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.

RENESAS

32

LCEVB-SH2

For SH2 Series Low-Cost Evaluation Board

Microcomputer Development Environment System

User's Manual

2003.4

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

Microcomputer Development Environment System

LCEVB-SH2
SH2 Evaluation Board
User's Manual

Renesas System Solutions Asia Pte. Ltd.

LCEVB-SH2 – Low-cost Evaluation Board for SH7040 Series Microcomputer

User's Manual

Published by : Renesas System Solutions Asia Pte. Ltd.
Date : April 1st, 2003, Version 1.0

Trademarks

(a) General

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

(b) Specific

Microsoft MS-DOS is registered trademark.

MS-Windows is a trademark of Microsoft Corporation.

Pentium is a registered trademark of Intel.

IMPORTANT INFORMATION

- **READ this user's manual before using this product.**

- **KEEP the user's manual handy for future reference.**

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

LCEVB-SH2 Evaluation Board:

Throughout this document, the term "LCEVB-SH2" shall be defined as the LCEVB-SH2 emulation system produced only by Renesas System Solutions Asia Pte. Ltd. excluding all subsidiary products.

Purpose of LCEVB-SH2:

This emulation product is a software and hardware development tool for application systems employing the SH2 series microcomputer. It should 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 emulation products. 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 Emulation Product:

User of this emulation product should have carefully read and thoroughly understood the information and restrictions contained in the user's manual before using it. Do not attempt to use the emulation product until you fully understand its mechanism.

It is highly recommended that users who know how to operate this emulation product give proper training to users who are not familiar with the operation of this product.

LIMITED WARRANTY

Renesas warrants its emulator products to be manufactured in accordance with published specifications and free from defects in material and/or workmanship. Renesas, at its option, will repair or replace any emulator products returned intact to the factory, transportation charges prepaid, which Renesas, upon inspection, shall determine to be defective in material and/or workmanship. The foregoing shall constitute the sole remedy for any breach of Renesas warranty. This warranty extends only to you, the original Purchaser. It is not transferable to anyone who subsequently purchases the emulator product from you. Renesas is not liable for any claim made by a third party or made by you for a third party.

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 emulation 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 damage 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 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

This guide explains how to setup and use the LCEVB-SH2 for the SH7040 series of MCU.

Section 1 Introduction

Gives an introduction to the system, package, specification and functions.

Section 2 Start Up Instructions

Explains how to setup and install LCEVB-SH2.

Explains how to setup and install Hitachi Debugging Interface program (HDI).

Section 3 Operation

Explains operation of each component.

Section 4 Board Options

Describes various user-settable functions used in LCEVB-SH2.

Section 5 Hitachi Debugging Interface (HDI)

Introduces HDI and provides a step by step guide in using the LCEVB-SH2 to perform emulation.

Section 6 Tutorials

Provides a step by step guide in using the LCEVB-SH2 to perform emulation.

Related Manuals:

- SuperH RISC engine C/C++ Compiler, Assembler, Optimizing, Linkage Editor User's Manual
- SH-1/SH-2 Series Programming Manual
- SH7040 Series Hardware Manual

Contents

Figures & Tables

Section 1 Introduction

The SH7043 Evaluation Board (LCEVB-SH2) is an inexpensive demonstration and evaluation tool for the SH7043 family of RISC microcomputers. Figure 1.1 shows the physical layout of the LCEVB-SH2.

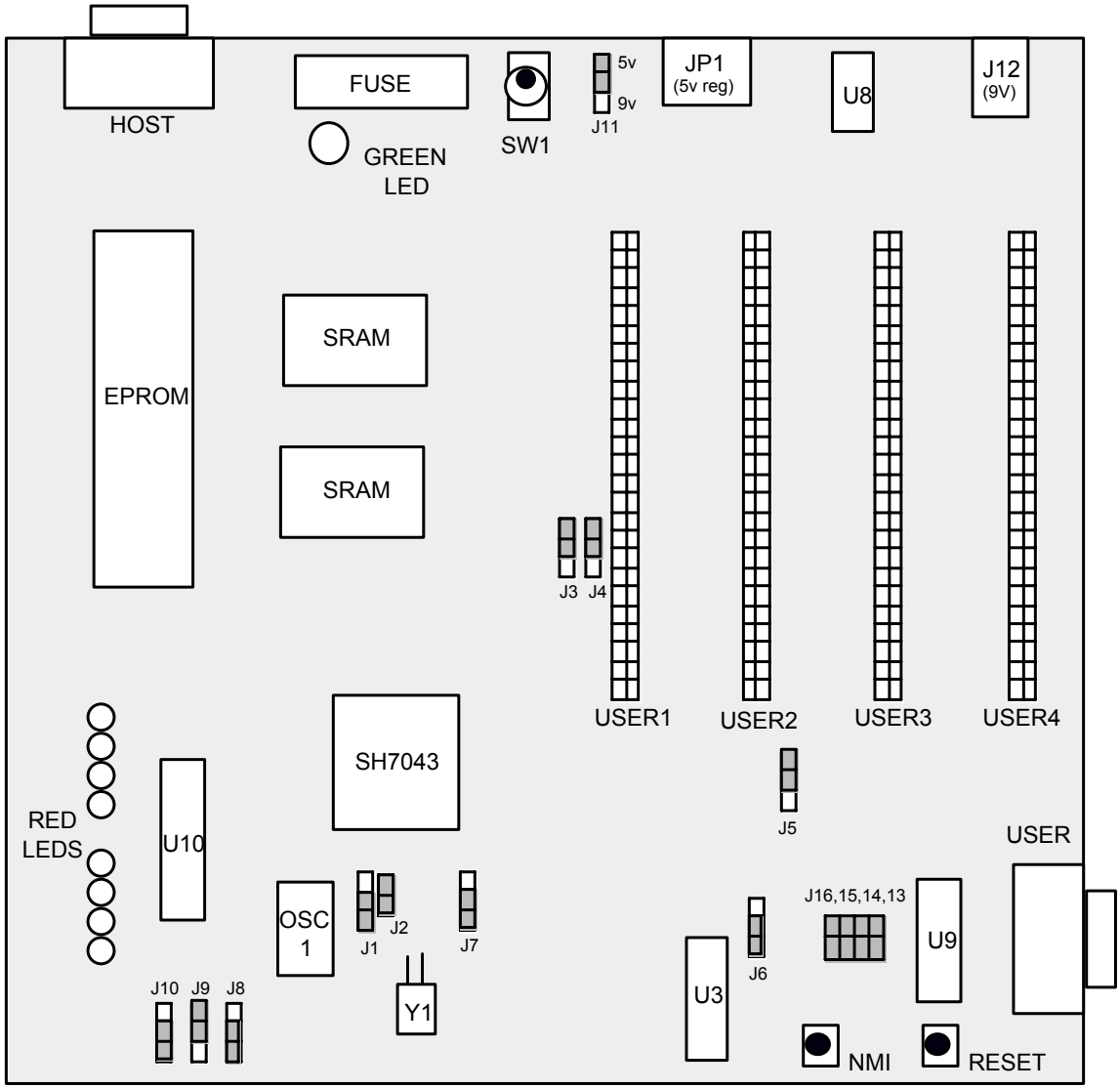


Figure 1.1 LCEVB-SH2 Layout (not drawn to size)

Schematic diagrams are provided at the back of this manual.

1.1 Functional Blocks

At the top level, the LCEVB-SH2 is composed of the SH7043 microcomputer, ROM, RAM, and two serial ports, as shown in figure 1.2.

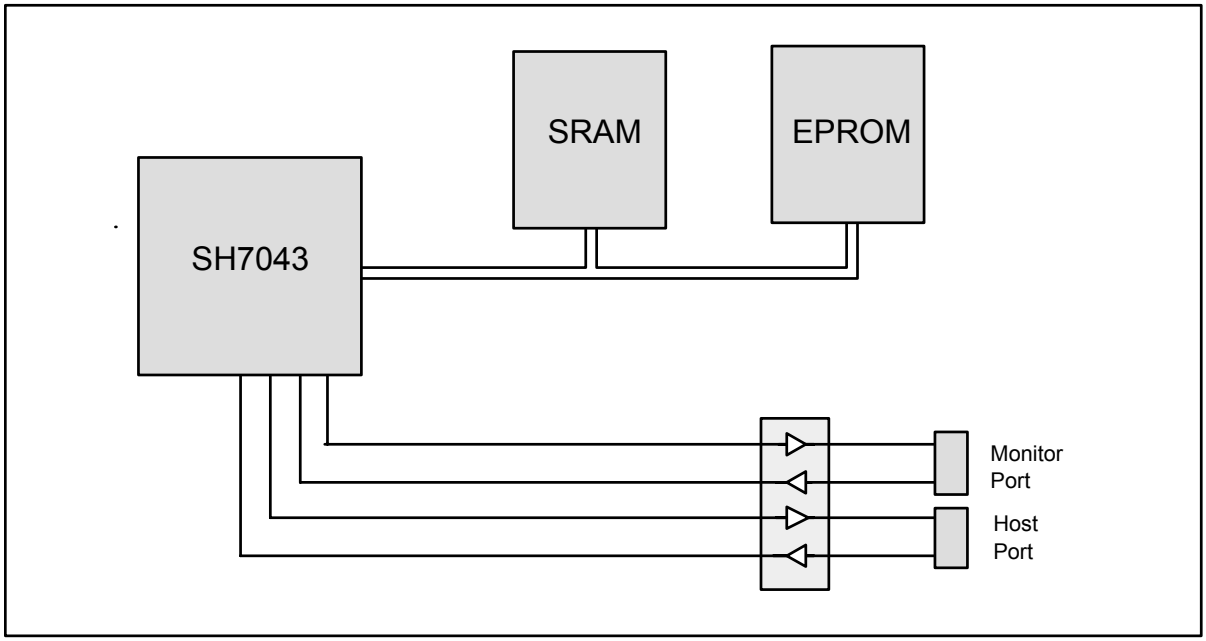


Figure 1.2 LCEVB-SH2 Functional Block Diagram

The SH7043 microcomputer contains most of the decoding and other glue logic necessary to implement an expanded memory SH7043 based system. Read-only memory (EPROM) contains the firmware monitor program, CMON. Two byte-wide RAM blocks are used side-by-side to provide word-wide reads and writes. A serial transceiver supports two three-wire serial ports using the two on-board the SH7043 UARTs. One port is dedicated to the on-board CMON monitor, and the second is available to the user. A PC running a terminal emulation program is typically used to communicate with CMON.

Users reconfiguring LCEVB-SH2 I/O ports are cautioned that pull-up resistors may be required for proper operation in some configurations. In particular, users adding external memory in area 3 should be aware that the chip selects provided by SH7043 are shared and may be floating until configured.

1.2 Specifications

1.2.1 General

- 28.6363 MHz SH7043 microcomputer
- 128 Kbytes Static CMOS RAM, accessed either as (16 bit) words or bytes
- 128 Kbytes of 16 bit EPROM
- CMON firmware monitor communicates with Hitachi Debugging Interface program (HDI)
- An 8 bit Buffer-driven LED port in area 1
- Most SH7043 signals available for user connection

1.2.2 Communications

- 57600-baud RS-232 host interface communications
- RS-232 connection via two DB-9S connectors

1.2.3 Power

- Power input either regulated 5V DC or unregulated 9V DC

1.2.4 Memory Map

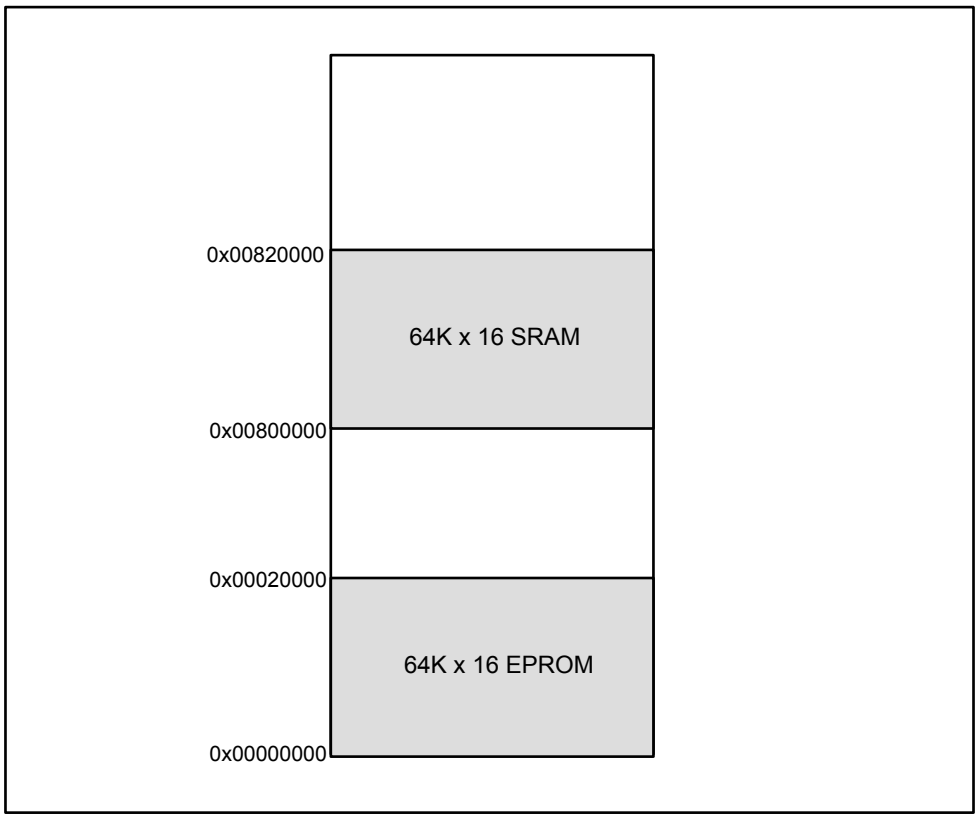


Figure 1.3 LCEVB-SH2 Memory Map

Section 2 Start-Up Instructions

2.1 Installing the LCEVB-SH2 Evaluation Board

Installing the LCEVB-SH2 requires connecting power and serial communications to a host computer. Host computer communication is facilitated with the HINT program resident on the host computer.

2.2 Power Supply

The LCEVB-SH2 hardware requires a regulated power supply of +5 V DC at approximately 100 mA supplied to JP1 or an unregulated +9V DC supply supplied to J12. Since total power consumption can vary widely due to external connections, the SH7043 port state, and memory configuration, use a power supply capable of providing at least 500 mA.

2.3 Power Connection

The LCEVB-SH2 requires an external Power supply or a DC adapter unit to operate. Figure 2.1 shows the necessary connection to the board. When connected correctly the GREEN power indicator LED should light up. To select between DC adapter or external power supply simply change the connection at Jumper J11. J11 (1-2) is the default connection and it connects the LCEVB-SH2 to an external 5V regulated power supply. J11 (2-3) is jumpered when a 9V DC adapter is used. The connection should not be left open.

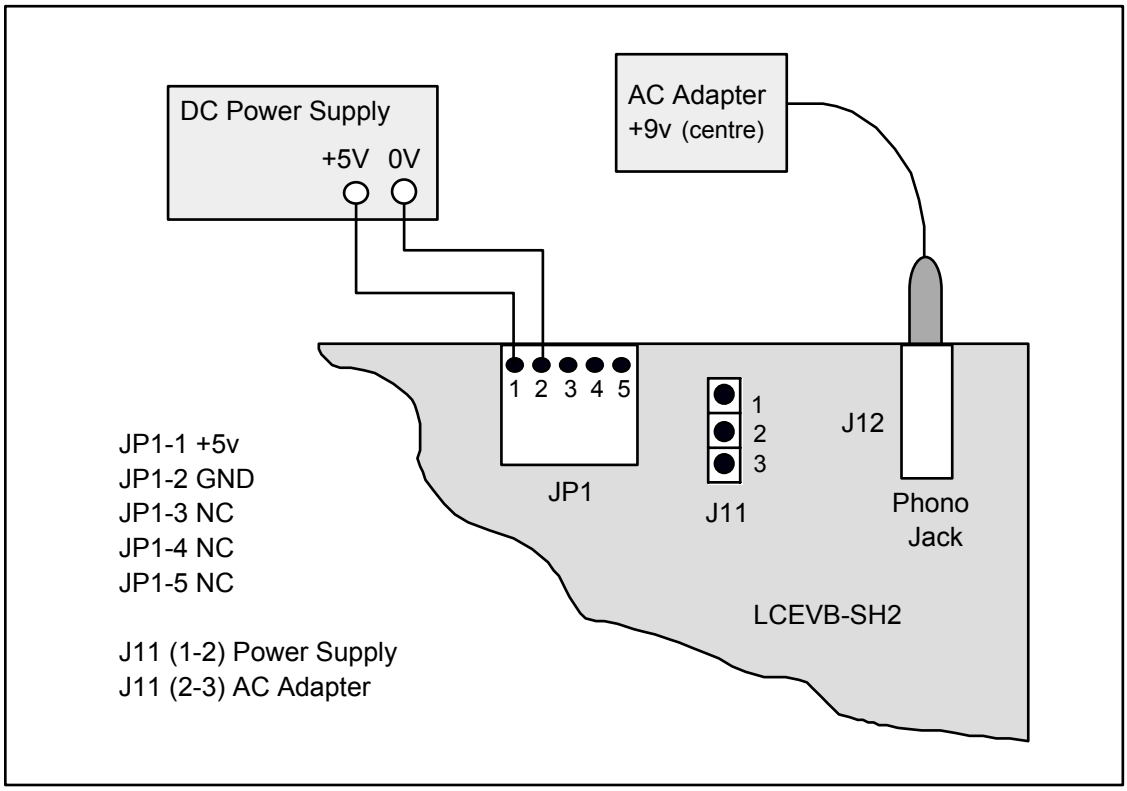


Figure 2.1 Power Supply Connection to LCEVB-SH2

2.4 HDI Installation

Please refer to Section 5.1.

2.5 Checking the System

The next step is to run the HDI software to check that the LCEVB-SH2 is working correctly.

- Switch on the LCEVB-SH2 and check that LED D15~D8 display 0x31.
- Execute the HDI program.

With everything set up correctly the HDI will be displayed, and the following sequence of messages will be shown in the status bar at the bottom of the window:

Figure 2.2 HDI Start-Up Messages

Finally the status bar will display Link up to indicate that everything is set up correctly, and the HDI screen will be displayed as shown below.

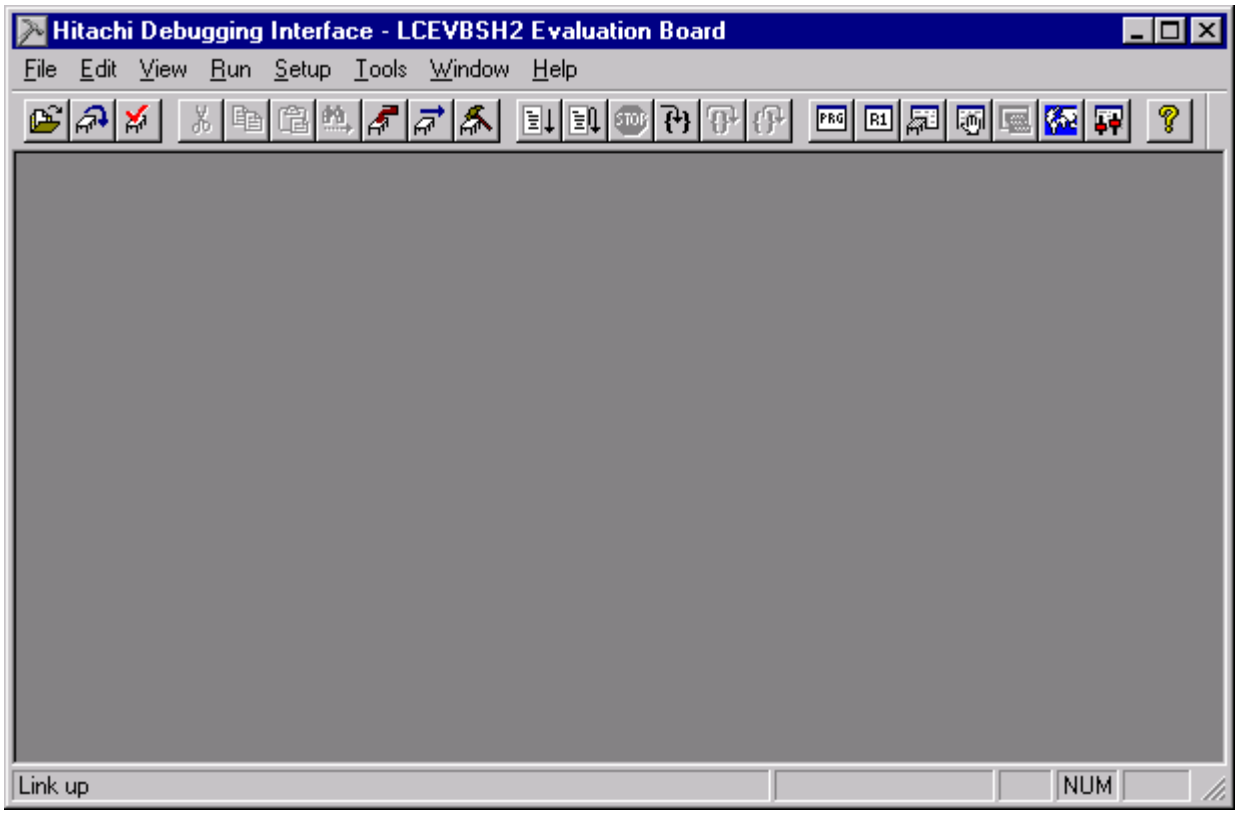


Figure 2.3 HDI Desktop

Perform the Diagnostic Test to verify the LCEVB-SH2 functionality. Please refer to Section 5.19 for detail.

Section 3 Operation

The LCEVB-SH2 includes the following components:

- SH7043 RISC Microcomputer
- Clock circuitry
- Reset circuitry
- NMI circuitry
- EPROM memory
- SRAM memory
- Serial interface
- LED drivers
- External user interface

Complete LCEVB-SH2 schematics are provided as part of the LCEVB-SH2 kit and are referenced throughout this chapter. Where loading options occur, the schematics always reference the alternative device with the most pins.

3.1 SH7043 RISC Microcomputer

Because the SH7043 provides on-board many of the functions required to implement an expanded-memory microcomputer system (for example, address area decoding), the amount of glue logic required is minimized.

3.2 Clock Circuitry

The LCEVB-SH2 may use one of two clock sources. The LCEVB-SH2 is delivered configured with an AT-cut parallel resonating crystal and this is the default clock source (Y1). Alternatively by changing the jumper connection, a standard TTL "can" oscillator may be used. Refer to section 4 for loading options.

3.3 Reset Circuitry

3.3.1 Reset Generator

The reset generator for the LCEVB-SH2 is a Dallas Semiconductor DS1233 "Econo Reset" device. The DS1233 monitors its supply voltage. When the supply voltage is out of tolerance, the DS1233 pulls its reset input/output line active-low. This condition continues indefinitely. After the voltage reaches tolerance, the reset is held low for an additional 350 ms to allow for final supply stabilization and microcomputer reset.

The DS1233 monitors its own reset output so that a push-button can be used as a reset source. The DS1233 debounces the switch and provides a 350-ms reset signal when the reset push-button is released. SW1 is used to activate this function.

Quickly switching power to the board off and then on again may not allow V_{CC} to fall low enough to generate a reset pulse. In practice, the SH7043 usually continues to operate normally. Rapid switching of the power supply stresses the integrated circuit components and is not recommended.

3.3.2 Reset and Non-Maskable Interrupt (NMI)

The SH7043 distinguishes between a power-on reset and a manual reset by sampling the state of the NMI input when the RESET line goes high. If NMI is high at this point, a power-on reset sequence is initiated internally, and the SH7043 is initialized throughout. If NMI is low, the manual reset sequence initiates the SH7043 excepting the bus state controller, the pin function controller, and I/O ports. The LCEVB-SH2 by default generates a power-on reset when power is applied and when the reset push-button is used, as detailed in section 3.4.

3.4 NMI Circuitry

The NMI input of the SH7043 is an independent edge-triggered input. NMI may be generated on the positive or negative-going transition, depending on the setting of the Interrupt Control register (ICR) NMIE bit.

The LCEVB-SH2 uses two NAND gates (U3A and U3B) as delay circuit to de-bounce the output of momentary push-button S2. In the quiescent case, the output of U3B (and thus the NMI input) is high. Closing S2 pauses the NMI signal to go low until S2 is released. The default value of ICR.NMIE is 0, and NMI is generated when NMI goes low. Multiple bounces of the switch on the normally open closure will have no further effect (switches bounce on the active closure only), and the NMI signal will stay low until S2 is released.

Since the quiescent state of NMI is high, closing the reset push-button (S3) always generates a power-on reset. In other words, when the board is reset, all SH7043 internal circuitry is normally affected. It is possible to generate a manual reset (leaving the bus state controller, pin function controller, and I/O port values untouched) with the following sequence:

1. Close the reset switch (S3), putting the SH7043 into reset state.
2. Close the NMI switch (S2), generating a negative-going edge on the NMI pin which is ignored.
3. Release the reset switch (S3), starting the SH7043 with the NMI pin low.
4. Release the NMI switch (S2), returning the NMIpin to its base state.

Alternatively, external connections can be used to affect the NMI signal if jumper J6 is changed from its default setting. Refer to section 4 for details.

3.5 ROM

The LCEVB-SH2's EPROM memory is provided by U4 which is a 64k × 16, 27C1024HG. Figure 1.3 shows the memory map. The EPROM is located in address area 0 of the SH7043 memory space, starting at location 0. U4 is always accessed two bytes at a time. The memory area select signal CS0 is generated by the SH7043 and is used to select the device. The value of the SH7043 wait state control register 1 (WCR1) bits W00 .. W03 control the number of wait states automatically inserted for accesses to area 0 and (WCR1) bits W20 .. W23 to area 2 by the SH7043 on-board bus state controller. Since RAM memory is located in area 2, the access time requirements for both RAM and can be set independently allowing for the use of high speed RAM.

3.6 RAM

The LCEVB-SH2's SRAMs are U5 and U6, which are a pair of 64k × 8, 62864-family static CMOS RAMs organized for word-wide access. Figure 1.3 shows the memory map. Reads from the RAM are always word-wide while the WRH and WRL write signals allow individual bytes to be written. The RAM memory is located in area 2 of the SH7043 memory space, starting at location H'800000. The value of the SH7043 wait state control register 1 (WCR1) bits W00 .. W03 control the number of wait states automatically inserted for accesses to area 0 and (WCR1) bits W20 .. W23 to area 2 by the SH7043 on-board bus state controller. Since RAM memory is located in area 2, the access time requirements for both RAM and can be set independently allowing for the use of high speed RAM.

3.7 Serial Interface

The LCEVB-SH2 supports two three-wire serial channels using the two identical SH7043 SCI UART-type devices, SCI-0 and SCI-1. Of these, SCI-1 (labeled HOST on the board) is normally dedicated to use by CMON for communications with a terminal host. SCI-0 (labeled USER on the board) is available to the user. CMON is capable of exchanging the assignments of SCI-0 and SCI-1.

U9 is a serial transceiver device that translates RS-232 signals to logic levels and vice-versa. This device provides two channels in each direction, enough to support TxD and RxD for each of two channels. As described below in section 4, U9 is loaded with a standard 16-pin MAX-232 device.

3.8 LED Driver

U10 is an octal latch (74HCT534) used to drive the bank of eight red LEDs and can be written by a write to any address in memory area 1 through the use of CS1. The bank of eight LEDs are written from data lines D8 .. D15 so for word writes the desired information should be in the upper byte.

3.9 External User Interface

The external user interface makes most SH7043 microcomputer signals available to users consistent with keeping:

- Signal lines short
- Board design simple
- Signals are assigned compatible with Japan User Cable
- Lines potentially used for analog signals isolated

The external user interface consists of 4 two-row connectors of 50 pins each. Table 3.1 lists LCEVB-SH2 connectors and signals.

Table 3.1 Address and Data Connectors and Signals

Connector*	Signals
UCN1	SH7043 data lines, (D0–D29)
UCN2	(PE0 .. PE6), the analog port (PF0 .. PF7), analog ground (AVSS), port A lines (PA16, PA17), user NMI (UNMI), and the clock (CK)
UCN3	Address lines (A0 .. A7), port A lines (PA0 .. PA5, PA21 .. PA23), port E lines (PE7 .. PE15), analog reference (AVREF), and a analog power (AVCC)
UCN4	Address lines (A8 .. A17), data lines (D30, D31), port A lines (PA8, PA9, PA18 .. PA20), port B lines (PB2 .. PB9), chip selects (CS0 .. CS3), RD-, WRH-, WRL-, and WDT0VF-

Note: Each of these external user interface connectors includes V_{CC}, normally at +5 V. Trivial external circuitry may use V_{CC} from the LCEVB-SH2. External circuits drawing >50 mA at +5 V should be powered by an independent power supply.

Note the positioning of pin 1 on each connector. The pins are numbered odd-even as shown in figure 3.1.

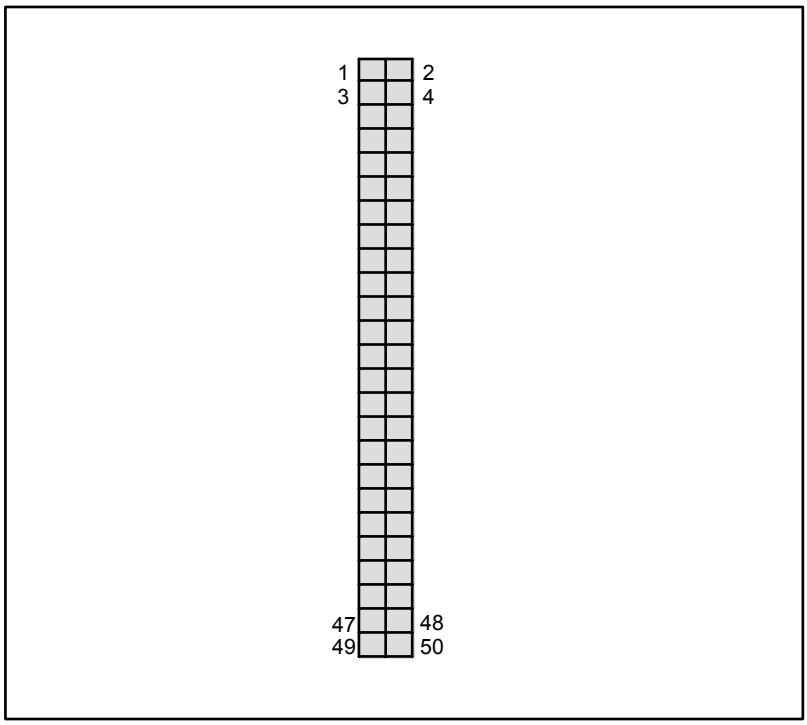


Figure 3.1 Connector Configuration

Table 3.2 on the next page describes each user connector for to LCEVB-SH2 in numerical order.

Table 3.2 SH7043 Connector Pin-out

UCN1

Pin No.	Signal Name	CPU	Pin No.	Signal Name	CPU
1	GND	NC	2	GND	NC
3	PD29	56	4	PD28	57
5	GND	NC	6	PD27	58
7	PD26	59	8	PD25	60
9	GND	NC	10	PD24	62
11	PD23	64	12	GND	NC
13	PD22	65	14	PD21	66
15	GND	NC	16	PD20	67
17	PD19	68	18	GND	NC
19	PD18	69	20	PD17	70
21	GND	NC	22	PD16	72
23	PD15	73	24	GND	NC
25	PD14	74	26	PD13	75
27	GND	NC	28	PD12	76
29	PD11	78	30	GND	NC
31	PD10	80	32	PD9	81
33	GND	NC	34	PD8	82
35	PD7	83	36	GND	NC
37	PD6	84	38	PD5	86
39	GND	NC	40	PD4	88
41	PD3	89	42	GND	NC
43	PD2	90	44	PD1	91
45	PD0	92	46	GND	NC
47	GND	NC	48	GND	NC
49	GND	NC	50	GND	NC

UCN2

Pin No.	Signal Name	CPU	Pin No.	Signal Name	CPU
1	GND	NC	2	VCC	NC
3	NMI	98	4	PA16	100
5	PA17	101	6	GND	NC
7	GND	NC	8	GND	NC
9	GND	NC	10	VCC	NC
11	GND	105	12	GND	NC
13	PA15	107	14	VCC	NC
15	GND	NC	16	PE0	109
17	PE1	110	18	GND	NC
19	PE2	111	20	PE3	113
21	GND	NC	22	PE4	114
23	PE5	115	24	PE6	116
25	GND	NC	26	PF0	118
27	PF1	119	28	GND	NC
29	PF2	120	30	PF3	121
31	GND	NC	32	PF4	122
33	PF5	123	34	AVSS	124
35	PF6	125	36	PF7	126
37	GND	NC	38	VCC	NC
39	VCC	NC	40	VCC	NC
41	GND	NC	42	NC	NC
43	GND	NC	44	NC	NC
45	GND	NC	46	NC	NC
47	GND	NC	48	NC	NC
49	GND	NC	50	NC	NC

UCN3

Pin No.	Signal Name	CPU	Pin No.	Signal Name	CPU
1	GND	NC	2	PC7	16
3	PC6	15	4	GND	NC
5	PC5	13	6	PC4	11
7	GND	NC	8	PC3	10
9	PC2	9	10	GND	NC
11	PC1	8	12	PC0	7
13	GND	NC	14	PE15	5
15	PA21	4	16	GND	NC
17	PA22	3	18	PE14	2
19	GND	NC	20	PA23	1
21	PE13	144	22	GND	NC
23	PE12	143	24	PE11	142
25	GND	NC	26	PE10	140
27	PE9	139	28	GND	NC
29	PE8	138	30	PE7	137
31	GND	NC	32	PA5	136
33	PA4	134	34	GND	NC
35	PA3	133	36	PA2	132
37	GND	NC	38	PA1	131
39	PA0	130	40	GND	NC
41	AVcc	128	42	AVref	127
43	GND	NC	44	VCC	NC
45	VCC	NC	46	VCC	NC
47	GND	NC	48	NC	NC
49	GND	NC	50	NC	NC

UCN4

Pin No.	Signal Name	CPU	Pin No.	Signal Name	CPU
1	GND	NC	2	GND	NC
3	PA6	54	4	PA7	53
5	GND	NC	6	PA8	52
7	PA9	51	8	GND	NC
9	PA10	50	10	PA11	49
11	GND	NC	12	PA12	48
13	PA13	47	14	GND	NC
15	PD30	46	16	PD31	45
17	GND	NC	18	WDTOVF	44
19	PA14	NC	20	GND	NC
21	PB9	41	22	PB8	39
23	PB7	38	24	GND	NC
25	PB6	37	26	PB5	36
27	GND	NC	28	PB4	34
29	PA18	33	30	GND	NC
31	PB3	32	32	PB2	31
33	GND	NC	34	PA19	30
35	PA20	29	36	GND	NC
37	PB1	27	38	PB0	25
39	GND	NC	40	PC15	24
41	PC14	23	42	GND	NC
43	PC13	22	44	PC12	21
45	GND	NC	46	PC11	20
47	PC10	19	48	GND	NC
49	PC9	18	50	PC8	17

Section 4 Board Options

The LCEVB-SH2 provides a number of user-settable optional configurations. All of these are chosen by jumper settings.

4.1 Jumpers

LCEVB-SH2 jumpers allow users to configure the board as required for testing and evaluation. For simplicity, all jumpers are three-pin header or two-pin header. In every case, the default jumper setting is pin 1 to pin 2 (figure 4.1). For most LCEVB-SH2 uses, these settings need not be changed.

Table 4.1 summarizes jumper settings.

Table 4.1 Jumper Settings and Options

Jn	Use	Default (1-2)	Alternate (2-3)
J1		Resonating XTAL	TTL XTAL
J2			No connection
J3	A_{VCC}	= digital V_{CC}	Set externally
J4	A_{VREF}	= digital V_{CC}	Set externally
J5	A_{VSS}	= digital V_{SS}	Set external
J6	NMI	internal	external
J7	MD0	(Mode 0)	As set (see table 4.2)
J8	MD1		
J9	MD2		
J10	MD3		
J11	Power	DC Adapter	Power Supply
J13	TxD1	PA0 connected	PA0 not connected
J14	RxD1	PA1 connected	PA1 not connected
J15	TxD0	PA3 connected	PA3 not connected
J16	RxD0	PA4 connected	PA4 not connected

Do not connect the jumper wire if PB11, PB10, PB9 and PB8 are to be left open.

The following sections describe each jumper and its alternative settings.

4.1.1 NMI (Jumper J6)

Default (1-2) Setting: The SH7043 NMI input is controlled by the set-reset flip-flop de-bounce circuit implemented with NAND gates U3A and U3B.

Alternate (2-3) Setting: The SH7043 NMI input is controlled by an external signal. An on-board pull-up is provided.

Open Setting: Not recommended. The SH7043 NMI signal should be driven in all conditions. Failure to do so may cause the board to operate erratically.

4.1.2 Setting SH7043 Operating Mode (Jumpers J7, J8, J9, J10)

As described in section 3 of the *SH7043 RISC Hardware Manual,* the operating mode of the SH7043 microcomputer is set at device initialization time by the settings of the three mode inputs, MD0, MD1, and MD2. These settings should not be changed while the SH7043 is running. Table 4.2 lists jumper settings for these modes. Leaving any of these jumpers open is not recommended. Settings not shown in table 4.2 are currently undefined. J9 and J10 are primarily used for setting the phased locked loop frequency multiplier for the input clock. Note that table 4.2 is for the 144 pin version of the SH7043, the 112 pin package has different values for the CS0 area bus width values.

Table 4.2 Mode Settings for the SH7043 144 pin package

Mode	J7(1)	J8(1)	J9	J10	Mode name	On Chip ROM	CS0 Area	Implementation
0	1-2	1-2	x	x	MCU mode 0	Not active	16 bit space	Default
1	2-3	1-2	x	x	MCU mode 1	Not active	32 bit space	Not supported
2	1-2	2-3	x	x	MCU mode 2	Active	8/16/32 bits(2)	Not supported
3	2-3	2-3	x	x	Single chip mode	Active	NA	NA
4	2-3	2-3	2-3	2-3	Prom Mode	ACtive	NA	NA

Notes: 1) MD2 and MD3 select the clock mode in modes 0-3 (table 4.3), 2) Set by BCR2 of BSC register

Table 4.3 Clock Mode Settings

Mode	J7	J8	Clock Mode
0	1-2	1-2	PLL ON x 1
1	1-2	1-2	PLL ON x 2
2	1-2	2-3	PLL ON x 4
7	2-3	2-3	Not supported

4.1.3 Analog Reference and Supply (Jumpers J3, J4, and J5)

As described in section 15 of the *SH7043 RISC Hardware Manual,* the eight port F bits of the SH7043 microcomputer may be configured as analog inputs. In this case, reference voltages for analog signals become important. The default settings of these three jumpers route on-board digital references and the digital V_{CC} to the SH7043 analog subsystem. For demonstration purposes, this configuration may be sufficient. However, to demonstrate the full capabilities of the the SH7043 analog subsystem, as well as to reduce noise in the analog subsystem, it may be desirable to use external sources for some or all of these signals.

If an external analog V_{CC} (AV_{CC}) is provided to the SH7043 on User Connector 3, Pin 41, set J3 (2-3).

If an external analog reference voltage (V_{REF}) is provided to the SH7043 on User Connector 3, Pin 42, set J4 (2-3).

If an external analog ground (AV_{SS}) is provided to the SH7043 on User Connector 2, Pin 34, set J5 (2-3).

Leaving any of these jumpers open is not recommended.

4.1.4 Serial Port Disconnects (Jumpers J13, J14, J15, and J16)

UART1 is dedicated by default to the monitor CMON. UART0 is unassigned. The port pins (TxD0, RxD0 and TxD1, and RxD1) associated with transmitting and receiving data for both UARTs are connected to a serial transceiver device.

In some applications it may be necessary to use some or all of these pins for another purpose, in which case the connections of these port pins to the transceiver device should normally be disconnected.

These jumpers may be left open because the logic inputs of the MAX232 transceivers are internally pulled up weakly to V_{CC}. Alternate devices may not include these pull-ups.

To free PA0, remove Jumper at J16 (1-2). To free PA1, remove Jumper at J15 (1-2).

To free PA3, remove Jumper at J14 (1-2). To free PA4, remove Jumper at J13 (1-2).

This will disable serial communications between the LCEVB-SH2 and its host.

4.1.5 Serial Port Hardwiring Options

As supplied, the LCEVB-SH2 supports three-wire serial communication. No direct provision is made for additional handshaking signals that may be required by host computers or terminals in some configurations.

4.1.6 Crystal Clock source

The LCEVB-SH2 comes with two types of clock sources, an AT-cut parallel resonant crystal and a TTL Can crystal oscillator. Either Clock source may be used but not both. To make a selection, simply make the necessary connection on jumper J1 & J2. The default crystal clock source is the AT-cut parallel resonant crystal with J1 (1-2) and J2 (1-2) connected. To use the TTL can crystal simply change connection to J1 (2-3) and remove Jumper at J2 (1-2).

The LCVEB-SH2 evaluation board is delivered with a 14.31816 MHz. crystal and with no TTL can crystal oscillator. J9 and J10 have been set for 2x PLL operation resulting in an internal CPU clock of 28.63636 MHz. When choosing a different frequency care must be taken in setting J9 and J10 correctly as well as the baud rate divisor settings in the serial bit rate (SBR) register.

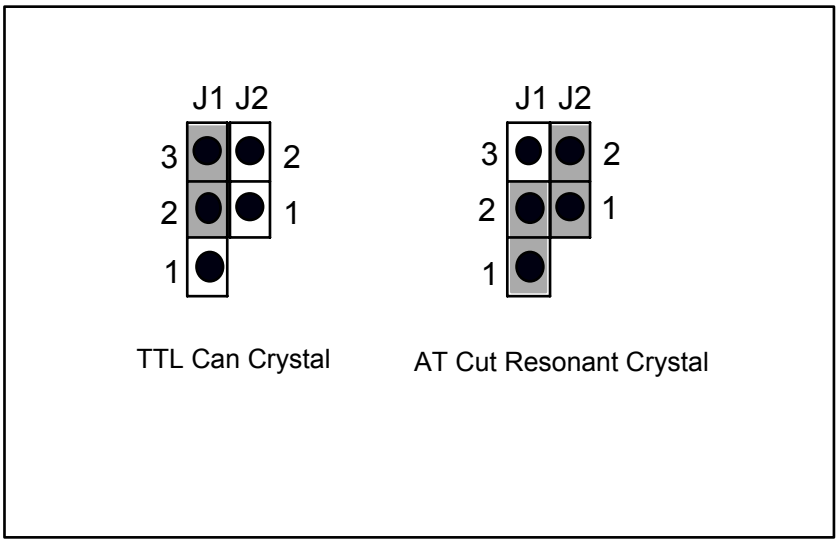


Figure 4.1 Crystal Source Selection

Section 5 Hitachi Debugging Interface (HDI)

On the LCEVB-SH2, you can only do your program debugging with the HDI. This section answers the following questions:

- What is HDI?
- How to install HDI?
- How do I download, run and debug my program with HDI on LCEVB-SH2?

5.1 Introduction to HDI

The Hitachi Debugging Interface (HDI) is a Graphical User Interface intended to ease the development and debugging of applications written in C/C++ and assembly language for Hitachi microcomputers. Its aim is to provide a powerful yet intuitive way of accessing, observing and modifying the debugging platform in which the application is running.

5.2 Installation

There are two installation disks. First install the HDI software from the installation disk proceed as follows:

- Insert the HDI installation disk #1.
- Run Windows 95/98/NT/2000 if it is not already running.
- Close all other applications that are running.
- Choose **Run** from the **Program Manager File** menu.
- Type `a:\Lsh2_xxx.exe (eg. Lsh2_101.exe)` and click **[OK]** button.

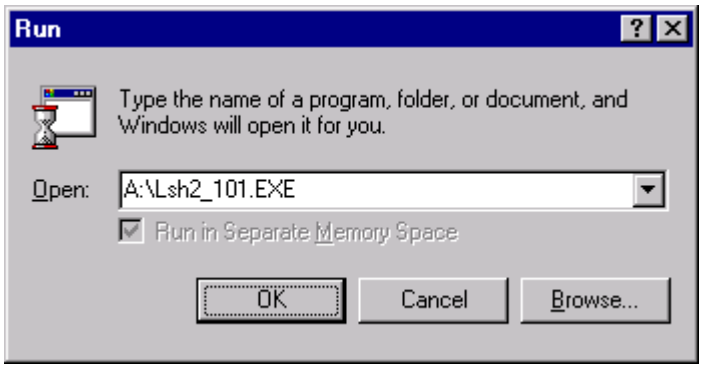


Figure 5.1 Run Dialogue box

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

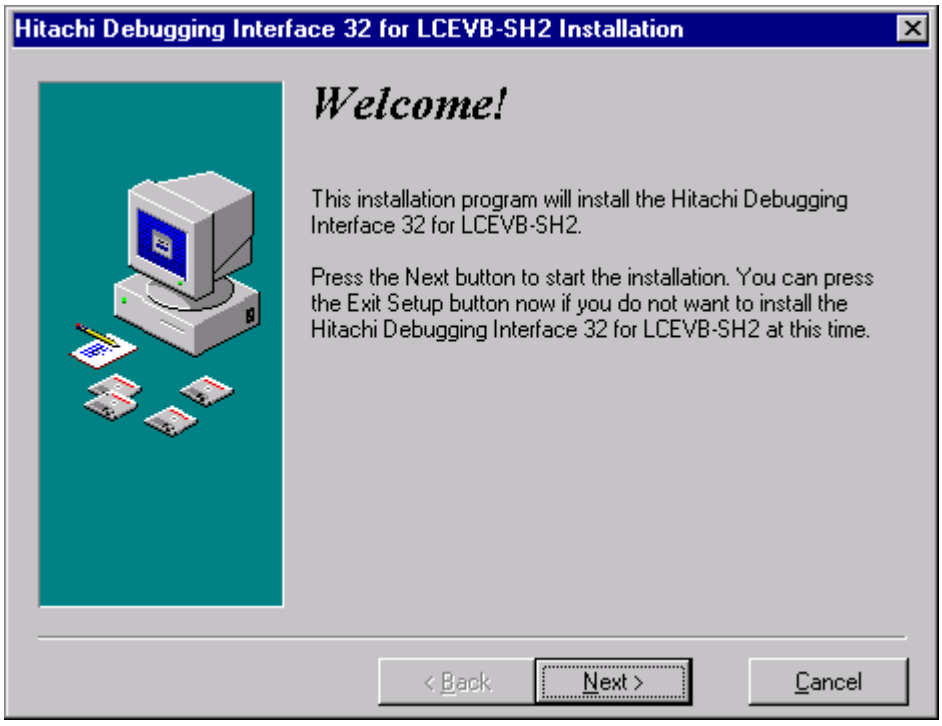


Figure 5.2 HDI Installer Welcome! Screen

- Click **[OK]** button to proceed with the installation.
- Check the `ReadMe` file for any important information regarding the installation and then click **[Next>]** button to proceed.

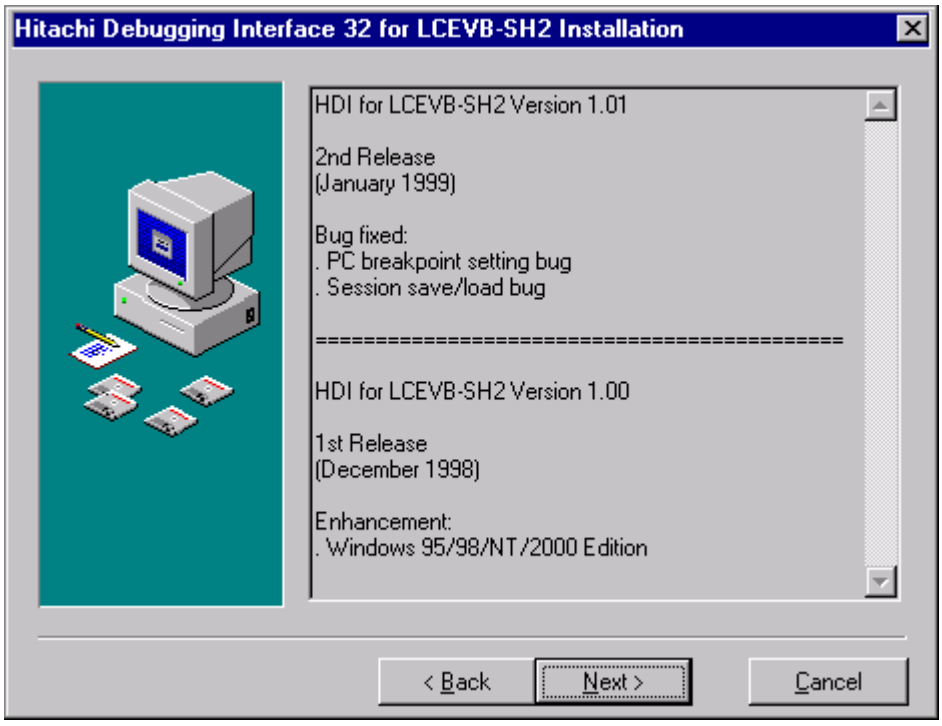


Figure 5.3 Update Information Screen

The following dialogue box then allows you to select a directory in which to install HDI.

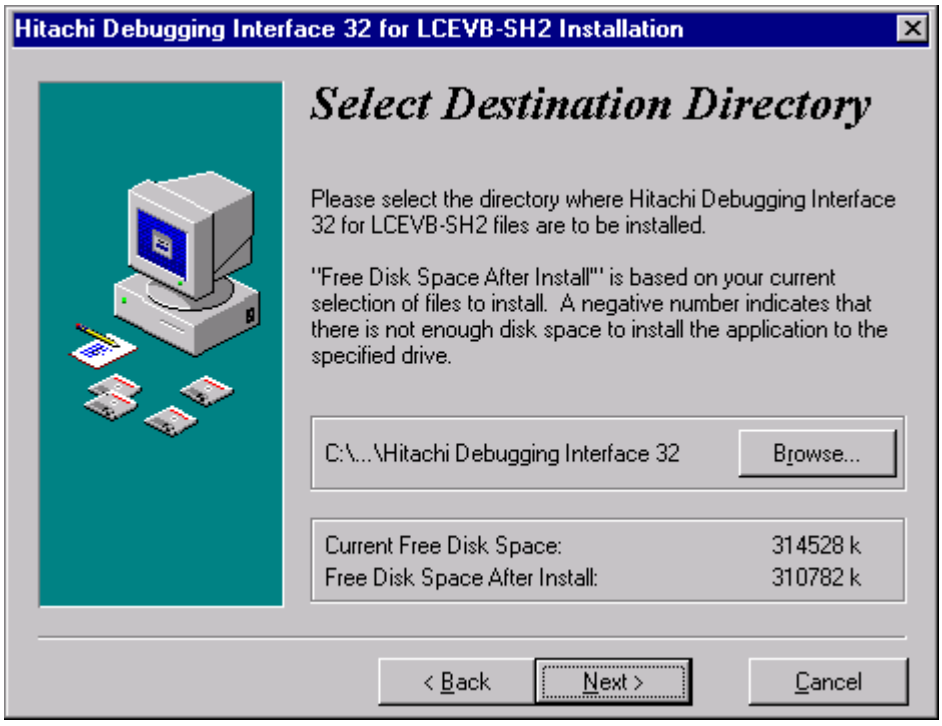


Figure 5.4 Select Destination Directory Screen

- Click Next to install into the default directory `C:\Program Files\Hitachi Debugging Interface 32`, or specify an alternative directory and click **[Next>]** button.

The following dialogue box allows you to specify which Program Group into insert icons for HDI. The default Program Group is `Hitachi Debugging Interface 32`.

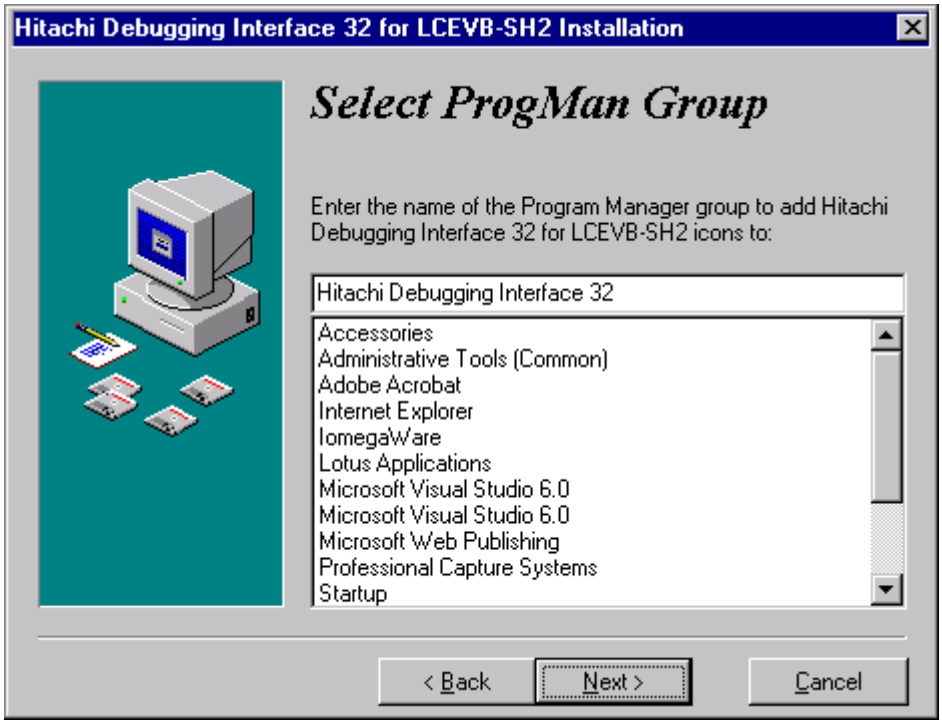


Figure 5.5 Select Program Group Screen

- Enter the directory you want to use and click **[Next>]** button.

The installer then copies the HDI files to the specified directory after prompting a confirmation dialog. Insert installation disk #2 and click **[OK]** button when the following dialogue is prompted.

Figure 5.6 Insert New Disk Dialogue

Finally icons for HDI will be created into the Program Group specified earlier. The system restart request dialog will be prompted in the end of the installation, press **[OK]** button to restart your windows.

5.2.1 Installation Details

The installer creates the following icons in the program group you specified, by default HDI

Figure 5.7 HDI Icons

These icons have the following functions:

“Hitachi Debugging Interface” the main HDI program.

“Uninstall Hitachi Debugging Interface” will remove HDI, and its associated files, if you need to uninstall it at any stage.

5.3 System Overview

HDI is a modular software system, utilizing self-contained modules for specific tasks. These modules are linked to a general purpose Graphical User Interface, which provides a *common look & feel* independent of the particular modules with which the system is configured.

1. User Interface

The HDI Graphical User Interface is a Windows application that presents the debugging platform to you and allows you to set up and modify the system.

2. Toolbar

The toolbar provides convenient buttons as shortcuts for the most frequently used menu commands.

3. Status Bar

The status bar displays the status of the LCEVB-SH2 evaluation board. For example cause of last break and so on.

4. Help

HDI has a standard Windows context sensitive help system. This provides on-line information about using the debugging system.

Help can be invoked by pressing the **F1** key of *via* the Help menu. Additionally, some windows and dialog boxes have a dedicated help button to launch the help file at the appropriate location.

To get help on a specific item in the HDI, a help cursor can be used. To enable the help cursor, press **SHIFT+F1**.

Your cursor the changes to include a question mark. You can then click on the item for which you require help and the help system will be opened at the appropriate location.

5.4 Preparing to Debug

This subsection describes all the functions that are available in HDI for LCEVB-SH2.

5.4.1 Compiling for Debugging

In order to be able to debug your program at C/C++ source level, the compiler must provide information about your C/C++ program to the debugging platform *via* the object file. When this option is enabled, the compiler puts all the information necessary for debugging your C/C++ code into the object file, which is then usually called a *debug object file*.

Make sure you have the debug option enabled on your compiler and linker, when you generate an object file for debugging.

If your debug object file does not contain any debugging information, you can still load it into the debugging platform but will only be able to debug at assembly-language level.

5.5 Selecting a Debugging Platform

If you have only installed HDI for LCEVB-SH2, then it will automatically link to he LCEVB-SH2 when launched. Otherwise, you should select LCEVB-SH2.

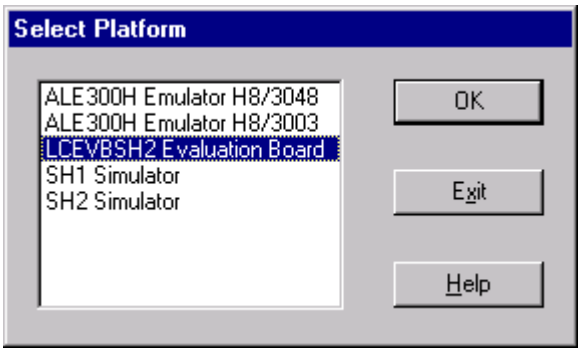


Figure 5.8 Select Platform Dialogue Box

HDI will load the target module and establish communications with the LCEVB-SH2. As the module loads, it will reset the LCEVB-SH2. When the LCEVB-SH2 has been successfully initialized HDI will report “Link Up” on the status bar.

5.6 Configuring the Debugging Platform

All the configurations of the LCEVB-SH2 are done by the firmware, jumper setting on the board or the HDI software. Therefore it is not necessary for the user to configure the LCEVB-SH2 through HDI.

5.6.1 Setup

To view the LCEVB-SH2 configuration, invoke the **[Setup->Configure Platform…]** menu option. The following dialog box will be displayed.

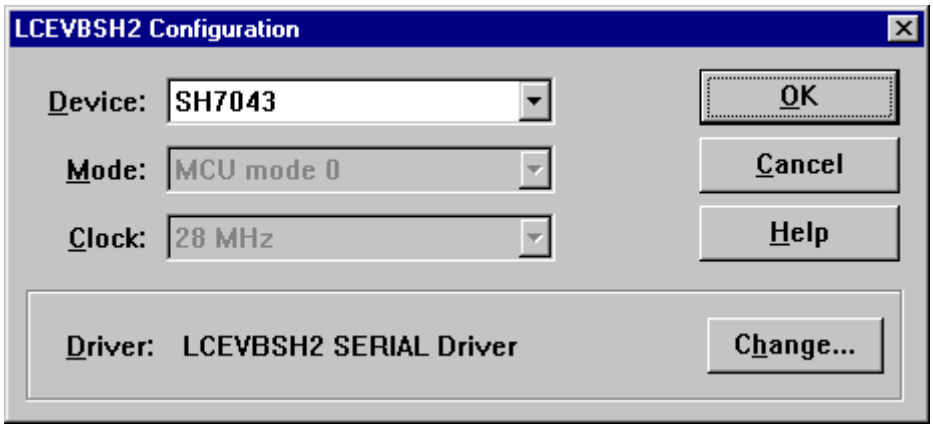


Figure 5.9 Target Configuration Dialogue Box

This configuration dialogue box shows:

- Devices supported by the HDI. User can select device from SH7040 series microcomputer.
- CPU operating mode of the CPU. This is not settable via HDI.
- Clock mode of the CPU. This is not settable via HDI.

To show the configuration of the driver, click the **[Change…]** button.

User cannot configure the serial driver because the parameters are decided by the firmware. HDI can auto detect serial port which is connected to the LCEVB-SH2.

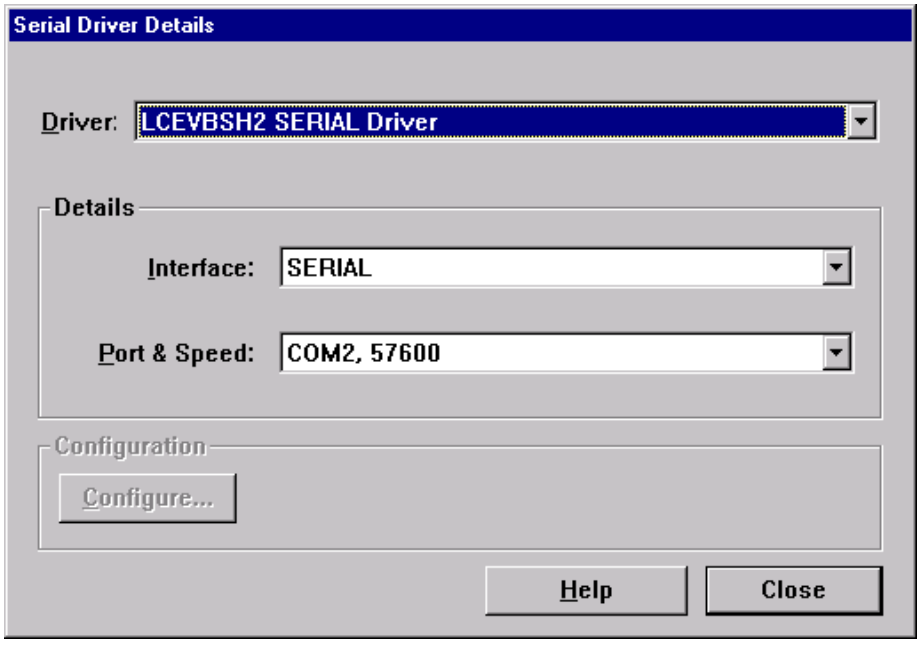


Figure 5.10 Driver Dialogue Box

5.6.2 Memory Mapping

To display the memory mapping of LCEVB-SH2, either:

- Click the **[View->Memory Mapping Window]** menu option.
- Click the *open memory mapping* icon in the toolbar.

Note that this Memory Mapping Window is only displayed for reference. You are forbidden to modify the memory map of LCEVB-SH2.

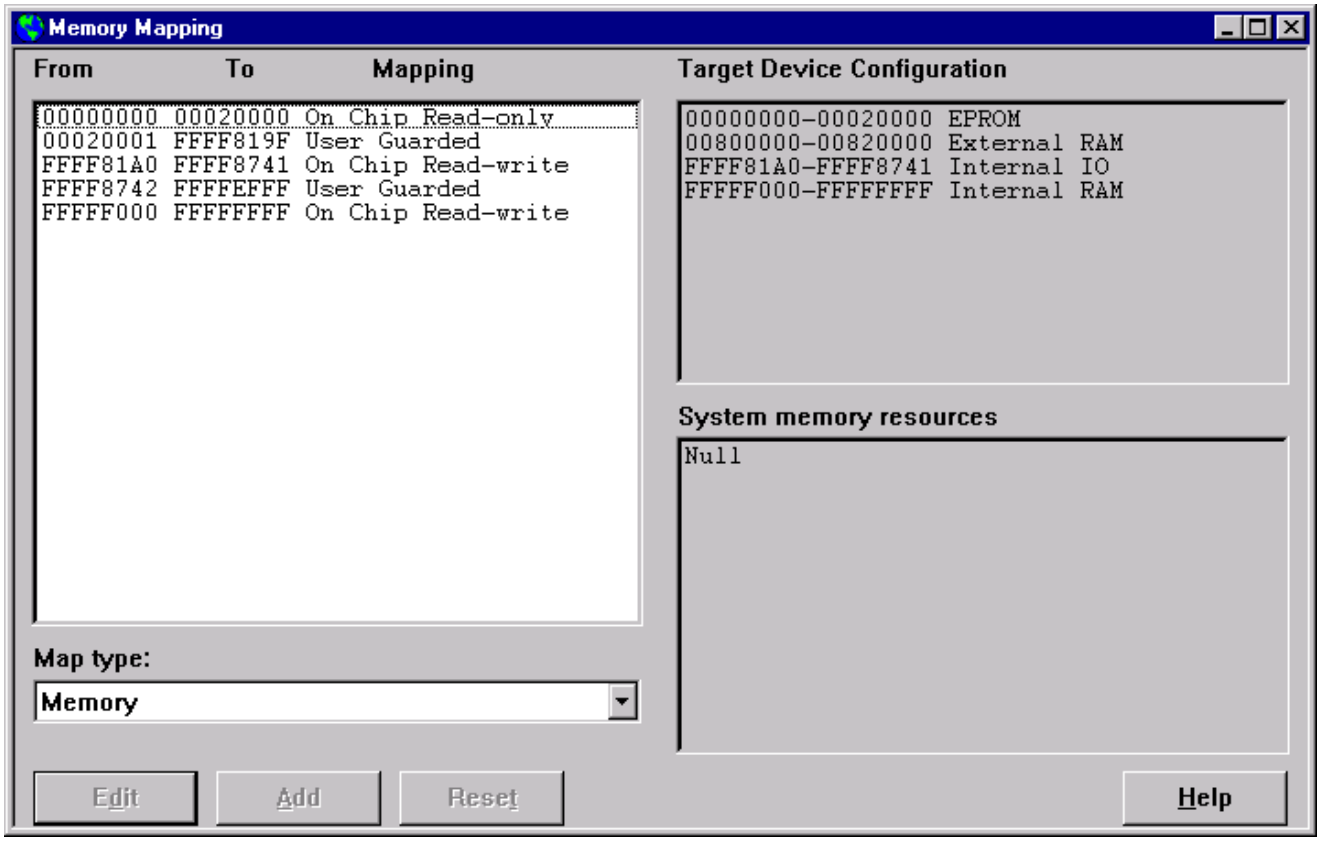


Figure 5.11 Memory Mapping Window

5.6.3 Status

You can check he configuration and status of the LCEVB-SH2 by looking in the System Status Window. To open the window, either:

- Click the **[View->Status Window]** menu option.
- Click the *open status window icon* in the toolbar.

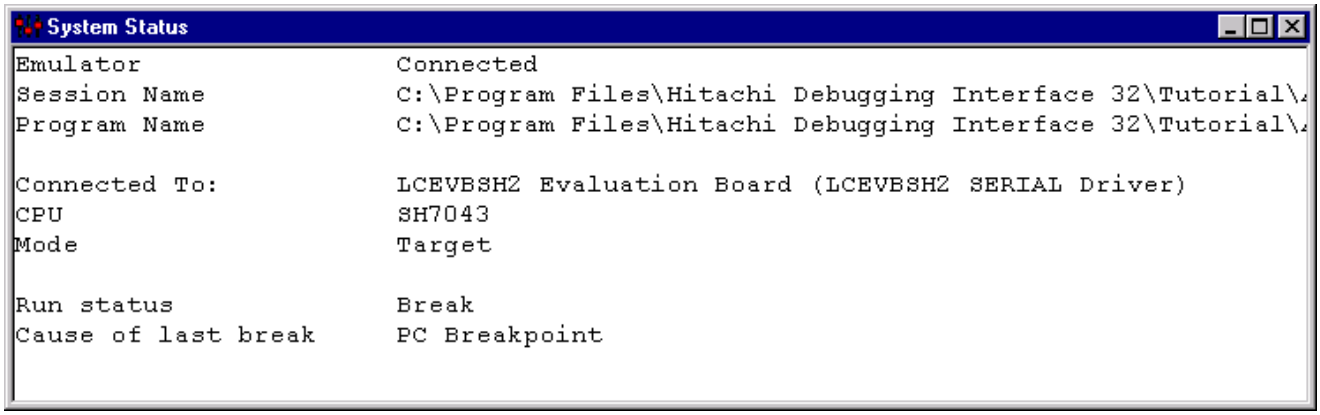


Figure 5.12 System Status Window

5.7 Downloading User Program

Once the LCEVB-SH2 is set up you can download the object program you want to debug. To invoke the Load Object File dialogue box, either:

- Click on the **[File->Load Program…]** menu option.
- Click *load code and symbols* icon in the toolbar.

5.7.1 Selecting a File Type

You have to download Sysof type file (*.abs) if you want to do C/C++ source level debugging. Otherwise just select Motorola S-type Record file (*.mot).

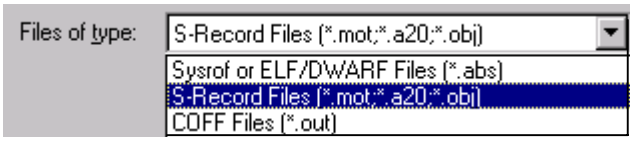


Figure 5.13 File Type Selection

When the file has been loaded, the Reset Start Address dialogue box will prompt. You must define the default reset start address for you program. The address will be stored in the Reset Vector location.

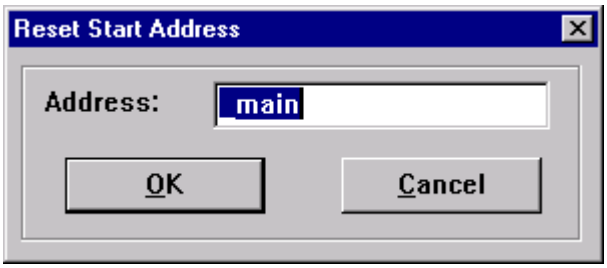


Figure 5.14 Reset Start Address Dialogue Box

After setting default reset start address, the dialogue box shown in the following figure displays information about the memory areas that have been filled with the program code.

Figure 5.15 Program Downloading Information Dialogue Box

5.8 Reset LCEVB-SH2

To reset the LCEVB-SH2, select **[Run->Reset CPU]** menu option.

The internal state of the firmware will be returned to default (power-up) values. Any existing breakpoint(s) will be cleared. A Code Window will be displayed and PC register will be set to address stored in the Reset Vector location. Common register R15 will be set to *820000h*, which is the end address of the external SRAM.

5.9 Display the Program Listing

HDI allows you to display and debug a program as source level or assembly language mnemonics. In source level display, you can see a listing of the program alongside the disassembled code as you debug. To do this you need to read in a copy of the source program from which the object file was compiled.

To display downloaded program, open the Program Window by either:

- Selecting the **[View->Program Window…]** menu option.
- Clicking on the *program window* icon in the toolbar.
- Press **Ctrl+K**.
- Selecting the **[Run->Reset CPU]** menu option.

Select your source file and HDI will open a Program Window.

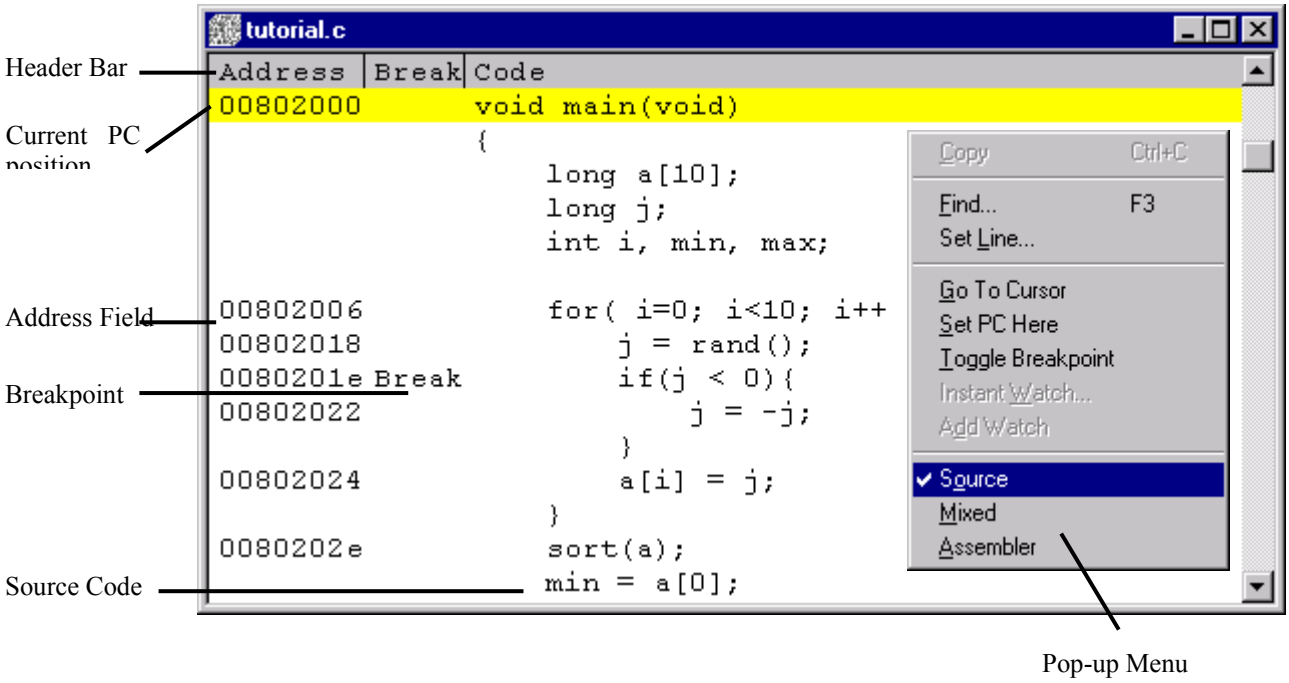


Figure 5.16 Program Windows in Source Code Display

5.9.1 Viewing Assembly Language Code

If you wish to view code at assembly language level or do not have a C/C++ source file, select the Address radio button in Figure 5.15. HDI will display the Program Window as below.

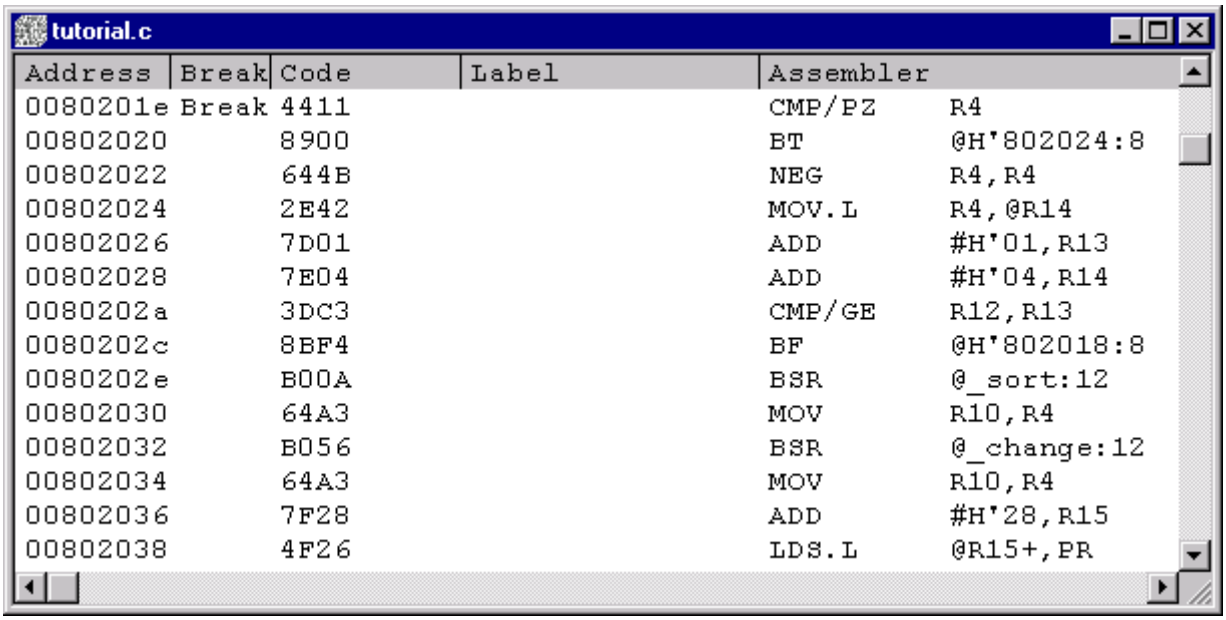


Figure 5.17 Program Window in Assembly Language Display

5.9.2 Modifying Assembly Language Code

You can modify the assembly language code by double clicking on the instruction that you wish to change. The Assembler dialogue box will appear.

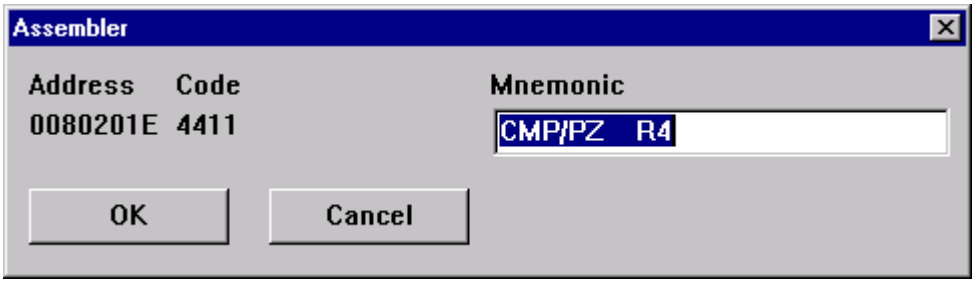


Figure 5.18 Assembler Dialogue Box

Note that the assembly language display is disassembled from the actual machine code in the LCEVB-SH2 memory. If the memory contents are changed, the display will show the corresponding new assembly language code, but will not match the text shown in the source display.

5.10 Symbols

Symbols or labels are text names that represent an address in the program. You will only see symbols in the Label field in the Program Window displayed in assembly language format.

5.10.1 Listing Symbols

To see a list of all the symbols defined in the current session, open the Symbols Window by either:

- Selecting the **[Tools->Symbols…]** menu option.
- Press **Ctrl+L**.

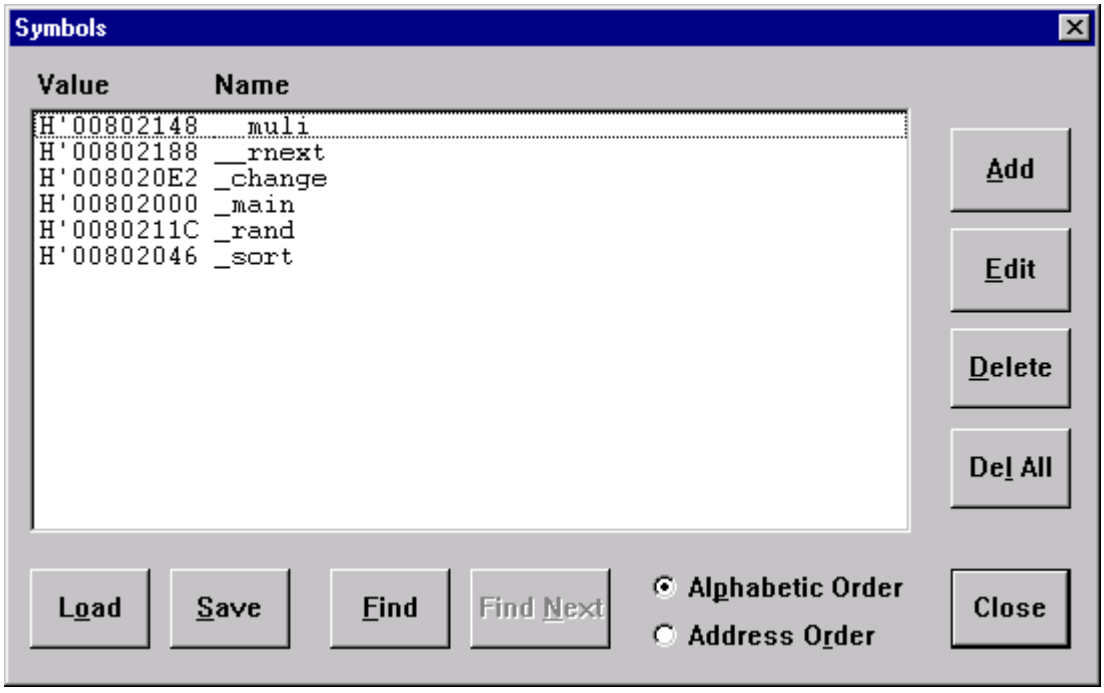


Figure 5.19 Symbols Window

5.11 Working with Memory

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

5.11.1 Displaying Memory

To display an area of memory, open a Memory Window by either:

- Selecting the **[View->Memory Window]** menu option.
- Clicking the *open memory window* icon in the toolbar.
- Press **Ctrl+M**.

You will be presented with an Open Memory Window dialogue box.

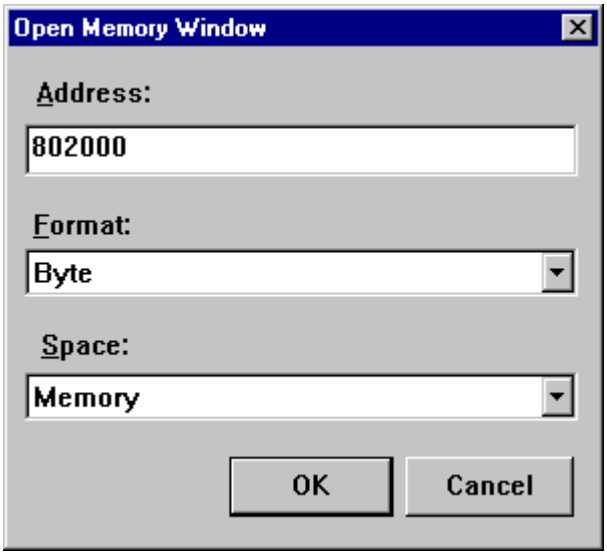


Figure 5.20 Open Memory Window Dialog Box

Type in the start address or equivalent symbol for the window display in the Address field and select the required display format from the Format list.

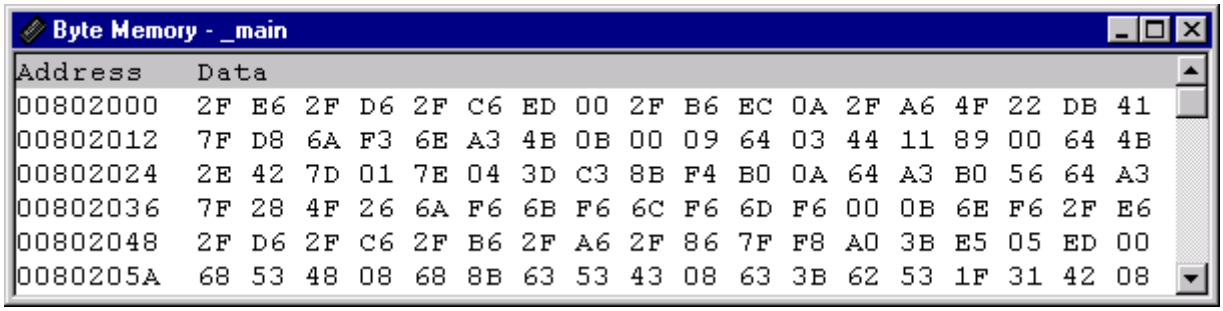


Figure 5.21 Memory Window in Byte Format

5.11.2 Modifying Memory Contents

To modify the contents of memory *on* the Memory Window, either:

- Enter value by typing directly
- Press ↵ to invoke the Edit dialogue box.

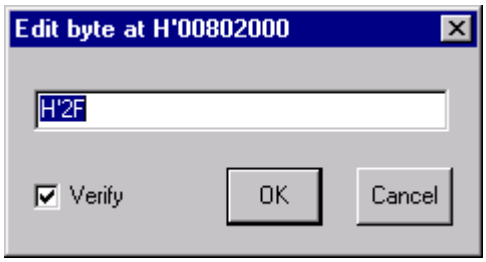


Figure 5.22 Edit Dialogue Box

5.11.3 Filling Memory

To fill an area of memory with a value, either:

- Invoke the **[Edit->Filling Memory…]** menu option.
- Click the *fill memory area* icon in the toolbar.

You will be presented with a Fill Memory dialogue box.

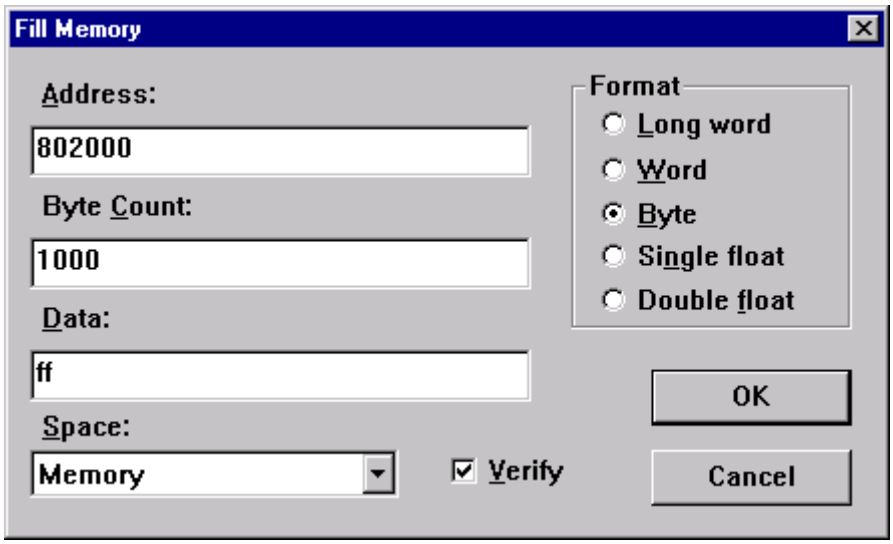


Figure 5.23 Fill Memory Dialogue Box

Fill in all the blank fields with necessary values and click the **[OK]** button.

There is another way to fill memory if the memory address range is in the Memory Window. You can select the range by dragging the mouse, invoke the pop-up menu by clicking the right mouse button and click on **[Fill]**.

5.11.4 Moving an Area of Memory

To move memory, either:

- Invoke the **[Edit->Move Memory…]** menu option.
- Click the *move a block of memory* icon in toolbar.
- Invoke the pop-up menu and click on **[Move]** after selecting a memory range in the Memory Window.

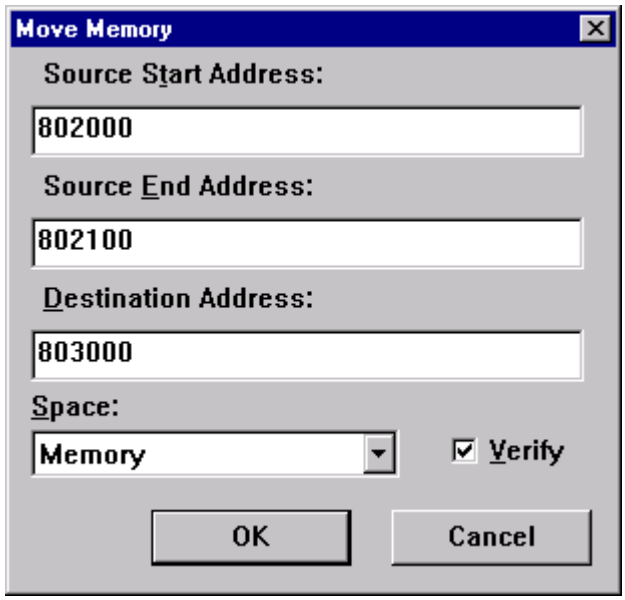


Figure 5.24 Move Memory Dialogue Box

Fill in all the blank fields with necessary values and click the **[OK]** button.

5.11.5 Testing Memory

To test an area of memory, either:

- Invoke the **[Edit->Test Memory…]** menu option.
- Click on the *test memory area* icon in the toolbar.
- Invoke the pop-up menu and click on **[Test]** after selecting a memory range in the Memory Window.

You will be presented with a Test Memory dialogue box after a warning message box.

Fill all the blank fields with necessary values and click the **[OK]** button

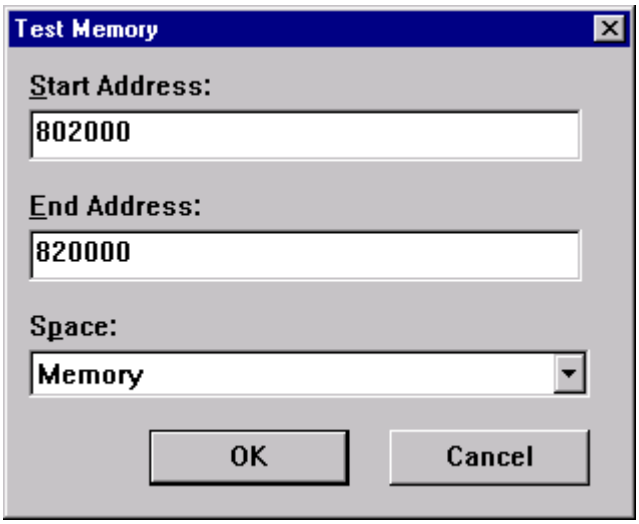


Figure 5.25 Test Memory Dialogue Box

An information message box will appear to report the test result.

Figure 5.26 Memory Test Result Confirmation Dialogue Box

5.11.6 Saving Memory

To save an area of memory, either:

- Invoke the **[File->Save Memory…]** menu option.
- Click on the *save memory area to file* icon in the toolbar.
- Invoke the pop-up menu and click on **[Save]** after selecting a memory range in the Memory Window.

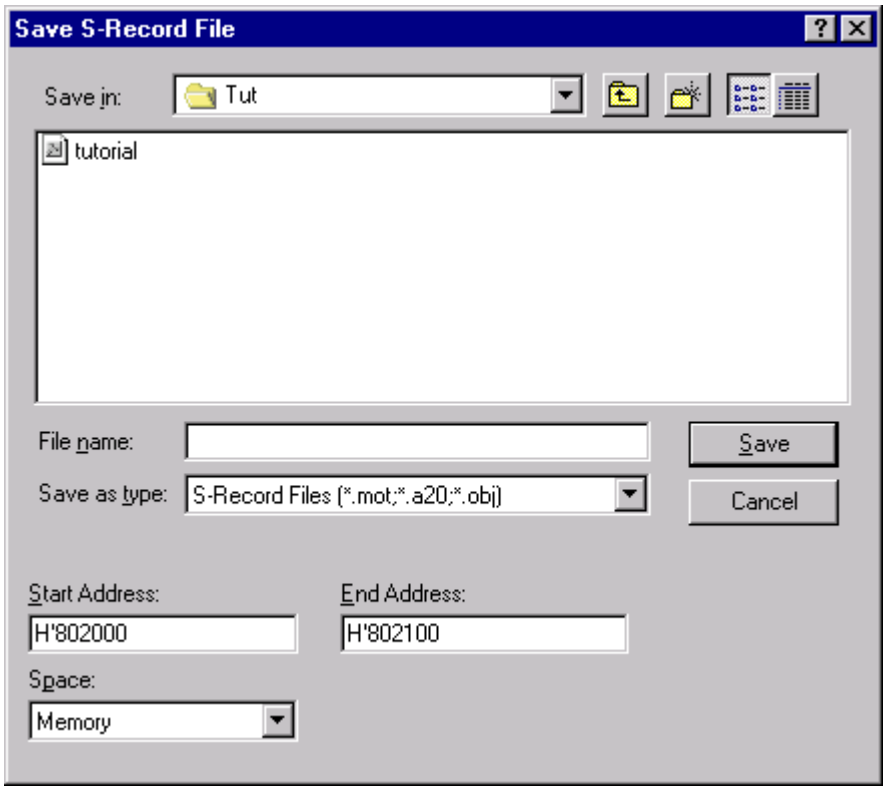


Figure 5.27 Save S-Record File Dialogue Box

5.11.7 Loading Memory

Please refer to Section 5.7.

5.11.8 Verifying Memory

To verify an area of memory in the address space against a S-Record disk file, either:

- Select **[File->Verify Memory…]** menu option.
- Click the *verify memory area with file* icon in the toolbar.

An information message box will appear as a result.

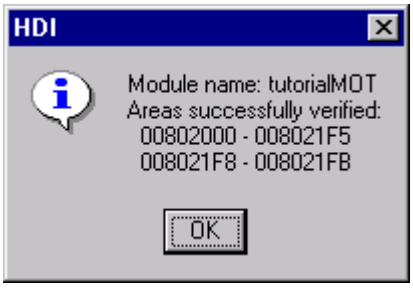


Figure 5.28 Verify S-Record File Information Dialogue Box

5.12 Working with Variables

As you step through a program it is useful to be able to watch the values of variables used in your program, to verigy that they change in the way that you expected.

5.12.1 Instant Watch

To open an Instant Watch Window,

1. Click to position the cursor on the variable which you want to watch in the Program Window.
2. Click in the program window with the right mouse button to display a pop-up menu and choose **[Instant Watch…]**.

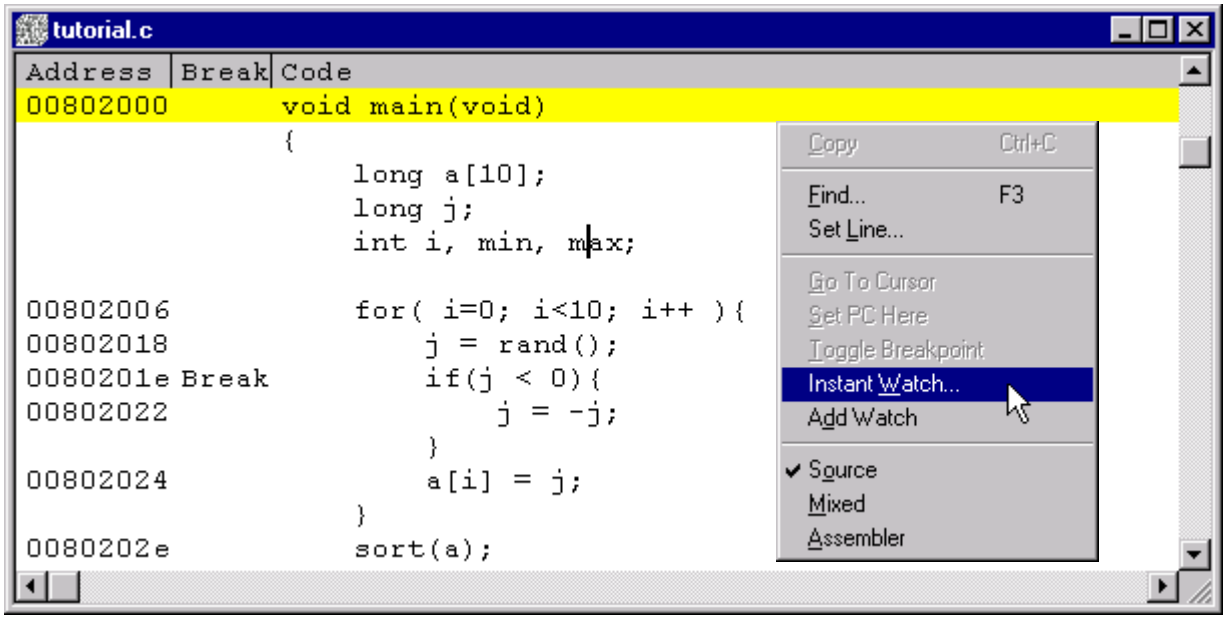


Figure 5.29 Procedure to open the Instant Watch Window

The Instant Watch Window will be presented to you.

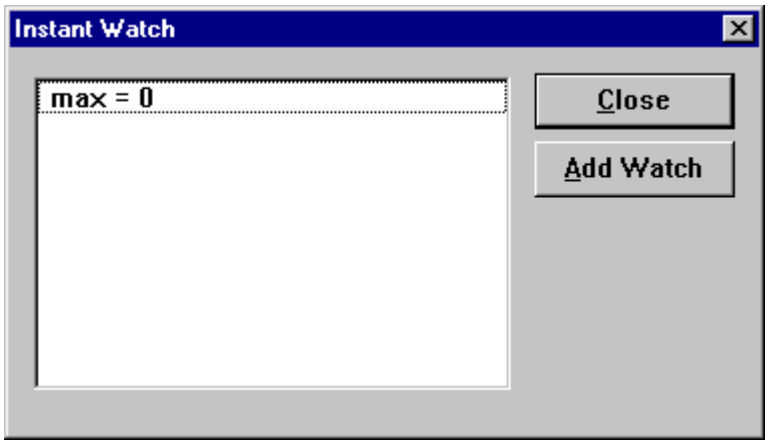


Figure 5.30 Instant Watch Window

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

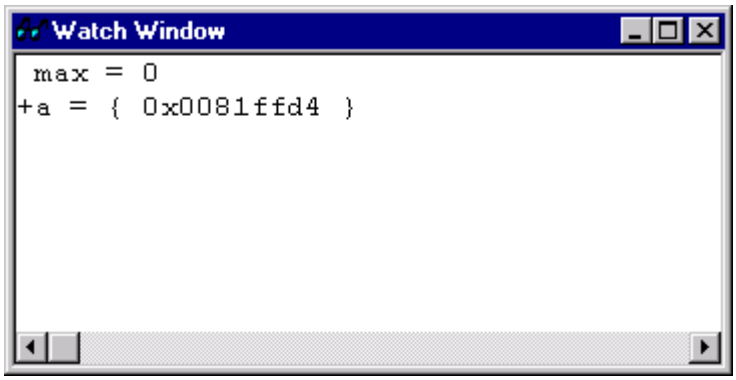


Figure 5.31 Watch Window

In the Watch Window, you can double-click the + symbol to the left of any symbol to expand it and display the individual elements in the array. You can also add/delete a watch to it b right click on the mouse button. If you choose **[Add Watch…]** from the pop-up menu, an Add Watch dialogue box will be presented to you.

5.13 Executing User Program

LCEVB-SH2 supports several Run and Step functions

5.13.1 Running from Reset

To reset the LCEVB-SH2 and run the user program from the Reset Vector Address, either:

- Select the **[Run->Go Reset]** menu option.
- Click on the *run program from reset* icon in the toolbar.

Note that the program will start running from whatever address is stored in the Reset Vector location. Therefore it is important to make sure that this location contains the address of your startup code. The initial value of the address is zero.

5.13.2 Continuing Run

To execute your program from the current Program Counter (PC), either:

- Select the **[Run->Go]** menu option.
- Press the **F5** key.
- Click on the *run from current PC* icon in the toolbar.

5.13.3 Running to the Cursor

To use this function,

1. Position the cursor on the address at which you want to stop in the Program Window.
2. Invoke the pop-up menu by clicking the right mouse button (or pressing **SHIFT+F10**) and selecting the **[Go To Cursor]** menu option

The program will be executed from the current PC register value.

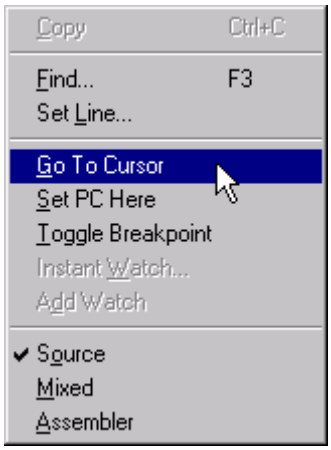


Figure 5.32 Selecting Go To Cursor Pop-up Window

The following dialog box is displayed when user program is under execution. You have to press RESET of NMI button on the LCEVB-SH2 to halt the program.

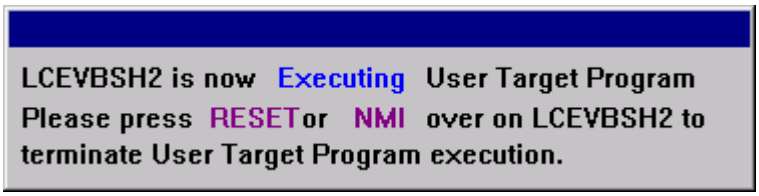


Figure 5.33 Executing User Program Dialogue Box

5.13.4 Single Step

LCEVB-SH2 only supports Step Into function. To step into a routine, either:

- Select the **[Run->Step In]** menu option.
- Click the *step into functions* icon in the toolbar.
- Press **F8** key.

A highlighted yellow line corresponding to current PC value is displayed on the Program Window.

Step Out and Step Over functions are not presently supported.

5.13.5 Multiple Steps

To step several instructions at a time, select the **[Run->Step…]** menu option. You will be presented with a Step Program dialogue box. Specify the number of steps and step rate.

The Step Over Calls and Source Level Step functions are not supported by LCEVB-SH2.

Figure 5.34 Step Program Dialogue Box

5.14 Stopping User Program

To halt user program from executing, either:

- Select the **[Run->Halt Program]** menu option.
- Click the *stop running program* icon in the toolbar if it is enabled (a red stop sign).
- Press the **ESC** key.
- Set a breakpoint at a specified address. This is described in the next section.

However, if the dialogue box in Figure 5.33 is being displayed, you have to press RESET or NMI button on the LCEVB-SH2 to halt your program.

5.15 Setting Breakpoints

The simplest debugging aid is the program breakpoint, which lets you halt execution 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.

LCEVB-SH2 supports 20 PC Breakpoints and 1 User Breakpoint respectively. To set a breakpoint, either:

- Select the **[View->Breakpoint Window]** menu option.
- Click the *open breakpoint window* icon in the toolbar.
- Press **Ctrl+K**.

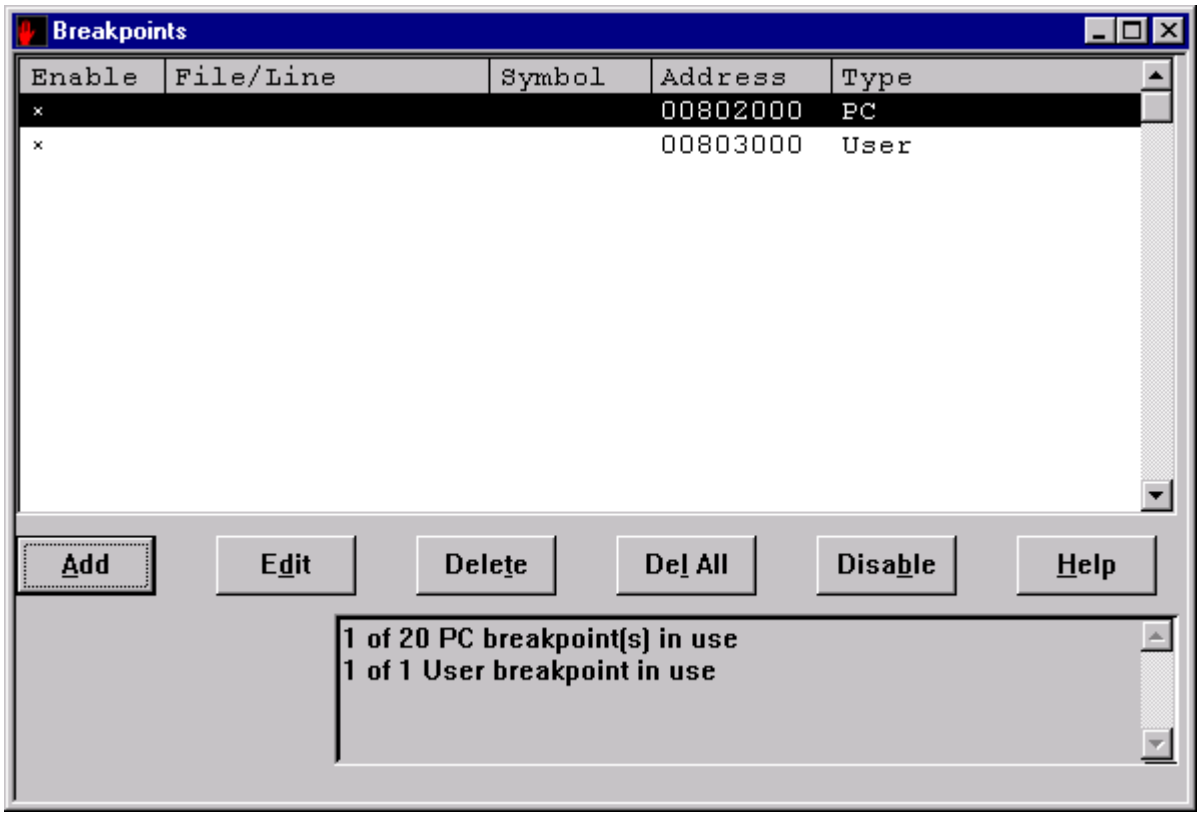
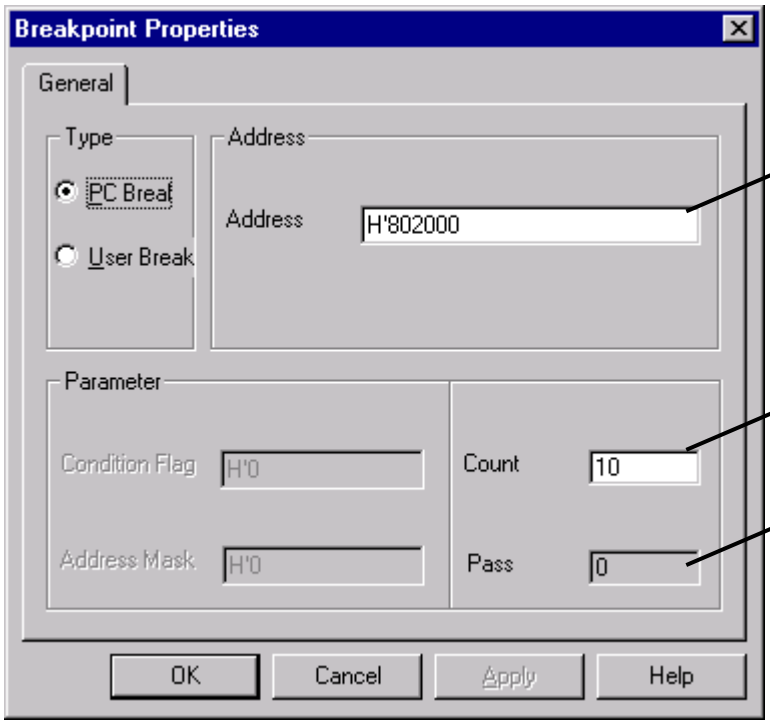


Figure 5.35 Breakpoints Window

Selecting a PC Breakpoint allows the Address and Count and Pass parameters to be set



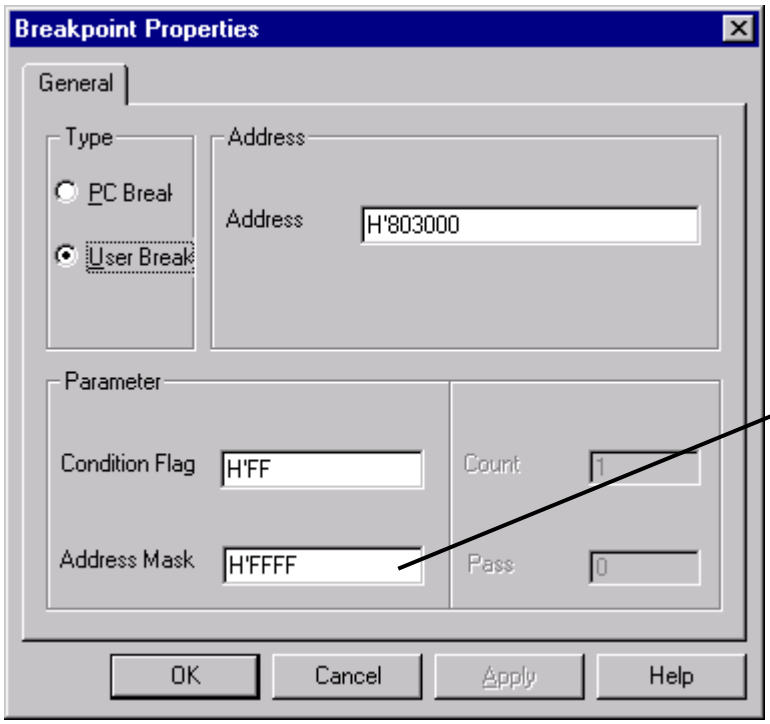
Address of the breakpoint.

Number of times the PC will pass through address before a break is triggered. Default value is 1.

Number of times the PC has passed through the specified address.

Figure 5.36 PC Breakpoint Properties Window

Selecting an USER breakpoint allows the Address, Condition Flag and Address Mask parameters to be set.



Masks the corresponding bit in the address rendering it as ‘don’t care’.

Figure 5.37 User Breakpoint Properties Window

Section 6 Tutorials

This section answers, in tutorial form, the most common questions asked about using this evaluation board:

- How do I download, and run a simple program?
- How do I write my own serial interface software?
- How do I use the analog input function?
- How can I gain access to some of the interrupt vectors?

Files referenced in this chapter are included in the distribution on the diskette labeled *Supplementary Tools and Tutorials*. The tutorial examples in this chapter assume that installation procedures described in section 2 have been completed.

Note: Source code listings in this manual are for explanation purposes only. Due to software revisions, the listings may not be identical to the listings on the disk.

6.1 Tutorial A: “ON, OFF, RUN”

The LCEVB-SH2 is equipped with bank of 8 red LEDs that may be program controlled. These LEDs are driven buy an octal latch on the high byte of the systems data bus (D8 .. D15). Any write to memory area 1 (CS1) will write to the latch. The first tutorial example shows how to turn these LEDs on and off. In the process, you’ll also see:

- How to access memory space 1 (CS1)
- How to write the LEDs
- How to download and run a simple program

Files associated with Tutorial A are located in a sub-directory called “A”.

6.1.1 Source Files for ON and OFF

Here is a listing of the source file ON.C:

```
#include <machine.h>
#define LED_PORT (unsigned short *)0x400000

main()
{
volatile short *p;
p=LED_PORT;
*p=0x5500;    /* value is 0x0000 for off.c */
exit();
}

exit()
{
trapa(11);    /* exception 11, software NMI to exit */
}

int __main() {}    /* now required by compiler; see release notes */
```

The contents of OFF.C differ only by a single value. Here's a detailed look at the components of this file: The Header file machine.h contains the intrinsic functions trapa( ). For more details on the intrinsic functions available refer to SH Series C Compiler (Document no. ADE-702-095).The #define statement at the beginning declare the location of the memory-mapped external octal latch. Accessing memory-mapped I/O requires special consideration, because compilers can detect storage that is written but is never read, and often eliminate the writes entirely. The specification "volatile" tells the compiler that code writing each of these locations should never be optimized away. The specifications "unsigned short" declare each as 16-bit locations.

Usually C programs can return to a monitor or operating system environment by "running off the end" of the main function. For this example, need to provide an explicit return to CMON. In the next tutorial section, you'll see how to make this return processing implicit. If return processing of any kind is omitted, "running off the end" means executing random instructions or data in memory following main. Control may or may not eventually return to CMON, and you can't be sure what damage has been done in the meantime.

As can be seen, the function exit contains simply the trapa (11) which generates an hardware exception 11, (NMI), which will cause CMON to do a warm start. A more harsh method of returning is to execute a trapa(0) exception 0, (RESET), which will reset the monitor to its default state including the LEDS. In the following tutorial, you'll see a gentler way of re-entering the monitor.

6.1.2 Running ON and OFF

Refer to section 5.13 for detail

6.1.3 Source for RUN

RUN.C goes one step further and continually displays a count value on the LEDs. Since the body of the program is an infinite loop, the RESET or NMI button must be pressed to return to the monitor program. Here is a listing of the source file RUN.C:

```
#include <machine.h>
#define LED_PORT (unsigned short *)0x400000
main()
{
long count;
int volatile *p;
long val;

p=LED_PORT;
val=1;
while(1){
*p=val++;
count=20;
while(count)count--;  /* wait loop */
        }
}
exit()
{
trapa(11);
}
int __main() {}     /* now required by compiler; see release notes */
```

6.1.4 Running RUN

Refer to section 5.13 for detail.

6.2 Tutorial B: “Hello World”

This tutorial shows how to output the familiar “Hello World” string from the the SH7043 serial port used by CMON. In the process, you’ll see how to:

- Setup an on-board the SH7043 serial port
- Use a monitor trap to do character I/O to either serial port
- Re-enter the monitor via a warm start re-entry
- Implement implicit return processing

Files associated with tutorial B are located in a sub-directory TUTOR_B.

6.2.1 Source File

This is the listing of the source file HELLO.C:

```
#include “montraps.h”
#include <machine.h>
main()
{
int port = CMON_PORTB ;
CMONtrap(CMON_PUTSTR, port, “\nHello World\n\n”);
exit();
}

CMONtrap(a1,a2,a3)
{
trapa(33);    /* arguments are passed implicitly */
}

int exit()
{
CMONtrap (CMON_EXIT, 0); /* trap exit to monitor, w/error status */
return(0);
}
int __main() {}    /* now required by compiler; see release notes */
```

Let’s discuss the function main() first. There are two serial ports on the SH7043, and the declaration of “port” chooses one, PORTB, which is the same one used to communicate with the host computer. Then the program references the CMONtrap() function to output a simple string to that port. The specific trap function is chosen by the first function argument in all CMONtrap calls, and the other arguments vary according to the trap function. The predefined constants shown in all caps are defined in the include file montraps.h.

Let’s look at the function CMONtrap() next. This function is defined with three generalized input arguments and no specification of a return value. We’re being unusually informal here because the number of input arguments to CMONtrap varies with the trap function selected. In C language the formal declarations describing this would be so complicated as to wholly obscure the simple interface implemented here. As a result, argument typing is omitted, and the compiler doesn’t do any type-checking of arguments.

The body of CMONtrap is simply an invocation of the SH assembly language to generate trap 33, the CMON function entry. The trap 33 handler inside CMON takes advantage of the fact that the first argument in the list is always passed in R4, the second in R5, and so forth. There’s no reason to mention the arguments again inside the definition of CMONtrap.

Note that exit() simply invokes the monitor exit function. The second argument is a return code that may be used for any purpose; upon return, CMON displays its value on the console. The return value zero usually denotes success.

6.2.2 Running HELLO

Refer to section 5.13 for detail.

6.3 More Advanced Examples

The tutorial directory also contains three more advanced examples in the sub-directories C\, D\, and E\. These three examples include:

- C\TRAPS.C, a demonstration of user program IO to the host computer using additional CMON trapa() function calls
- D\SERIAL.C, a demonstration of how to communicate directly with the serial ports without using CMON's trapa() function calls
- E\DEMO.C, a user application shell that communicates directly with the host, and which demonstrates the use of an additional trapa() function call to re-direct a hardware interrupt (from timer 0 in this case) to a user interrupt service routine

Code for these examples begins on the following page.

6.3.1 Code for TRAPS.C

```
/*************************************************************************
TRAPS.C
This program demonstrates the use of the software interrupt trapa(33)
vector to access some simple service routines in CMON for doing IO
to the serial ports. These traps are defined in "montraps.h"
*************************************************************************/
#include "montraps.h"
#include <machine.h>

/*-----------------------------------------------------------------------*/
main()
{
char c;int x;
CMONtrap(CMON_PUTCHAR,CMON_PORTB,'?');
c=CMONtrap(CMON_GETCHAR,CMON_PORTB);
CMONtrap(CMON_PUTCHAR,CMON_PORTB,c);
CMONtrap(CMON_PUTDEC,CMON_PORTB,0x0ff);
CMONtrap(CMON_PUTHEX,CMON_PORTB,0x0ff);
CMONtrap(CMON_PUTSTR,CMON_PORTB," good by!\n\r");
CMONtrap(CMON_EXIT, 0);
}
/*-----------------------------------------------------------------------*/
CMONtrap(a1,a2,a3)
{
trapa(33);
}
/*-----------------------------------------------------------------------*/
int __main() {              /* now required by compiler; see release notes */
/*-----------------------------------------------------------------------*/
```

6.3.2 Code for SERIAL.C

```
/***********************************************************************
SERIAL.C
This program demonstrates the use of raw hardware serial ports as
compared with using the CMON trapa(33) routines.
***********************************************************************/

#include "montraps.h"
#include <machine.h>

#define SMR0 (*(volatile char *)(0xFFFF81A0)) /* Channel 0  serial mode register */
#define BRR0 (*(volatile char *)(0xFFFF81A1)) /* Channel 0  bit rate register */
#define SCR0 (*(volatile char *)(0xFFFF81A2)) /* Channel 0  serial control register */
#define TDR0 (*(volatile char *)(0xFFFF81A3)) /* Channel 0  transmit data register */
#define SSR0 (*(volatile char *)(0xFFFF81A4)) /* Channel 0  serial status register */
#define RDR0 (*(volatile char *)(0xFFFF81A5)) /* Channel 0  receive data register */
#define SMR1 (*(volatile char *)(0xFFFF81B0)) /* Channel 1  serial mode register */
#define BRR1 (*(volatile char *)(0xFFFF81B1)) /* Channel 1  bit rate register */
#define SCR1 (*(volatile char *)(0xFFFF81B2)) /* Channel 1  serial control register */
#define TDR1 (*(volatile char *)(0xFFFF81B3)) /* Channel 1  transmit data register */
#define SSR1 (*(volatile char *)(0xFFFF81B4)) /* Channel 1  serial status register */
#define RDR1 (*(volatile char *)(0xFFFF81B5)) /* Channel 1  receive data register */
/*Serial control register bits*/
#define SCI_TIE         0x80    /* Transmit interrupt enable */
#define SCI_RIE         0x40    /* Receive interrupt enable */
#define SCI_TE          0x20    /* Transmit enable */
#define SCI_RE          0x10    /* Receive enable */
#define SCI_MPIE        0x08    /* Multiprocessor interrupt enable */
#define SCI_TEIE        0x04    /* Transmit end interrupt enable */
#define SCI_CKE1        0x02    /* Clock enable 1 */
#define SCI_CKE0        0x01    /* Clock enable 0 */

/* Serial status register bits*/
#define SCI_TDRE        0x80    /* Transmit data register empty */
#define SCI_RDRF        0x40    /* Receive data register full */
#define SCI_ORER        0x20    /* Overrun error */
#define SCI_FER         0x10    /* Framing error */
#define SCI_PER         0x08    /* Parity error */
#define SCI_TEND        0x04    /* Transmit end */
#define SCI_MPB         0x02    /* Multiprocessor bit */
#define SCI_MPBT        0x01    /* Multiprocessor bit transfer */

#define USE_PORT1
#ifdef USEPORT1
  #define SCR_PORT        SCR1
  #define SMR_PORT        SMR1
```

```
  #define BRR_PORT        BRR1
  #define PB_TXD_PORT     PB_TXD1
  #define PB_RXD_PORT     PB_RXD1
  #define RDR_PORT        RDR1
  #define SSR_PORT        SSR1
  #define TDR_PORT        TDR1
#else
  #define SCR_PORT        SCR0
  #define SMR_PORT        SMR0
  #define BRR_PORT        BRR0
  #define PB_TXD_PORT     PB_TXD0
  #define PB_RXD_PORT     PB_RXD0
  #define RDR_PORT        RDR0
  #define SSR_PORT        SSR0
  #define TDR_PORT        TDR0
#endif
/*------------------------------------------------------------------------*/
main()
{
int c=0;
while(c!='q'){
c=hw_rx_char();
hw_tx_char();
       }
exit();
}
/*------------------------------------------------------------------------*/
CMONtrap(a1,a2,a3)
{
trapa(33);
}
/*------------------------------------------------------------------------*/
int exit()
{
CMONtrap(CMON_EXIT, 0);
return(0);
}
/*------------------------------------------------------------------------*/
int __main() {  }          /* now required by compiler; see release notes */
/*------------------------------------------------------------------------*/
int hw_rx_ready()
{
char mySSR;
mySSR = SSR_PORT & ( SCI_PER | SCI_FER | SCI_ORER );
return SSR_PORT & SCI_RDRF ;
}
/*------------------------------------------------------------------------*/
```

```
int hw_rx_char()
{
char ch,mySSR;
while ( ! hw_rx_ready());
ch = RDR_PORT;
SSR_PORT &= ~SCI_RDRF;
mySSR = SSR_PORT & (SCI_PER | SCI_FER | SCI_ORER);
return ch;
}
/*------------------------------------------------------------------------*/
int hw_tx_ready()
{
return (SSR_PORT & SCI_TDRE);
}
/*------------------------------------------------------------------------*/
int hw_tx_char (char ch)
{
while(!hw_tx_ready());
TDR_PORT = ch;
SSR_PORT &= ~SCI_TDRE;
return 0;
}
/*------------------------------------------------------------------------*/
```

6.3.3 Code for INTER.C

```
/*****************************************************************************

MM.C
This program demonstrates two new techniques, 1) A small user program
shell that can communicate with the host computer and process simple
commands, and 2) the use of user interrupt vectors to call interrupt
service routines. CMON allows the user to specify a vector to a user
interrupt service routine. Vectors can be assigned to IRQ0 .. IRQ7 and
to the MTU's counter 0 trigger interrupt conditions TGR0A, TGR0B, TGR0C,
and TGR0D.
The serial interface from the previous example has been incorporated
into this program.
demo()
This function sets up timer 0, enables the interrupt for TGR0A and sets
the interrupt priority to 15 (required because the SR is usually set at
14). It then sets the TGR0A vector to point at myvector() which counts
the upper 4 LEDS down. It then starts the counter which will  generate
interrupts to be serviced by myvector(). Finally demo() enters a loop
which will count the lower 4 LEDS up. The effect is the both the upper
and lower 4 LEDS will be counting simultaneously, but in opposite
directions.
demo2()
demo2() is similar to demo() except that it returns to the main program
after setting up the interrupt for myvector.

*****************************************************************************/
#include "montraps.h"
#include "iosh7043.h"
#include "userints.h"
#include "serial.h"
#include <machine.h>
#include <string.h>

/*#define m_getch()  CMONtrap(CMON_GETCHAR,CMON_PORTB)
#define m_putch© CMONtrap(CMON_PUTCHAR,CMON_PORTB,c)   */
#define m_getch()  hw_rx_char()
#define m_putch© hw_tx_char©

#define message(a) (*(short *)0x400000=(a)<<8)
#define LEDS *(short *)0x400000
#define CRLF {m_putch(0x0a);m_putch(0x0d);}
/*---------------------------------------------------------------------*/
char cmd[100];
int leds;
int myvector();
/*---------------------------------------------------------------------*/
```

```
main()
{
while(1){
getcommand(cmd);
if (strncmp(cmd,"quit",4)==0)break;
else if (strcmp(cmd,"demo")==0)run_demo();
else if (strcmp(cmd,"demo2")==0)run_demo2();
else if (cmd[0]==0);
else m_puts("unknown command\n");
    }
exit();
}
/*---------------------------------------------------------------------*/
run_demo()
{
int i,count2;
m_puts("demo running...\n");
count2=0;
set_user_vec(UV_TGI0A,myvector);
settimer();
while(1){
leds=(leds&0xf0)|(0x0f&count2);
for(i=0;i<0xffff;i++);
count2++;
    }
}
/*---------------------------------------------------------------------*/
run_demo2()
{
m_puts("demo2 running...\n");
set_user_vec(UV_TGI0A,myvector);
settimer();
}
/*---------------------------------------------------------------------*/
myvector()
{
static int count1=3;
int i;

    i=MTU_TSR0;                     /*clear status       */
    MTU_TSR0 =0;                    /*clear status       */
    MTU_TCNT0=0x00;                 /*set counter to 0   */
count1--;
leds=(leds&0x0f)|(0xf0&count1);     /*count the leds down*/
message(leds);
}
/*---------------------------------------------------------------------*/
```

```
settimer()
{
    MTU_TCR0 =0x23;                    /*control reg to p/64 */
    MTU_TCNT0=0x00;                    /*set counter to 0    */
    MTU_TGR0A=0xfff;            /*set compare register*/
    MTU_TSR0 =0;                       /*clear status        */
    MTU_TSTR =1;                       /*start counter 0     */
    INTC_IPRD=0xf000;                  /*set intrerupt level */
    MTU_TIER0=0x41;                    /*set interrupt       */
}
/*---------------------------------------------------------------------------*/
getcommand(char *s)
{
char c;
m_putch('$');
while(1){
c=m_getch();
m_putch©;
if(c==0x0A)continue;
if(c==0x0D){*s=0;CRLF;return;}
*s++=c;
   }
}
/*---------------------------------------------------------------------------*/
CMONtrap(a1,a2,a3)
{
trapa(33);
}
/*---------------------------------------------------------------------------*/
int exit()
{
CMONtrap(CMON_EXIT,0);
return(0);
}
/*---------------------------------------------------------------------------*/
int __main(){}             /* now required by compiler; see release notes */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/*                     SERIAL COMMUNICATION                                  */
/*---------------------------------------------------------------------------*/
m_puts(char *s){
int c;
while(*s){
c=*s++;
m_putch©;
if(c==10)m_putch(13);
    }
```

```
}
/*-------------------------------------------------------------------------*/
int hw_rx_ready()
{
char mySSR;
mySSR = SSR_PORT & ( SCI_PER | SCI_FER | SCI_ORER );
return SSR_PORT & SCI_RDRF ;
}
/*-------------------------------------------------------------------------*/
int hw_rx_char()
{
char ch,mySSR;
while ( ! hw_rx_ready());
ch = RDR_PORT;
SSR_PORT &= ~SCI_RDRF;
mySSR = SSR_PORT & (SCI_PER | SCI_FER | SCI_ORER);
return ch;
}
/*-------------------------------------------------------------------------*/
int hw_tx_ready()
{
return (SSR_PORT & SCI_TDRE);
}
/*-------------------------------------------------------------------------*/
int hw_tx_char (char ch)
{
while(!hw_tx_ready());
TDR_PORT = ch;
SSR_PORT &= ~SCI_TDRE;
return 0;
}
/*-------------------------------------------------------------------------*/
```

APPENDIX A: Frequently Asked Questions

This section contains a list of frequently asked questions about developing and evaluating programs using the LCEVB-SH2.

1. When I turn the LCEVB-SH2 board the LEDs blink, what does that mean

2. How do I write to the SH7043 serial ports?

3. Why is RAM on the LCEVB-SH2 at such a high address?

4. Why doesn't the firmware use the SH7043 on-board RAM?

5. Why does the LCEVB-SH2 have word-wide instead of byte-wide RAM?

6. My program (or the monitor) is crashing randomly. What might be wrong?

7. How can I time my benchmarks?

8. Does an application program need to establish its own stack for proper operation on the LCEVB-SH2?

9. I just entered a very simple program loop using the CMON assembler and it crashed for no reason I can see. What's going on?

10. My benchmark shows that the SH7043 doesn't run as fast as I think it should. Why?

1. **When I turn the LCEVB-SH2 board the LEDs blink, what does that mean**

See the tutorial example SERIAL.C. Refer to the also Appendix C.

2. **How do I write to the SH7043 serial ports?**

The LCEVB-SH2 RAM starts at H'0800000. Since the monitor must reside at 0 (area 0) to provide for system boot-up, system RAM must be located somewhere else. Areas 2 is a logical place for RAM. Area is always read as word wide but can be written as words or bytes. In the case of byte reads the CPU automatically discards unused information.

3. **Why is RAM on the LCEVB-SH2 at such a high address**?

When the LCEVB-SH2 board is powered on or reset the system outputs diagnostic information during boot up. If the board is working correctly the last value put to the LEDs should be 0x31.

4. **Why doesn't the firmware use the SH7043 on-board RAM?**

On-board the SH7043 RAM is the fastest possible, being accessible in 16-bit chunks without wait states (when the bus controller register is set correctly). Committing this RAM to the monitor might interfere with using this area for full-speed benchmarks.

5. **Why does the LCEVB-SH2 have word-wide instead of byte-wide RAM?**

Word-wide accesses are faster. This choice was an inexpensive way of providing faster program execution.

6. **My program (or the monitor) is crashing randomly. What might be wrong**?

Check these possibilities:

— Check that your program isn't affecting the monitor RAM area. Use the CMON status command to find the current limit of monitor RAM use. Locate the code, data, and stack above that area.

— Check that there is actually RAM in the areas you are using.

— Make sure your power supply is more than sufficient for the needs of the LCEVB-SH2. Power supplies operating at or near their limits can sometimes cause programs to operate strangely.

— Make sure that the stack pointer (R15) is pointing to RAM. A simple way of doing this is to reset the monitor.

7. **How can I time my benchmarks?**

Every benchmark is different. One approach is to use an I/O bit, for example, by setting it low upon entering the code-of-interest and high again when leaving.

8. **Does an application program need to establish its own stack for proper operation on the LCEVB-SH2?**

Trivial programs don't. There's room on the monitor stack for user programs, and we've used simple programs without declaring a distinct user stack, but we'd prefer that you establish your own stack. See the tutorials for examples.

9. **I just entered a very simple program loop using the CMON assembler and it crashed for no reason I can see. What's going on?**

If your program ends with a branch, and there's garbage in memory following the branch, you've likely found an often misunderstood the SH7043 feature: "delayed branches". In order to be as efficient as possible, in the case of some program branches, the instruction following the branch is also executed even if the branch is taken. The BRA instruction is one of these. If you write a simple loop, it is likely to end with a BRA back to beginning, and it's likely that the garbage following the BRA caused the problem. You'll crash somehow, possibly generating an exception and a message like "INVALID INSTRUCTION" or "INVALID SLOT" or "CPU BUS ERROR". A full discussion of this subject is beyond the scope of this manual. An excellent rule of thumb is: when in doubt, follow all branches with innocuous instructions (such as NOPs).

10. **My benchmark shows that the SH7043 doesn't run as fast as I think it should. Why?**

For maximum flexibility, CMON accepts the default setting of WCR3, that is, the LCEVB-SH2 automatically inserts the maximum numbers of wait states into area 0 and area 2 accesses. This will certainly make the the SH7043 run slower. You can adjust the value of the W23, W22, W21, and W20 bits in WCR1 wait state register, consistent with the operating speed of the SH7043 microcomputer and the memory currently installed.

Appendix B Assembler Commands

This appendix lists assembler command syntax, sorted according to different categories and types.

B.1 Legend

Table below lists command syntax abbreviations and their meanings.

Abbreviation	Meaning
Rn	A numbered register
Rm	Another numbered register
#imm	Immediate data
disp	Displacement
disp8	8-bit displacement
disp12	12-bit displacement

B.2 Commands Sorted Alphabetically

```
add #imm,Rn
add Rm,Rn
addc Rm,Rn
addv Rm,Rn
and #imm,R0
and Rm,Rn
and.b #imm,@(R0,GBR)
bf disp8
bf.s disp8
bra disp12
braf Rm
brsf Rm
bsr disp12
bt disp8
bt.s disp8
clrmac
clrt
cmp/eq #imm,R0
cmp/eq Rm,Rn
cmp/ge Rm,Rn
cmp/gt Rm,Rn
cmp/hi Rm,Rn
cmp/hs Rm,Rn
cmp/pl Rn
cmp/pz Rn
cmp/str Rm,Rn
div0s Rm,Rn
div0u
div1 Rm,Rn
dt Rm
dmuls.l Rm,Rn
dmulu.l Rm,Rn
exts.b Rm,Rn
exts.w Rm,Rn
extu.b Rm,Rn
extu.w Rm,Rn
jmp @Rn
jsr @Rn
ldc Rn,GBR
ldc Rn,SR
ldc Rn,VBR
ldc.l @Rn+,GBR
ldc.l @Rn+,SR
ldc.l @Rn+,VBR
lds Rn,MACH
lds Rn,MACL
lds Rn,PR
lds.l @Rn+,MACH
lds.l @Rn+,MACL
lds.l @Rn+,PR
mac.w @Rm+,@Rn+
mov #imm,Rn
mov Rm,Rn
mov.b Rm,@(R0,Rn)
```

B.2 Commands Sorted Alphabetically (cont)

```
mov.b Rm,@-Rn
mov.b Rm,@Rn
mov.b @(disp,Rm),R0
mov.b @(disp,GBR),R0
mov.b @(R0,Rm),Rn
mov.b @Rm+,Rn
mov.b @Rm,Rn
mov.b R0,@(disp,Rm)
mov.b R0,@(disp,GBR)
mov.l Rm,@(disp,Rn)
mov.l Rm,@(R0,Rn)
mov.l Rm,@-Rn
mov.l Rm,@Rn
mov.l @(disp,Rn),Rm
mov.l @(disp,GBR),R0
mov.l @(disp,PC),Rn
mov.l @(R0,Rm),Rn
mov.l @Rm+,Rn
mov.l @Rm,Rn
mov.l R0,@(disp,GBR)
mov.w Rm,@(R0,Rn)
mov.w Rm,@-Rn
mov.w Rm,@Rn
mov.w @(disp,Rm),R0
mov.w @(disp,GBR),R0
mov.w @(disp,PC),Rn
mov.w @(R0,Rm),Rn
mov.w @Rm+,Rn
mov.w @Rm,Rn
mov.w R0,@(disp,Rm)
mov.w R0,@(disp,GBR)
mova @(disp,PC),R0
movt Rn
mul.l Rm,Rn
muls Rm,Rn
mulu Rm,Rn
neg Rm,Rn
negc Rm,Rn
nop
not Rm,Rn
or #imm,R0
or Rm,Rn
or.b #imm,@(R0,GBR)
rotcl Rn
rotcr Rn
rotl Rn
rotr Rn
rte
rts
sett
shal Rn
shar Rn
shll Rn
shll16 Rn
shll2 Rn
shll8 Rn
shlr Rn
shlr16 Rn
shlr2 Rn
shlr8 Rn
sleep
stc GBR,Rn
stc SR,Rn
stc VBR,Rn
stc.l GBR,@-Rn
stc.l SR,@-Rn
stc.l VBR,@-Rn
sts MACH,Rn
sts MACL,Rn
sts PR,Rn
sts.l MACH,@-Rn
sts.l MACL,@-Rn
sts.l PR,@-Rn
sub Rm,Rn
subc Rm,Rn
subv Rm,Rn
swap.b Rm,Rn
swap.w Rm,Rn
tas.b @Rn
trapa #imm
tst #imm,R0
tst Rm,Rn
tst.b #imm,@(R0,GBR)
xor #imm,R0
xor Rm,Rn
xor.b #imm,@(R0,GBR)
xtrct Rm,Rn
```

B.3 Commands Sorted by Type

B.3.1 Data Transfer

```
mov #imm,Rn
mov Rm,Rn
mov.b Rm,@(R0,Rn)
mov.b Rm,@-Rn
mov.b Rm,@Rn
mov.b @(disp,Rm),R0
mov.b @(disp,GBR),R0
mov.b @(R0,Rm),Rn
mov.b @Rm+,Rn
mov.b @Rm,Rn
mov.b R0,@(disp,Rm)
mov.b R0,@(disp,GBR)
mov.l Rm,@(disp,Rn)
mov.l Rm,@(R0,Rn)
mov.l Rm,@-Rn
mov.l Rm,@Rn
mov.l @(disp,Rn),Rm
mov.l @(disp,GBR),R0
mov.l @(disp,PC),Rn
mov.l @(R0,Rm),Rn
mov.l @Rm+,Rn
mov.l @Rm,Rn
mov.l R0,@(disp,GBR)
mov.w Rm,@(R0,Rn)
mov.w Rm,@-Rn
mov.w Rm,@Rn
mov.w @(disp,Rm),R0
mov.w @(disp,GBR),R0
mov.w @(disp,PC),Rn
mov.w @(R0,Rm),Rn
mov.w @Rm+,Rn
mov.w @Rm,Rn
mov.w R0,@(disp,Rm)
mov.w R0,@(disp,GBR)
mova @(disp,PC),R0
movt Rn
swap.b Rm,Rn
swap.w Rm,Rn
xtrct Rm,Rn
```

B.3.2 Arithmetic Operations

```
add #imm,Rn
add Rm,Rn
addc Rm,Rn
addv Rm,Rn
cmp/eq #imm,R0
cmp/eq Rm,Rn
cmp/ge Rm,Rn
cmp/gt Rm,Rn
cmp/hi Rm,Rn
cmp/hs Rm,Rn
cmp/pl Rn
cmp/pz Rn
cmp/str Rm,Rn
div0s Rm,Rn
div0u
div1 Rm,Rn
dmuls.l Rm,Rn
dmulu.l Rm,Rn
dt Rn
exts.b Rm,Rn
exts.w Rm,Rn
extu.b Rm,Rn
extu.w Rm,Rn
mac.w @Rm+,@Rn+
mul.l Rm,Rn
muls Rm,Rn
mulu Rm,Rn
neg Rm,Rn
negc Rm,Rn
sub Rm,Rn
subc Rm,Rn
subv Rm,Rn
```

B.3.3 Logical

```
and #imm,R0
and Rm,Rn
and.b #imm,@(R0,GBR)
not Rm,Rn
or #imm,R0
or Rm,Rn
or.b #imm,@(R0,GBR)
tas.b @Rn
xor #imm,R0
xor Rm,Rn
xor.b #imm,@(R0,GBR)
```

B.3.4 Shift/Rotate

```
rotl Rn
rotr Rn
rotcl Rn
rotcr Rn
shal Rn
shar Rn
shll Rn
shll2 Rn
shll8 Rn
shll16 Rn
shlr Rn
shlr2 Rn
shlr8 Rn
shlr16 Rn
```

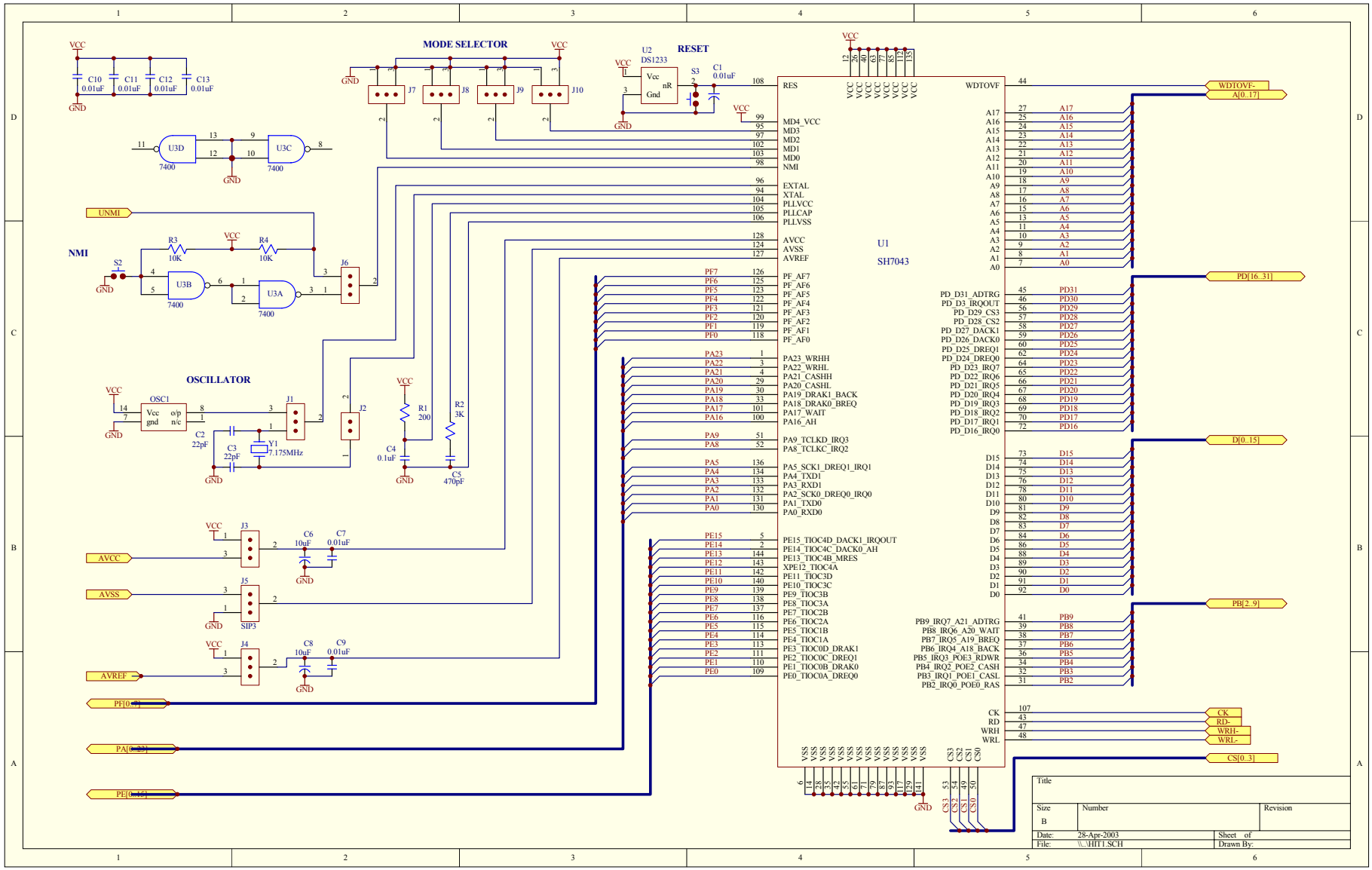
B.3.5 Branches

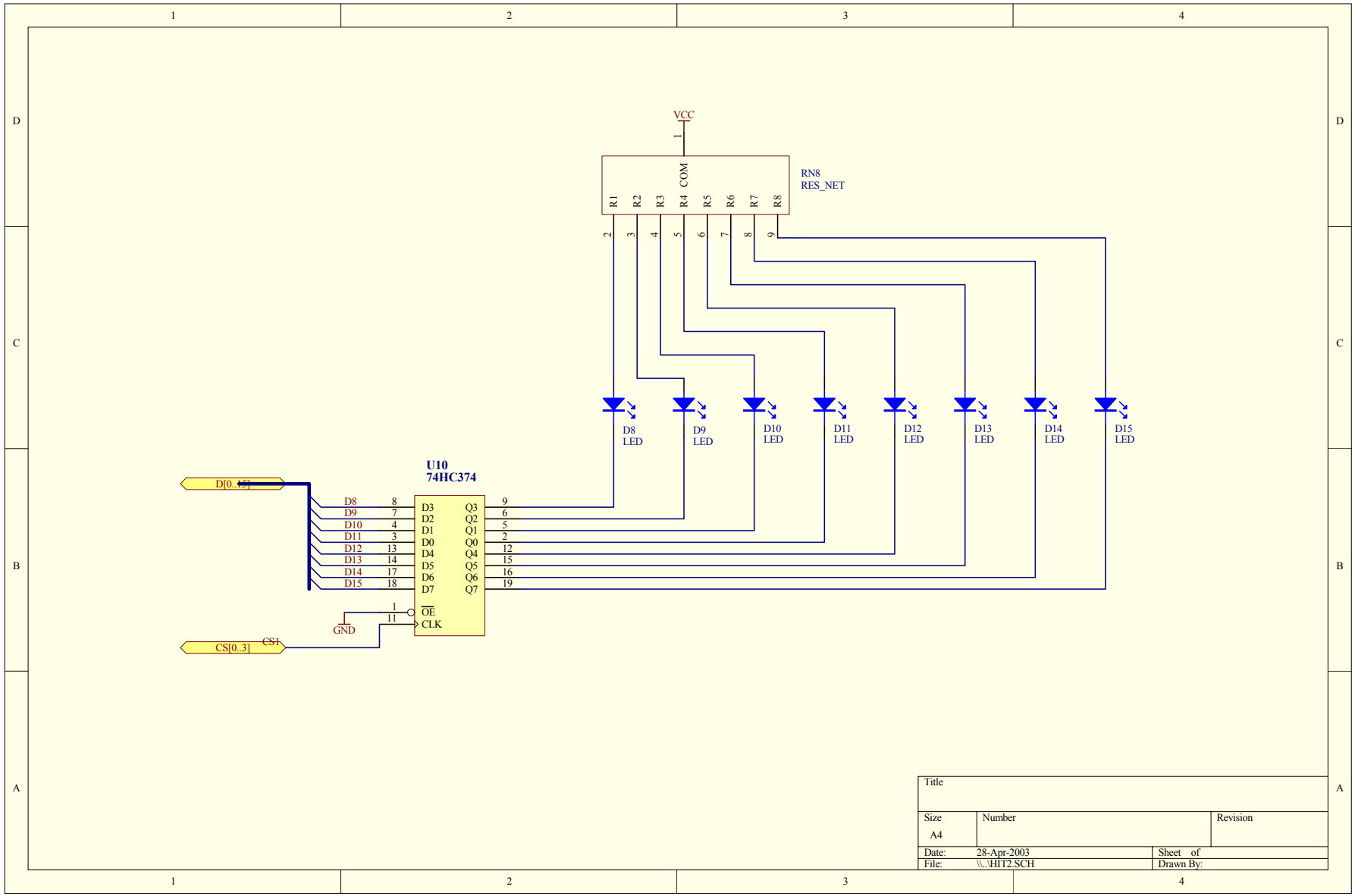
```
bf disp8
bf.s disp8
bt disp8
bt.s disp8
bra disp12
braf Rm
brsf Rm
bsr disp12
jmp @Rn
jsr @Rn
rts
```

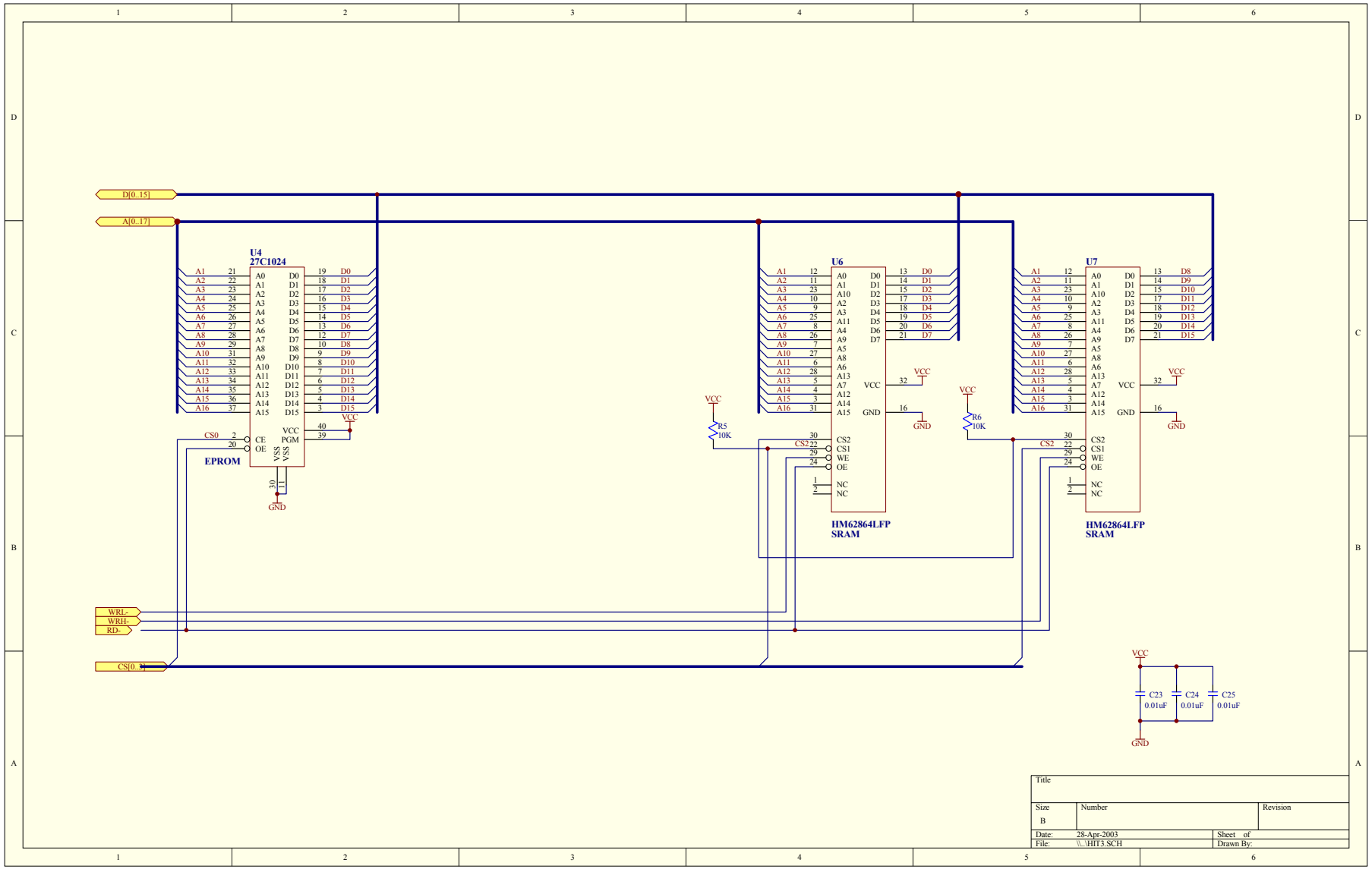
B.3.6 System Control

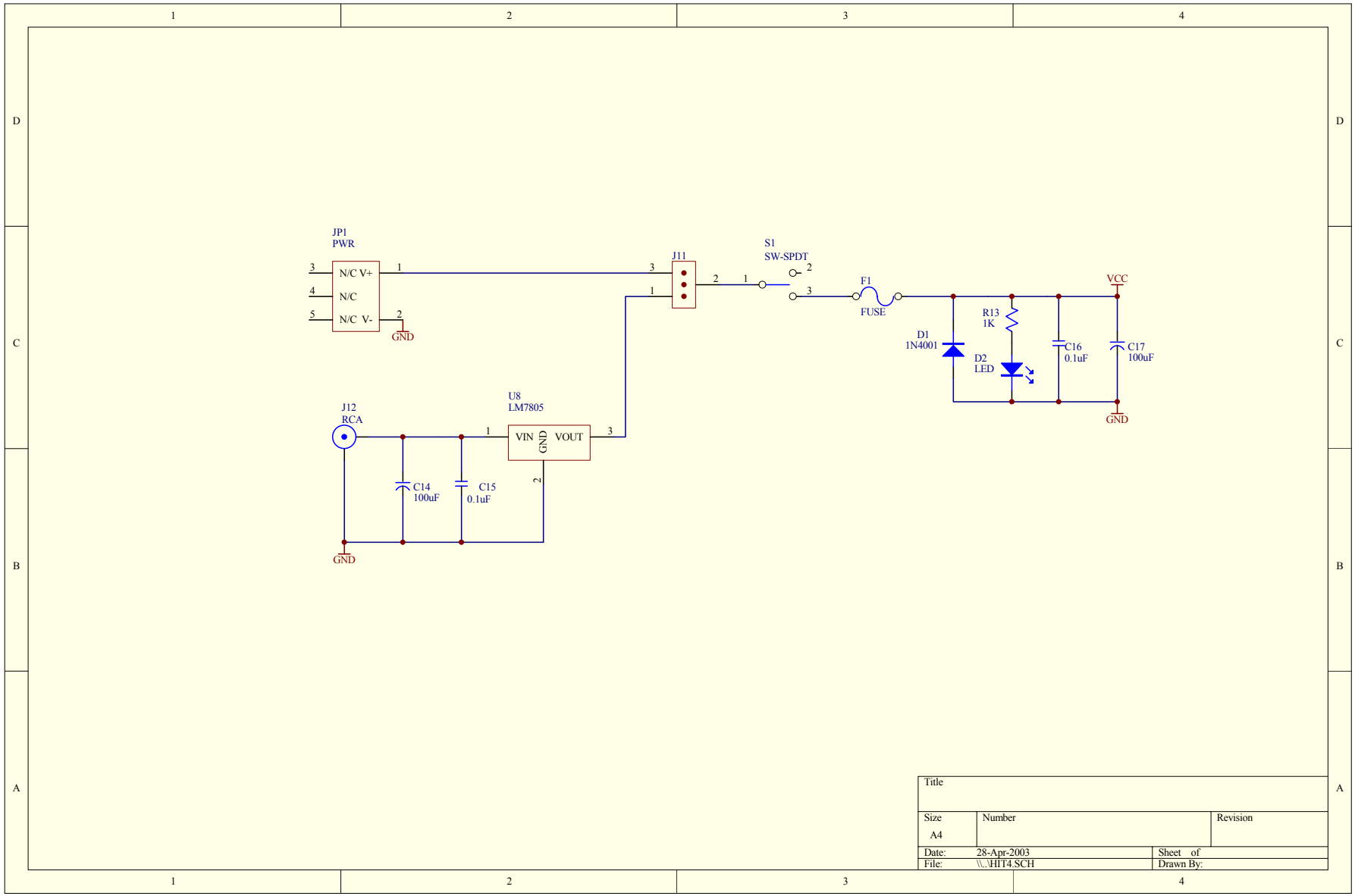
```
clrt
clrmac
ldc Rn,GBR
ldc Rn,SR
ldc Rn,VBR
ldc.l @Rn+,GBR
ldc.l @Rn+,SR
ldc.l @Rn+,VBR
lds Rn,MACH
lds Rn,MACL
lds Rn,PR
lds.l @Rn+,MACH
lds.l @Rn+,MACL
lds.l @Rn+,PR
nop
rte
sett
sleep
stc GBR,Rn
stc SR,Rn
stc VBR,Rn
stc.l GBR,@-Rn
stc.l SR,@-Rn
stc.l VBR,@-Rn
sts MACH,Rn
sts MACL,Rn
sts PR,Rn
sts.l MACH,@-Rn
sts.l MACL,@-Rn
sts.l PR,@-Rn
trapa #imm
tst #imm,R0
tst Rm,Rn
tst.b #imm,@(R0,GBR)
```

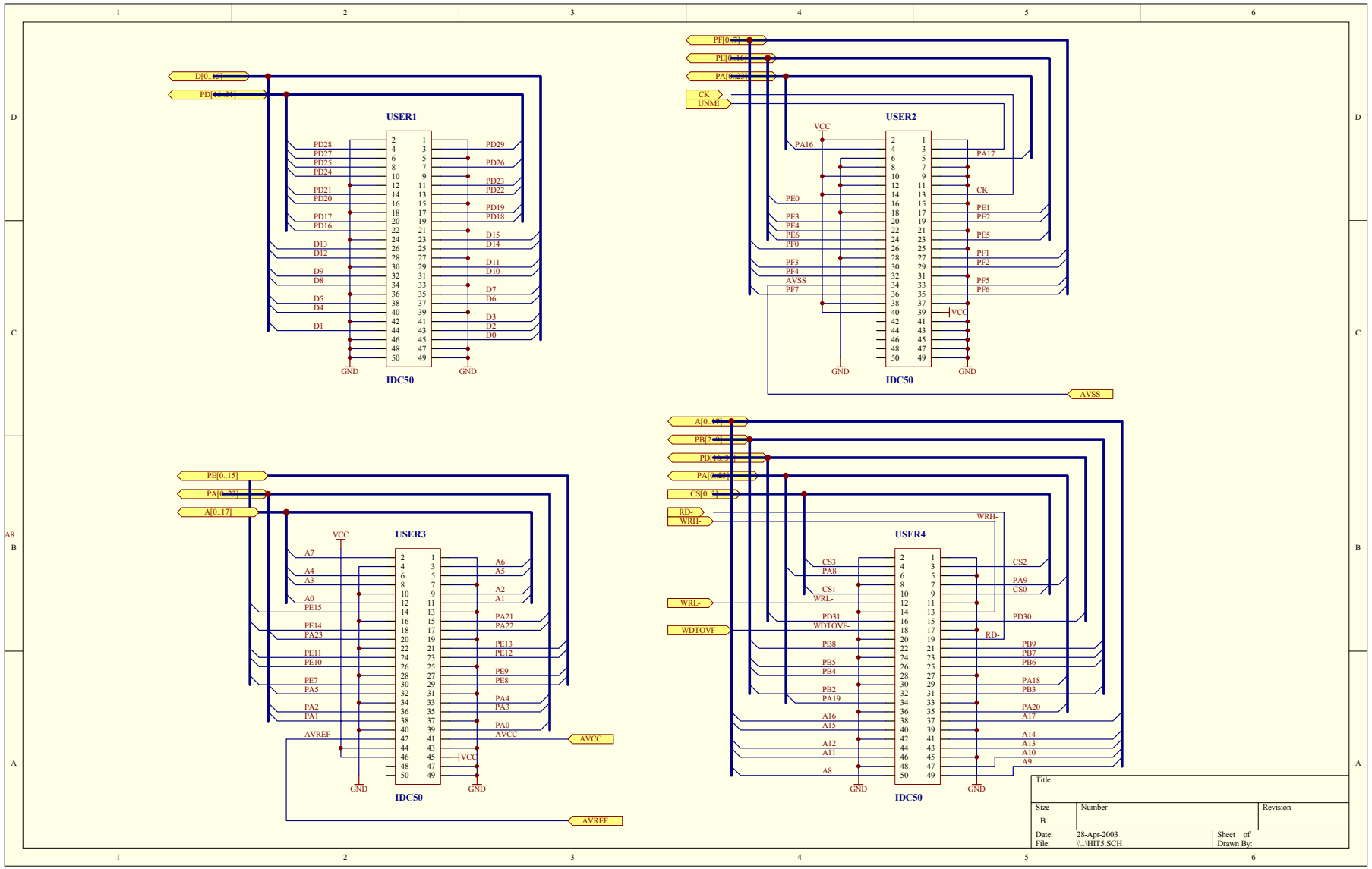
Appendix C LCEVB-SH2 Schematic Diagram

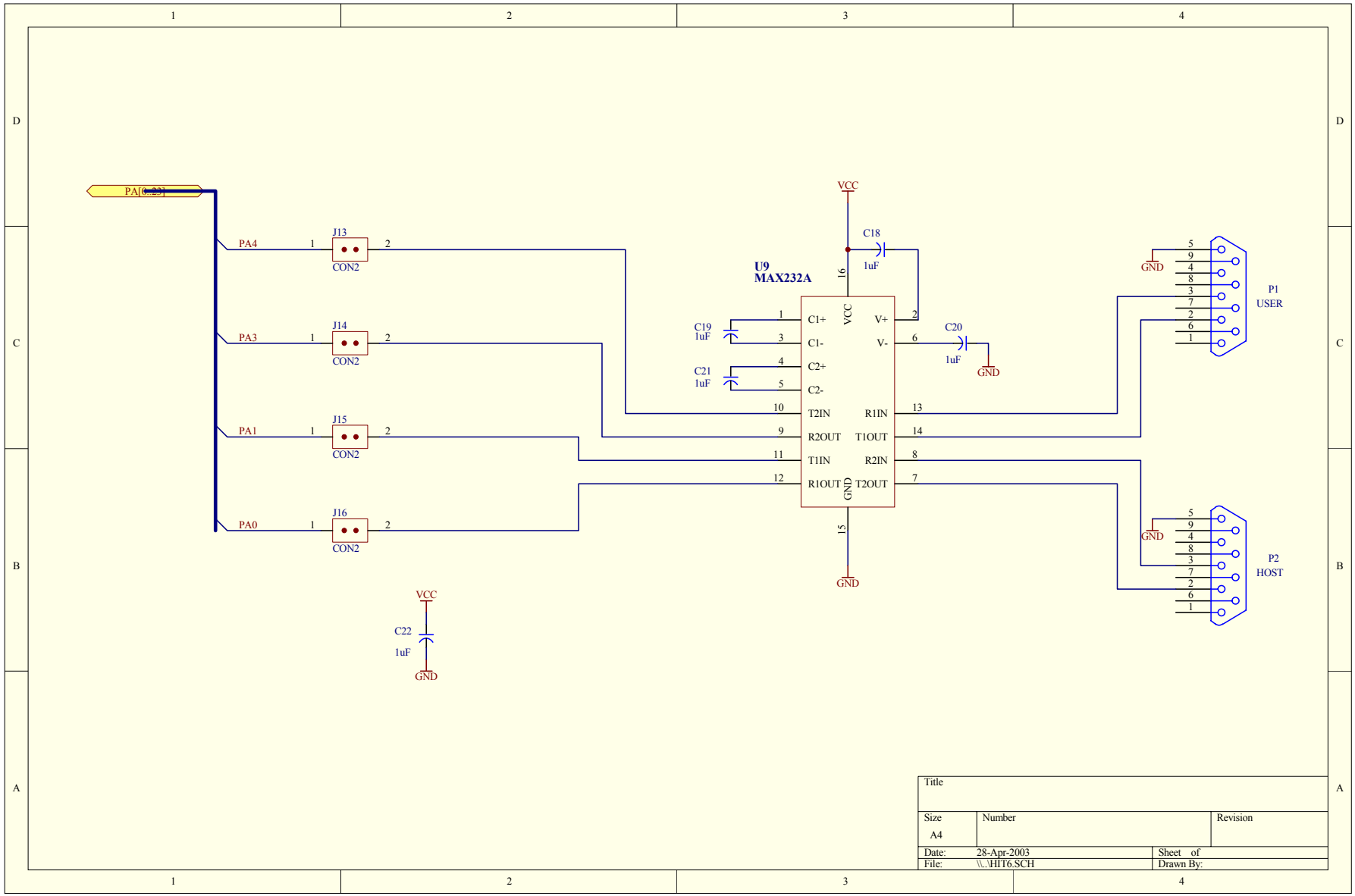



VCC
1
COM
RN8
RES_NET
R1
R2
R3
R4
R5
R6
R7
R8
2
3
4
5
6
7
8
9
D8 LED
D9 LED
D10 LED
D11 LED
D12 LED
D13 LED
D14 LED
D15 LED
U10
74HC374
D[0..15]
D8 8
D9 7
D10 4
D11 3
D12 13
D13 14
D14 17
D15 18
D3
D2
D1
D0
D4
D5
D6
D7
Q3 9
Q2 6
Q1 5
Q0 2
Q4 12
Q5 15
Q6 16
Q7 19
1
OE
11
CLK
GND
CS[0..3]
CS1
Title
Size
A4
Number
Revision
Date: 28-Apr-2003
Sheet of
File: \\..\HIT2.SCH
Drawn By:


D[0..15]
A[0..17]
U4
27C1024
EPROM
CS0
CE
OE
VCC
PGM
VSS
GND
U6
HM62864LFP
SRAM
U7
HM62864LFP
SRAM
R5
10K
R6
10K
CS2
CS1
WE
OE
NC
WRL-
WRH-
RD-
CS[0..3]
C23
0.01uF
C24
0.01uF
C25
0.01uF
Title
Size
B
Number
Revision
Date: 28-Apr-2003
File: \\..\HIT3.SCH
Sheet of
Drawn By:


JP1
PWR
N/C V+
N/C
N/C V-
GND
J11
S1
SW-SPDT
F1
FUSE
VCC
R13
1K
D1
1N4001
D2
LED
C16
0.1uF
C17
100uF
GND
J12
RCA
U8
LM7805
VIN
GND
VOUT
C14
100uF
C15
0.1uF
GND
Title
Size
A4
Number
Revision
Date:
28-Apr-2003
Sheet of
File:
\\..\HIT4.SCH
Drawn By:


USER1
IDC50
USER2
IDC50
USER3
IDC50
USER4
IDC50
VCC
GND
AVSS
AVCC
AVREF
Title
Size
B
Number
Revision
Date: 28-Apr-2003
File: \\..\HIT5.SCH
Sheet of
Drawn By:


PA[0..23]
PA4
PA3
PA1
PA0
J13
J14
J15
J16
CON2
U9
MAX232A
VCC
GND
C18
1uF
C19
1uF
C20
1uF
C21
1uF
C22
1uF
C1+
C1-
C2+
C2-
V+
V-
T2IN
R2OUT
T1IN
R1OUT
R1IN
T1OUT
R2IN
T2OUT
P1
USER
P2
HOST
Title
Size
A4
Number
Revision
Date: 28-Apr-2003
File: \\..\HIT6.SCH
Sheet of
Drawn By:

Renesas Technology (Asia Sales Offices)

URL: http://www.renesas.com

Renesas System Solutions Asia Pte. Ltd.

LCEVB-SH2

Renesas System Solutions Asia Pte. Ltd.