 to Pin 2 [Default]	Enable FWE
J3	MD1	Short Pin 1 to Pin 2	To place CPUBD into Boot mode

Table 2.5 Boot Mode Selection Jumpers

2.5.5. Flash Write Enable

FWE (Flash Write Enable) provides hardware protection for flash programming (writing) and erasing. When enabled (FWE pin pulled HIGH), hardware protection is disabled, enabling programming and erasing of flash memory.

When disabled (FWE pin pulled LOW), it prevents accidental flash programming and erasing, hereby protecting the user's program.

Jumper Designator	Jumper Selection	Descriptions
J2	Short Pin 1 to Pin 2	Enable FWE to allow Flash Programming & Erasing operation
	Short Pin 2 to Pin 3	Disable FWE to prevent accidental Flash Programming & Erasing operation

Table 2.6 Flash Write Enable Jumpers

NOTE: FWE must always be enabled to allow Flash Programming & Erasing operation during debugging.

2.5.6. User LED Selection

This jumper switch enables the use of the two user LEDs, D1 and D2, connected to P10 and P11 respectively. By disabling the connection to the LEDs, the user is able to make use of these IO port pins.

Jumper Designator	Jumper Selection	Descriptions
J7	Short Pin 1 to Pin 2	Enable D1
	Short Pin 2 to Pin 3	Disable D1, enable P10
J8	Short Pin 1 to Pin 2	Enable D2
	Short Pin 2 to Pin 3	Disable D1, enable P11

Table 2.7 LED Selection Jumpers

2.6. Installation of HEW (Pure Debugger) for CPU Board

To install the HEW (Pure Debugger) for CPUBD from the installation disk, proceed as follows:

- ❑ Insert the HEW (Pure Debugger) for CPUBD installation CD.
- ❑ Run Windows if it is not already running.
- ❑ Close all other applications that are running.
- ❑ Choose *Run* from the Program Manager File menu.
- ❑ Type *Setup* and click OK:

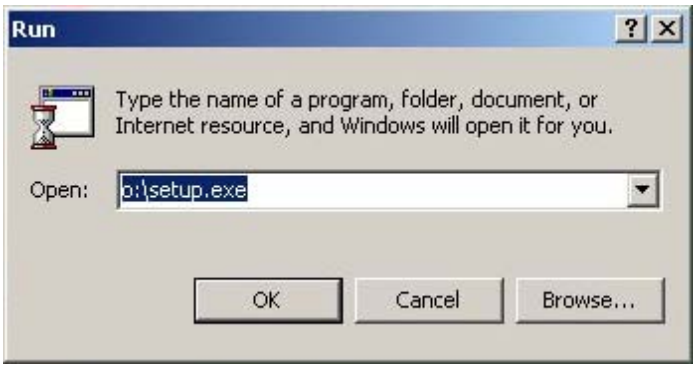


Figure 2.4 Run Dialogue Box

This runs the HEW (Pure Debugger) for CPUBD installer, and the following Welcome! Screen is displayed:

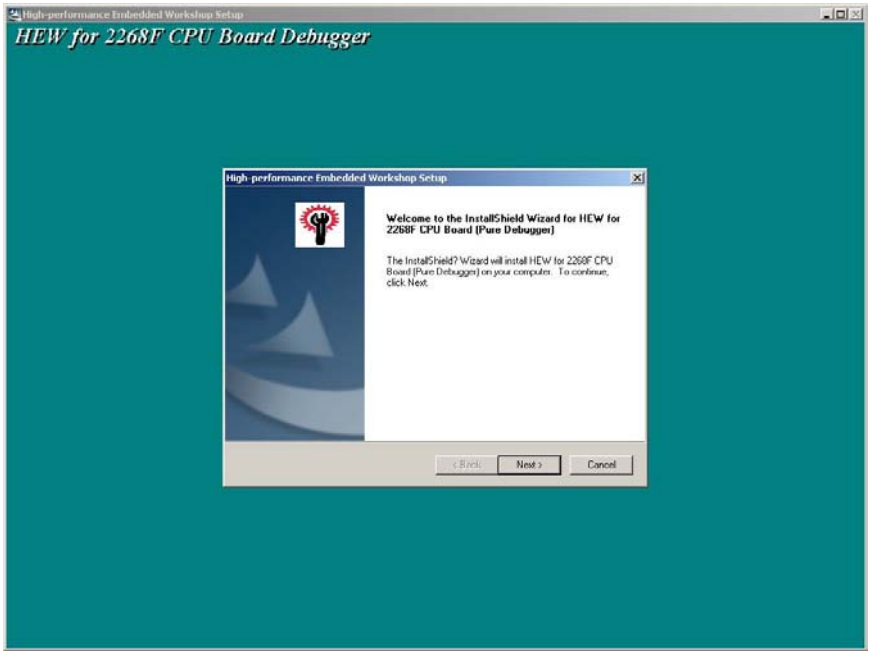


Figure 2.5 HEW for CPUBD Installer Welcome! Screen

- ❑ Click *Next* to proceed with the installation.

❑ Check the *License Agreement* concerning installation and then click *Yes* to proceed.

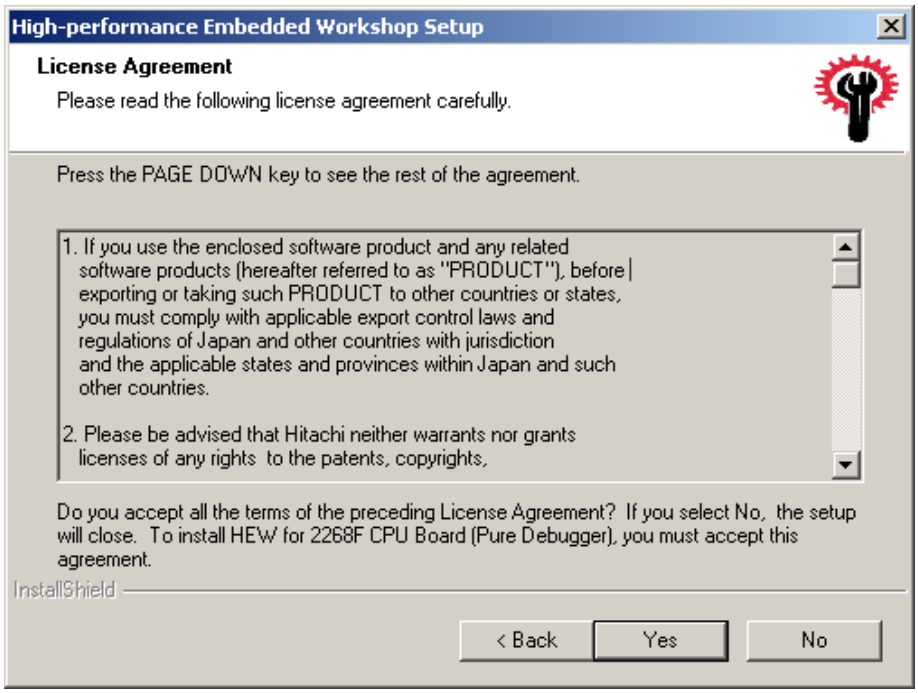


Figure 2.6 License Agreement Dialogue Box

The following dialogue box enables the selection of directory in which user can install the HEW (Pure Debugger) for CPUBD.

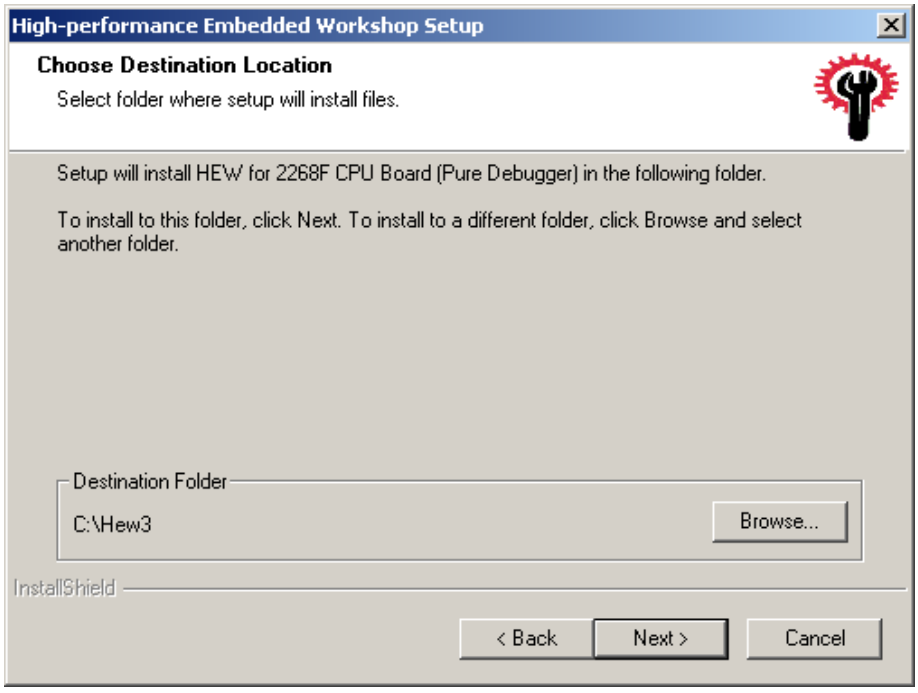


Figure 2.7 Select Destination Directory Screen

- 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 component to the respective HEW tool administration menu.**

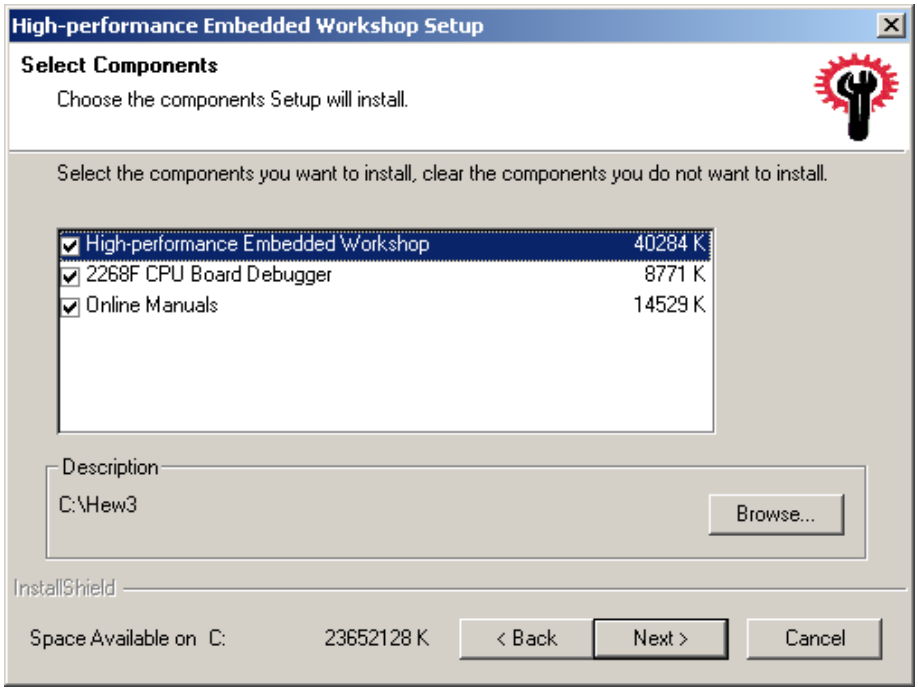


Figure 2.8 Select Components

- Select the components to be installed.
- Ensure each selection is selected in turn to confirm the correct directory it is installing into.

If user chooses *Next*, the following dialogue box will confirm each installation directory you selected. Always ensure that all components are installed in the same required directory.

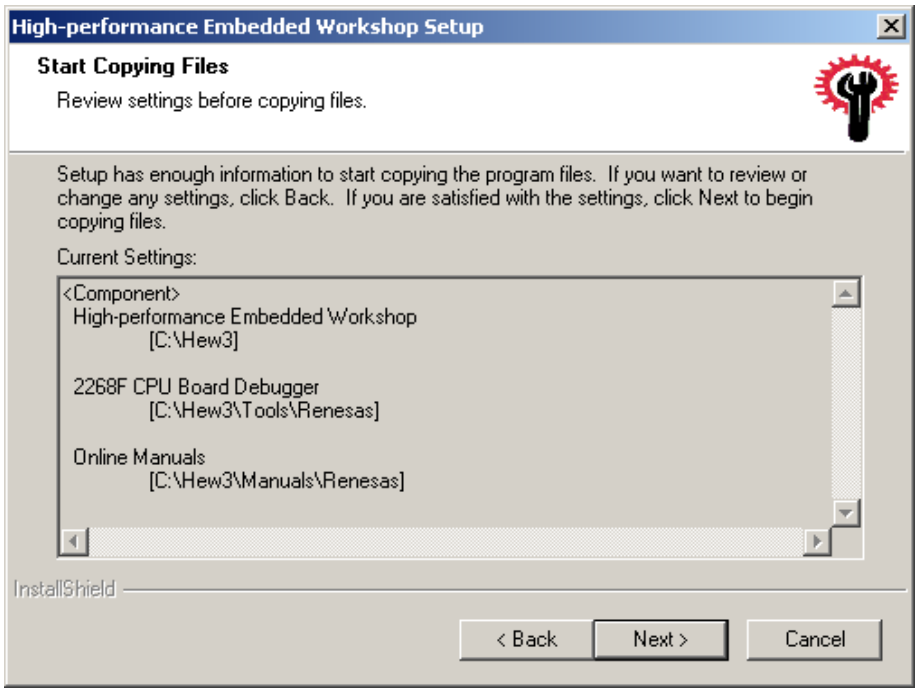


Figure 2.9 Directory Confirmation Screen

- Click *Next* to begin installation.

The installer then copies the HEW (Pure Debugger) for CPUBD files to the specified directory:

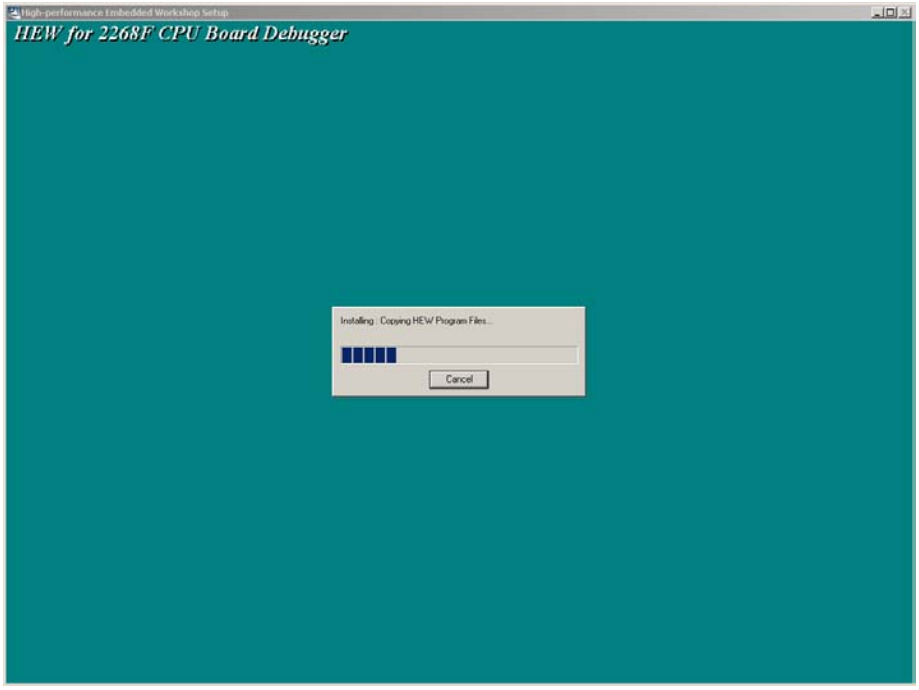


Figure 2.10 Installing Screen

The installation will complete with the Completion screen:

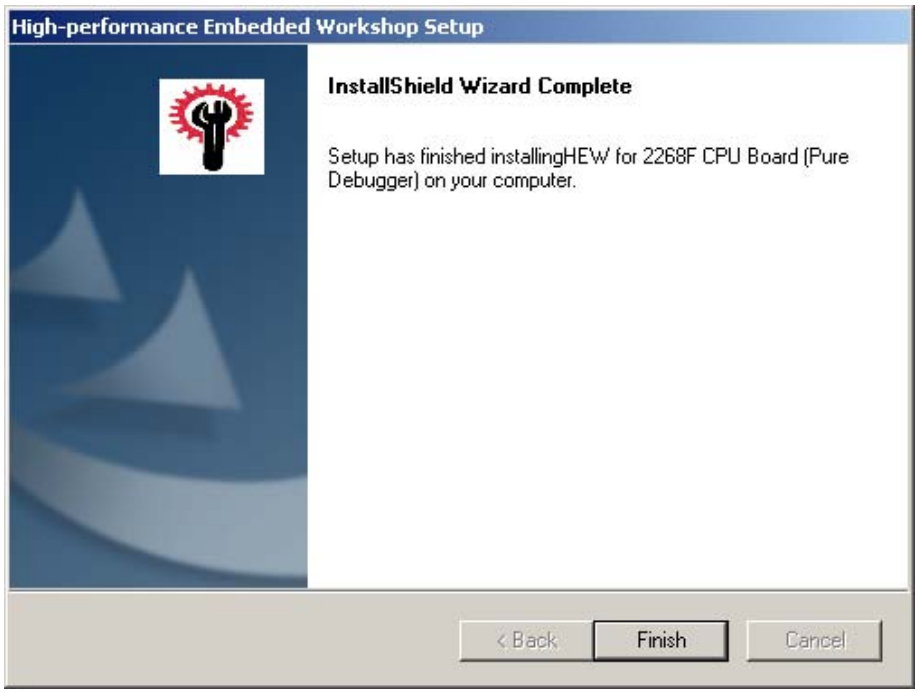


Figure 2.11 Completion Screen

At the end of the installation, icons for HEW (Pure Debugger) CPUBD will be created into the *Start Menu* and ready for execution.

2.7. Registering Toolchains

The HEW (Pure Debugger) for the 2268F CPU Board does not come with any free toolchain. If the user has previously installed other versions of HEW with toolchains, these toolchains can be registered and used on the current installed version. Toolchain is a HEW component that enables the user to create, compile and like a project.

- Click on *Administration* in the *Tools* menu.
- Click on the (+) to expand the tree to view the detail components.

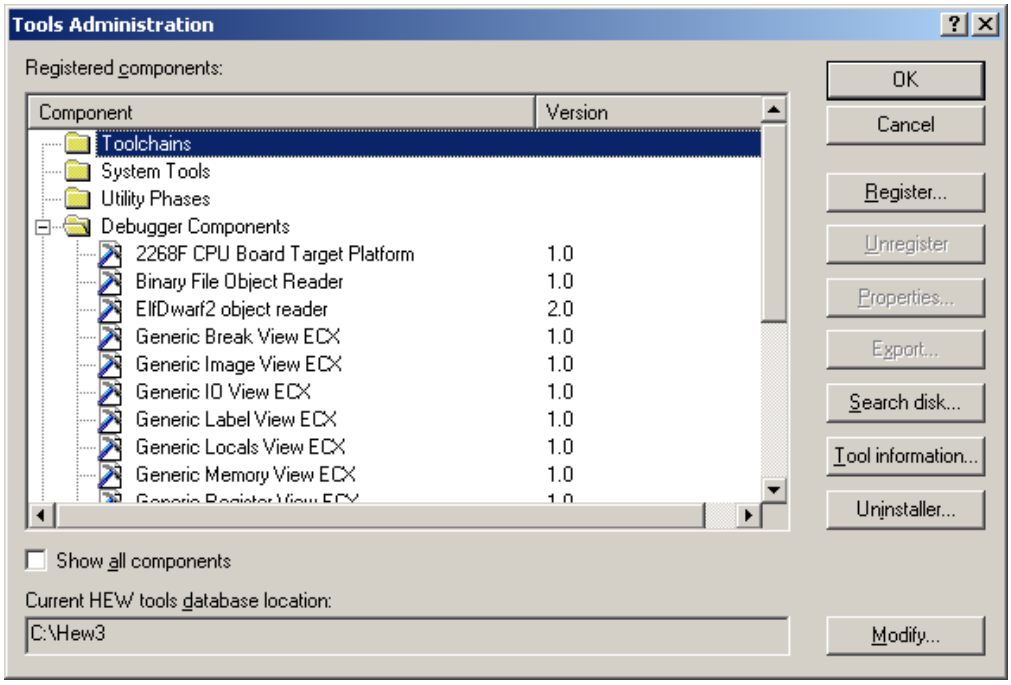


Figure 2.11 Tools Administration Window

- Click on Search disk…
- Click on Browse and select the location of the previous version of HEW.

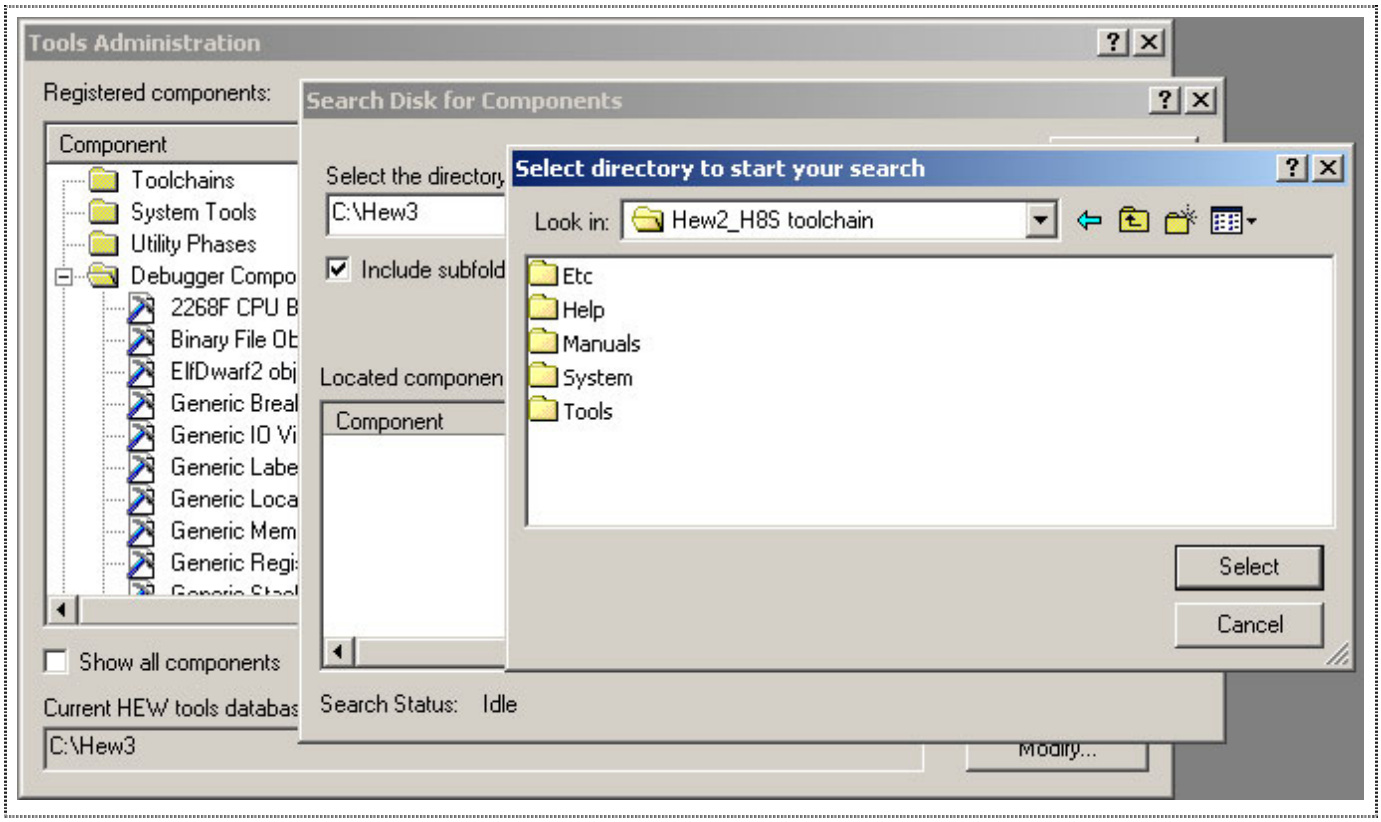


Figure 2.12 Locating Previous Versions of HEW

- Click on Start to start searching for components in the selected directory.
- Select the required toolchain and click Register.

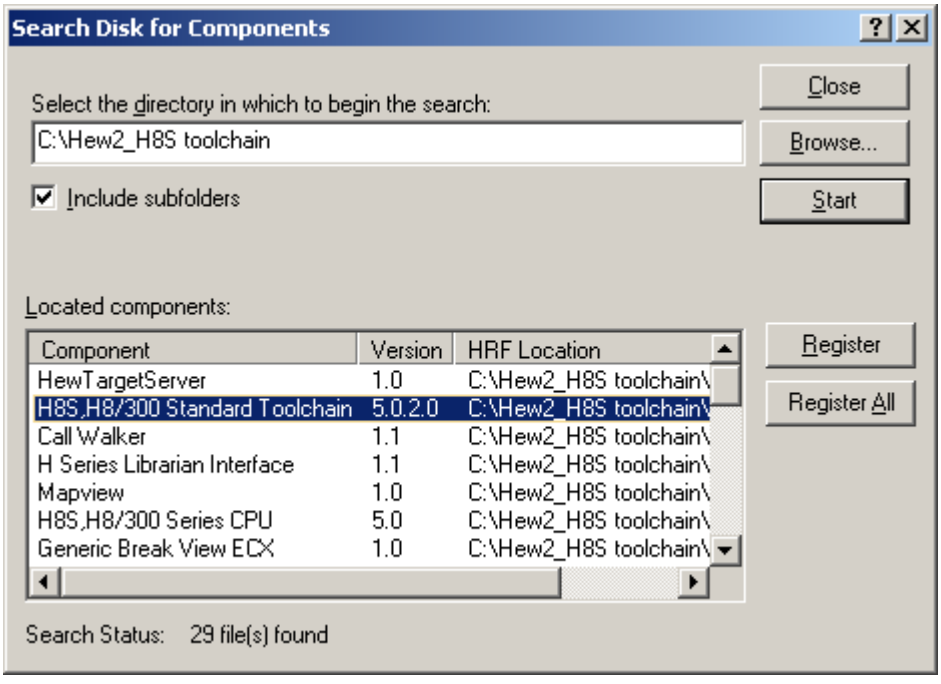


Figure 2.13 Searching for HEW Components

- Click OK to exit the Tools Administration window. The toolchain is now installed.

Section 3. Setup of HEW (Pure Debugger) for CPU Board

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.

Ensure that the CPUBD is linked up i.e. the serial cable is linked between the CPUBD and PC, and the CPUBD is powered up.

3.1. Running HEW (Pure Debugger) for CPU Board

- Execute HEW (Pure Debugger) for CPUBD by selecting HEW.

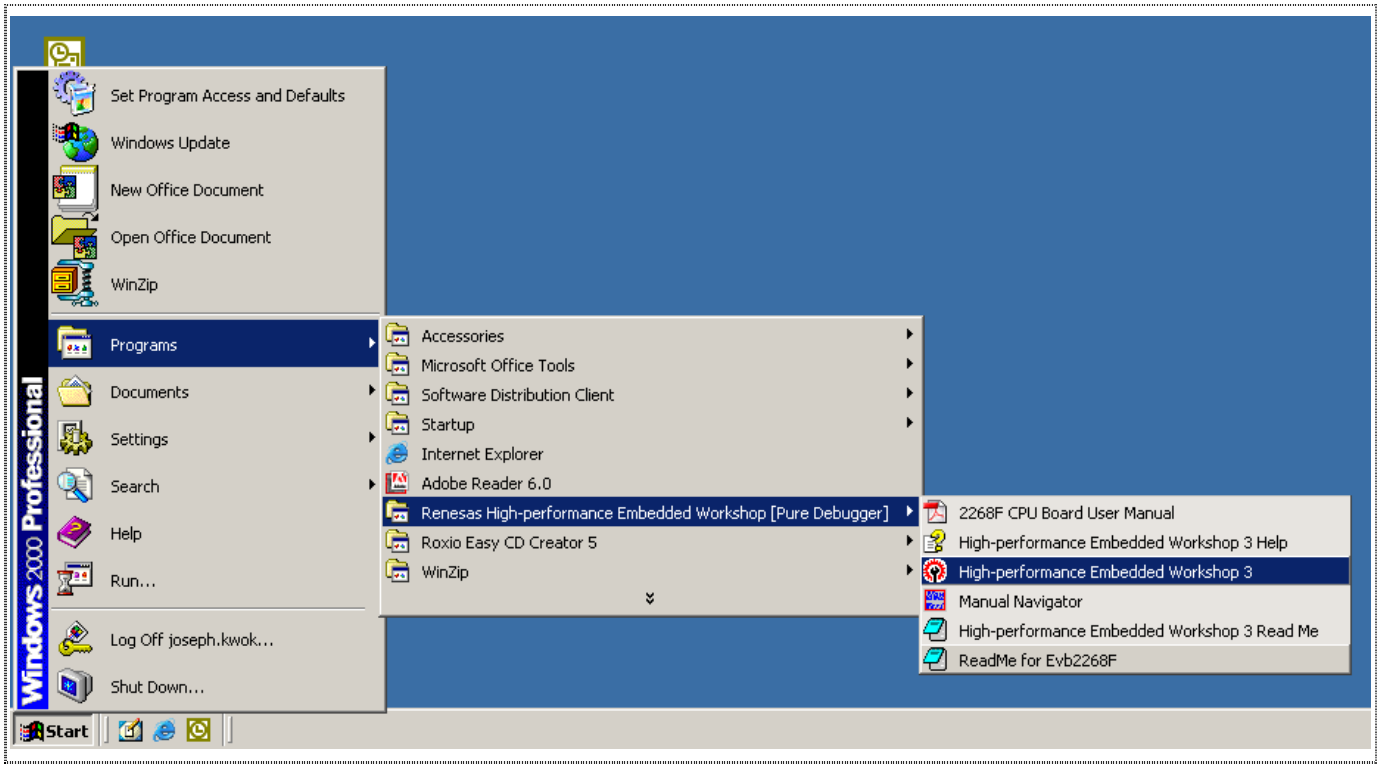


Figure 3.1 HEW (Pure Debugger) for CPUBD Icon

3.2. Creating a New Workspace

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 enables the user to create, compile and link a project.

3.2.1. 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]

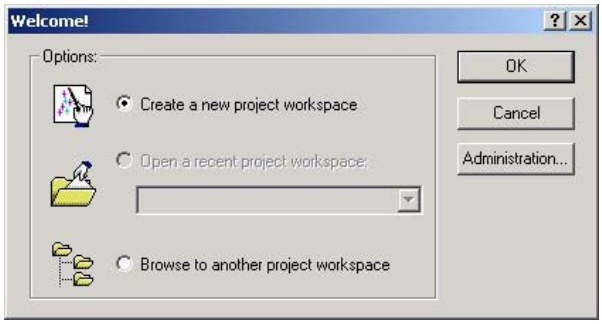


Figure 3.2 Select Platform Dialogue Box

- Select a directory and key the workspace name as required

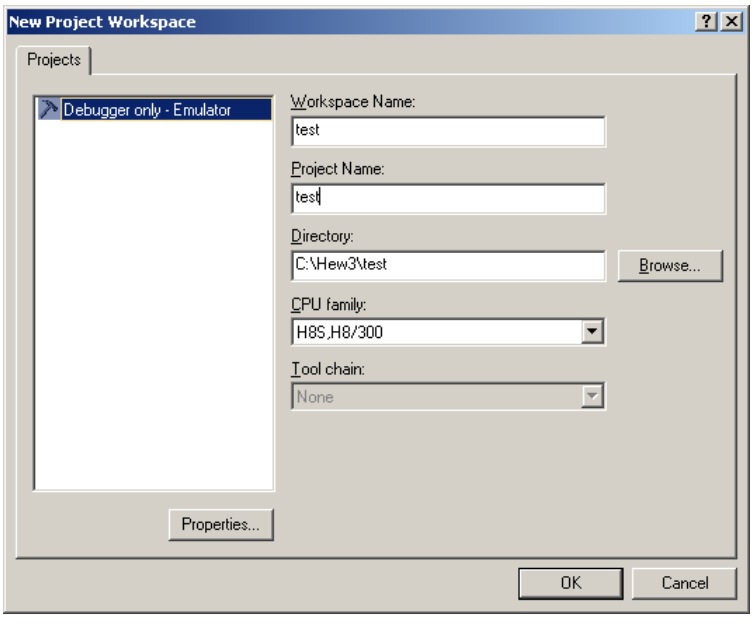


Figure 3.3 HEW Start-Up Window (without toolchain)

- Select 2268F CPU Board as the target

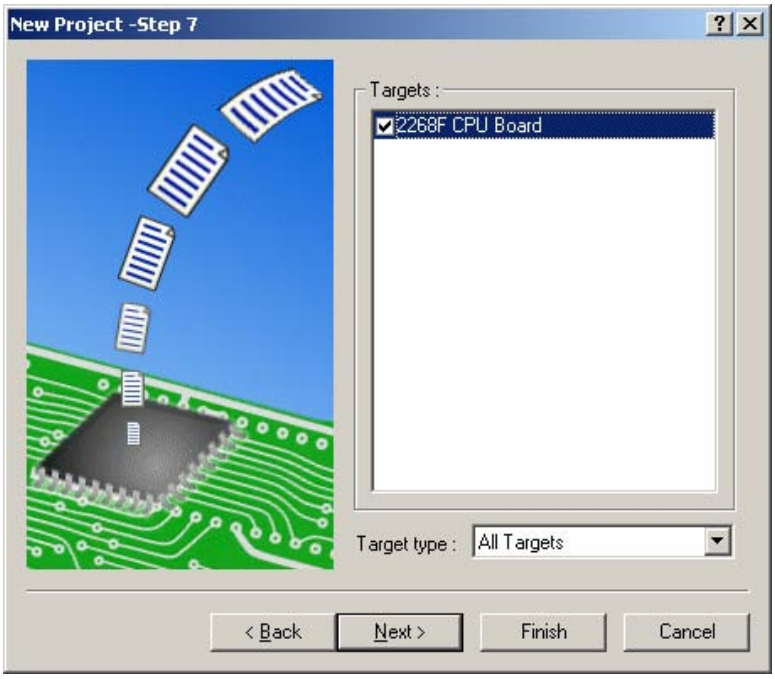


Figure 3.4 Select Target

- Click on [Next>] button

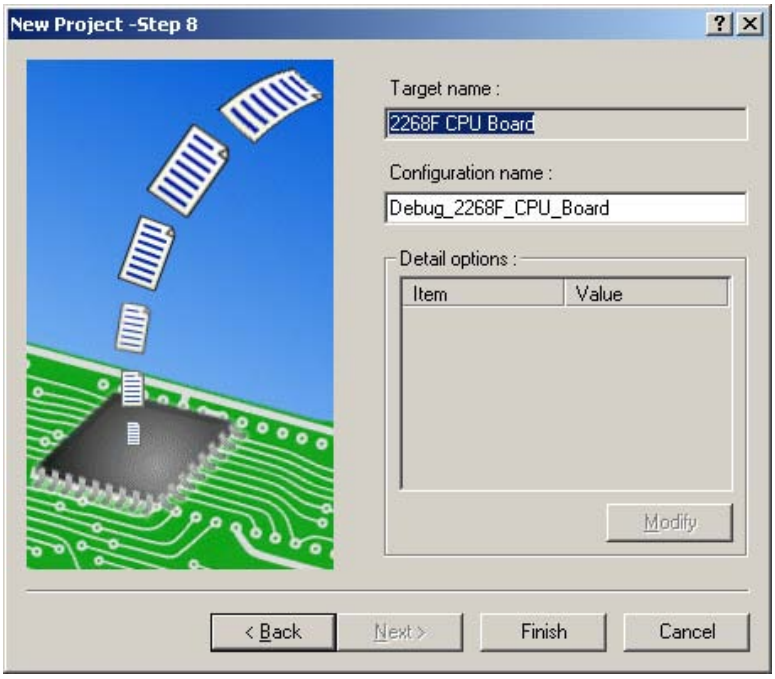


Figure 3.5 Debugger Configuration

- Click on [Finish] button

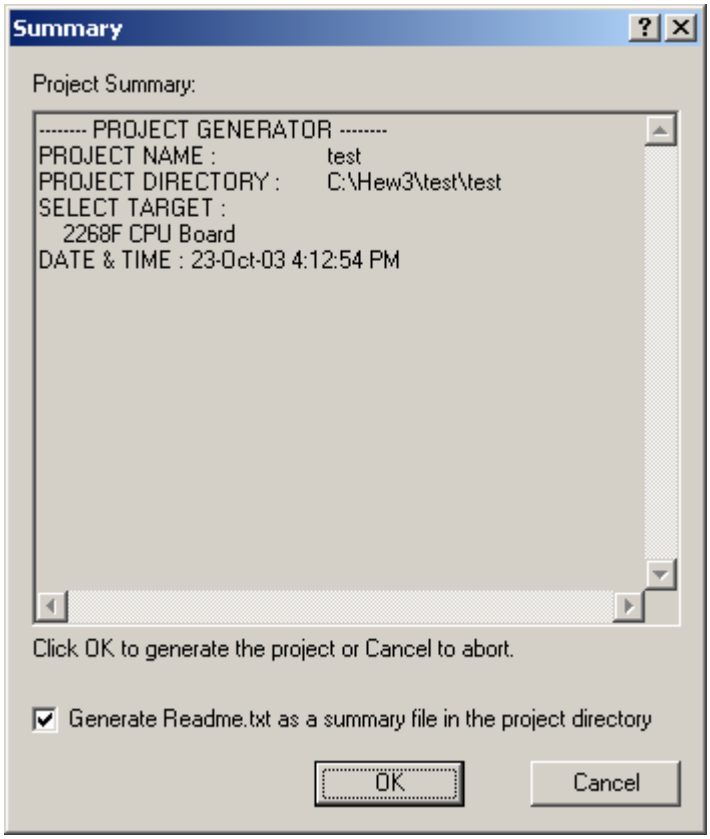


Figure 3.6 Debugger Setting Summary Window

- A summary window shows the project files that will be generated
- Click OK to proceed

Refer to Section 3.3 to proceed with the HEW setup.

3.2.2. With Toolchain

When a toolchain is registered (refer to Section 2.7 for registering toolchains), user is able to create and compile codes. Creating and compiling of codes is detailed in the HEW user manual.

❑ Click on [Create a new project workspace]

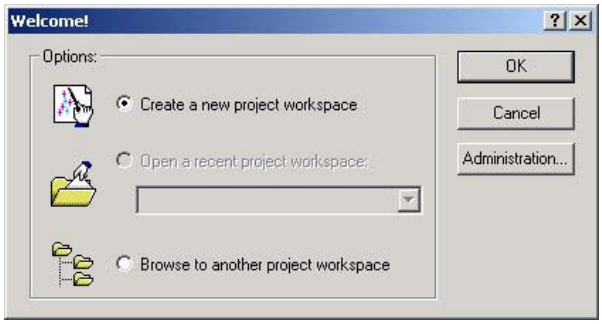


Figure 3.7 Select Platform Dialogue Box

❑ Select a directory and key the workspace name as required

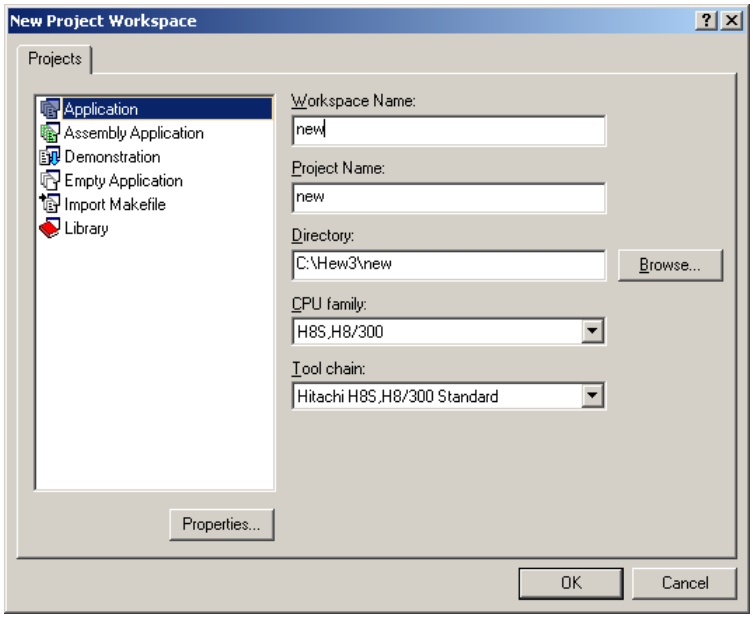


Figure 3.8 HEW Start-Up Window (without toolchain)

- Select 2268F CPU Board as the target by selecting
  - CPU Series: 2000
  - CPU Type: 2268

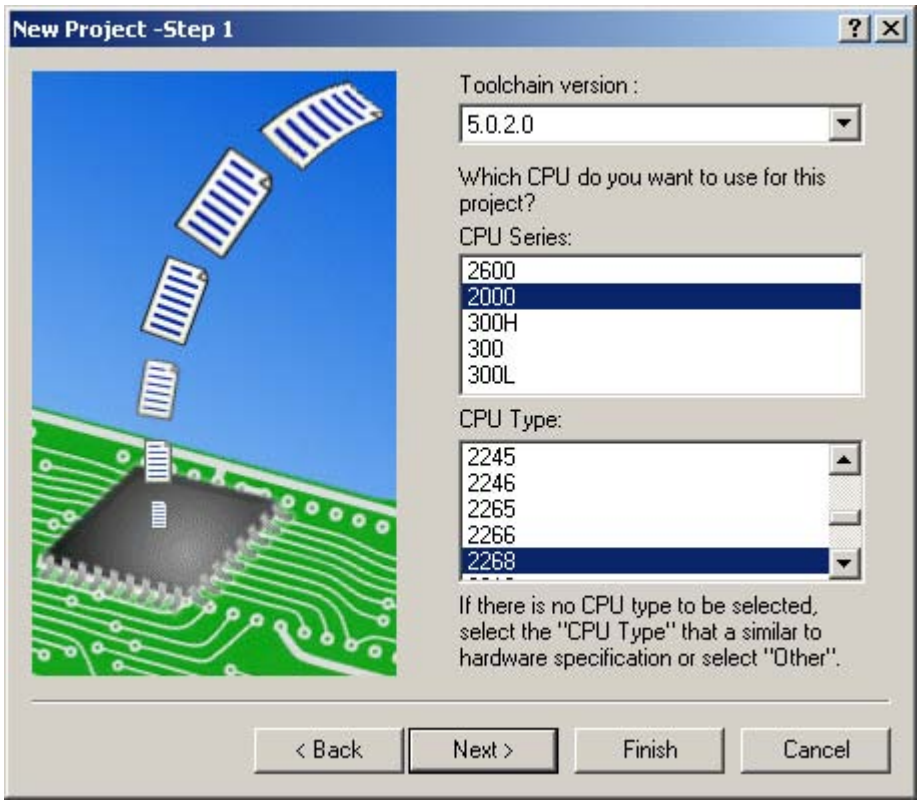


Figure 3.9 Select Target

- Complete the workspace setup by clicking on [Finish] button
- A summary window will pop up, showing the project files that will be generated
- Click OK to proceed

3.3. Selecting the Target (Debug Settings)

HEW (Pure Debugger) for CPUBD can be extended to support multiple target emulators or platforms (if the system is setup for more than one platform), user will have to choose a platform for the session from *Debug Settings…* in the *Options* menu.

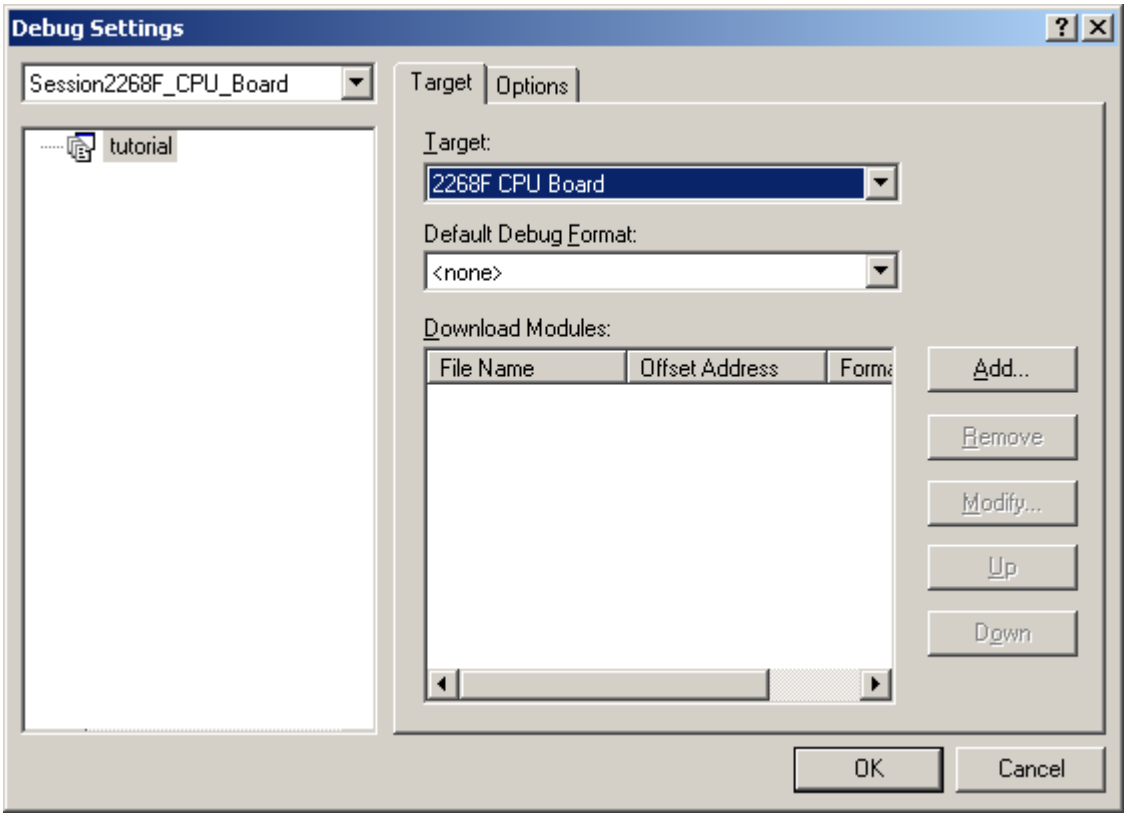


Figure 3.10 Select Platform Dialogue Box

- Select *'2268F CPU Board'* and click OK to continue
- A warning message will pop up. Click "OK" to proceed

NOTE: User can change the target platform at any time by choosing *Debug Settings…* from the *Options* menu. Under the *Download Modules*, User can also define the Download Module/s for Debugging.

When the emulator has been successfully setup, the HEW (Pure Debugger) for CPUBD desktop window will be displayed. A message *Connected* is displayed in the Output Window.

Section 4. Performing Emulation

4.1. High-performance Embedded Workshop

The following shows a snap shot of the HEW (Pure Debugger) desktop Window:

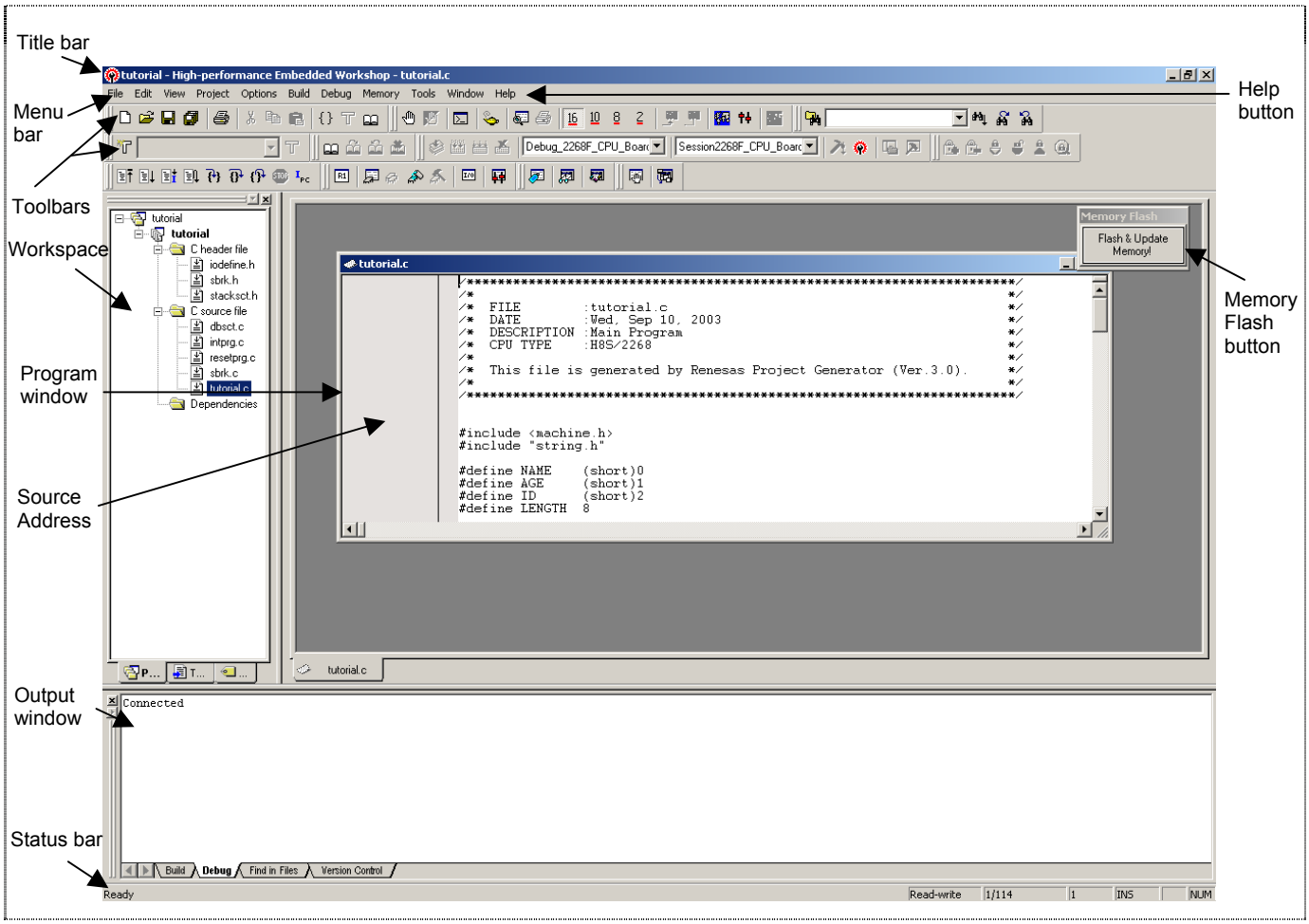


Figure 4.1 High-Performance Embedded Workshop Window

The key features of HEW (Pure Debugger) for CPUBD are described in the following sections:

Title Bar : Displays the name of the currently open workspace, project and file.

Menu Bar : Give you access to the HEW (Pure Debugger) for CPUBD debugging commands for controlling CPUBD.

Toolbars : 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.

Program Window : Displays the source code of the program being debugged as well as the source address.

Workspace : Display the detail of current workspace, and provide a quick & easy mean of navigation.

Output Window : Displays the various outputs from HEW. For example, build details, results of find files.

Status Bar : Displays the status of the CPUBD. For example, progress information about downloads.

Help Button : Activates context sensitive help about any feature of the HEW (Pure Debugger) for CPUBD software.

Memory Flash Button : Flash contents of the memory window for on-chip ROM area into the MCU. User is required to press this button when he/she manually updates the contents of the memory window for on-chip ROM* area. This is not required for RAM* area.

NOTE: * Please refer to the *Appendix B – H8/2268F Memory Map* for the on-chip ROM and RAM areas.

The major topics are highlighted as follows.

	Menu	General Description	Sub Menu	Usage
1	Option	Emulation Setting	Debug Settings	Target Selection
			Emulator	View memory mapping and Configure Platform
2	View	MCU related information	Disassembly	View disassembly window
			CPU	Register, memory, Status, I/O
			Symbol	Label
			Code	Breakpoints
3	Memory	MCU memeory manipulation	Fill	
			Refresh	
4	Debug	Execution of MCU Code	Reset CPU	
			Go/Reset Go /Go to Cursor/ Set PC to Cursor /Run	
			Step In/ Over/ Out/ … Step mode	
			Initialize	

4.2. Compiler Configuration & Debugger Session

In HEW compiler, every setting is stored in a configuration. The configurations available when a toolchain is registered are different to that when a toolchain is not available.

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.

4.2.1. Session Without Toolchain

At the HEW (Pure Debugger) environment without any toolchain, a default debugger **Session**, [Session2268F_CPU_Board] is created to store information of

- Target platform
- Downloadable program
- Window positioning
- Registers value settings

The default configuration created is [Debug_2268F_CPU_Board].

Figure 4.2 Toolbar Showing the Session and Configuration without Toolchain

4.2.2. Session With Toolchain

At the HEW (Pure Debugger) environment with a toolchain, a default debugger **Session**, [Session2268F_CPU_Board] is created to store information of

- Target platform
- Downloadable program
- Window positioning
- Registers value settings

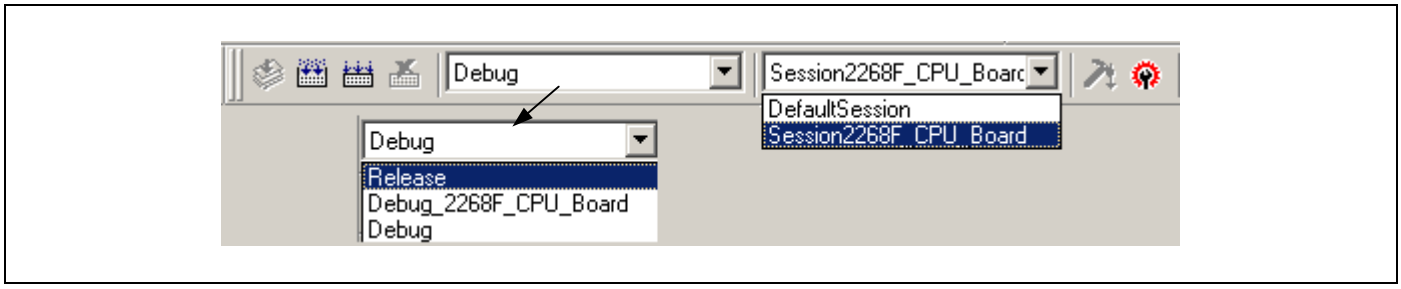
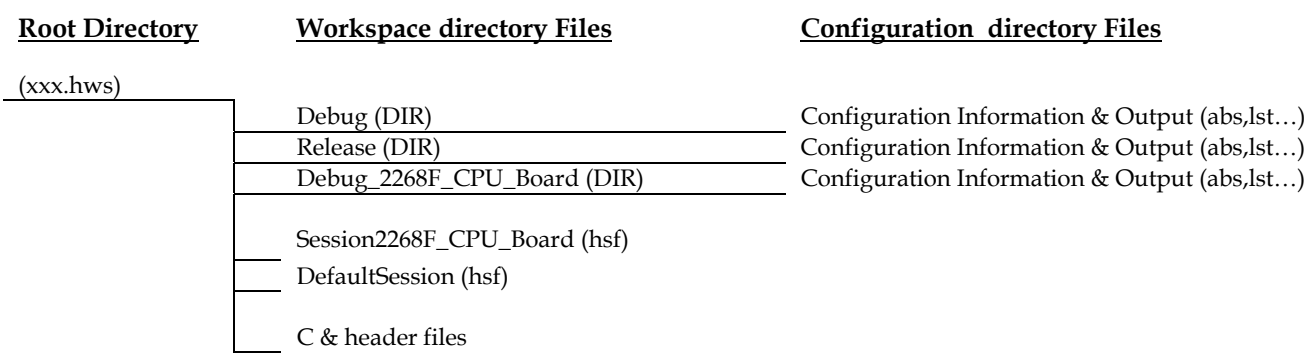


Figure 4.3 Toolbar Showing the Sessions and Configurations with a Toolchain

Generally, the HEW organized the configuration & session of a workspace as follows



Example of usage:

User may use [Session2268F_CPU_Board] to link to CPUBD, & [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]. Otherwise, when the session is switch, the download module will not be able to switch correctly.

4.3. Debug Settings

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

In HEW Pure Debugger without toolchain, users have been provided with the session

- Session2268F_CPU_Board

Whereas, in HEW Pure Debugger with a toolchain, users have been provided with two sessions

- Session2268F_CPU_Board
- DefaultSession

In each session, users are to set

- Target (2268F, Simulator…)
- Default Debug Format (Elf\Dwaf2, S-record, IntelHex…)
- Download module ($(CONFIGDIR)\$(PROJECTNAME).abs)

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 (Pure Debugger) environment with a toolchain, user is provided with two sessions

- Session2268F_CPU_Board (linking with emulator)
- DefaultSession (no target)

Thus by switching between the sessions, 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 done before any emulation.

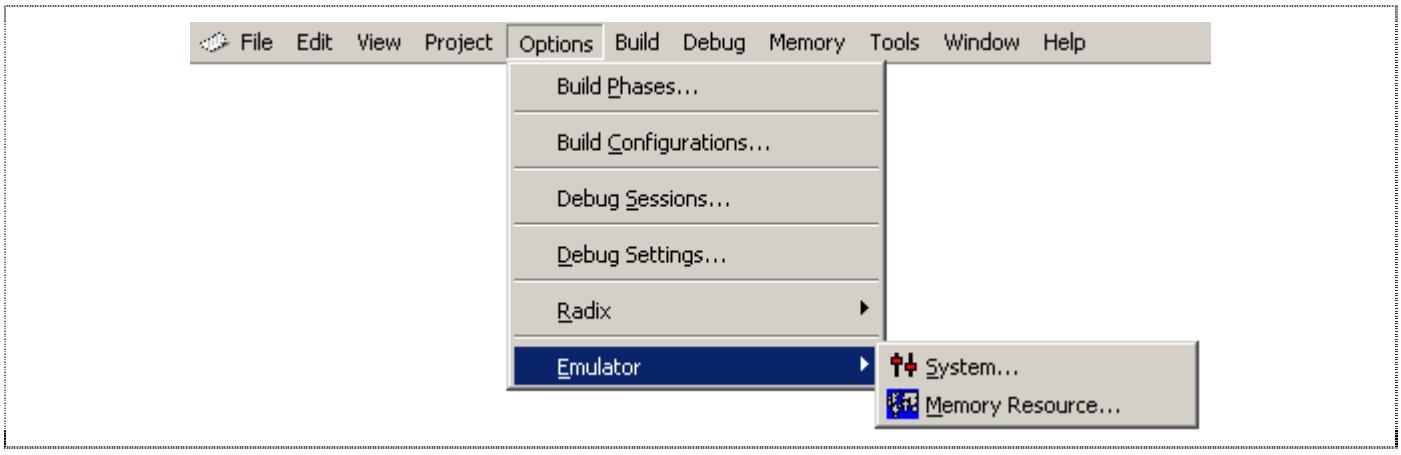


Figure 4.4 Option - Emulator

4.5.1. Configure Platform

The configure platform enables the user to set their target device and mode at startup.

To setup the system configuration:

- From the *Options* menu, choose *Emulator*, *System*… or click on the following icon on the Toolbar:

- The following Configure Platform dialogue will appear:

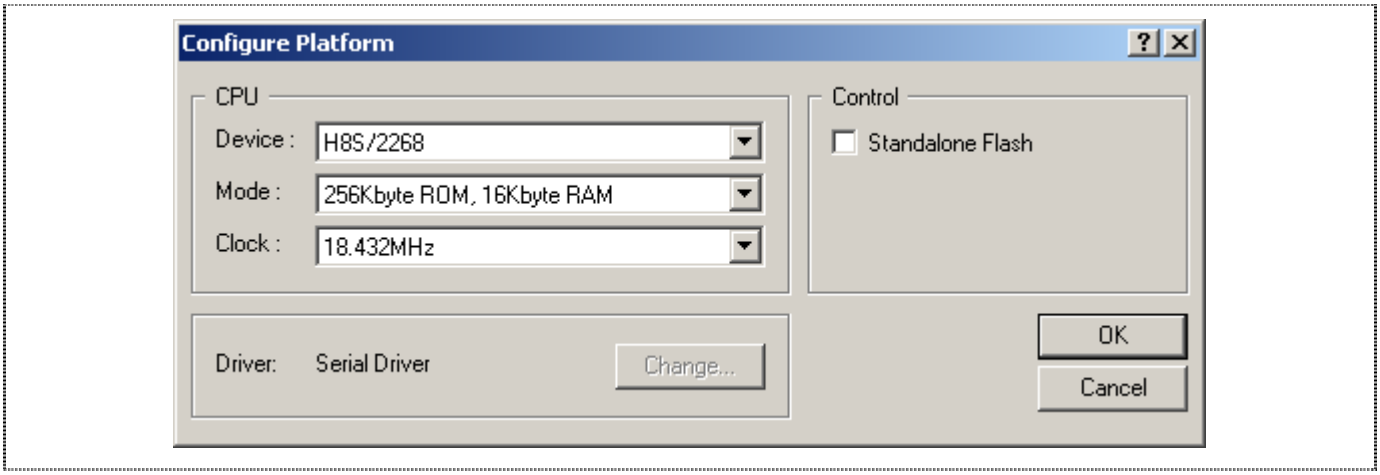


Figure 4.5 Target Configuration Dialogue Box

The user has the option of using standalone flashing by enabling the Standalone Flash in the Control option.

4.5.1.1. Standalone Flash

Standalone Flashing downloads the user target program directly into the memory. Monitor program would not reside in the memory and hence no debugging is available if this option is used. This option should only be used when the user has finalized his/her user target program and wants to run it on the CPU Board.

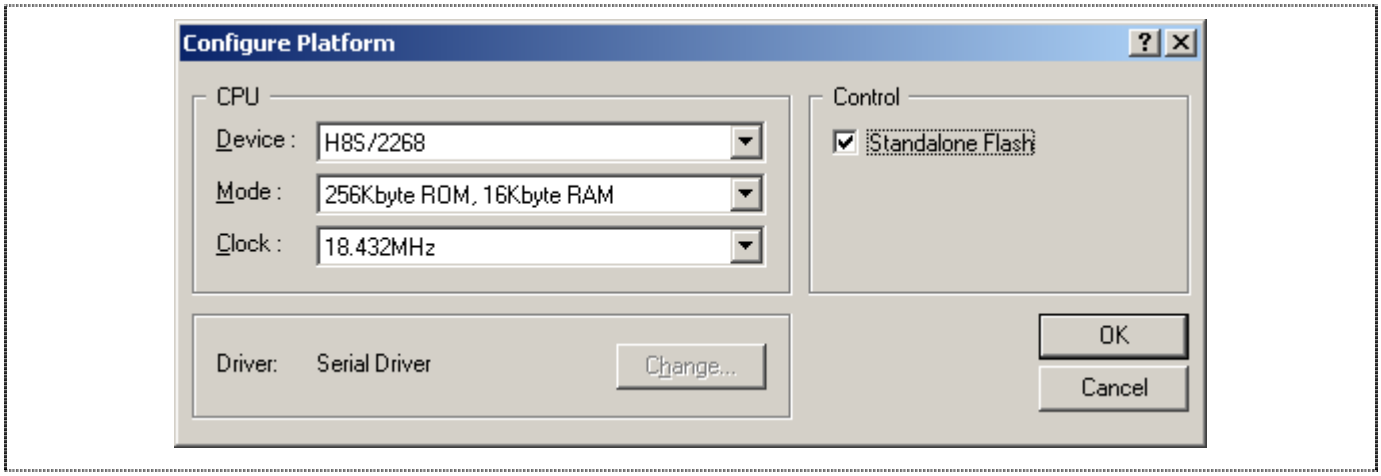


Figure 4.6 Enabling Standalone Flash option

- Click on the check box and click OK to enable standalone flashing.

When user downloads the selected object file, the following dialogue box would appear, prompting the user to switch to Boot Mode to download the user target program.

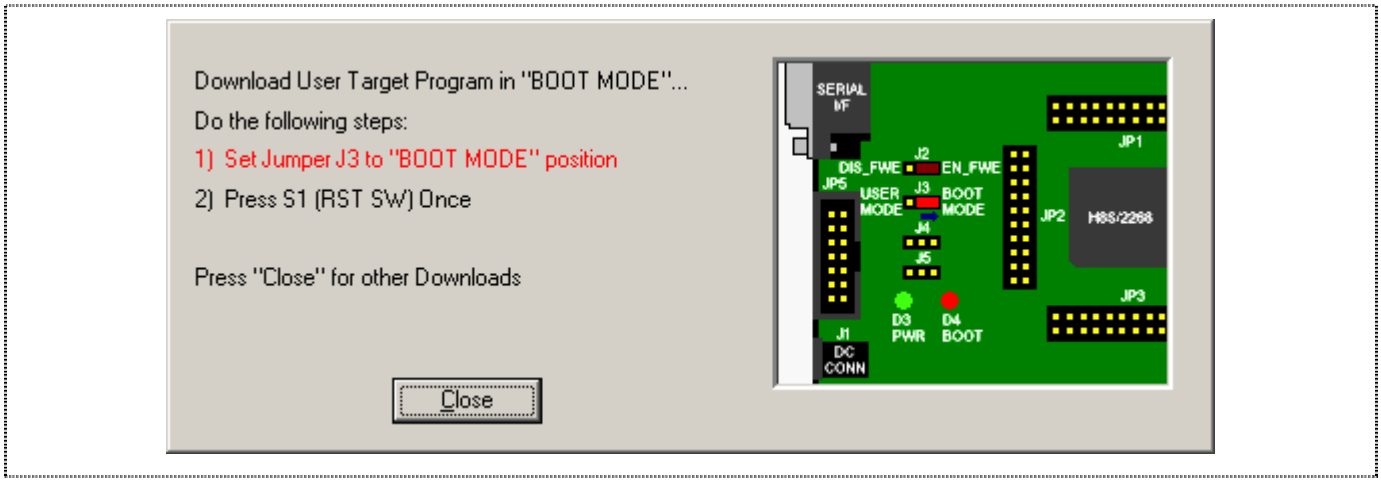


Figure 4.7 Dialogue box for downloading user target program

After downloading the user target program, the dialogue box would prompt the user to switch to User Program Mode to run the user target program. The user can either click YES to exit HEW or click NO to re-download the user target program or Flash monitor Program.

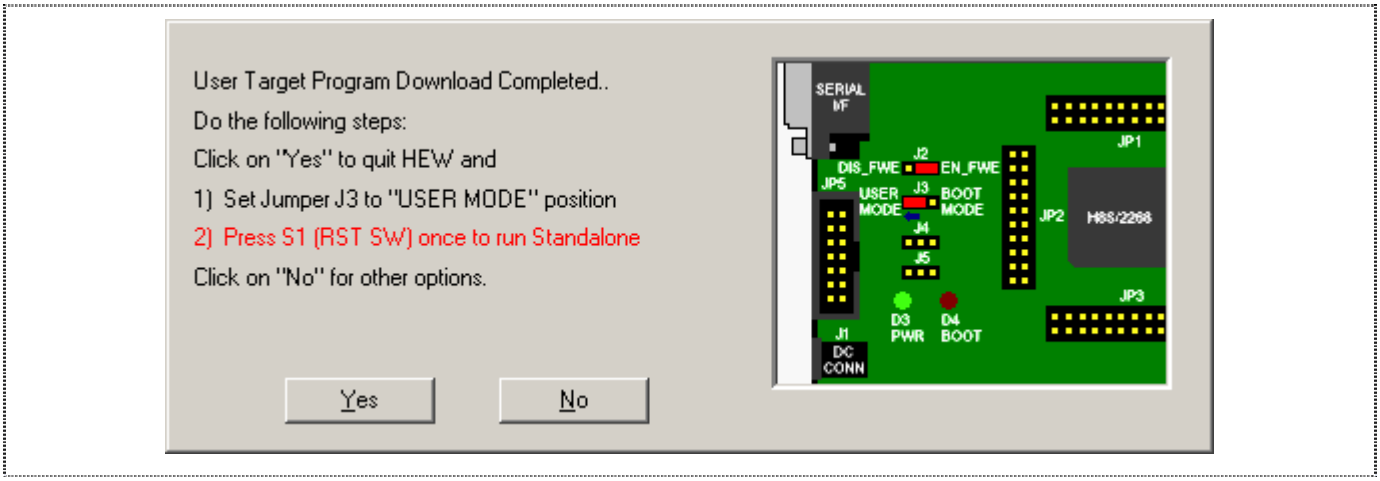


Figure 4.8 Dialogue box for running user target program

NOTE: After pressing the reset switch when jumper J3 is in the User Mode position, the user target program will run in standalone mode, that is, no connection to HEW is required to run the user target program , no debugging is available to user.

Having successfully downloaded the user target program, user can disable FWE to enable Flash Program/ Erase Protection by switching jumper J2 to *DIS_FWE* position, so as to prevent accidental flashing or erasing of the user target program.

4.5.2. Memory Mapping

Once the device and operating mode are selected, the default memory mapping will be set. The main objective of memory mapping is to ensure that the emulator has the correct internal memory (Internal ROM, RAM, IO) access.

To display the current memory mapping:

- From the *Options* menu, choose *Emulator*, *Memory resource*… or click the Open memory mapping button in the toolbar:

The memory mapping is shown in the following figure:

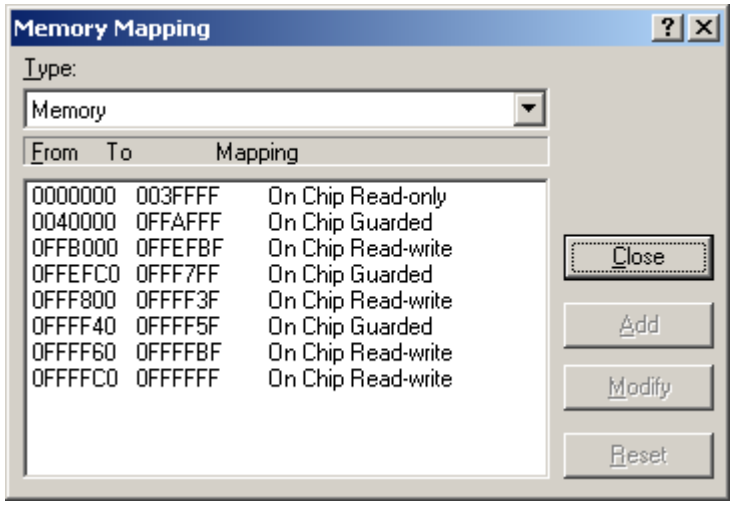


Figure 4.9 Memory Mapping Dialogue Box

Alternatively, the CPU memory map can be viewed from the status window:

- From the *View* menu, choose *CPU* then *Status*, or click the View Status button in the toolbar:

- Select the Memory tab in Status window to show the Memory Mapping configured:

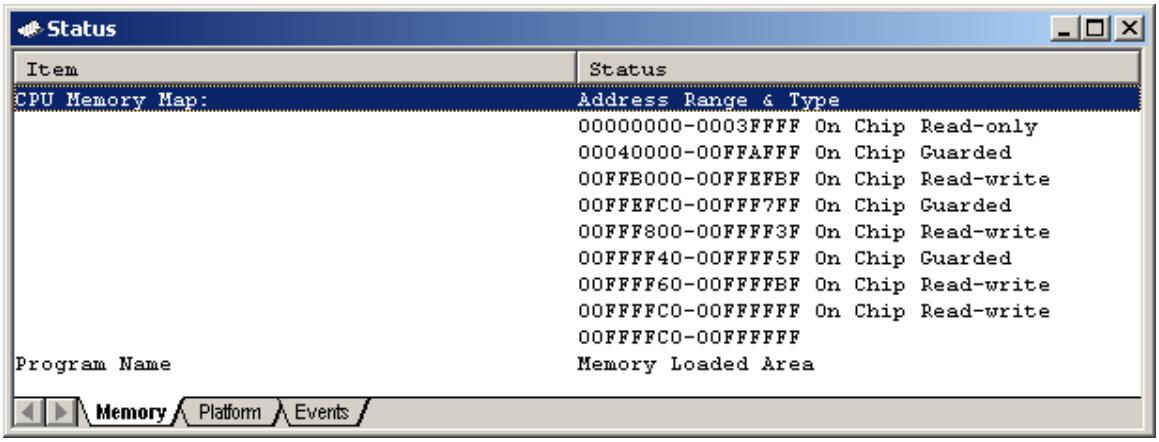


Figure 4.10 Target Memory Configuration Dialogue

NOTE: CPUBD Memory Map is for display and information purpose, user cannot configure it.

The following explains the target memory configuration dialogue:

CPU Memory Map	:	Display the memory configuration of the specific target device selected.
Program Name	:	Display the Downloaded Module's name (User Target Program) and the memory space that it has occupied

4.6. Viewing of Program

Programs can be viewed as

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

4.6.1. Source Code level

Users may double-click on the file located in the workspace window to open and 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 dialogue box to allow user to manually locate the file.**

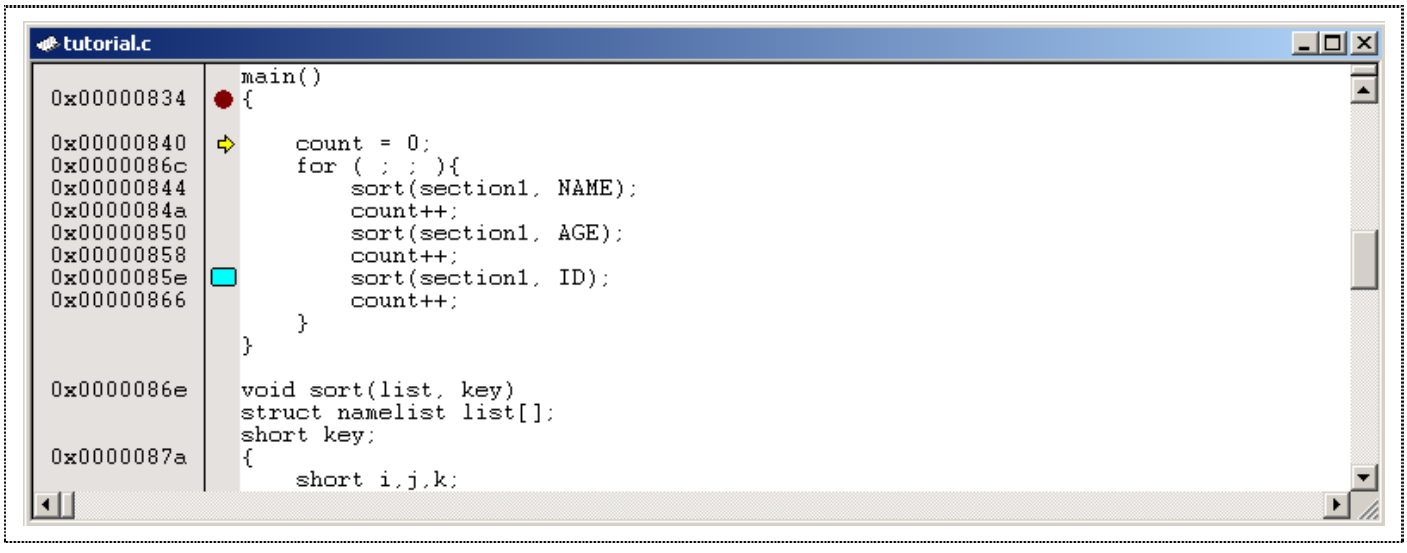


Figure 4.11 Source Level

Information available:

Corresponding address for source file

PC location

Bookmark

Breakpoint

4.6.2. Disassembly level

User can open the disassembly window:

- Choose *Disassembly* from the *View* Menu, or right click on the source window, and select *Goto Disassembly*

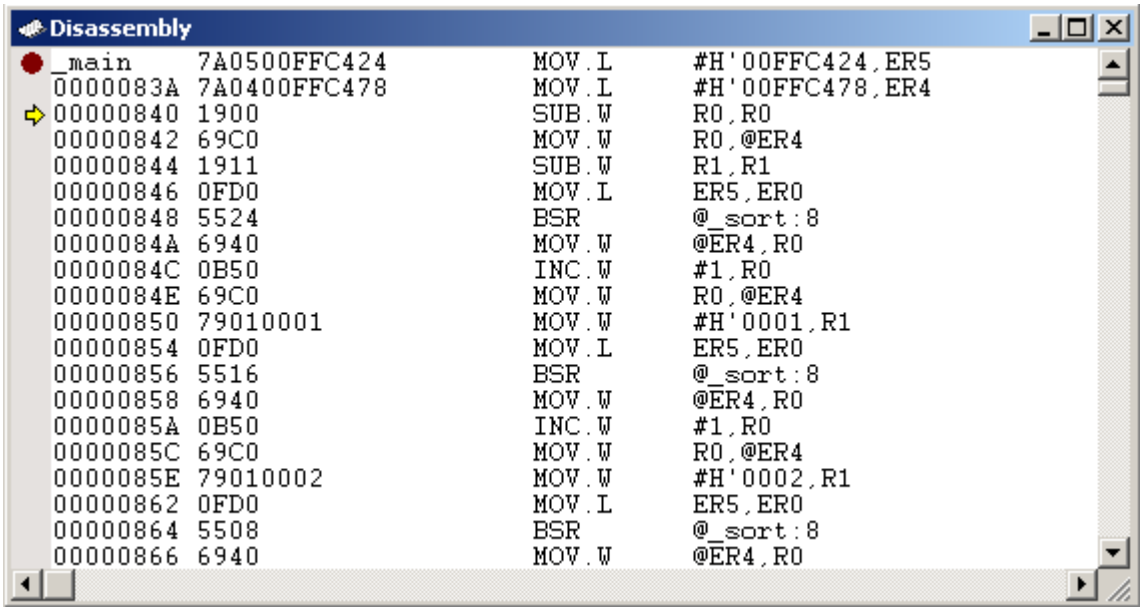


Figure 4.12 Disassembly Window

4.7. MCU related information

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

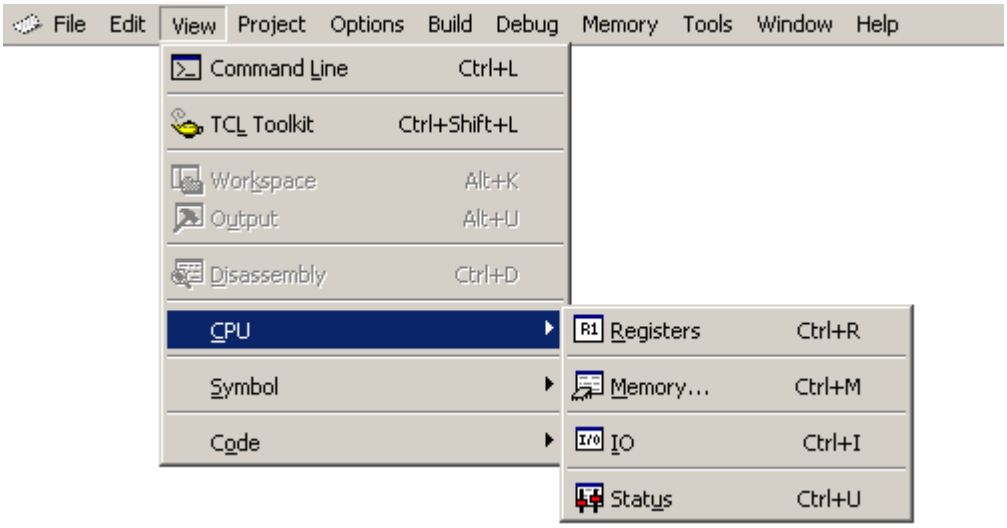


Figure 4.13 View - CPU

4.7.1. Registers

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

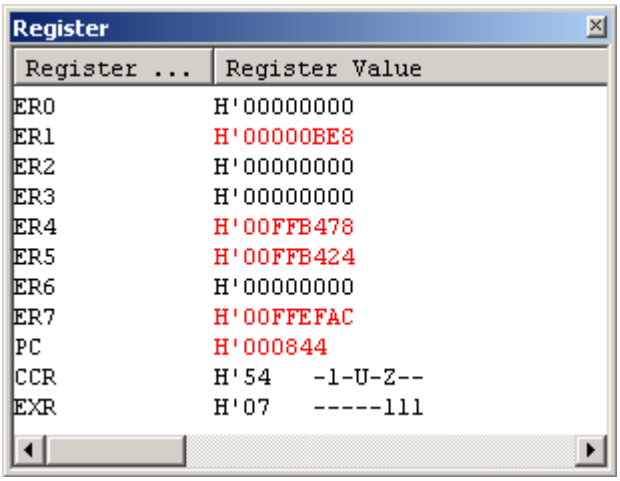


Figure 4.14 Register

4.7.2. Memory

Users will have to set a pre-defined address range to be monitored, 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.

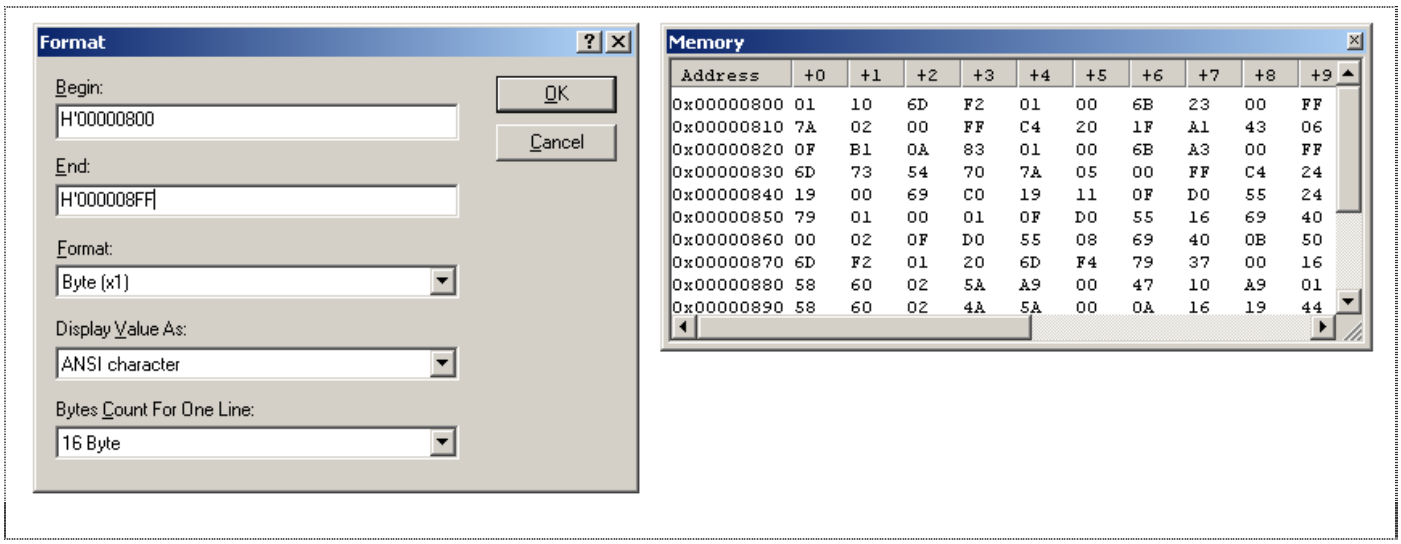


Figure 4.15 Set Memory

4.7.3. I/O

The IO window provides an easy access to MCU IO registers. The Address & Data values of respective peripherals & MCU control registers are displayed in the IO window.

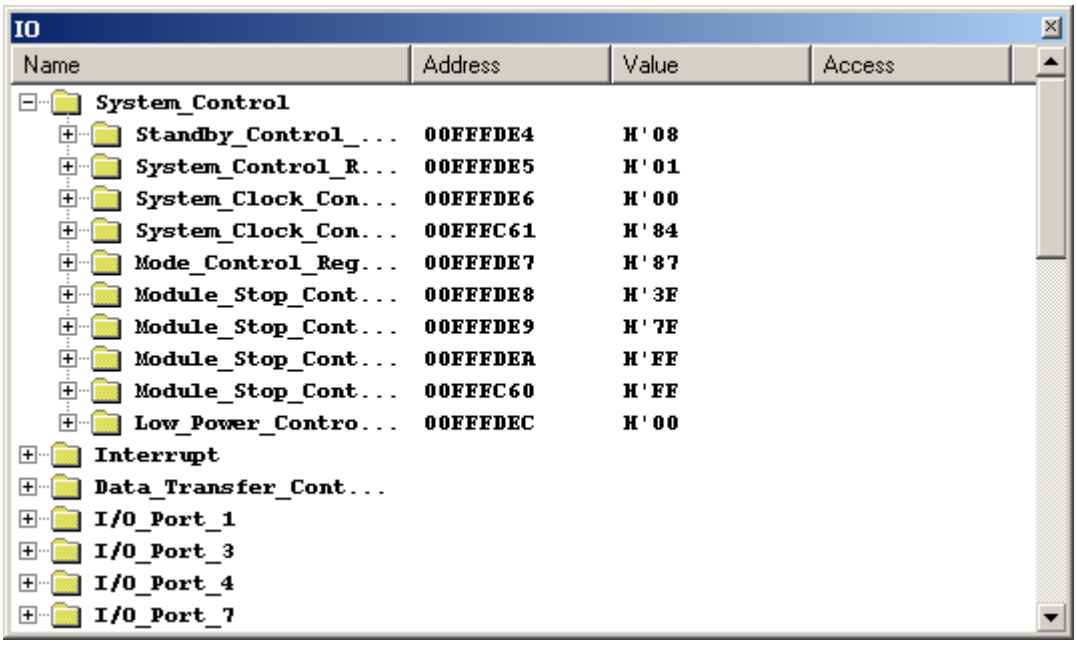


Figure 4.16 Input and Output Register

4.7.4. Status

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

4.7.4.1. Status - Memory

The memory tab display

- the available memory setting for the selected target device & mode.
- the address range where the User Target Program is loaded

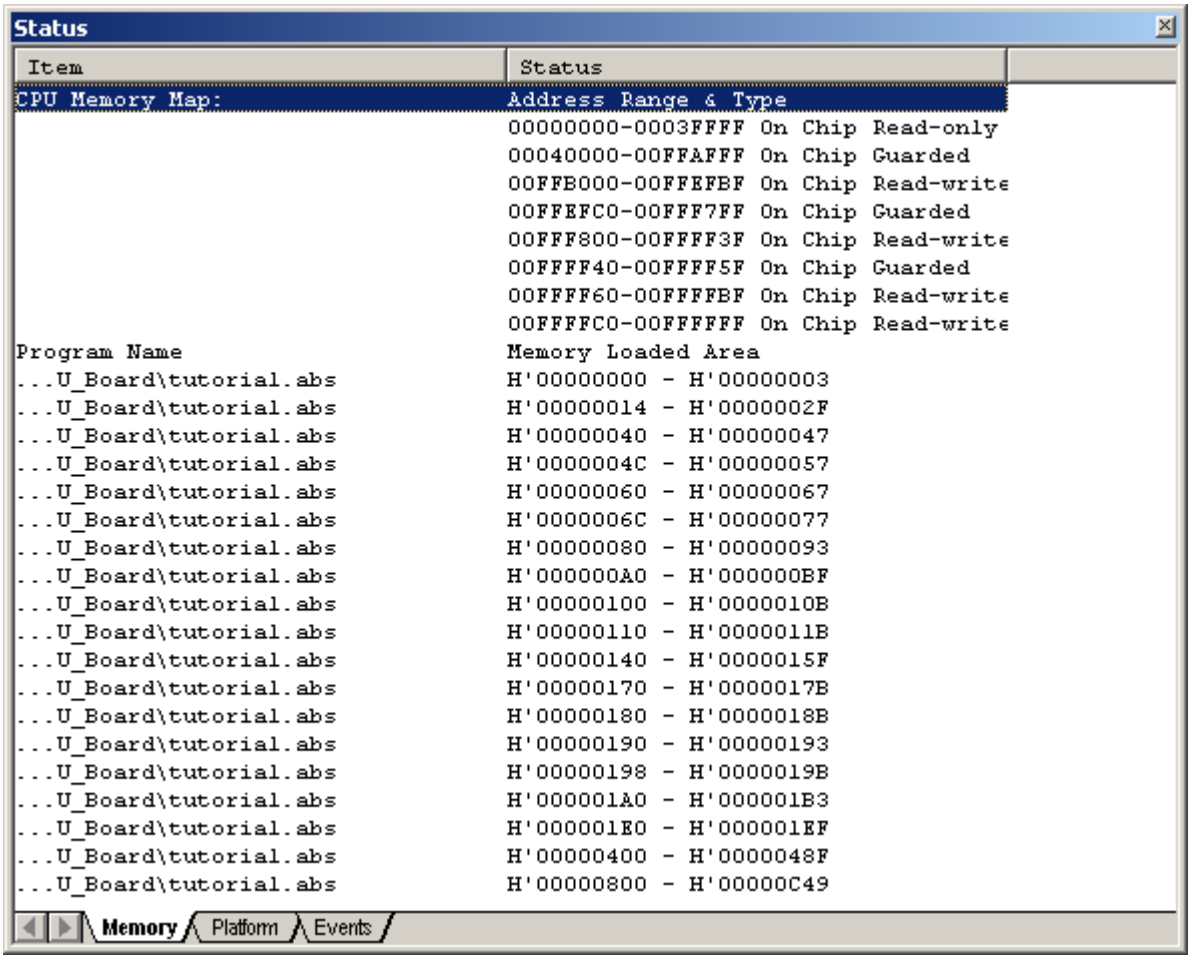


Figure 4.17 Status – memory window

4.7.4.2. Status - Platform

This platform tab shows the current emulation condition

- Target device
- CPU
- Run Status
- Break Cause

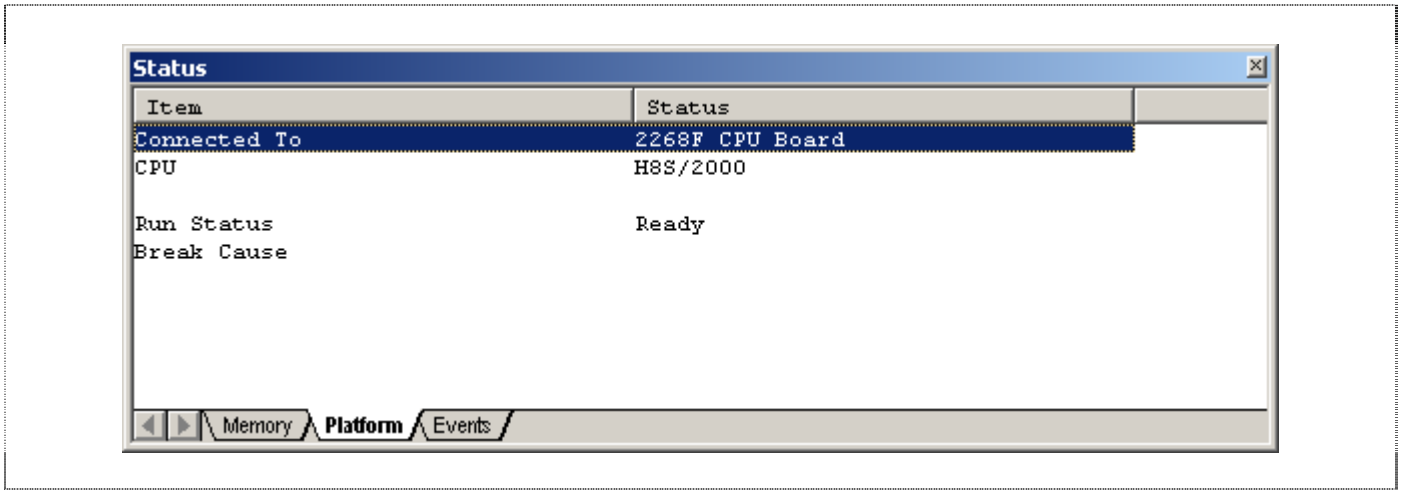


Figure 4.18 Status – Platform window

4.7.4.3. Status - Events

The events tab shows the usage of

- PC Breakpoints

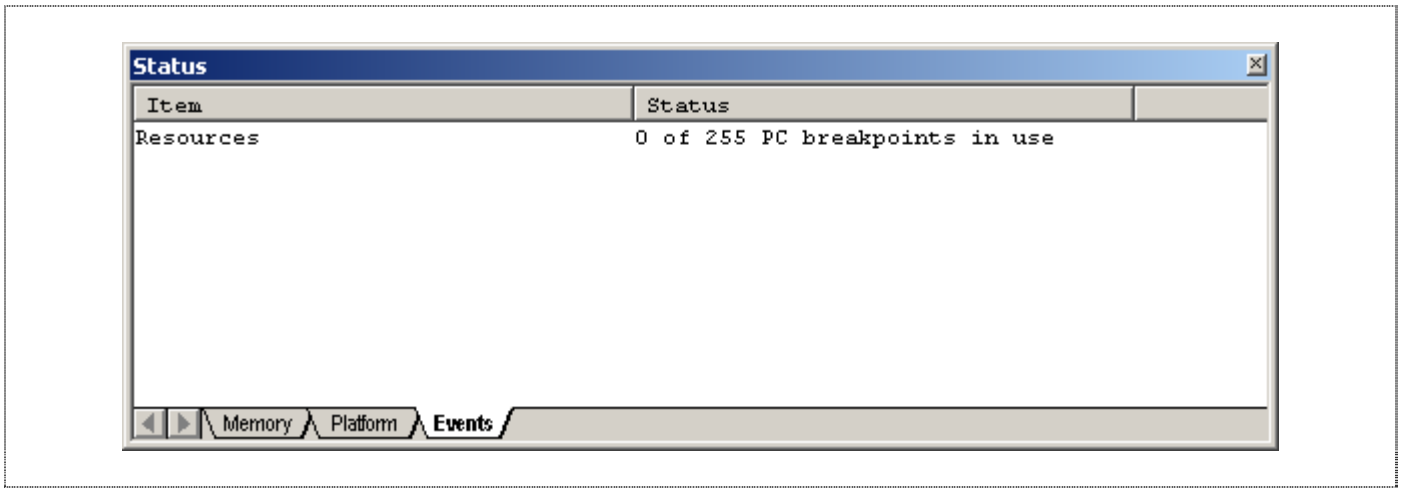


Figure 4.19 Status – Events window

4.7.5. Symbol

This enables easy monitoring of declared variables in the assembly or C files. If debug information is not included, the Watch and Locals sub menus will not appeared.

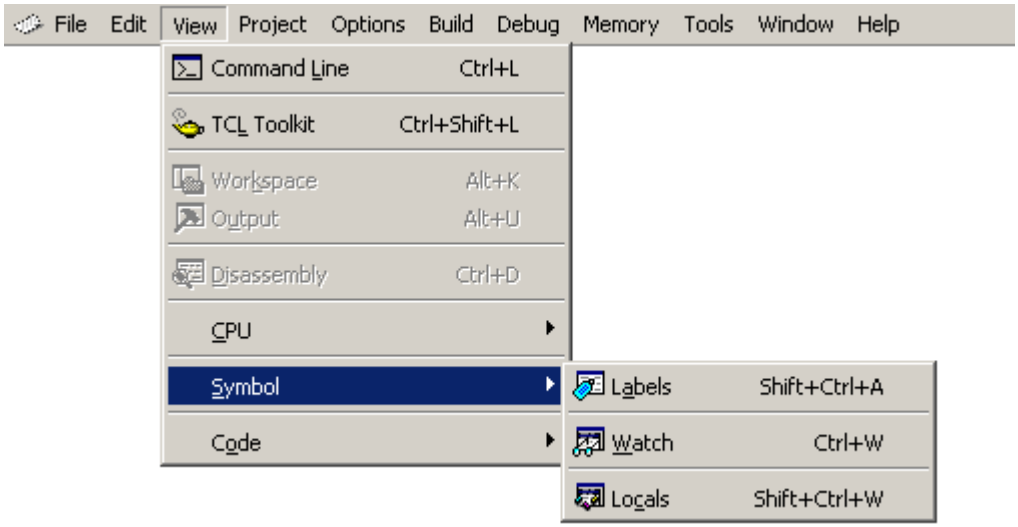


Figure 4.20 View - Symbol

4.7.5.1. 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.

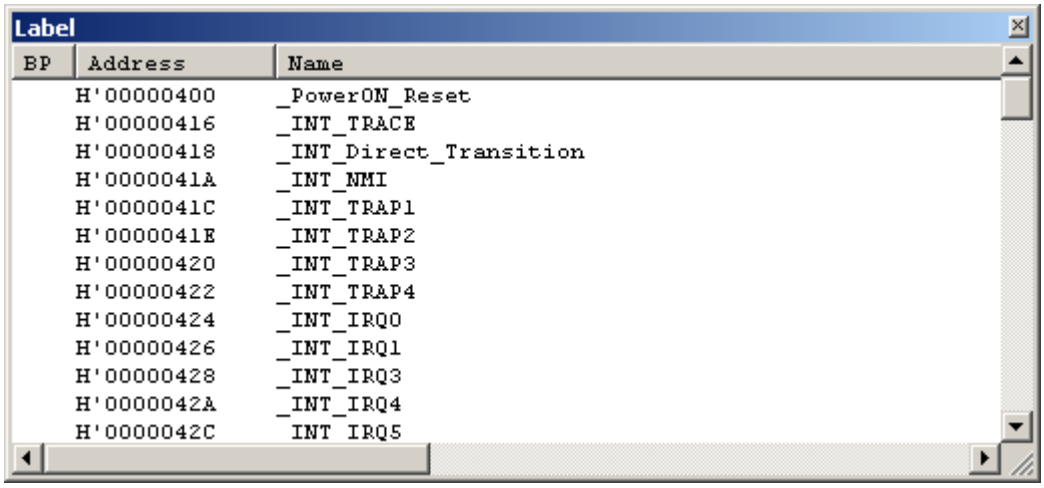


Figure 4.21 Label

NOTE: When a label value matches 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.

4.7.5.2. Watch

User will have to add the variables into the watch window.

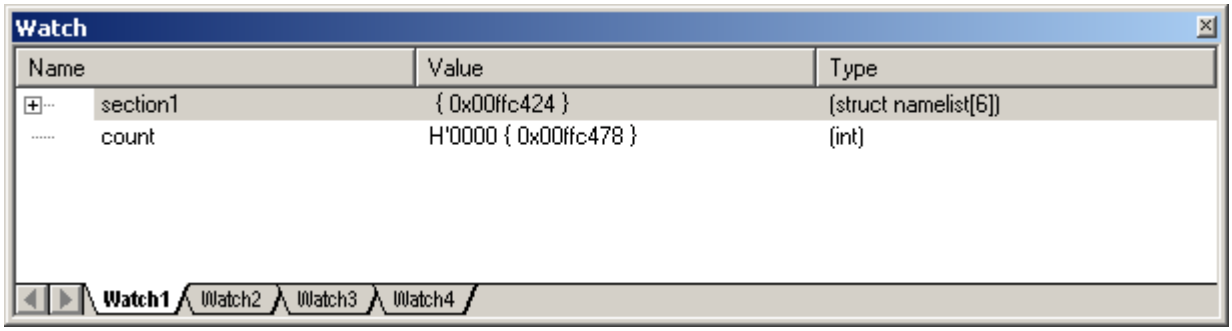


Figure 4.22 Watch

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 complier optimization

- The variables are not cleared as macro.

4.7.5.3. 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.

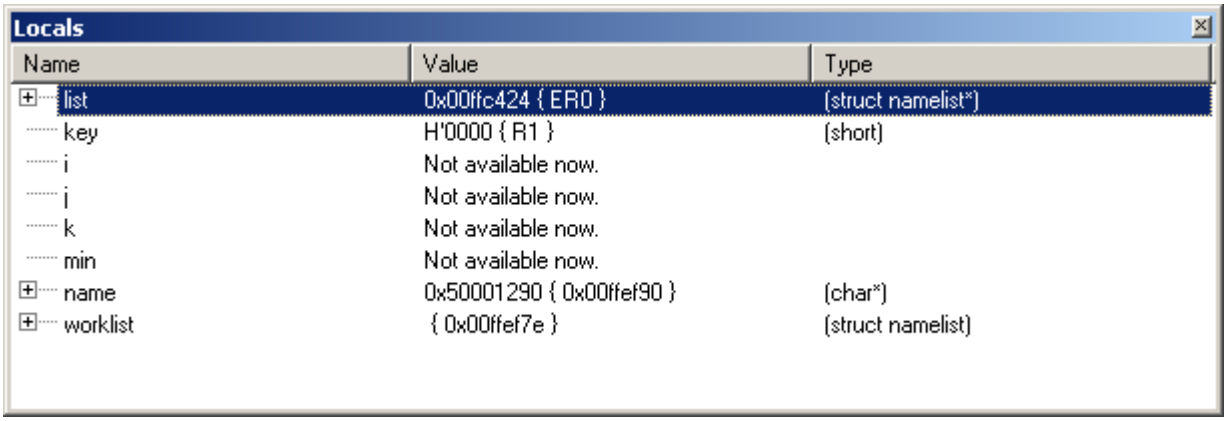


Figure 4.23 Locals

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

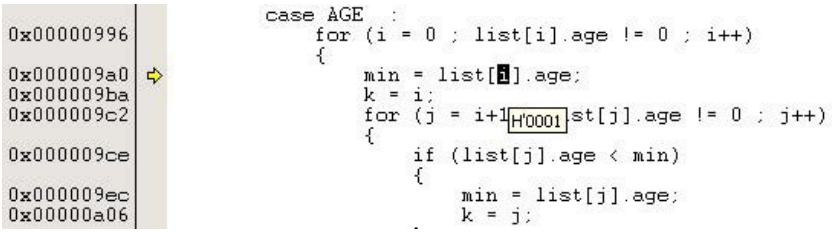


Figure 4.24 Tooltip

4.7.6. Break Functions

Various breakpoints setting are discussed as follows.

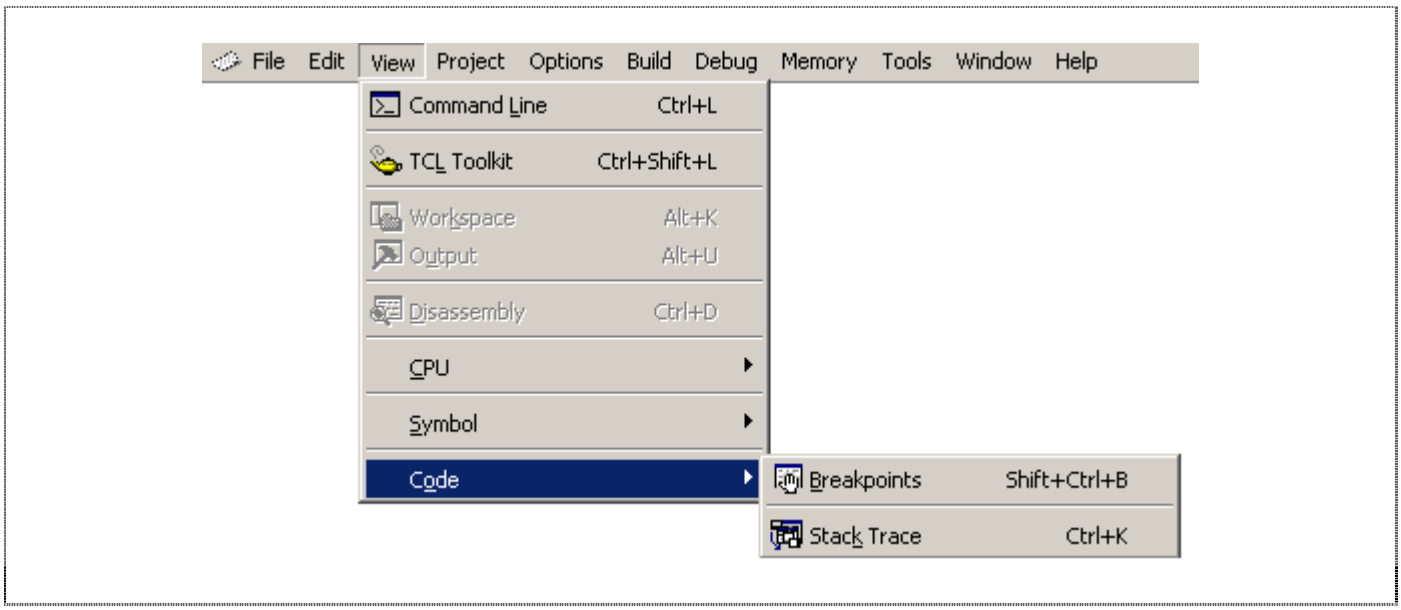


Figure 4.25 View Code

Breaks are events used to intercept the normal program execution when a specific condition is matched. There are two types of break in the 2268F CPU Board, hardware and software 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.

	Types of Break	Description
1	PC Break (Software Break)	A break occurs at the program address specified by PC Break 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.
2	User Break (Hardware Break)	There are 3 scenarios when a hardware break occurs: Pressing the ESC key of the host PC Pressing STOP button of HEW Pressing reset switch of CPUBD

Table 4.1 Types of Breaks Encountered During Emulation

4.7.7. Stack Trace

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

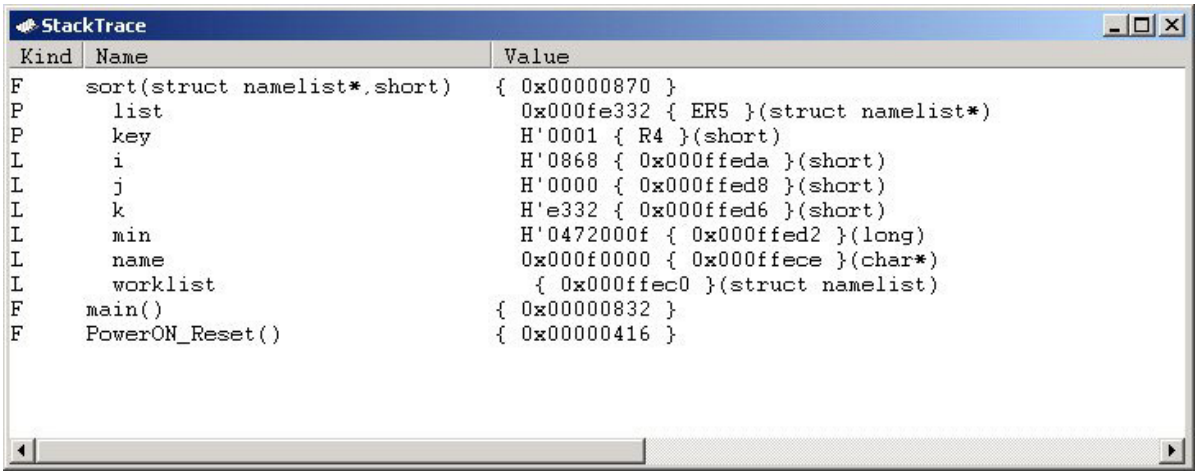


Figure 4.26 Stack Trace

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…*.

4.8. MCU memory manipulation

General supported functions are

- fill
- refresh

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

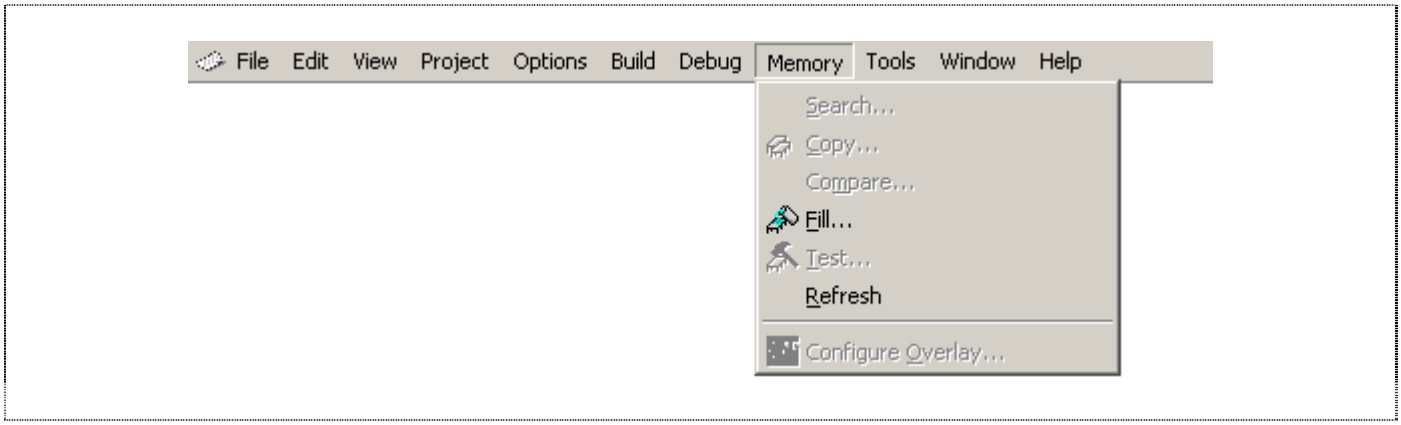


Figure 4.27 Memory Functions

4.9. Execution of MCU Code

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

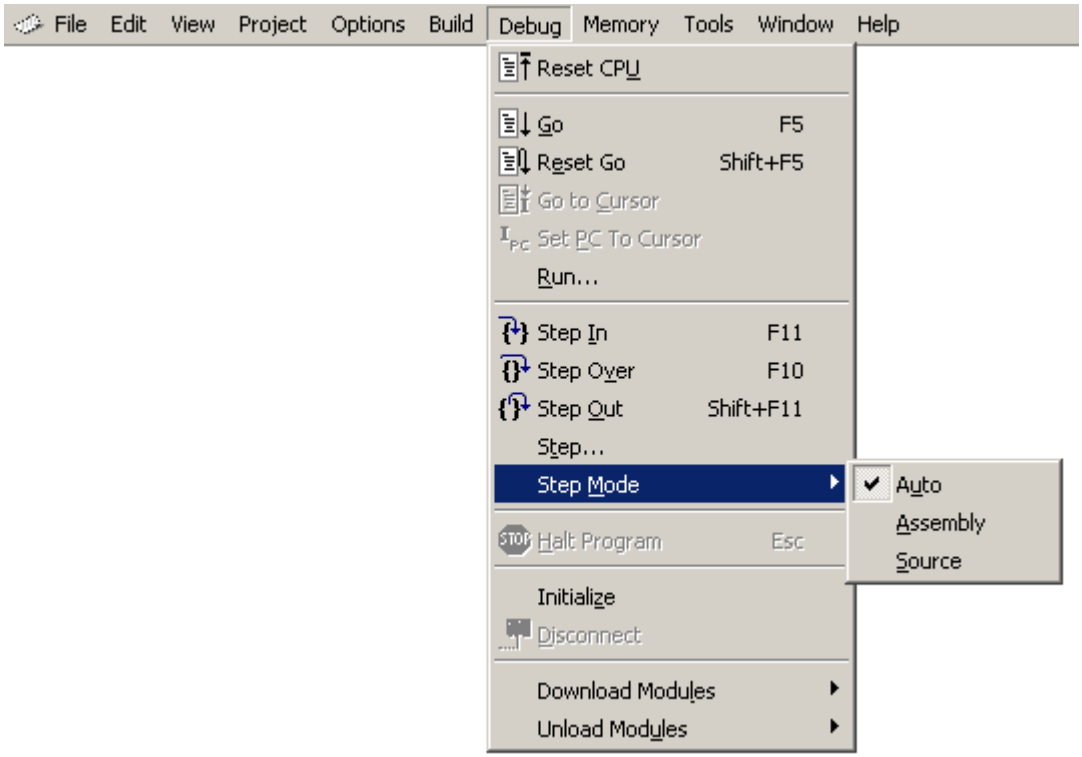


Figure 4.28 Debug Functions

4.9.1. Reset CPU

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

PC	=	Power on Reset vector value
ER7	=	H’FFFFEFC0
ER0-6	=	H’00000000
CCR	=	no change
EXR	=	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…

Near Real-time execution [Debug] by the MCU based on the user setting. These commands will cause the HEW Debugger to steal a cycle from the running chip, in order to probe a response from the MCU to verify that the communication link between the PC and CPUBD is still active.

NOTE: [Go To Cursor] will not halt if the running program never executes the code at the cursor. Stopping of the execution is possible via [ESC] key, pressing the RESET switch on the CPUBD or STOP button of HEW.

4.9.3. Step Functions

There are four types of Step Functions:

- 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. It is used to perform stepping to exit from the subroutine. Instructions in the subroutine function will be executed and PC will be set to the line of code after the subroutine return instruction RTS.

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 enables a visual view of the code running sequence.

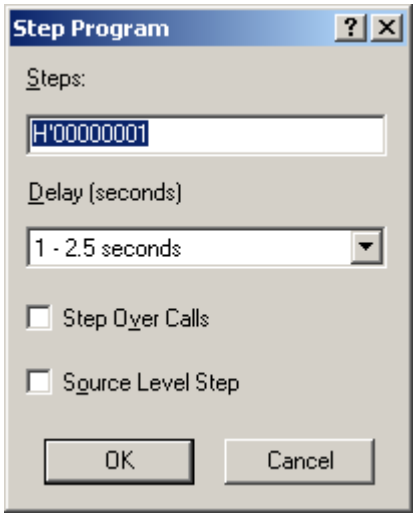


Figure 4.29 Step program

Step Mode setting configures how the step instruction operates.

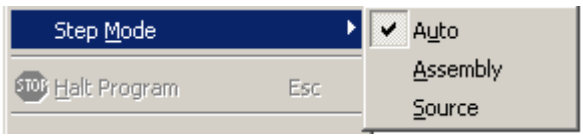


Figure 4.30 Step Mode

- *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 (when a toolchain is used) 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,
- Go To label (address),
- View local
- Instant/add watches (local and user defined)
- Stack Trace

In other words, C-source Level debugging is only available when a ELF/DWARF2 (.abs) file is downloaded. User would not be able to perform debugging if other file formats like S-Record, Intel Hex and Binary are used.

Section 5. Usage Precautions

Users may need to observe several precautions while operating the CPUBD. They are described as below:

5.1. Corruption of Monitor Software

The monitor software occupies predefined locations in the flash memory area as the user target program. Due to unforeseen reason, user might access to this area and corrupt the monitor code. As a result, HEW might not be able to communicate with the CPUBD and debugging could not be performed.

Please refer to the *Appendix B – H8S/2268F Memory Map* to take note of the area occupied by the monitor code.

- ❑ User target program must not reside in H'03E000 to H'03FFFF of the on-chip ROM as this memory space contains the Monitor Program.
- ❑ User must not use H'FFB000 to H'FFB4FF of the RAM as this area is reserved for Monitor RAM.

If a Renesas Standard Toolchain is used, go to *Options* menu and select the toolchain used. View the *Section* of the program by selecting Section in the *Link/Library* tab.
Figure 5.1 shows an example of the Section of a program.

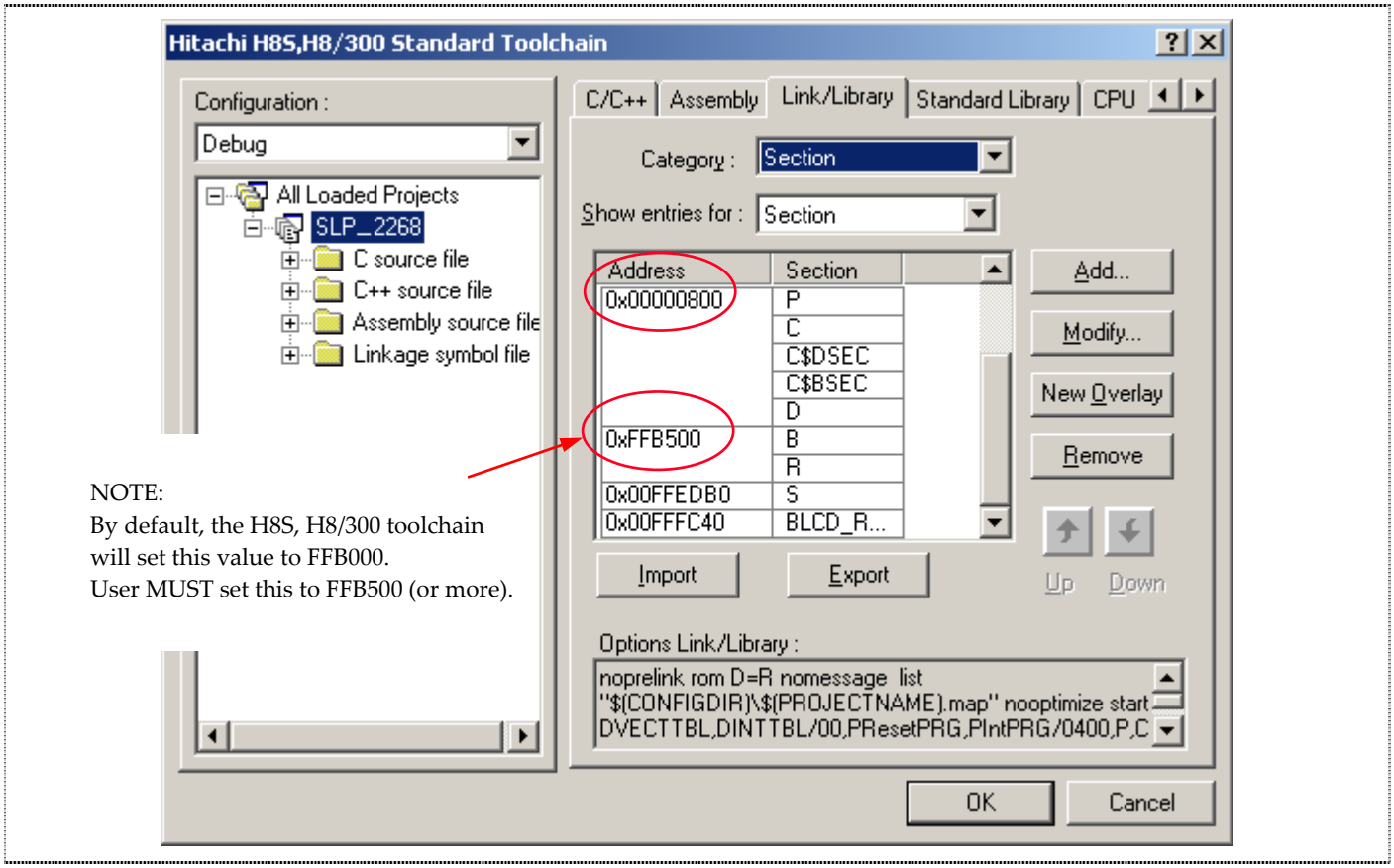


Figure 5.1 Program Section

Ensure that the circled addresses do not overlap with the RAM and ROM used by the Monitor Program.

5.2. Interrupt

- Interrupt Control Mode 2* is not supported by HEW.

- Users, who want to perform debugging operation on the CPUBD, must enable interrupts.

- The example provided below, would result in a loss of communication between HEW and CPUBD:

Referring to the following code, after single stepping the line, *set_imask_ccr(1);*, I bit is set to '1', disabling interrupts.
Therefore, if another single step is performed, SCI0 interrupt would not occur and HEW will timeout and a dialogue box "Error in communication" will be displayed as follow:-

```
set_imask_ccr(1);
light_LED();
```

5.3. Timing Issues

- Execution time to complete an interrupt subroutine must not be longer than 3sec, else HEW will timeout and a dialogue box "Error in communication" will be displayed.
- If the frequency of interrupts generated is less than 300msec, MCU will not be able to respond to the SCI0 interrupt sent by HEW. This will also cause HEW to timeout.

The following shows the timing diagram when using HEW.

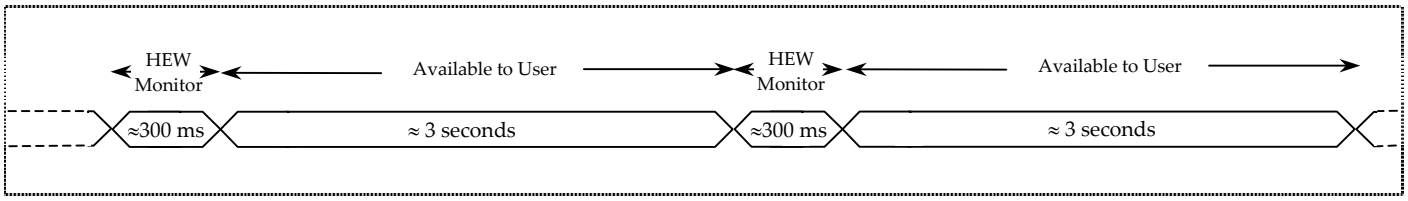


Figure 5.2 Timing diagram of HEW

5.4. Watchdog timer

Watchdog timer must not be used to generate an internal reset when performing debugging operation. This is because when counter in watchdog timer overflows, a signal is generated, resetting the MCU. At this instance, if HEW performs a debug operation, the operation will not be completed as the MCU has been reset, resulting in a loss in synchronization. This will result in a timeout in HEW.

5.5. SCI0, PC Break* and TRAP1

- PC Break* and TRAP1 are used by monitor program.
- The user is not allowed to use PC Break* and TRAP1 exception handling if debugging operation is performed.

User is not allowed to make use of SCI0 in his/her program because SCI0 is used by HEW to communicate with the CPUBD.

5.6. Software Breakpoint

- User shall not set a software breakpoint in the following address:
  - An area other than the flash memory or RAM
  - An area of address H'03E000 to H'03FFFF [Monitor code resident]
  - An area of address H'FFB000 to H'FFB4FF [Monitor work area]
- User shall not set any breakpoints in the interrupt service routines.
- When execution resumes from the breakpoint address, single-step execution is performed before execution resumes.

NOTE: * Only available for H8S/2268F MCU

5.7. Step

- Step function (step in, step out and step over) is implemented by the conventional hardware break mechanism.
- No interrupt will be serviced during stepping.
- Do not step into interrupt service routines.
- Stepping of SLEEP and TRAPA instructions are not allowed in HEW. User needs to use "Go to cursor" in order to proceed to the next instruction.

5.8. Power-Down Modes

User must not place the MCU in any of the following power-down modes when performing debugging operation:

- Watch Mode
- Sub-active Mode
- Subsleep Mode
- Software Standby Mode
- Hardware Standby Mode

Serial Communication function is disabled in these modes, hence HEW is unable to communicate with the CPUBD.

5.9. E7 Interface [Applicable with H8S/2264F MCU Only]

When interfacing with E7, the following limitations have to be observed:
The FWE is not available for use because it is dedicated to E7

- The Port H pin 7, Port 7 pin 1, and Port 7 pin 4 are also not available for use. To use these pins, additional hardware is required on the user's board.

5.10. Other Constraints

- When viewing memory content in HEW, user may access to memory area above the available memory area on H8S/2268F MCU. This is because the H8S/2268F MCU has only 16MByte address space so the top bits of any address above 24 bits are ignored. This results in address error if data is written to these wrong addresses.
- User must be aware that they are not allowed to place the MCU into hardware standby mode as this condition is exited by reset interrupt only. This would restart the monitor software, and DESTROYS the current context of the user target program. Software standby mode may be entered, but may not be exited by the use of the reset interrupt for the same reason mentioned.
- When SLEEP instruction is executed, the MCU is unable to stay in SLEEP mode as HEW will send data via SCI 0 and wake up the MCU.

NOTE: * Only available for H8S/2268F MCU.

Section 6. Hardware

The CPUBD comprises of the following blocks:

- Microcomputer
- Power Supply circuitry
- Reset circuitry
- Clock circuitry
- Serial Communication block [via SCI0]
- Flash ROM and RAM
- LEDs
- Boot Mode Enable
- E7 Emulator Interface
- External User Interface
- Optional Components

6.1. Microcomputer

The H8S/2268F and H8S/2264F series has a system-on-chip architecture that includes peripheral functions and can be used as embedded microcomputer in application systems. Its on-chip ROM offers flexibility as it can be reprogrammed in no time to cope with all situations from early stages of mass production to full-scale mass production. Users reconfiguring processor I/O ports are cautioned that pull-up resistors may be needed for proper operation in some configurations.

6.2. Power Supply Circuitry

The power supply circuitry supplies the DC power to the CPUBD from an external power supply. This is also known as the system DC power. The CPUBD either accepts +7.5 DC to +9V DC voltage supply. This power input is further stepped down to either +3.3V DC or +5.0V DC, via two voltage regulators, that is acceptable by the MCU. In addition, user can select the source of power supply to the MCU via a jumper selection between the system power supply or from a target system.

When power is supplied to the MCU, the green LED, D3 lights up.

6.3. Reset Circuitry

The reset circuitry comprises of RC circuit and a push button, S1 also known as the RST SW. During power-on, the RC circuit asserts a reset signal to MCU to reset the MCU. If the RST SW, S1, is pressed, a reset signal of approximately 20msec. duration is generated to allow proper reset to be performed. The reset switch allows user to manually reset the CPU board when abnormal situation occurs and during flash programming control.

6.4. Clock Circuitry

The clock circuitry comprises of a quartz crystal of 18.432MHz, the system clock oscillator. A sub clock is also provided by a quartz crystal of 32.768KHz on the CPUBD.

6.5. Serial Communication Block [via SCI0]

The CPUBD supports a three-wire serial channel using the on-chip serial communication channel [SCI0] on the MCU. SCI0 is used, both to flash the device using a flash programming software and to connect to HEW. If neither flashing nor debugging with HEW is required, then the serial channel is available to user. The SCI0 port provides transmit and receive signals to the RS3232 transceiver device on the board. The transmit and receive signals from the transceiver device is then connected to the 9-pin D-type connector, P1 on the CPUBD. The RS3232 transceiver device translates the RS232 signals to logic levels and vice versa.

6.6. FLASH ROM & RAM

The MCU does not have any interface to external memory; it could only be used in single chip mode. The H8S/2268F has 256Kbytes of FLASH ROM and 16Kbytes of RAM for user and the H8S/2264F has 128Kbytes of FLASH ROM and 4Kbytes of RAM. If debugging by user is necessary, a monitor software would be downloaded together with the user target program. A total of 8Kbytes of FLASH ROM and 1.25Kbytes of RAM must be reserved for the monitor software.

6.7. LEDs

There are two red LEDs on the CPUBD available to user. LED D1 can be driven by port 1 bit 0 of the MCU. This can be selected by a jumper selection of J7 header, see section *2.5.6*.

The second LED D2 can be driven by port 1 bit 1 of the MCU. This can be selected by a jumper selection of J8 header, see section *2.5.6*.

HIGH output level from the MCU will set the LED ON and a Low output level would set the LED OFF.

6.8. Boot Mode Enable

Boot Mode is necessary to flash the FLASH kernel software and monitor software or user target program if required into the FLASH ROM when the CPUBD is placed into Boot mode. This is via the Mode Selection jumper, J2 & J3. Boot mode is required at the Power-On stage and standalone programming only. For the jumper selection, see section *2.5.4.*

A red LED, D4, lights up when Boot Mode is selected.

6.9. E7 Interface [Applicable with H8S/2264F MCU only]

This interface allows user to extend the debugging function of the CPUBD if an E7 emulator is available. When the user wants to make use of the E7 emulator for debugging, the following components have to be mounted:

Component Designator	Descriptions	Remarks
R8	Zero Ohm resistor	Not provided (0805)
R9	Zero Ohm resistor	Not provided (0805)
R10	Zero Ohm resistor	Not provided (0805)
J1	2x7-way Box Header	Provided

Table 6.1 Components for E7 Emulator

6.10. External User Interface

The external user interface makes all signals of the MCU available to user. These signals are connected to the following connectors.

- Four 2x 13-pin connector [JP1 ~ JP4]
- Two 2x30-pin socket connector [CON1, CON2]

The four 2x 13-pin connectors [JP1~JP4] are placed closed to the MCU on the CPUBD.

The two 2x30-pin socket connector [CON1, CON2] is placed to the edge of the CPUBD for ease of connection to an external system. These connectors should be mounted on the solder side of the CPUBD.

Both connector types use commonly available 2.54mm[0.100inch] pitch male header and female socket with 0.635mm[0.025inch] square posts.

These connectors are all connected to the MCU, and can be used to access the pins of the chip and labeled with reference to the actual chip QFP 100B pin-out.

See appendix *C*, appendix *D* for the pin assignment for JP1~JP4 and CON1, CON2.

NOTE: External interface should be powered by an independent power supply.

6.11. Optional Components

NMI and STBY pins are pulled HIGH on the CPU Board. However, user could mount a push button switch along with a 0.1μF capacitor (for de-bouncing) to pull the signal level LOW.

Pressing SW2 switch would send the MCU to hardware standby mode.

Pressing SW3 switch would generate a nonmaskable external interrupt.

Section 7. Monitor Software

7.1. Introduction to Monitor software

The Monitor Software is a FLASH-resident debugging program hosted on the CPUBD. Monitor software may be used to download, run, and debug programs developed on a PC. The monitor software provides all the necessary control and communications to operate under the HEW. This allows users to perform high-level C debugging on the CPUBD.

Using the powerful debugging features of HEW, user may explore features of the MCU and the CPUBD by directly running sample programs.

The CPUBD comprises of limited RAM and is also a single chip MCU. To debug the user target program, both the user code and the monitor software must be programmed into the FLASH ROM. The monitor software is built separately from the user target program into S-record format. Without the monitor software flashed into the FLASH ROM of the MCU, no debug can be performed with the HEW software.

7.2. Program Development

The tutorial program which accompanied the CPUBD contain examples you may use as a basic reference code to explore and evaluate the architecture of the MCU.

When you install the High-Performance Embedded Workshop [HEW], user obtains faster turn-around-time for steps: 'Download S-record/ Elf-Dwarf2 file to MCU' → 'Execute User target program' → 'Debug User target program' within an integrated environment (*HEW with 2268F pure debugger).*

7.3. Monitor software Requirements

The monitor software makes use of the following peripheral function and input/output pins of MCU, which cannot be used by user target program during debugging. These are:

- SCI0 Port for communication to the PC running HEW
- IO Port 1 Pin 0
- IO Port 1 Pin 1

Refer to Section 5 on usage precautions and limitations for more information.

7.4. Mode Transition

The CPUBRD operates in two modes: Boot Mode and User Program Mode.

In Boot Mode, user can either download the monitor program or user target program (for Stand-alone flash operation).

In User Program Mode, monitor program is being executed. User target program can be downloaded for debugging purposes in User Program Mode.

The MCU loops in the Break Mode of the monitor program while waiting for commands from HEW. Commands from HEW generate SCI interrupts to perform the required tasks.

To execute the downloaded user target program, user can either *Run at current program counter*, *Reset Go* or perform Step functions *(Step-In, Step-Over and Step-Out)*. This will cause it to operate in the User Target Mode.

To terminate the User Run state, a break condition has to be asserted to bring the MCU to the Break Mode. This can either be a preset condition (eg. PC Break, Event Break) or a force break condition (Hit ESC key or press STOP button). The MCU also returns to Break Mode automatically after completing Step functions.

Figure 7.1 illustrates the mode transition diagram.

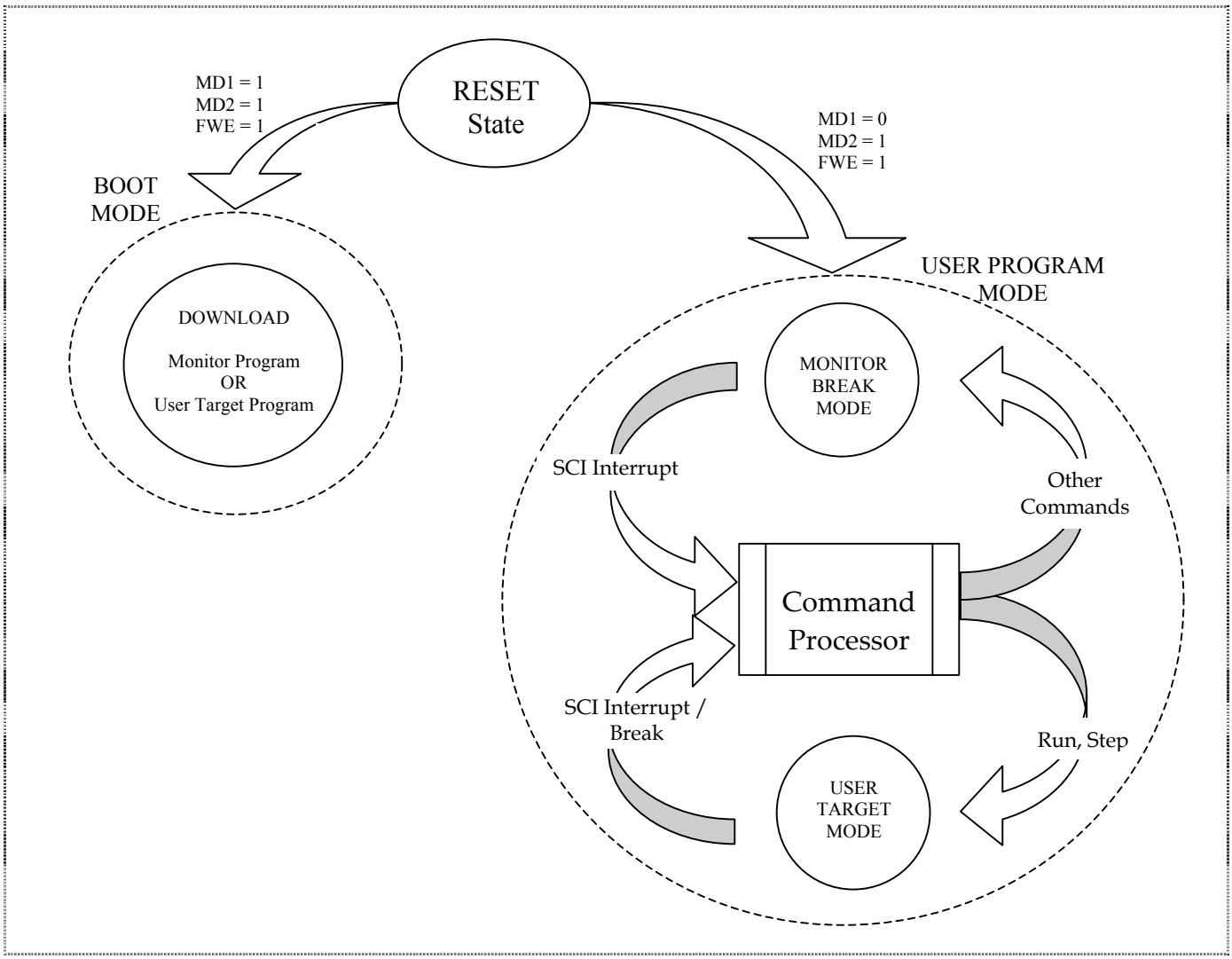


Figure 7.1 Mode Transition Diagram

7.5. Using Monitor software

The monitor software is used with the CPUBD. All monitor software functions are accessed through the HEW graphical user interface and they are not accessible by user commands via the serial interface.

The following functions are supported by monitor software:

- Program Download - Supported file formats are:
  - Elf/Dwarf2
  - Motorola S-Record
  - SYSROF format
- Breakpoint - Maximum of 256 breakpoint is allowed at a time when executing with the monitor software
- Types of Execution - Three execution modes:
  - RUN
  - STOP
  - Step
- Memory Read/Write
  - Memory Write
  - Memory Read
  - Fill Memory
- Register Read/Write
  - Read CPU Register
  - Write CPU Register
- Others
  - Mapping
  - Read or Write I/O registers [I/O windows]

7.6. Interrupts used by the Monitor

The monitor uses several interrupts to communicate with the host PC and control user target program execution. The user is not allowed to use these interrupts if HEW is used for debugging.

The following lists the interrupts reserved by the monitor and their vector addresses:

Exception Source	*Vector Number*	*Vector address*
Reset	0	H'0000 to H'0003
Trap 1	8	H'0020 to H'0023
PC Break*	27	H'006C to H'006F
SCI channel 0 (ERI0)	80	H'0140 to H'0143
SCI channel 0 (RXI0)	81	H'0144 to H'0147

Table 7.1 Interrupts Used by Monitor Program

- **NOTE: * Only available for H8S/2268F MCU.**

7.7. Breakpoints

The CPUBD only allows a maximum of 256 breakpoints to be assigned at a time when executing with the monitor software.

The breakpoint is controlled through software means, the line of code where the breakpoint is placed is NOT executed and the program stops at the same instruction where the breakpoint is set.

NOTE:

- **When user inserts breakpoints, it is recommended to use the 'Disassembly window'.**
- **Beware of instruction pre-fetches after branch instructions.**

A breakpoint inserted on a branch instruction, will halt on the line of code where the instruction branches. A breakpoint inserted on a line of code after a conditional branch such as *BNE* may never be triggered because the line of code may always be pre-fetched and thus not seen by the break control.

Section 8. FLASH Programming

For programming of the FLASH ROM, FLASH Kernel software is developed. This FLASH Kernel is downloaded together with the monitor software to the FLASH ROM at power on. It performs Write or Erase control program operation in Boot mode and User Program mode.

The MCU's serial communication port, SCI0 is used for flash programming.

Please refer to specific device manual to enter boot mode.

8.1. FLASH Programming the CPUBD

There are several methods to flash the CPUBD

- 2268F HEW (pure debugger)
- FDT version 2.2
- E7 emulator (for H8S/2264F MCU only)

HEW is discussed in this user manual. As for the other methods, please refer to their respective user manuals for detailed operations.

Flash programming is performed in the HEW under the following modes:

- Boot mode – the writing or erasing is performed in batches,
- User program mode - the range of writing or erasing can be defined independently for each program block.

8.1.1. Boot Mode:

Boot Mode is necessary under the following operation:

- Upgrade or Recovery of monitor program
- Stand-alone flash operation of user target program

Hardware jumpers are required to be set accordingly to trigger MCU to enter boot mode. For jumper settings, please refer to section *2.5.4* "Boot Mode Selection Jumpers".

The sequence to trigger MCU into boot mode is described below:

- Short J2 [1-2 default] and short J3 [1-2]
- Power-on the CPU Board
- Press RST SW to put MCU in the boot mode

The boot program then start to transfers the write control program received from the host machine to the MCU internal ram. When the write control program has been received, the entire internal flash memory area is erased.

After entire flash memory has been erased, the execution is transferred from the boot program to the write control program, and the application program (Monitor program or user program) received from the host machine is written to the flash memory.

8.1.2. User Program Mode:

User Program mode is used only when the monitor program is resident in the flash memory.

Most of the time, user program mode is used to download user target program and modify Flash memory content.

The advantage of using user program mode is no jumper setting is needed and the range of writing or erasing can be defined independently for each program block (reduce programming time).

When monitor program is started, host machine sends flash memory command to MCU. The monitor program copies the write / erase control program into internal RAM, this is followed by having execution transferred to the write / erase control program.

HEW sends address that needs to be programmed and the entire flash memory block is erased. The MCU starts receiving program data from HEW and write to the flash memory. After completing the flash programming, write / erase control program returns the execution control to the monitor program waiting for debugging command from HEW.

8.2. Operation during Programming Kernel Execution

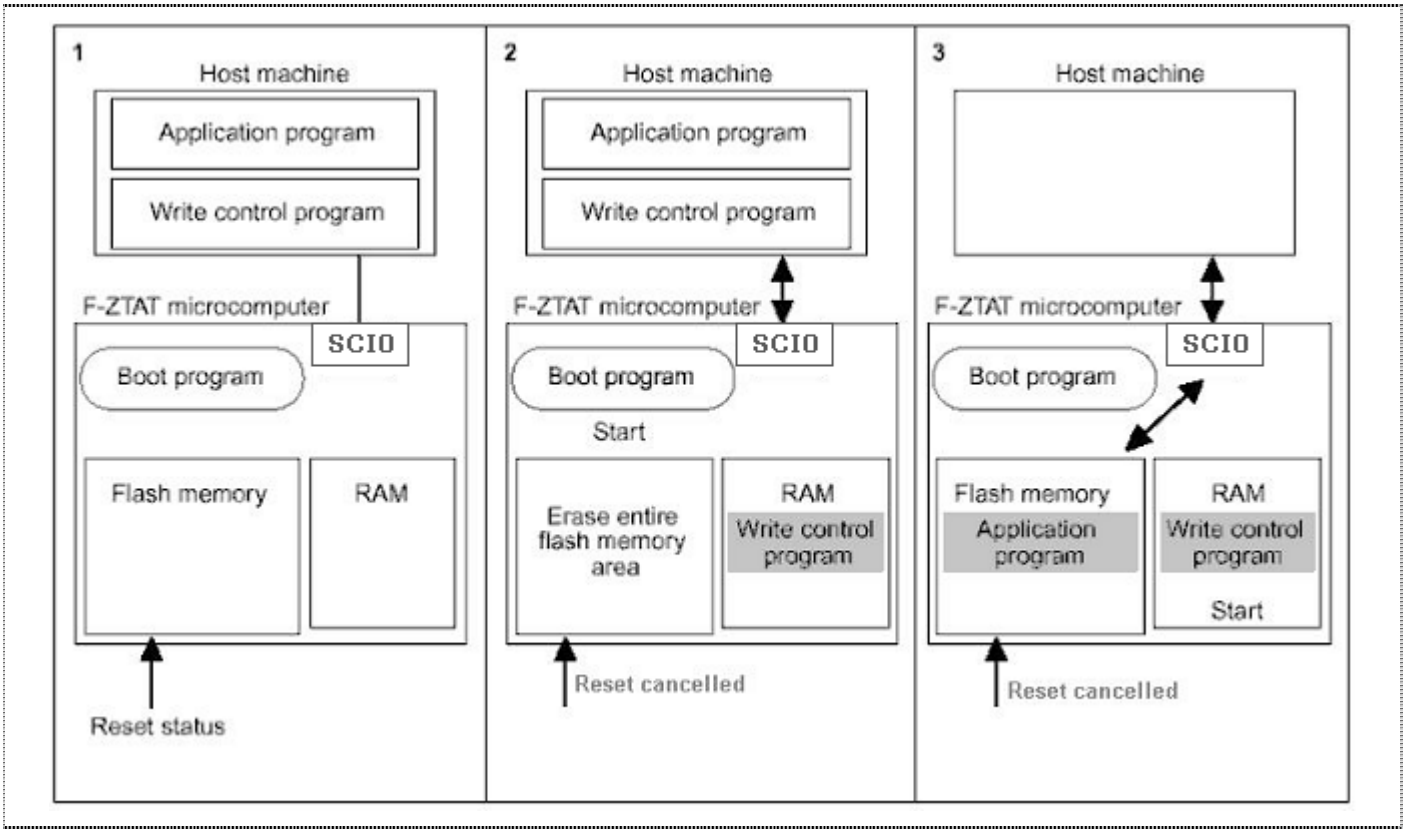


Figure 8.1 Overview of Boot Mode

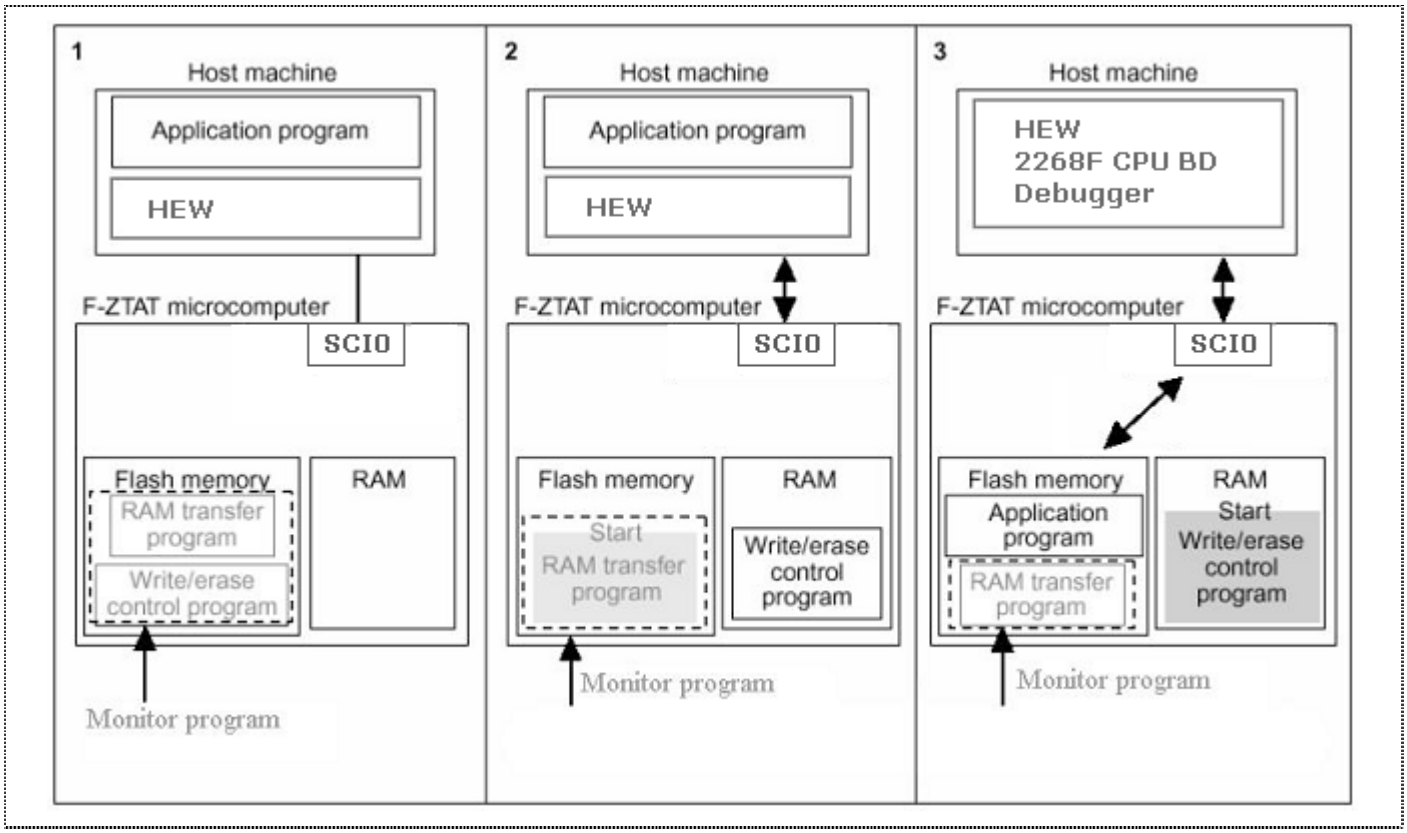


Figure 8.2 Overview of User Program Mode

Section 9. Tutorial

The following describes a simple debugging session, designed to introduce the main features of the CPUBD used in conjunction with the HEW (Pure Debugger) for CPUBD software.

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

User has to setup the CPUBD as stated in section 2 before the tutorial can begin.

9.1. Introduction

The tutorial is based on a simple Assembler / C program located in your installed directory *"…\Tools\Renesas\DebugComp\Platform\Emulator\Evb2268F\tutorial"*.

Before reading this chapter, ensure the followings would certainly ease the learning process:

- Setup the CPUBD and verify that it is working correctly with the HEW (Pure Debugger) software for CPUBD.
- User has to be familiar with the architecture and instruction set of the H8S Series MCU.

For more information please refer to the H8S/2600 Series, H8S/2000 Series Programming Manual and H8S/2268 Series, H8S/2264 Series Hardware Manual.

Refer to H8S, H8/300 Series High-Performance Embedded Workshop 3 in your installed directory (install directory/Manuals/Renesas/PDFS/EH8HTU36.pdf) for more detailed information on using HEW.

9.2. Overview

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

The tutorial workspace is provided on the installation CD. A compiled version of the tutorial is provided in Motorola S-Record in the file *tutorial.mot*.

- How the tutorial Program Works:

The first part of the program includes a series of header files:

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

The program then gives prototypes for the constants, structures, and function initial values:

```
#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();
```

Followed by the main program below.

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

The remainder of the program defines the functions called from main:

```
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;
  }

}
```

9.3. Tutorial Setup

Open tutorial workspace in:
"install directory\Tools\Renesas\DebugComp\Platform\Emulator\Evb2268F\tutorial".

NOTE: On a first time loading of the tutorial, a dialogue 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.

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

Before downloading a program to the CPUBD, check the following items and user target program (Download Module) to be debugged:

- ❑ Device type
- ❑ Memory map

NOTE: Refer to Section 4.5 for these emulation settings.

9.3.1. Downloading the tutorial Program

Once the emulation settings of the CPUBD have been setup, user can download the object program for debugging.

- ❑ First load the object file, as follows:
- ❑ Open the Debug Settings window by choosing *Options* menu and *Debug Settings*…
- ❑ Select Elf/Dwarf2 for the Default Debug Format.

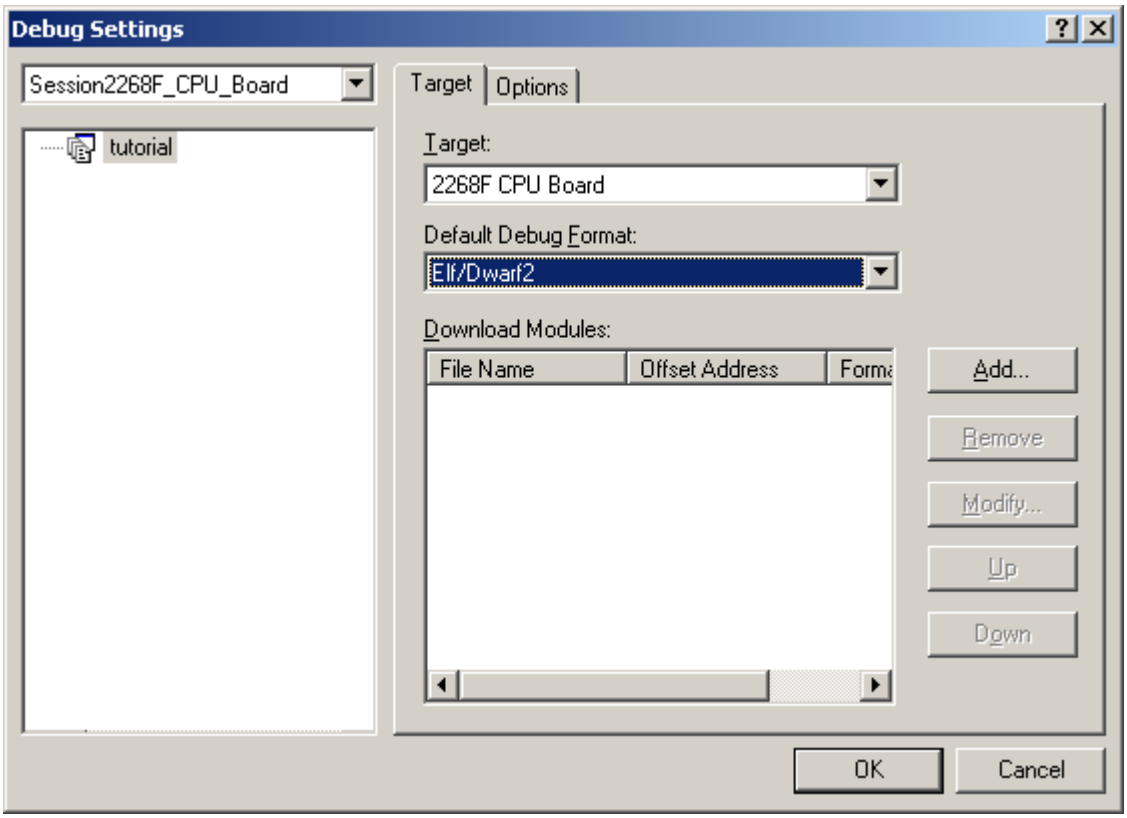


Figure 9.1 Debug Settings with Load Object File Dialogue

- Click on the Add… button.
- Select the download Format to be the ELF/DWARF2.
- Click the Browse button and select the file '*tutorial.abs*'.
- Click OK to exit from Download Module window and click OK again to exit the Debug Settings window.

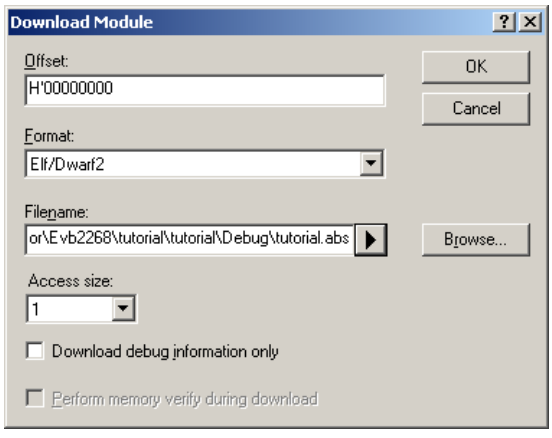


Figure 9.2 Configure Load Object File Dialogue

A new folder, Download Modules, with the '*tutorial.abs*' file is created in the workspace window.

Download the file into the memory as follows:

- Right click on the '*tutorial.abs*' in the workspace window and select Download module.

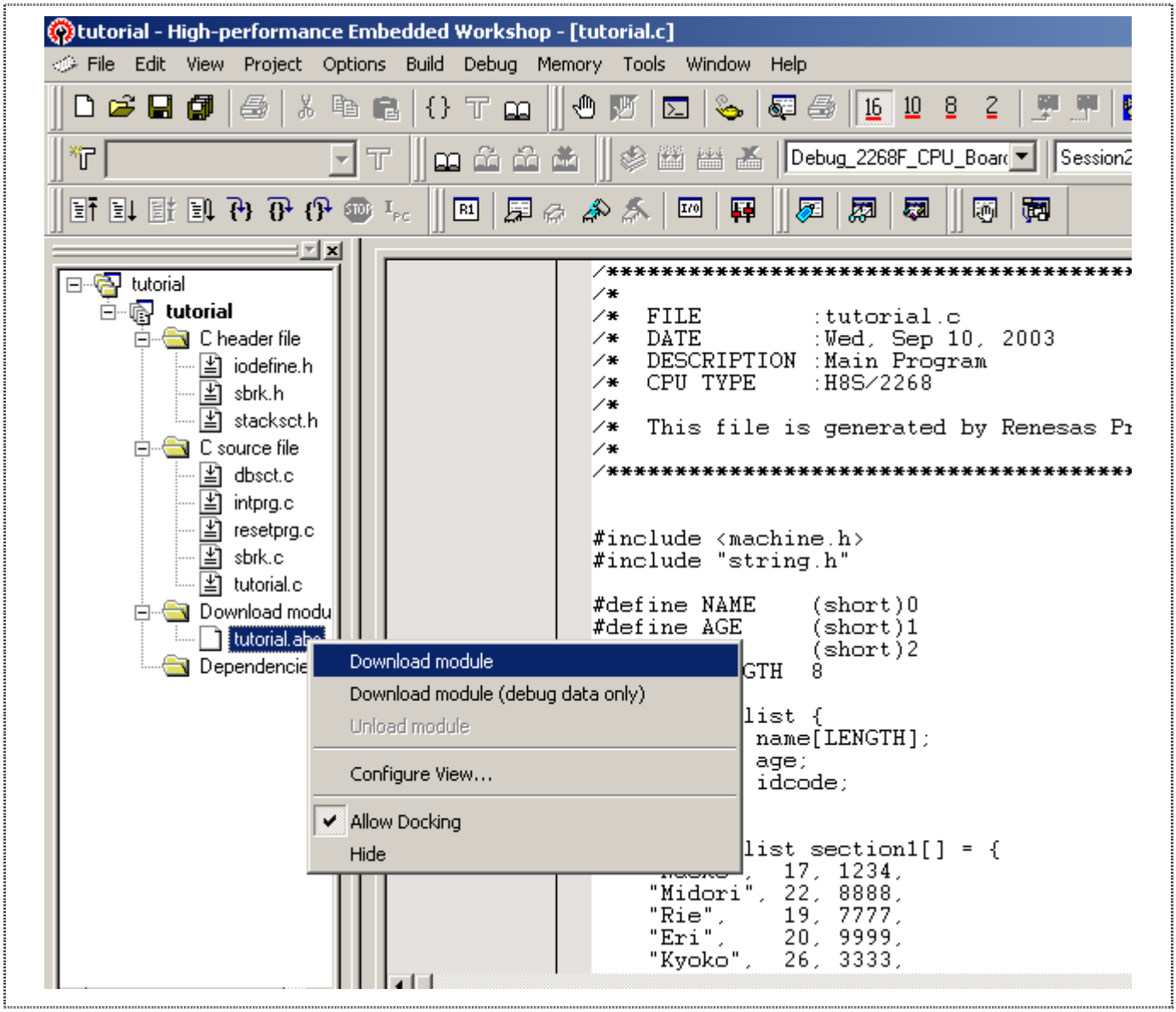


Figure 9.3 Download the Selected Object File

When the file has been downloaded, the Status-window Memory Tab will show the downloaded Memory Address.

NOTE: All the code should lie within the on-chip ROM.

9.3.2. Displaying the Program Listing

HEW (Pure Debugger) for CPUBD allows user to debug a program at source level, so that a listing of the program can be seen alongside the disassembled code. To do this, user needs to read in a copy of the source program from which the object file is compiled.

- ❑ Choose *Reset CPU* from the *Debug* menu.

User will be prompted for the '*Resetprg.c*' source file corresponding to the loaded object file if HEW could not automatically locate the required file.

Browse to the location of the file and double-click to open the file.

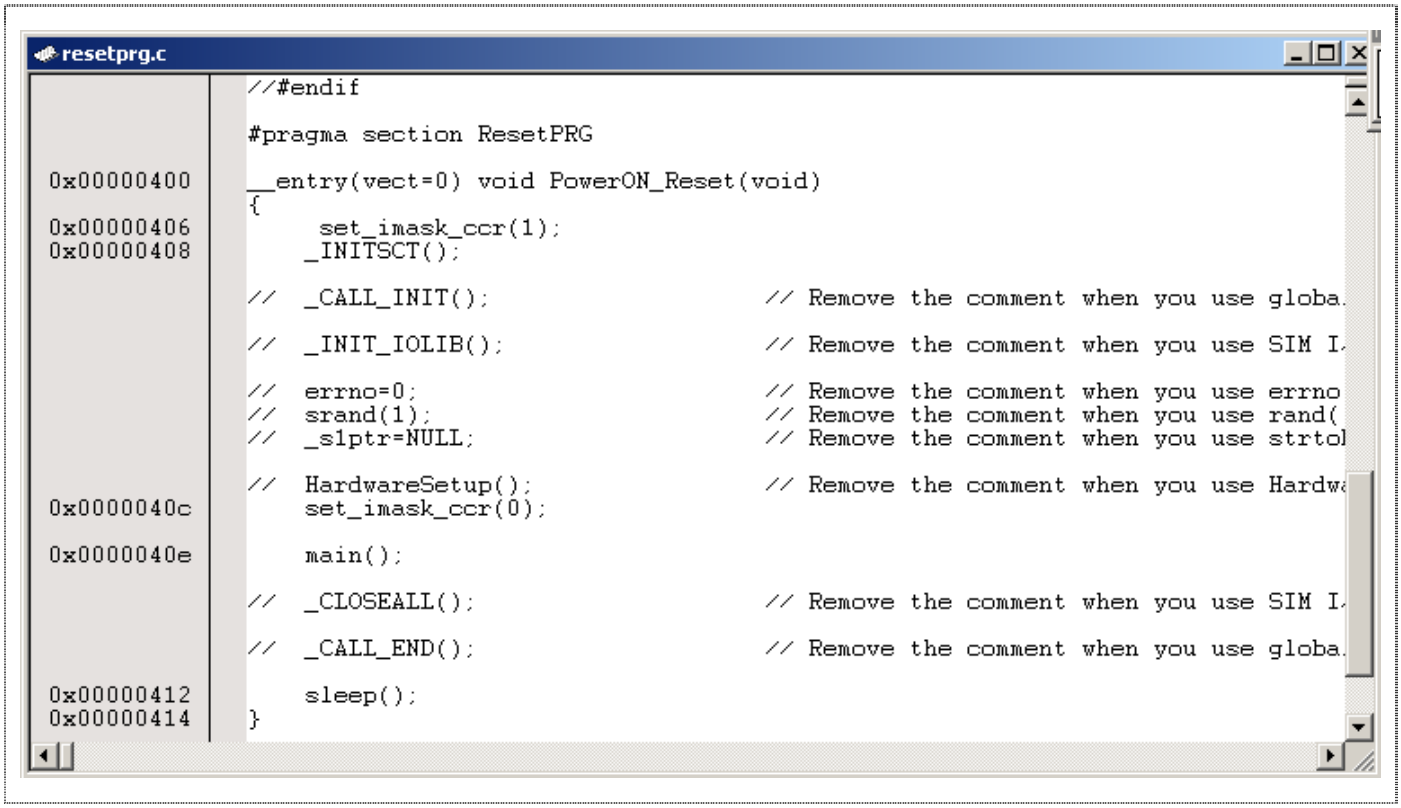


Figure 9.4 Source-window "Resetprg.c"

- ❑ Run the program until Address H'0000040e (Set breakpoint at H'0000040e and select Reset Go, see section 9.4).
- ❑ Single step (see section 9.6 for Single Step) again to jump into the tutorial.c main program window

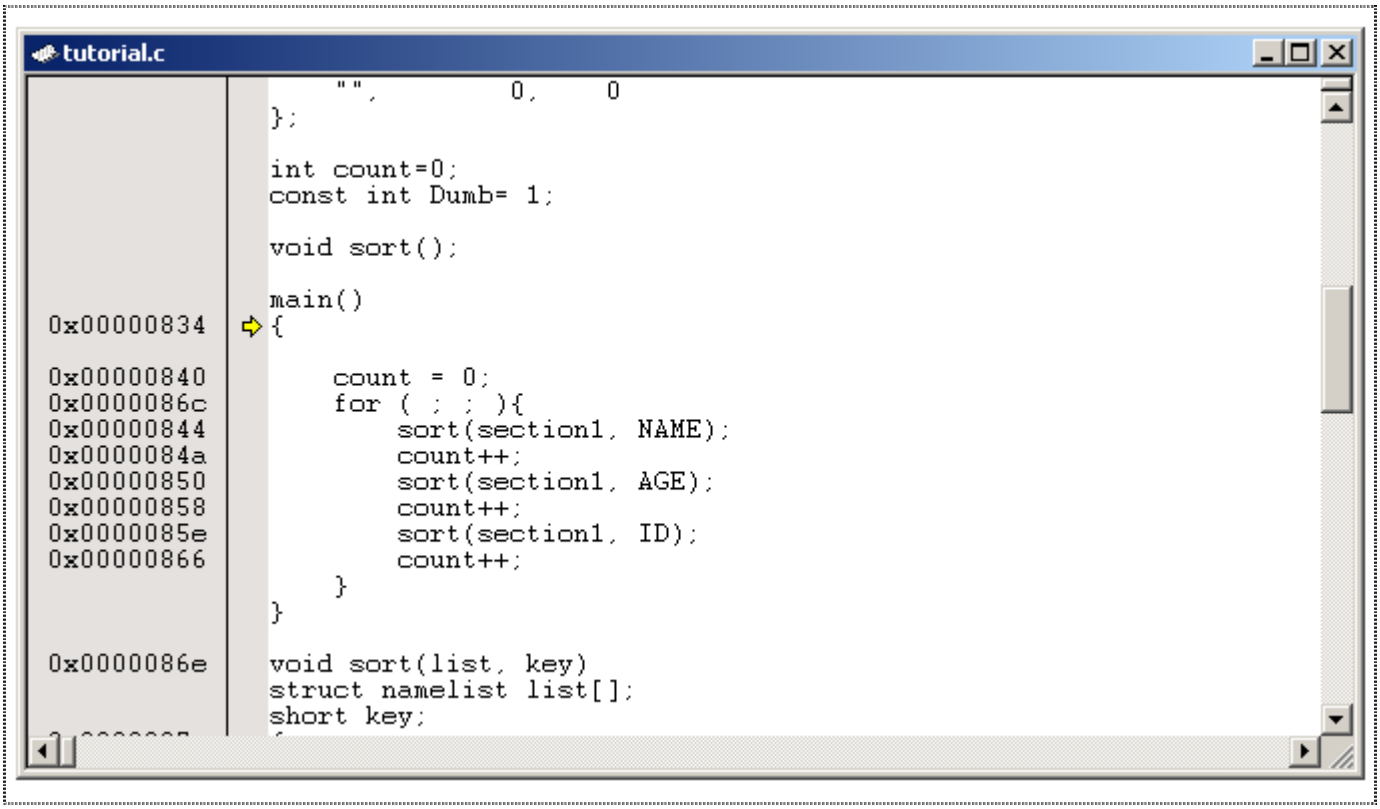


Figure 9.5 Source-window " tutorial.c"

- If necessary, choose *Format Views…* from the *Tools* menu to select a font and size suitable for your computer.

The above source-window has its font change to Courier New, 8-point font.

NOTE: If change of font or size did not take place in the window, close the window and re-open the file again.

9.4. Using Breakpoints

The simplest debugging aid is the program breakpoint (or PC 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.

9.4.1. 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 address H'00000844 as follows:

- Click once on the line containing address H'00000844 and right-click for the pop-up menu and select *Toggle Breakpoint* OR
- Click once on the line containing address H'00000844 and press F9.

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

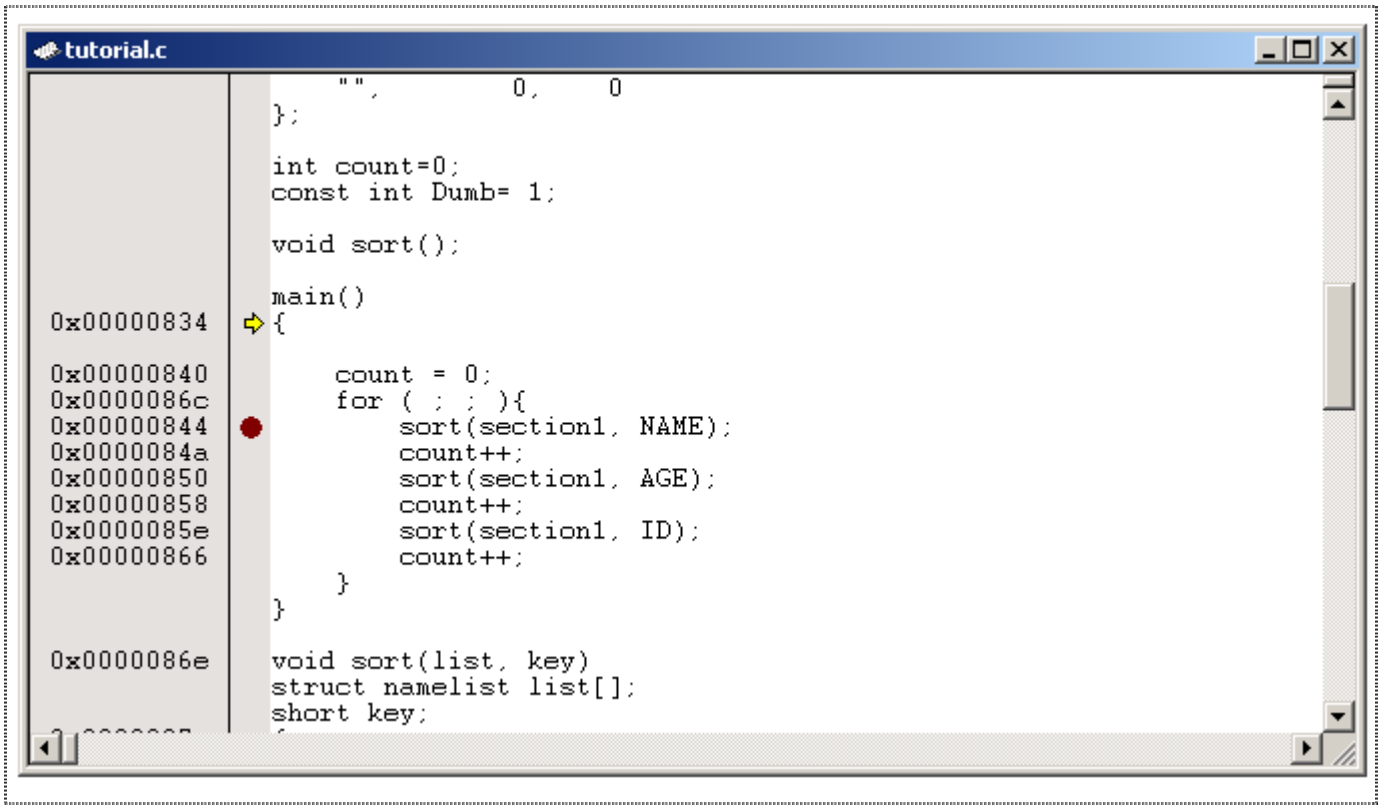


Figure 9.6 Setting a Breakpoint

9.4.2. Executing the Program

To run the program from reset:

- Choose *Reset Go* from the *Debug* menu, or click the Reset Go button in the toolbar icon.

The yellow arrow will appear on the read dot, indicating that the program is executed up to the breakpoint you have inserted.

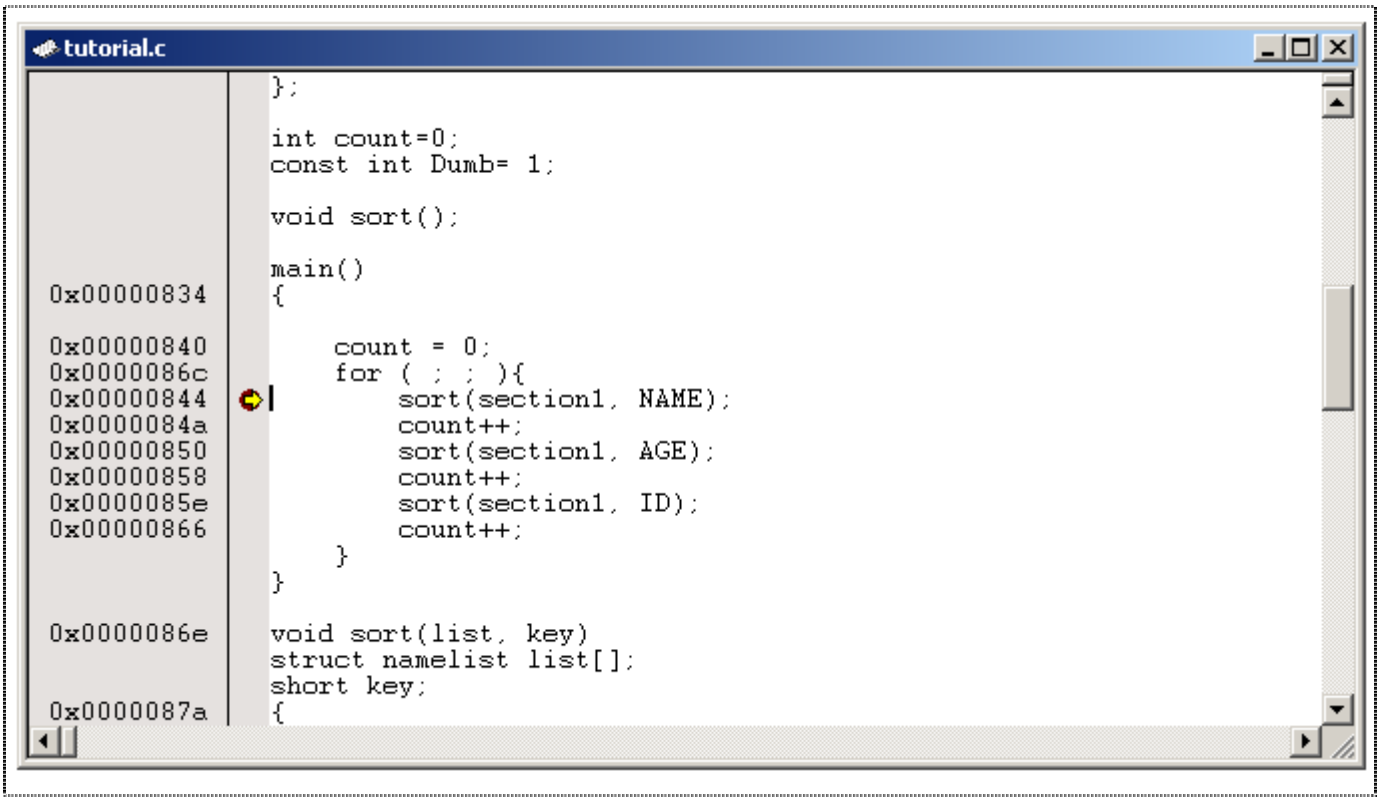


Figure 9.7 Program Break

The message *Break = PC Break* is displayed in the status bar to show the cause of the break.

This can be viewed under cause of the last break in the System Status window.

- From the *View* menu, choose *CPU* then *Status*, or click the Status Window button in the toolbar:

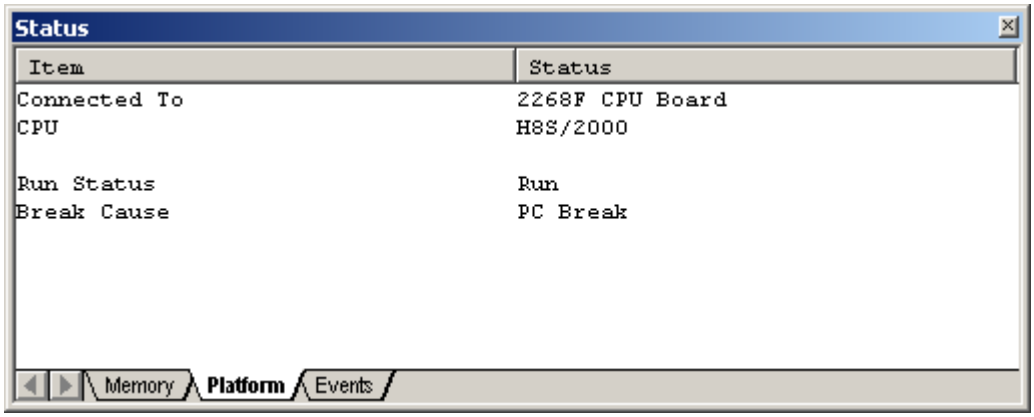


Figure 9.8 System Status Window

The cause of last break line shows that the break was a User PC Break.

9.4.3. Reviewing the Breakpoints

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

❑ Choose *Source Breakpoints* from the *Edit* menu, or click the Breakpoint Window button in the toolbar:

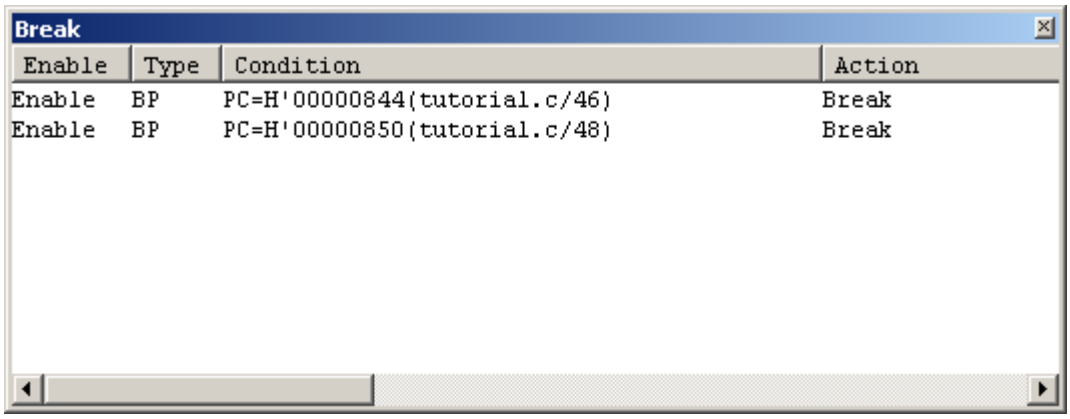


Figure 9.9 Breakpoints Window

The Breakpoints window also allows user to perform the following:

- Delete selected breakpoint
- Delete all existing breakpoints
- Disable existing breakpoint

❑ Right-mouse click on a breakpoint in the Breakpoint-window to show the following pop-up:

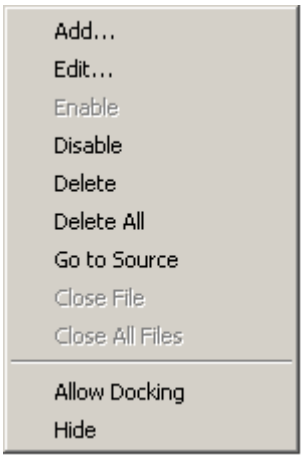


Figure 10.10 Popup in Breakpoints Window

9.4.4. Examining MCU Registers

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

- Choose *CPU*: *Registers* from the *View* menu, or click the Registers Window button in the toolbar:

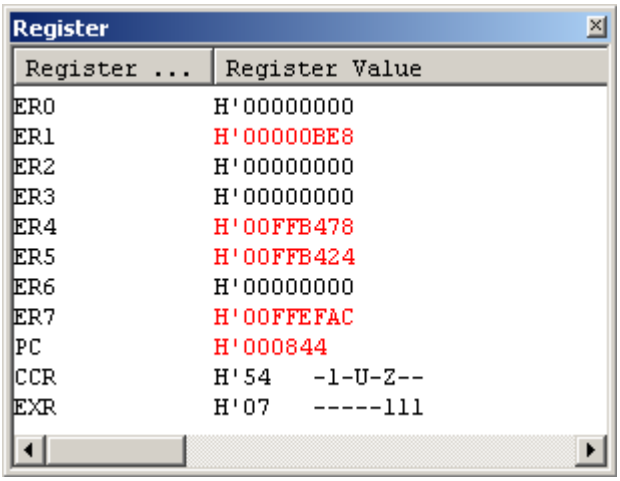


Figure 9.11 CPU Registers Window

As expected, the value of the program counter (PC) is the same as the position of the yellow arrow, H'00000844.

The registers' values can be changed from the Registers window by double-clicking on respective registers in the Registers window.

The Register-PC dialogue box allows you to edit the value.

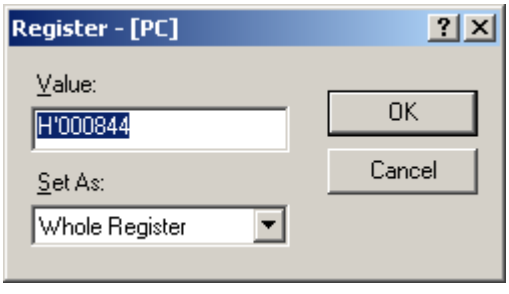


Figure 9.12 Changing Register Value

9.5. 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.

9.5.1. Viewing Memory

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

For example, to view the memory corresponding to the array section1 in ASCII:

- Choose *CPU*: *Memory*… from the *View* menu, or click the Memory Window button in the toolbar:

- Enter "_section1" (a label valid only after downloading of Download Module- .abs file) in the Begin Address field and "fff000" in the End field, and keep the Format as Byte (x1).

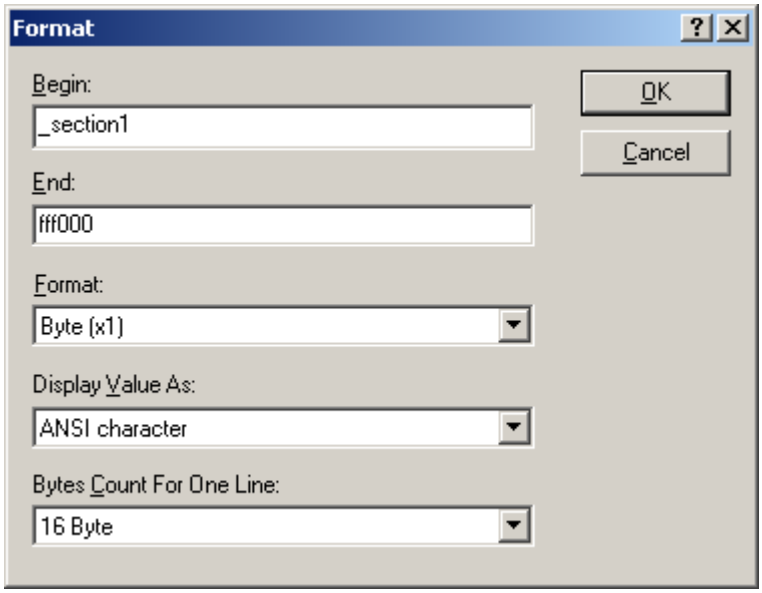


Figure 9.13 Open Memory Window

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

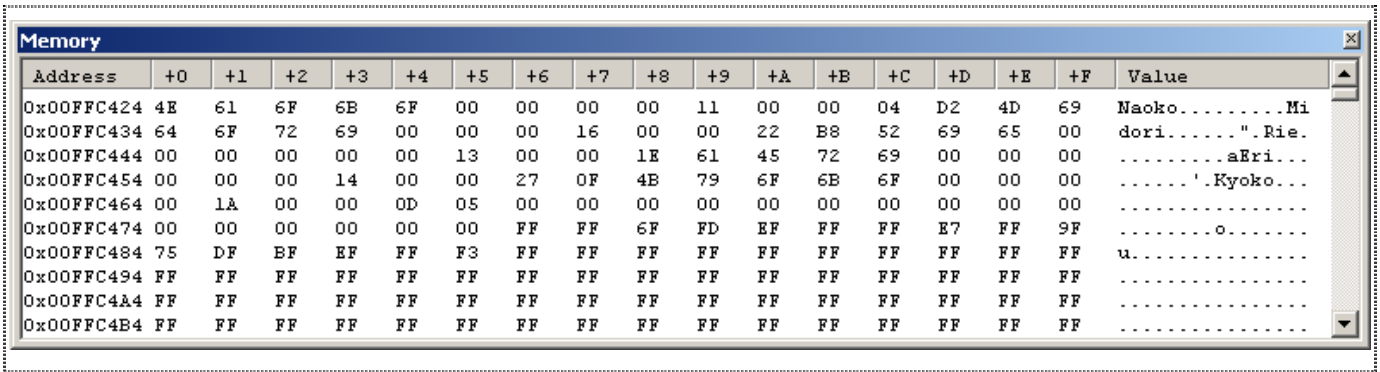


Figure 9.14 Memory Window

Leave the Memory window open so that you can monitor the contents of the array label "_section1".

9.5.2. Watching Variables

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

For example, set a watch on the structure (STRUCT) variable section1, which is declared at the beginning of the program, using the following procedure:

- Scroll up in the program window until you see the line:

  sort(section1, ID);

- In the Program windows, position the cursor on the word section1 and perform a right mouse button click to display a pop-up menu.
- Choose Instant Watch.

The Instant Watch dialogue box will be displayed:

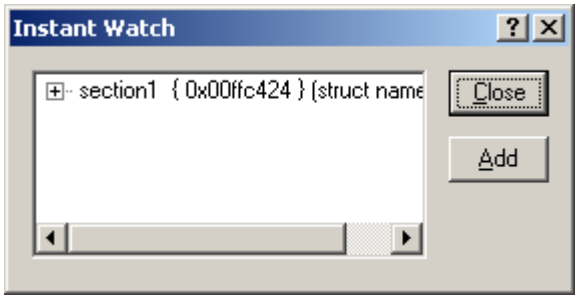


Figure 9.15 Instant Watch Dialogue Box

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

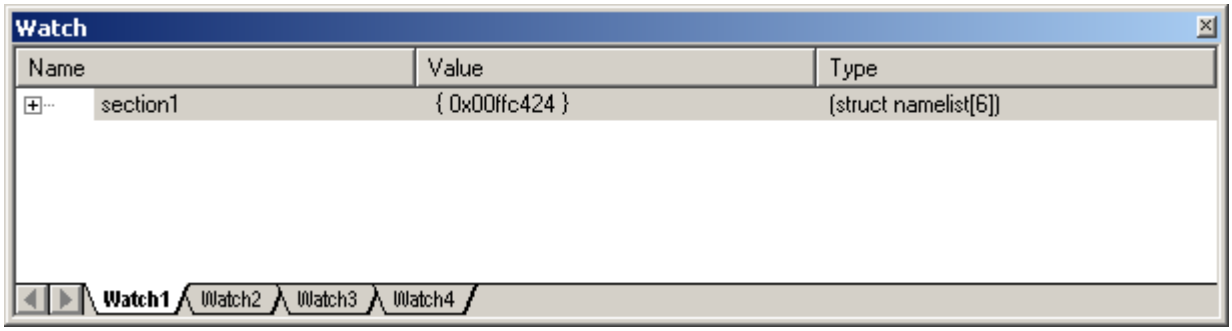


Figure 9.16 Watch Window

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

- Click with the right mouse button within the Watch window and choose Add Watch… from the pop-up menu.

The Add Watch dialogue box appears.

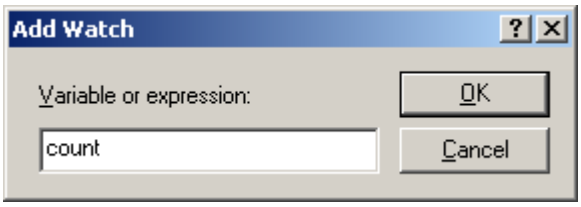


Figure 9.17 Add Watch Dialogue Box

- Type the variable 'count' in the Add Watch dialogue box and click OK.

The Watch Window will show the content of the variable label 'count'.

NOTE: You might be getting different result of 'count'.

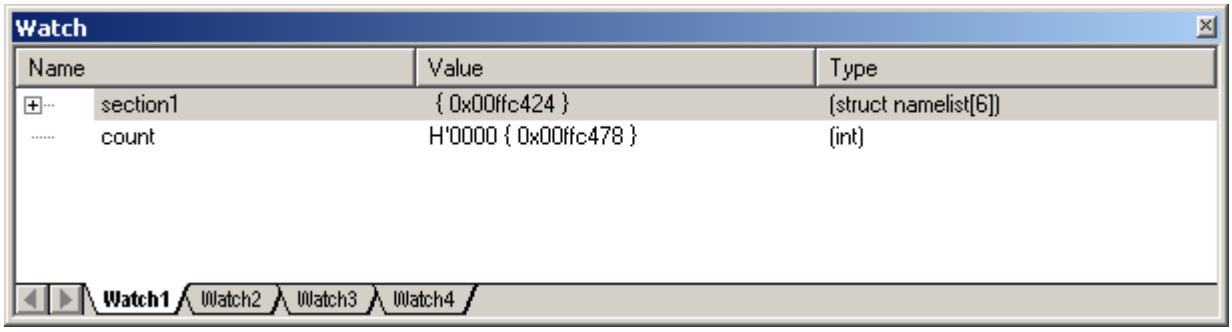


Figure 9.18 Watch Window

You can double-click on the '+' symbol to the left of any symbol in the Watch window to expand it and display the individual elements in the array.

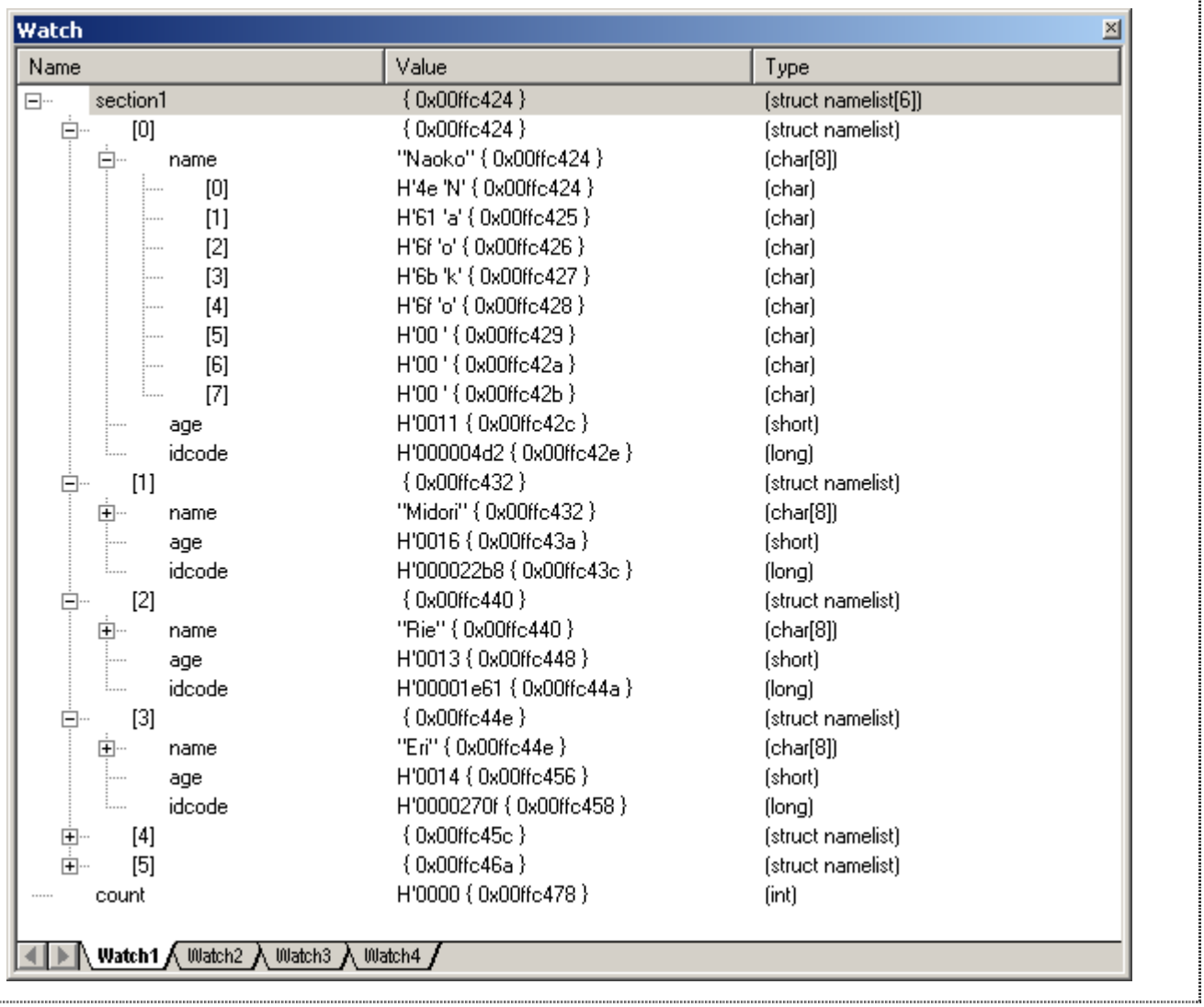


Figure 9.19 Displaying Individual Elements in an Array

9.6. Stepping Through a Program

The CPUBD provides a range of options for stepping through a program (Step In, Step Out and Step Over), executing an instruction or statement.

- Execute up to the breakpoint from the current position by choosing *Go* from the *Debug* menu, or clicking the Go button in the toolbar.

- Issue one *Step In* from the *Debug* menu, or click on the Step In button in the toolbar command to execute into the function sort(section1, NAME).

The yellow arrow will point to the first instruction in the function sort(section1, ID).

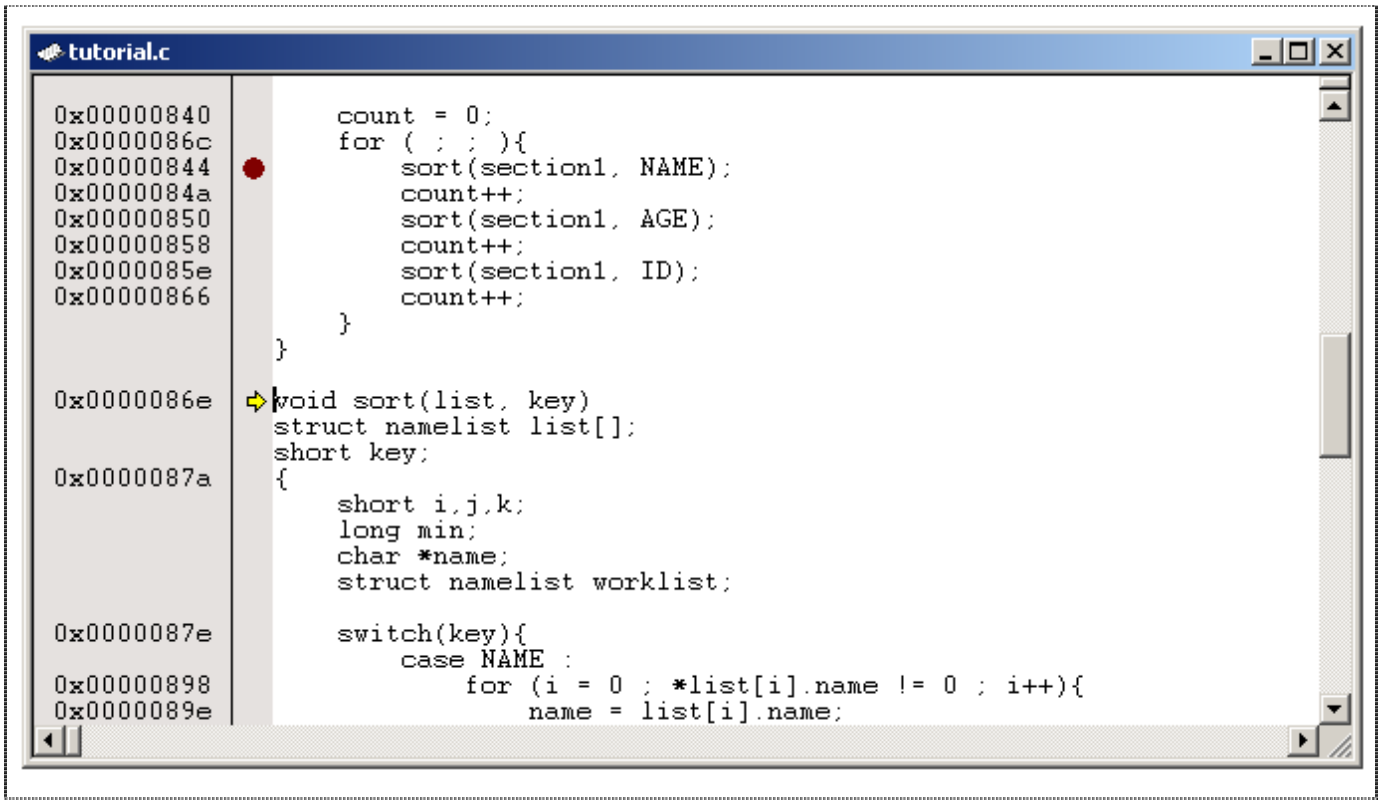


Figure 9.20 Executing up to a Function Call

- Issue another Step In command to execute the next instruction.
- User can also single step the assembly codes by selecting *Step Mode*: *Assembly* in *Debug* menu.

NOTE: After performing several Step In, there will be a time when the Code window will be displayed showing the assembled codes. These codes are included into the user target program to handle certain tasks such as saving or restoring CPU registers etc. C Compiler generates these codes automatically.

9.7. Watching Local Variables

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

For example, in order to examine the local variables in the function sort(), performs the following:

- Open the Locals window by choosing *Symbol*: *Local*… from the *View* menu or clicking the Locals Window button in the toolbar.

NOTE: The Local Window will be empty if there is no local variable declared or local variables have not yet been entered. In another words, user target program execution should halt within a function with local variables to show any variables within Locals Window.

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

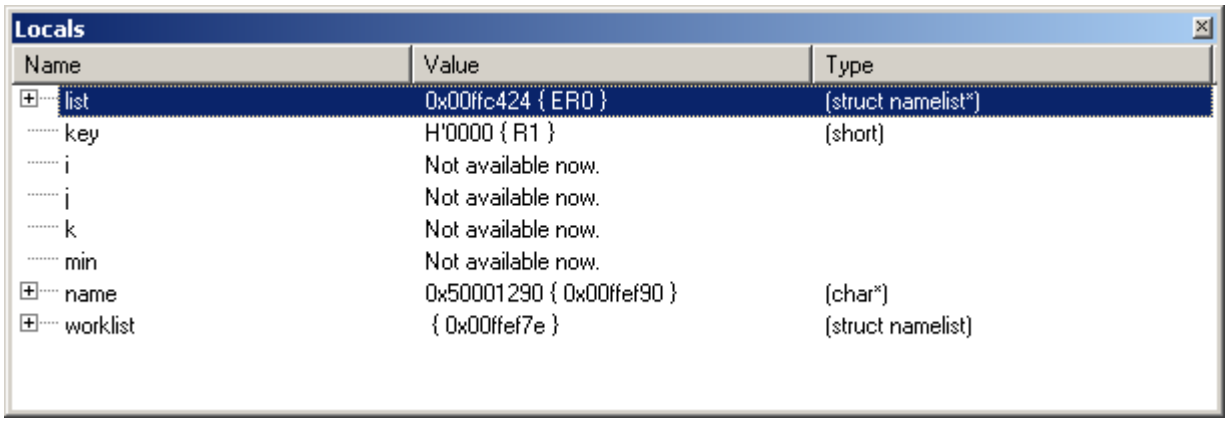


Figure 9.21 Locals Window

- Double-click on the '+' symbol in front of the variable 'list' in the Locals window to display the individual elements of the array 'list'.

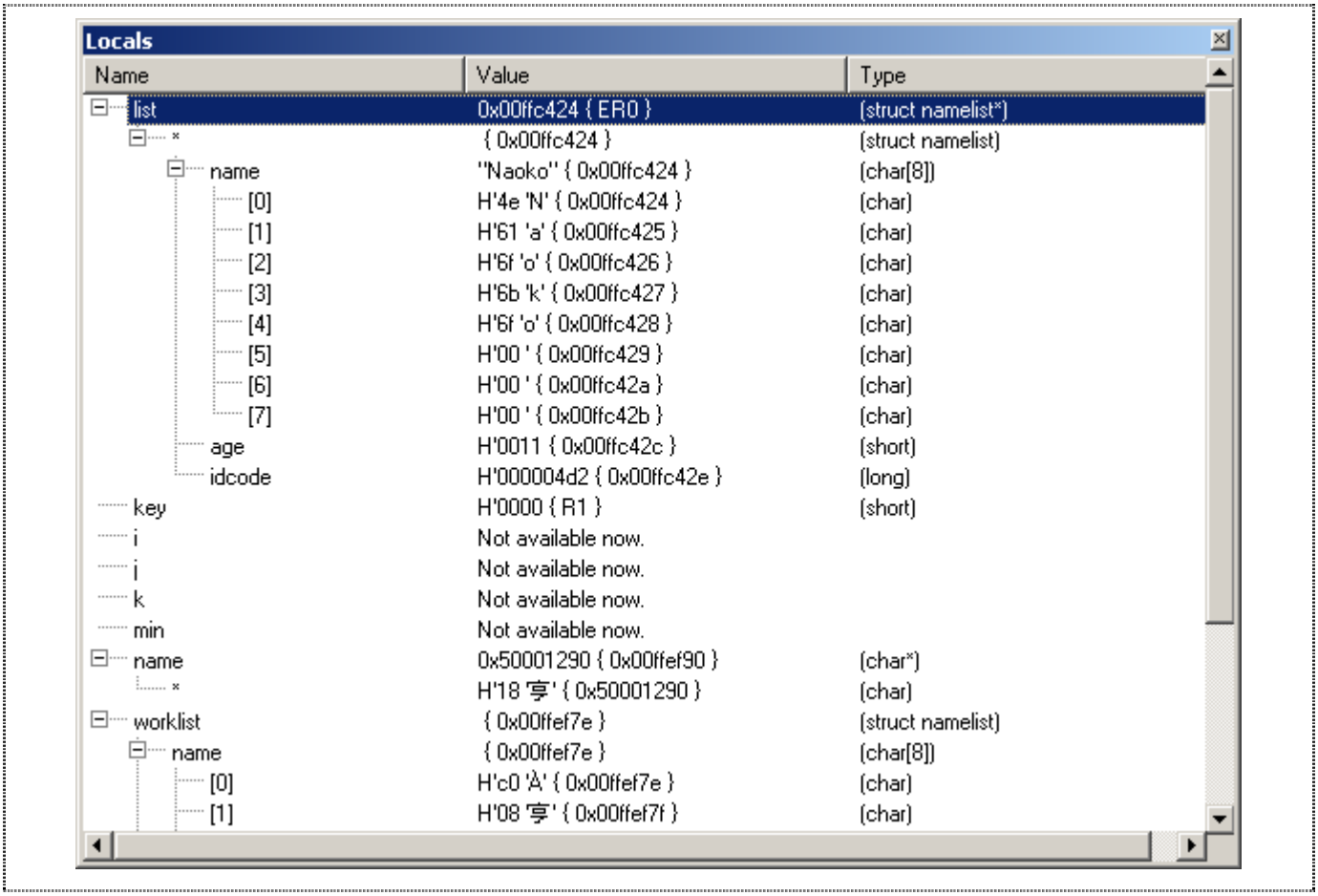


Figure 9.22 Displaying Individual Elements in an Array

9.8. Saves 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.

- Choose *Save Session* from the *File* menu.
- Choose *Exit* from the *File* menu to exit from HEW (Pure Debugger) for CPUBD.

9.9. What Next?

This tutorial has introduced the key features of the CPUBD, and their use in conjunction with the HEW (Pure Debugger) for CPUBD. By combining the debugging tools provided in the CPUBD, user can perform basic debugging to trace for any hardware and software problems by identifying the conditions under which they occur.

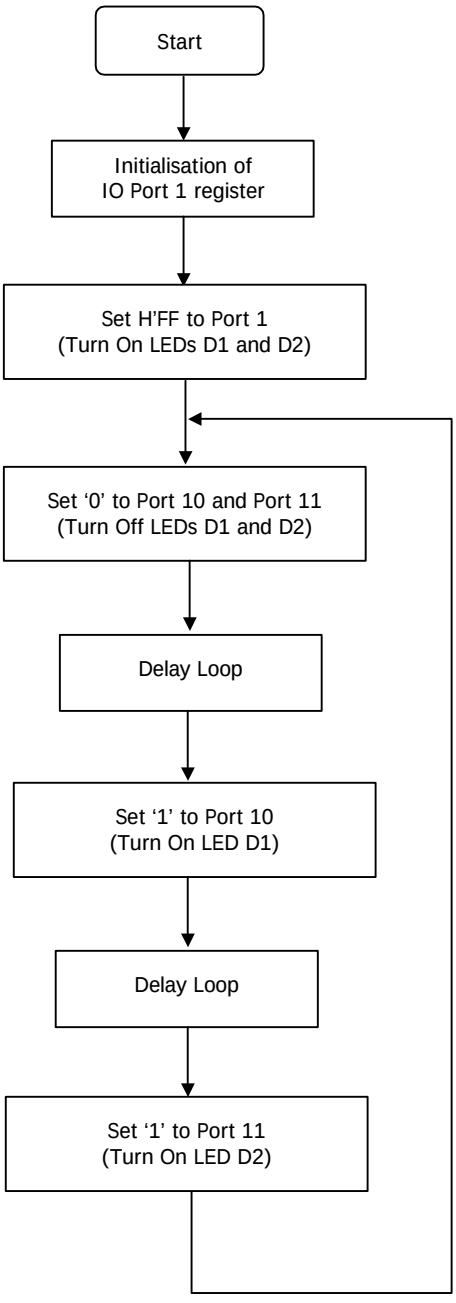
Section 10. Demonstration Program

There are two demonstration programs provided for user to have hands-on experience with the CPUBD in the installed directory:

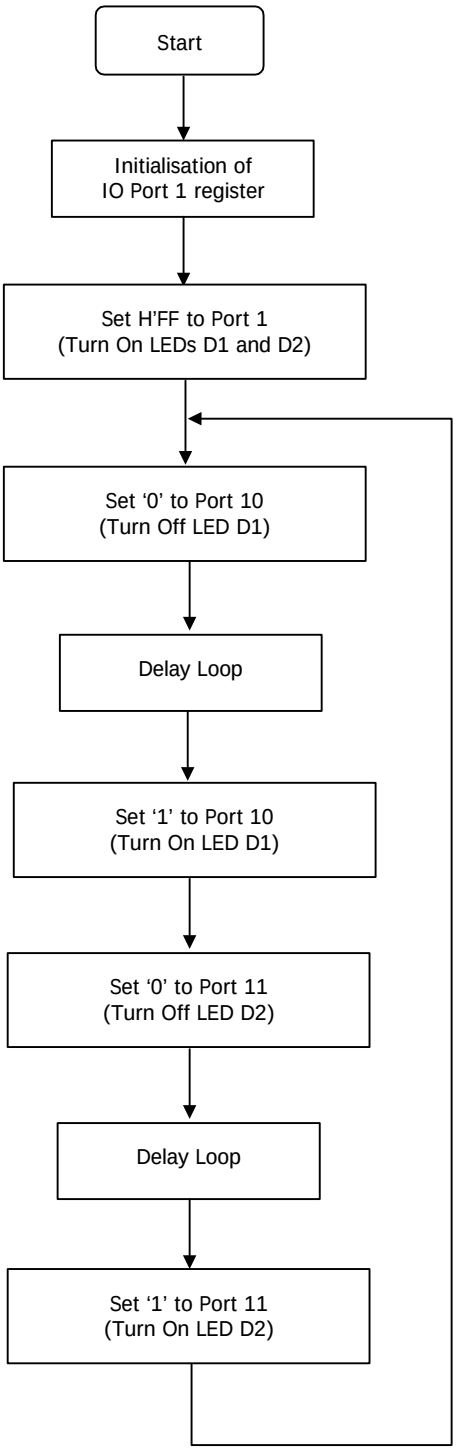
- "install directory*Tools**Renesas**DebugComp**Platform**Emulator**Evb2268F**Sample**Blinking_LED*"
  and
- "install directory*Tools**Renesas**DebugComp**Platform**Emulator**Evb2268F* *Sample**Running_LED*"

You may select to change the ON/OFF speed of the LEDs by changing the value in the delay routine.

10.1. Blinking LEDs



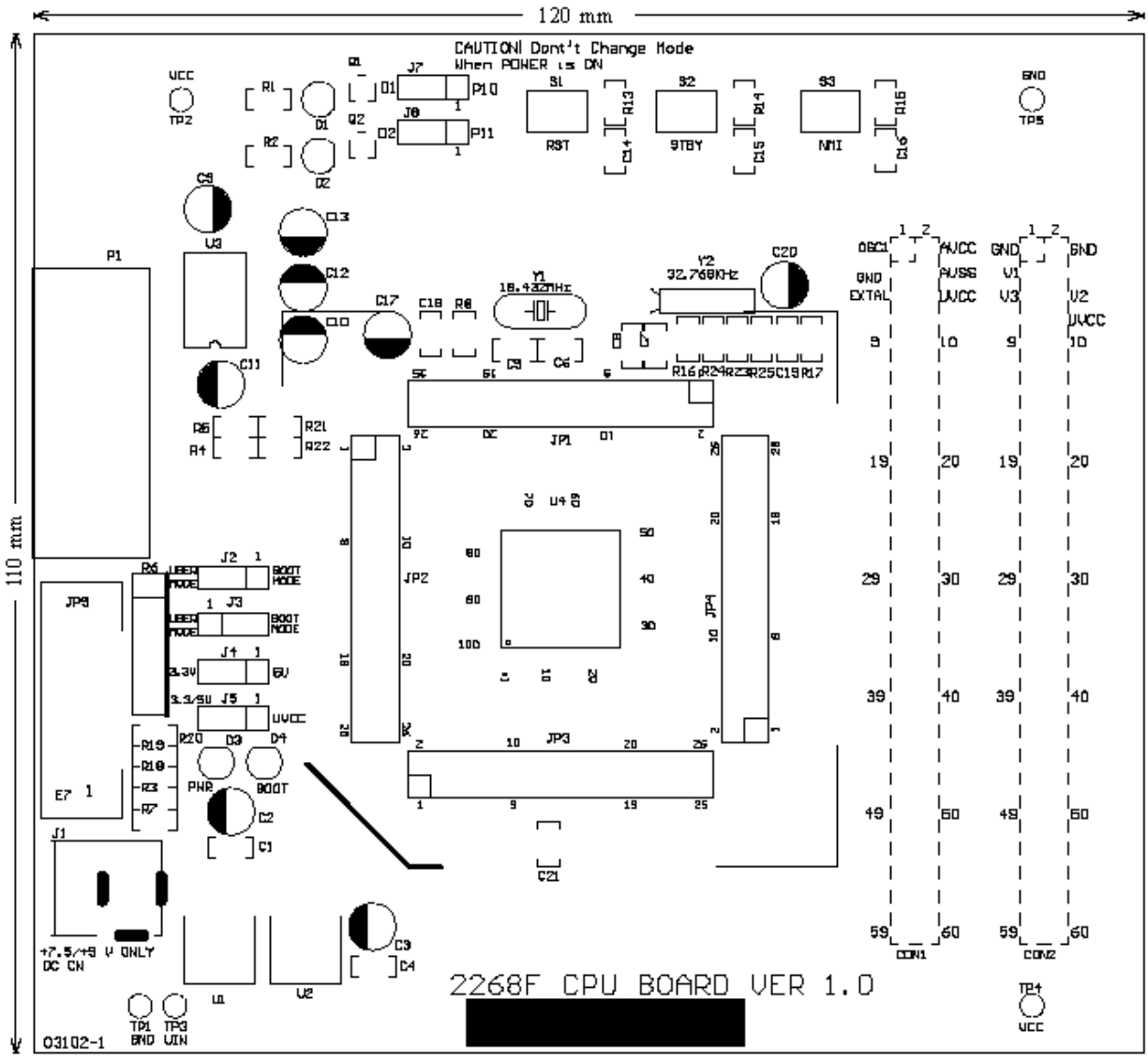
10.2. Running LEDs



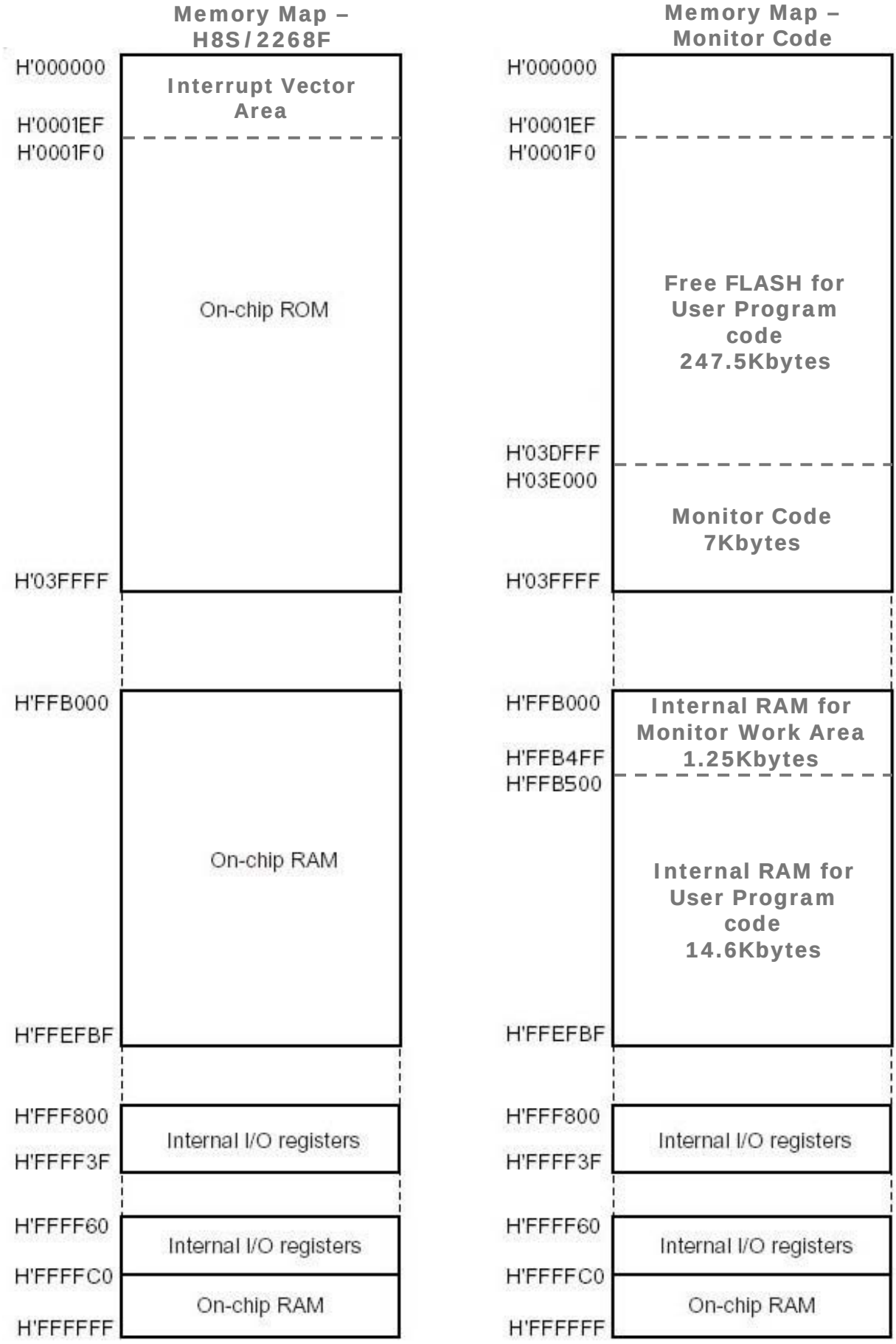
Section 11. Trouble-Shooting

Common Failures	Actions	Remarks
1. Wrong Settings of Jumpers and Switches	❑ Check the manual and set them accordingly.	
2. Power LED off	❑ Check DC input voltage (+7.5V /+9.0V) ❑ Check voltage of the voltage-regulators (≈5.0V) ❑ Check PWR LED D3	❑ If Power supply failure: measure TP2/4 = 5.0V? ❑ Regulator working? ❑ PWR LED broken?
3. Unable to detect CPUBD in "USER MODE"	❑ Check J2 1-2 short? ❑ Check J3 2-3 short? ❑ No monitor program at Flash Memory ❑ Check other software using communication port? ❑ Serial cable connected to P1? ❑ Check U2 pin 11 and 12 for serial data ❑ Check Y1 (18.432MHz) for clock oscillation?	
4. Unable detect CPUBD in "BOOT MODE"	❑ Check J2 1-2 short? ❑ Check J3 1-2 short? ❑ Check other software using communication port? ❑ Serial cable connects to COMM 1 ~ 8? ❑ Check U2 pin 11 and 12 for serial data ❑ Check Y1 (18.432MHz) for clock oscillation?	
5. Flashing Memory failure	❑ Time to change a new IC U1	Typical number of write cycle = 10,000 times
6. Current Overdrawn [Current draws more than 0.05 A]	❑ Identify short traces and then rework as accordingly.	Measure low resistance between Vcc with respect to the ground.

Appendix A CPUBD-2268F Board layout



Appendix B H8S/2268F Memory Map



Appendix C Pin Assignment for JP1~JP4

FP-100B	Descriptions	JP1		Descriptions	FP-100B
51	P41/AN1	1	2	P40/AN0	52
53	VREF	3	4	AVCC	54
55	PH7/TONED/TMCI4	5	6	MD1	56
57	OSC2	7	8	OSC1	58
59	RES_N	9	10	NMI	60
61	STBY_N	11	12	VCC	62
63	XTAL	13	14	VSS	64
65	EXTAL	15	16	FWE	66
67	MD2	17	18	P77/TxD2	68
69	P76/RxD2	19	20	P75/TMO3/SCK2	70
71	P74/TMO2	21	22	P73/TMO1	72
73	P72/TMO0	23	24	P71/TMRI23/TMC23	74
75	P70/TMRI0/TMCIO1	25	26	GND	-

FP-100B	Descriptions	JP2		Descriptions	FP-100B
76	P30/TxD0	1	2	P31/TxD0	77
78	P32/SCK0/SDA1/IRQ4	3	4	P33/TxD1/SCL1	79
80	P34/RxD1/SDA0	5	6	P35/SCK1/SCL0/IRQ5	81
82	PF3/ADTRG/IRQ3	7	8	C2	83
84	C1	9	10	V3	85
86	V2	11	12	V1	87
88	PH3/COM4	13	14	PH2/COM3	89
90	PH1/COM2	15	16	PH0/COM1	91
92	PN7/SEG40	17	18	PN6/SEG39	93
94	PN5/SEG38	19	20	PN4/SEG37	95
96	PN3/SEG36	21	22	PN2/SEG35	97
98	PN1/SEG34	23	24	PN0/SEG33	99
100	PM7/SEG32	25	26	GND	-

FP-100B	Descriptions	JP3		Descriptions	FP-100B
1	PM6/SEG31	1	2	PM5/SEG30	2
3	PM4/SEG29	3	4	PM3/SEG28	4
5	PM2/SEG27	5	6	PM1/SEG26	6
7	PM0/SEG25	7	8	PL7/SEG24	8
9	PL6/SEG23	9	10	PL5/SEG22	10
11	PL4/SEG22	11	12	CVCC	12
13	PL3/SEG20	13	14	VSS	14
15	PL2/SEG19	15	16	PL1/SEG18	16
17	PL0/SEG17	17	18	PK7/SEG16	18

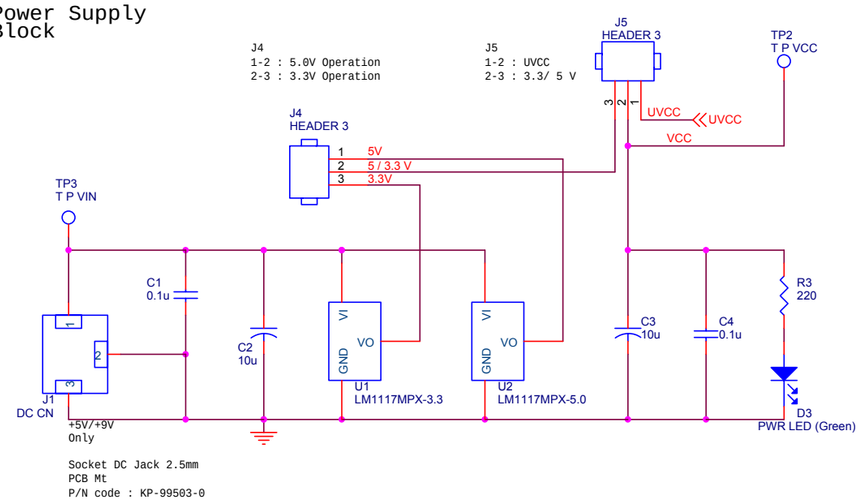
Appendix D Pin Assignment for CON1 & CON2

Signal Name	CON 1		Signal Name
OSC1	1	2	AVCC
GND	3	4	AVSS
EXTAL	5	6	UVCC
GND	7	8	GND
P11	9	10	P10
P13	11	12	P12
P15	13	14	P14
P17	15	16	P16
MD1	17	18	MD2
STBY_N	19	20	NMI
FWE	21	22	NC
NC	23	24	NC
P31	25	26	P30
P33	27	28	P32
P35	29	30	P34
NC	31	32	NC
RES_N	33	34	GND
NC	35	36	VREF
P41	37	38	P40
P43	39	40	P42
P45	41	42	P44
P47	43	44	P46
P71	45	46	P70
P73	47	48	P72
P75	49	50	P74
P77	51	52	P76
PH1/COM2	53	54	PH0/COM1
PH3/COM4	55	56	PH2/COM3
PH7	57	58	P96
P97	59	60	PF3

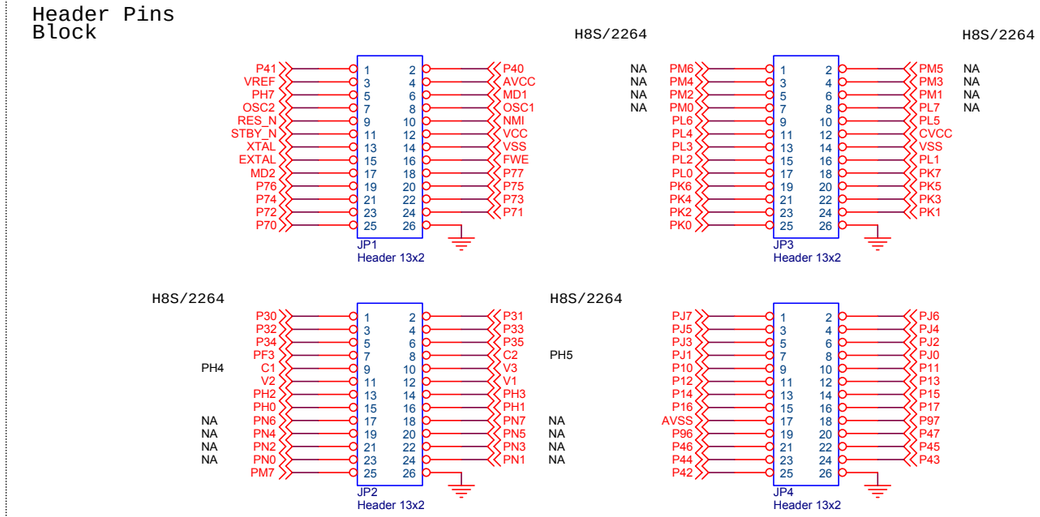
Signal Name	CON 2		Signal Name
GND	1	2	GND
V1	3	4	NC
V3	5	6	V2
GND	7	8	UVCC
C1	9	10	NC
C2	11	12	NC
PJ1	13	14	PJ0
PJ3	15	16	PJ2
PJ5	17	18	PJ4
PJ7	19	20	PJ6
PK1	21	22	PK0
PK3	23	24	PK2
PK5	25	26	PK4
PK7	27	28	PK5
PL1	29	30	PL0
PL3	31	32	PL2
PL5	33	34	PL4
PL7	35	36	PL6
NC	37	38	NC
PM1	39	40	PM0
PM3	41	42	PM2
PM5	43	44	PM4
PM7	45	46	PM6
PN1	47	48	PN0
PN3	49	50	PN2
PN5	51	52	PN4
PN7	53	54	PN6
NC	55	56	NC
NC	57	58	NC
GND	59	60	GND

Appendix E CPUBD-2268F Schematic Drawings

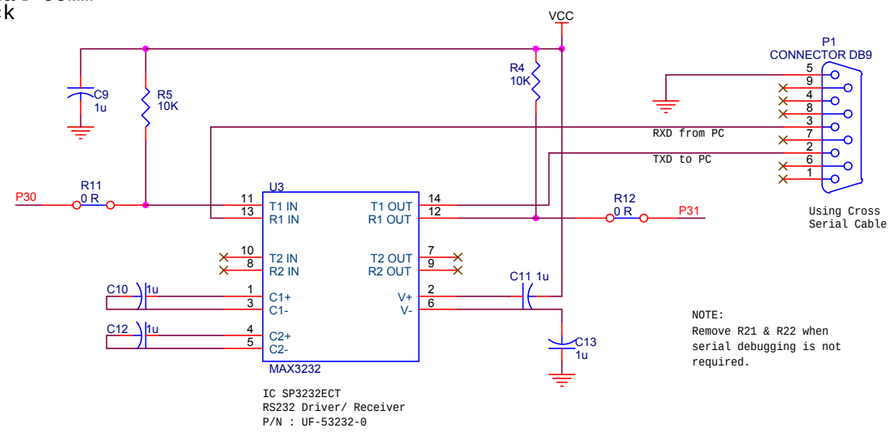
Power Supply Block



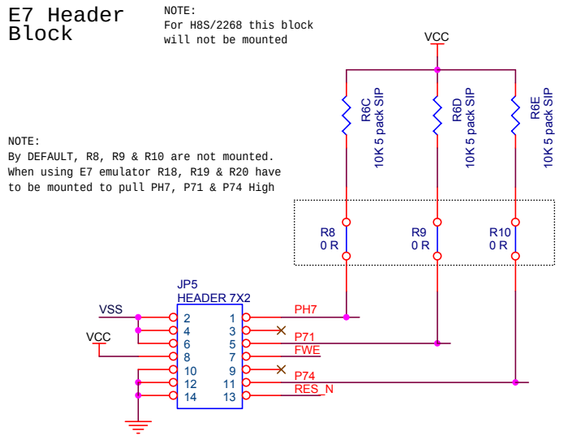
Header Pins Block



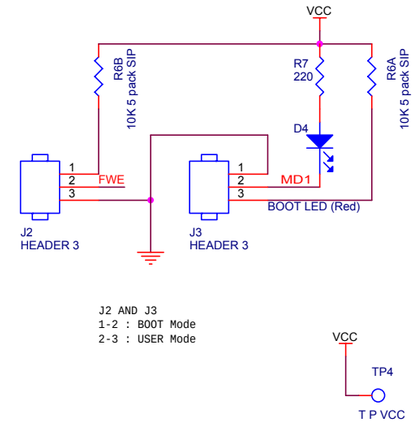
Serial Comm Block



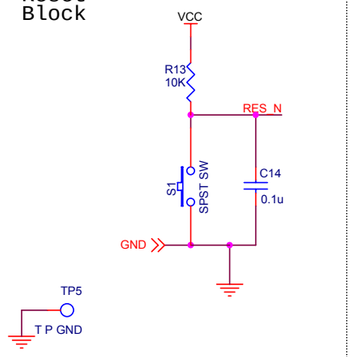
E7 Header Block



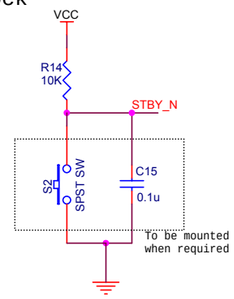
Boot Mode Enable Block



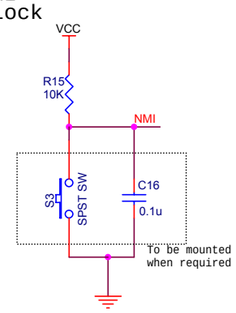
Reset Block



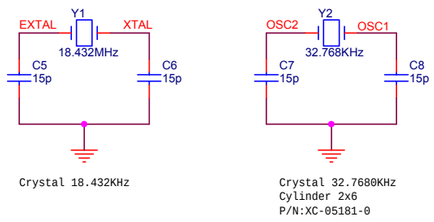
Standby Block



NMI Block

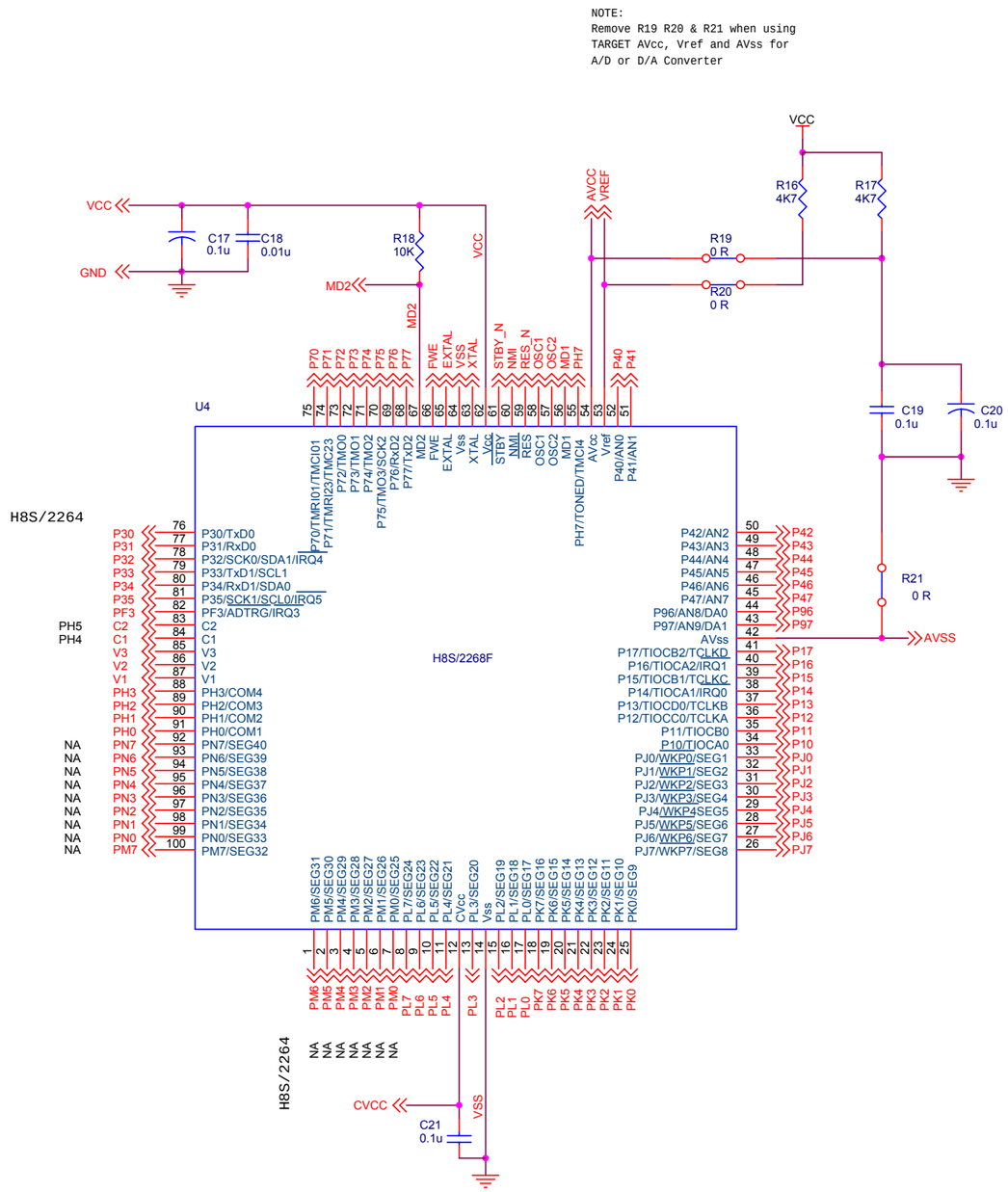


Crystal Oscillator Block

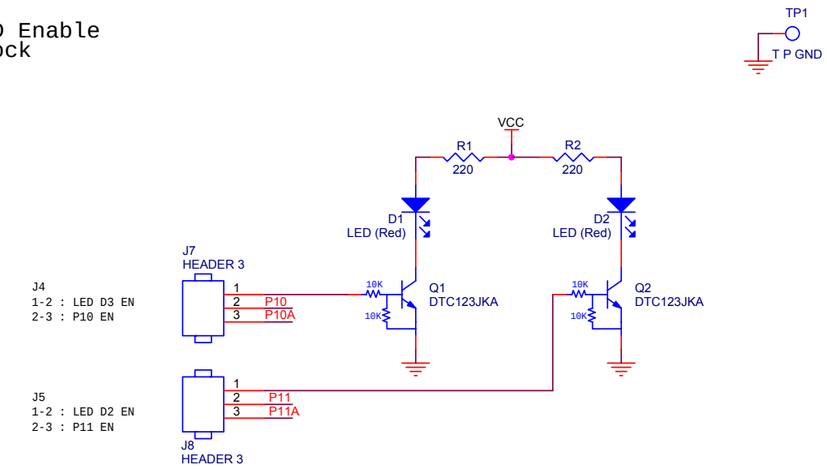


Hitachi Micro Systems Asia Pte Ltd - Singapore Design Centre			
Title: 2268F CPU Board Version 1.0			
Size: A3	Document Number: <Doc>		Rev: 1.0
Drawn By: J.C. Kwok		Checked By:	
Date: Thursday, June 19, 2003		Sheet 1 of 2	

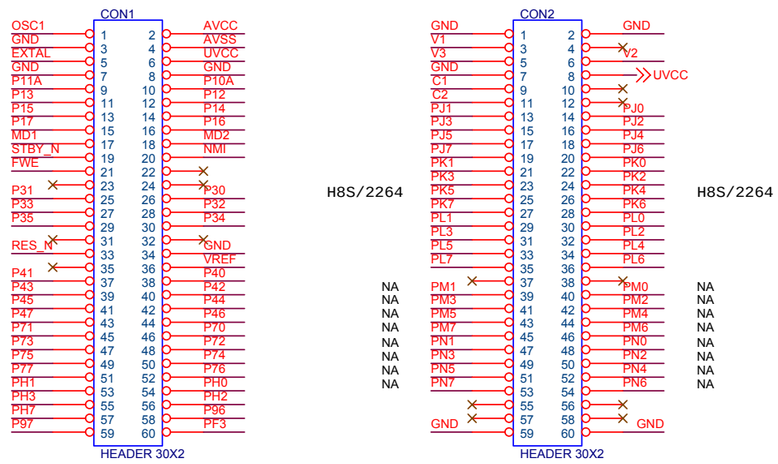
MCU Block



LED Enable Block



Sub-Board Interface Block



Hitachi Micro Systems Asia Pte Ltd - Singapore Design Centre		
Title: MPU		
Size: A3	Document Number: <Doc>	Rev: 1.0
Drawn By: J.C. Kwok		Checked By:
Date: Wednesday, September 17, 2003		Sheet 2 of 2

Appendix F Bill of Materials

Item	Designator					P/N Code	Mfg Name	Part Description	Qty	Package
A) H8S/2268F CPU Board										
1						AA-03102-1		PCB Board 2268 CPU Board Rev1.0	1	
2	C5	C6	C7	C8		CA-70151-6	ANY	Capacitor SMD 0805 15pF / 50V 5%	4	0805
3	C18					CA-73101-0	ANY	Capacitor SMD 0805 10nF / 50V 20%	1	0805
4	C1	C14	C19	C21	C4	CA-74101-3	ANY	Capacitor SMD 0805 100nF / 50V 10%	5	0805
5	C17	C20				CE-14105-2	ANY	Capacitor Ele GSM-R 100nF / 50V	2	Radial
6	C9	C10	C11	C12	C13	CE-15105-0	ANY	Capacitor Ele GSM-R 1uF / 50V	5	Radial
7	C2	C3				CE-16102-0	ANY	Capacitor Ele GSM-R 10uF / 16V	2	Radial
8	J2	J3	J4	J5	J7	KH-20103-1	AUK	Header Pin 0.100" 1x3-Way Gold	6	Thru-Hole
	J8									
9	J1					KP-99501-0	AUK	Connector DC Jack 2.1mm PCB Mt	1	Thru-Hole
10	JP1	JP2	JP3	JP4		KH-20163-1	AUK	Header Pin 0.100" 2x13-Way Gold	4	Thru-Hole
11	P1					KS-60309-0	AUK	Connector D-Sub Female 9-Way RA	1	Thru-Hole
12	D3					LE-03121-1	ROHM	LED 3mm Green Diffused-Rohm	1	Thru-Hole
13	D1	D2	D4			LE-03321-3	ROHM	LED 3mm Red Diffused-Rohm	3	Thru-Hole
14	Q1	Q2				QD-99123-1		Transistor DTC123JKA, SOT23	2	SOT23
16	R11	R12	R19	R20	R21	RA-20005-0		Resistor SMD 0805 1/10W 5% 0R	5	0805
15	R16	R17				RA-20005-0		Resistor SMD 0805 1/10W 5% 0R	2	0805
17	R1	R2	R3	R7		RA-23471-0	ANY	Resistor SMD 0805 1/10W 1% 470R	4	0805
18	R13	R14	R15	R18	R4	RA-25101-0	ANY	Resistor SMD 0805 1/10W 1% 10K	6	0805
	R5									
19	R6					RL-00641-0	ANY	Resistor Netwk-A SIL 1/8W 5% 10Kx6-P	1	Thru-Hole
20	S1					SP-10011-0	AUK	Switch Tactile Round	1	Thru-Hole
21	U3					UF-53232-1	SIPEX	IC SP3232ECT RS232 Driver /Receiver	1	16 pin-wide SOT
22	U2					UR-01117-5	ON-SEMI	IC LM1117DT5.0	1	TO-252
23	Y2					XC-05181-0		Crystal 32.7680 KHz Cylinder 2x6	1	Cylinder 2x6
24	Y1					XC-06850-0		Crystal 18.432MHz HC49/U-S	1	HC49/U-S
25	U4					***CSM		IC H8S/2268F, QFP100	1	QFP100
B) Packaging										
1						BA-61007-0	3M	Rubber Foot Stick On SJ5008	4	
2						BZ-00053-0		Plain Die Cut Box 9 3/4x 7x 3 1/2" S/W E	1	
3	JP7							The 2x7 Box header	1	
4	CON1		CON2			KH-27180-0		Connector PCB Mt 0.100" 2x30-Way	2	Thru-Hole
5								Label for Carton Box	1	
6								Checking List Form	1	
7								Packaging List Form	1	
C) Optional Items										
1	C15	C16				CA-74101-3	ANY	Capacitor SMD 0805 100nF / 50V 10%	2	0805
2	JP5					KH-20157-1	AUK	Header Pin 0.100" 2x7-Way Gold	1	Thru-Hole
3	S2	S3				SP-10011-0	AUK	Switch Tactile Round	2	Thru-Hole
4	R8	R9	R10			RA-20005-0	ANY	Resistor SMD 0805 1/10W 5% 0R	3	0805
5	U1					UR-01117-4	ON-SEMI	IC LM1117DT33	1	TO-252

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

CPUBD-2268F

Renesas System Solutions Asia Pte. Ltd.