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.

RENESAS

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-SH1

For SH1 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-SH1
SH1 Evaluation Board
User's Manual

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

LCEVB-SH1 – SH1-Series Low-cost Evaluation Board User's Manual

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

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

Trademarks

(a) General

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

(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-SH1 Evaluation Board:
Throughout this document, the term "LCEVB-SH1" shall be defined as the LCEVB-SH1 emulation system produced only by Renesas System Solutions Asia Pte. Ltd. excluding all subsidiary products.

Purpose of LCEVB-SH1:
This emulation product is a software and hardware development tool for application systems employing the SH1 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. See the Renesas warranty booklet for details on the warranty period. 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 emulator product without Renesas's prior written consent or any problems caused by the user system.

Restrictions:

1. Earthing (applies only to manual for Renesas hardware products)
   This hardware is designed for use with equipment that is fully earthed.
   Ensure that all equipments used are appropriately earthed.
   Failure to do so could lead to danger for the operator or damaged to equipments.

2. Electrostatic Discharge Precautions (applies only to manuals for Renesas hardware products)
   This hardware contains devices that are sensitive to electrostatic discharge.
   Ensure appropriate precautions are observed during handling and accessing connections.
   Failure to do so could result in damage to the equipment.

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

Other Important Things to Keep in Mind:

1. Circuitry and other examples described herein are meant merely to indicate the characteristics and performance of Renesas Technology's semiconductor products. Renesas assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples described herein.

2. No license is granted by implication or otherwise under any patents or other rights of any third party or Renesas.

3. MEDICAL APPLICATIONS: Renesas Technology's products are not authorized for use in MEDICAL APPLICATIONS without the written consent of the appropriate officer of Renesas Technology (Asia Sales company). Such use includes, but is not limited to, use in life support systems. Buyers of Renesas Technology's products are requested to notify the relevant Renesas Technology (Asia Sales offices) when planning to use the products in MEDICAL APPLICATIONS.

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

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

PREFACE

This guide explains how to setup and use the LCEVB-SH1 emulation system for the SH1 series of MCU.

Section 1	Introduction Introduction of LCEVB-SH1 system including its package, specification and functions.
Section 2	Functional Description Explain the functional blocks in LCEVB-SH1
Section 3	Board Options Configurable components in LCEVB-SH1
Section 4	Hardware Startup The necessary hardware installation to startup LCEVB-SH1
Section 5	Software Startup The software installation sequences
Section 6	LCEVB-SH1 System Check General check/diagnostic possible on LCEVB-SH1
Section 7	Tutorial Session Provides a step by step guide in using the LCEVB-SH1 to perform emulation.
Section 8	Troubleshooting Advises on some basic fault locating methods and commonly make mistakes.

Related Manuals:
- SuperH RISC engine C/C++ Compiler, Assembler, Optimizing, Linkage Editor User's Manual
- SH7034, SH7032 Hardware Manual
- SH7020, SH7021 Hardware Manual

Table of Contents

Section 1. Introduction

1.1 Overview

The SH1 Evaluation Board (LCEVB-SH1) is an inexpensive demonstration / evaluation tool for the SH7000 family of RISC micro-controllers. It supports two members of the SH-1 family:

- SH7020 Series : SH7021, SH7020
- SH7032 Series : SH7034, SH7032

Figure 1.1 shows the physical layout of the LCEVB-SH1 system not drawn to scale.

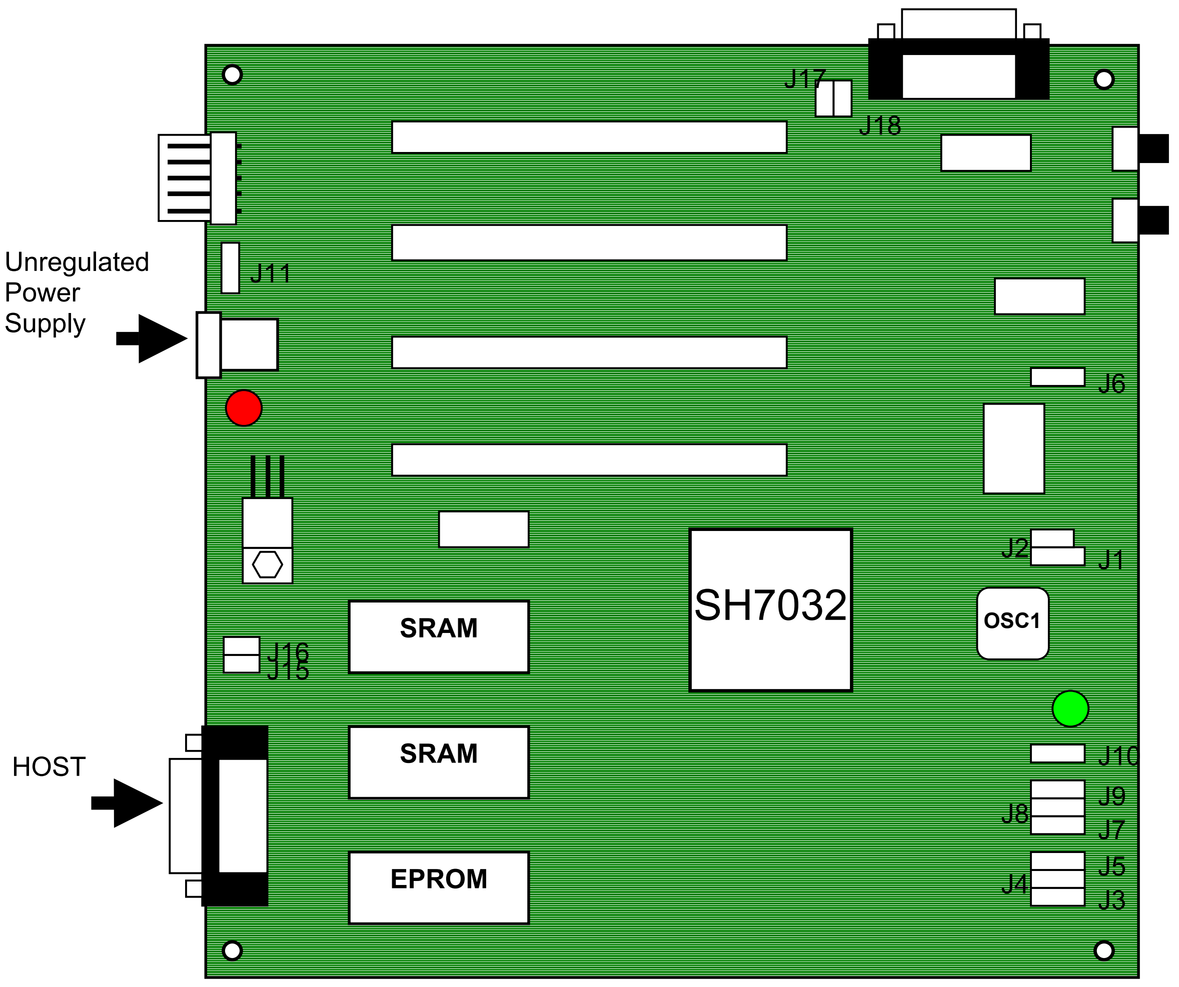


Figure 1-1 LCEVB-SH1 Layout

Schematic diagrams are provided at the back of this manual.

At the top level, the LCEVB-SH1 is composed of an SH processor, ROM, RAM, and two serial ports, as shown in figure 1.2.

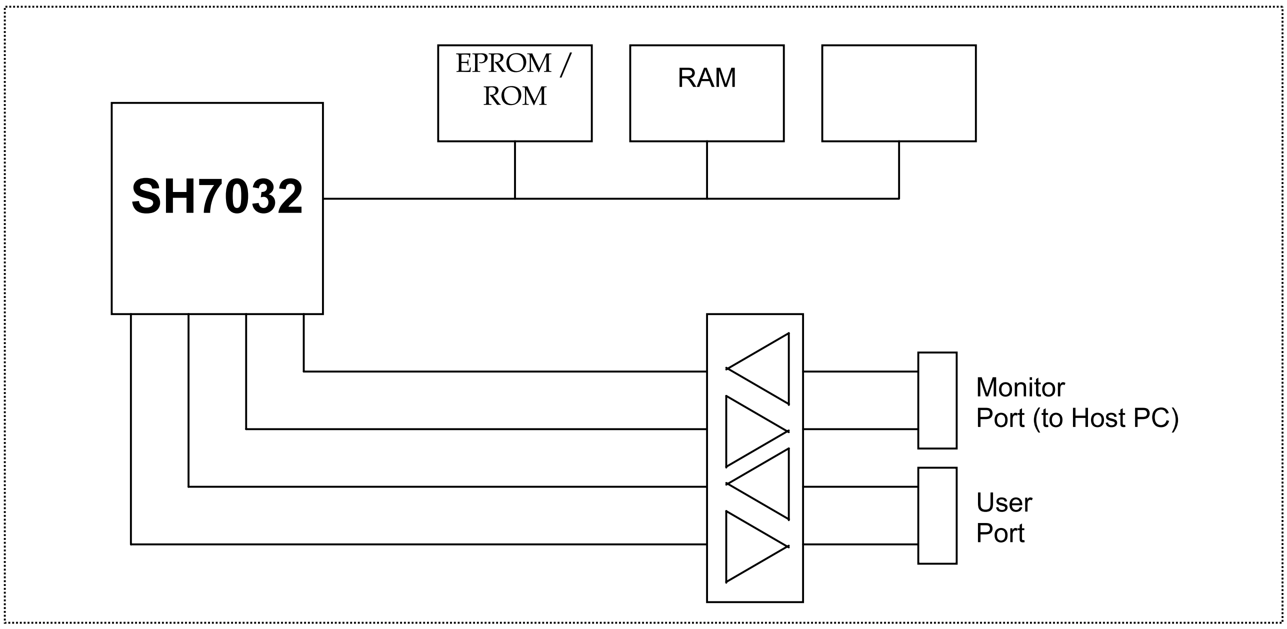


Figure 1-2 LCEVB-SH1 Functional Block Diagram

The SH1 processor contains most of the decoding and glue logic necessary to implement an expanded memory SH1-based system.

Read-only memory (ROM/EPROM) contains the Monitor Firmware. Two byte-wide (WORD) 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 SH1 Universal Asynchronous Receiver/Transmitters (UARTs). The respective usage of the ports are:

- 1 x Port is dedicated to the on-board Monitor Firmware for Host PC communication.
- 1 x Port is available to the user for user application system

The PC Host communication with LCEVB-SH1 is through PC interface program, HDI (Hitachi Debugging Interface).

Users re-configuring LCEVB-SH1 I/O ports should be caution that pull-up resistors may be required for proper operation in some the port configurations. In particular, users adding external memory in area 3 through 7 should be aware that the chip selects provided by SH1 are shared and may be floating until properly configured.

1.2 Package

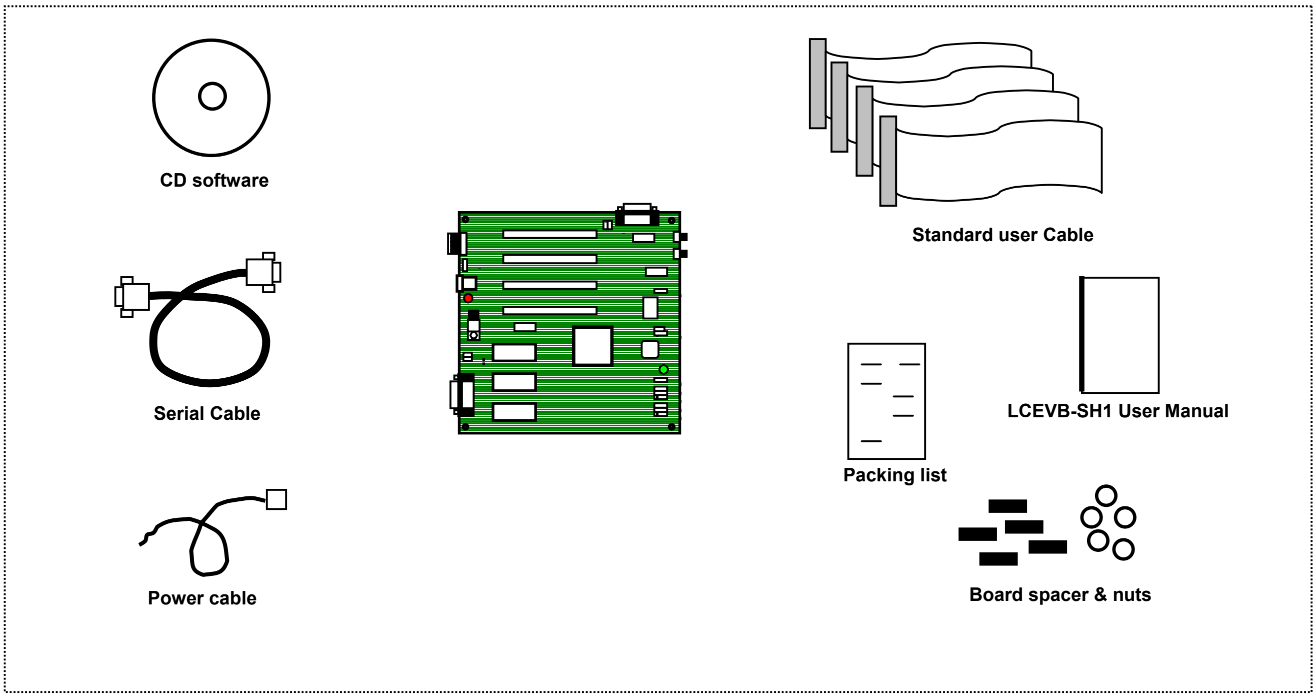


Figure 1-3 LCEVB-SH1 Package

1.2.1 Software Components

The software components included in the package are listed below.

- 1 x CD Software setup
  - Setup.exe for HDI setup
  - Adobe Reader Version 3.01
  - SH Series Hardware manuals in pdf format
  - SH Series Evaluation software and manuals in pdf format

Documents included in the package:

- 1 x User Manual
- 1 x Packing list

1.2.2 Hardware Components

The hardware components included in the package are listed below.

- 1 x LCEVB-SH1 System
- 4 x General User Cable
- 1 x Serial cable
- 1 x Power Supply Cable
- 5 x Board spacer & Nuts

1.3 Specifications

ITEM	SPECIFICATION
MCU series	• SH7034, SH7032, SH7021, SH7022
Host PC	• PC/AT i486 or above • Microsoft Windows 3.x / Window 95 • One free serial port • Approximately 4 Mbytes of free hard disk space
Host-Interface	• RS232C Serial Communication • Baud rate : 57600bps
Dimension	• 160 x 160 x 25 mm
Power supply requirement	• Power-jet (unregulated) Input : 7.5 ~ 9 VDC (1A) J11 set at 2-3 (input to regulator) Or • Direct DC (regulated) Input : 5 VDC (1A) J11 set at 1-2 (bypass regulator)
Environmental	• Operating Temperature: 10°C to 35°C • Humidity: 30% to 85% RH (no condensation) • Corrosive Gas: None

Table 1-1 LCEVB-SH1 Specification

1.4 Summary of LCEVB-SH1 Functions

ITEM	SPECIFICATION
Emulation	• Performs close to real-time emulation of a target program • High-level C debugging capability with SYSROF User Target Program • Performs simulated single step execution • Supports MCU frequency : 20MHz • Modifies and displays MCU registers • Resets MCU
File	• Loads User Target Program (SYSROF or Motorola S-type format) • Save target program in Motorola S-type format • Save Session
Memory functions	• Fixed User usable Emulation Memory Address (H'0A000000 to H'0A00FFFF) • Modifies and displays memory contents (including memory mapped peripheral registers) • Dumps a range of memory contents • Fills data with specified pattern • Standard 64Kbyte high-speed RAM for emulation
Breakpoint	• 20 PC breakpoints
Single step	• Performs simulated single step execution • Executes target program in step/s.

Table 1-2 LCEVB-SH1 Functions

Section 2. Functional Description

The LCEVB-SH1 includes the following components:

- SH1 (SH7032) RISC Microcomputer
- Clock circuitry
- Reset circuitry
- NMI circuitry
- ROM memory
- RAM memory
- Serial interface
- LED driver
- External user interface

Complete LCEVB-SH1 schematics are provided as part of the LCEVB-SH1 kit and are referenced throughout this chapter.

2.1 SH1 RISC Microcomputer

Because the SH1 (SH7032) provides many on-board functions required to implement an expanded-memory micro-controller system (for example, address area decoding), the amount of glue logic required is minimized.

2.2 Clock Circuitry

The LCEVB-SH1 may use one of two clock sources:

- AT-cut parallel resonating system
- Oscillator

The SH1 is designed to operate with an AT-cut parallel resonating crystal (default clock source, Y1).

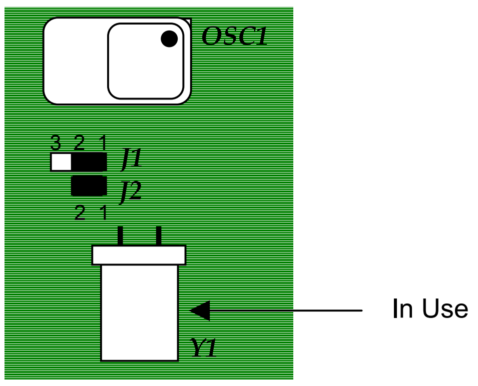


Figure 2-1 Selecting AC-cut parallel resonating system for system clock

Alternatively by changing the jumpers setting of both J1 and J2, a standard TTL “can” oscillator may be used.

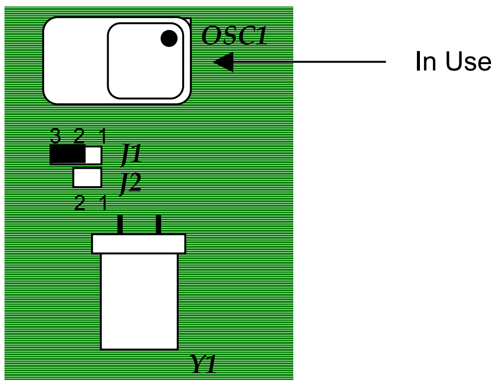


Figure 2-2 Selecting Oscillator system for system clock

The setting of J1 and J2 can be shown below:

Jumper setting		Assignment
J1	**J2**	
1-2	1-2	AT-cut parallel resonating system (Y1 & circuitry) (default)
2-3	NC	Oscillator (OSC1)

Table 2-3 Jumper J1 and J2 setting

2.3 Reset Circuitry

2.3.1 Reset Generator

The reset generator for the LCEVB-SH1 is a Dallas Semiconductor DS1233 “Econo Reset” device.

The DS1233 monitors its supply voltage. When the supply voltage is out of tolerance level, the DS1233 pulls its reset input/output line active-low. This condition continues indefinitely. After the voltage reaches tolerance level again, the reset is held low for an additional 350 ms to allow for final supply stabilization before release the processor from reset state.

The DS1233 monitors its own reset output so that a pushbutton can be used as a reset source. The DS1233 de-bounces the input from the pushbutton (S1) provides a 350-ms reset signal when S1 is released.

Quickly switching power off then on supplying to the board may not allow V_{CC} to fall low enough to generate a reset pulse. In practice, the SH1 usually continues to operate normally. Rapid switching of the power supply stresses the integrated circuit components and is not recommended.

2.3.2 Reset and Non-Maskable Interrupt (NMI)

The SH1 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 SH1 is initialised throughout. If NMI is low, the manual reset sequence initiates the SH1 except the following:

- bus state controller
- pin function controller
- I/O ports

The LCEVB-SH1 by default generates a power-on reset when:

- power is applied into the system
- reset pushbutton is depressed and released

2.4 NMI Circuitry

The NMI input of the SH1 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-SH1 uses two NAND gates (U3A and U3B) as an inverter to de-bounce the output of momentary pushbutton S2. In the quiescent case, the output of U3B (and thus the SH1 NMI input) is high. Closing/depressing S2 cause 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 pushbutton (S1) always generates a power-on reset. In other words, when the board is reset, all SH1 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 (S1), putting the SH1 into reset state.
2. Close the NMI switch (S2), generating a negative-going edge on NMI, which is ignored.
3. Release the reset switch (S1), starting the SH with NMI low.
4. Release the NMI switch (S2), returning NMI to its base state.

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

Jumper setting (J6)	Assignment
1-2	Internal usage
2-3	External usage (to S2) (default)

Table 2-4 Jumper J6 setting

2.5 EPROM/ROM

The LCEVB-SH1's EPROM/ROM memory is provided by U4, which is configured to contain 64k × 8, 27(C)512-family device. Below shows the memory map.

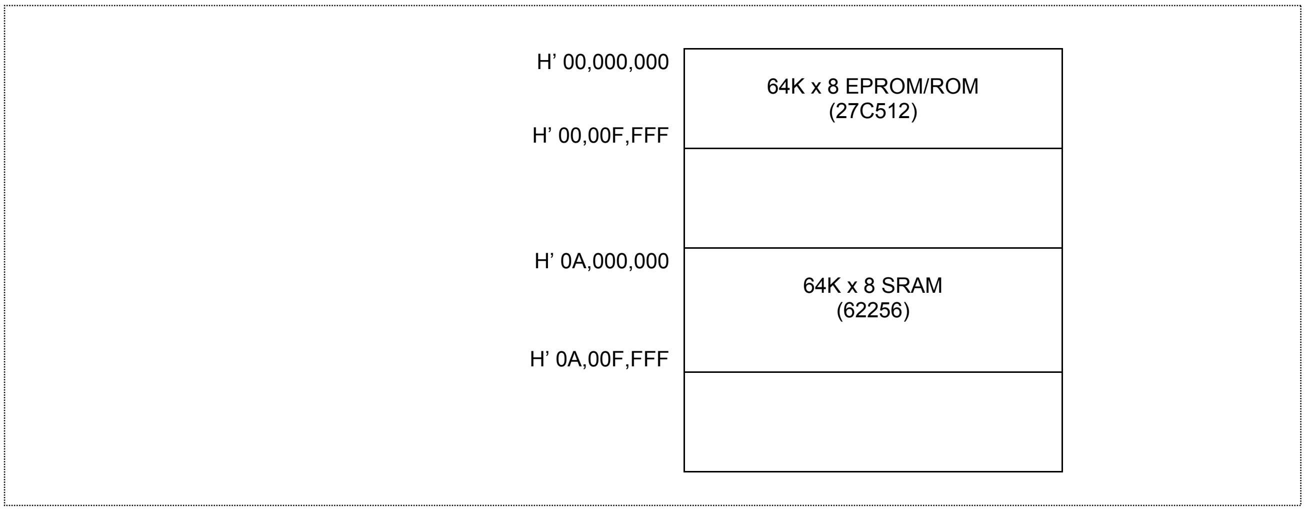


Figure 2-3 LCEVB-SH1 Memory Map

In either case, the EPROM/ROM is located in area 0 of the SH1 memory space, starting at location 0. U4 is always accessed a byte at a time. The memory area select signal, CS0 is generated by the SH1 and is sufficient to select either device.

The value of SH1 wait state control register 3 (WCR3) bits A02LW1 and A02LW0 control the number of wait states automatically inserted for accesses to area 0 and area 2 by the SH1 on-board bus state controller. Since RAM memory is located in area 2, the access time requirements for both RAM and EPROM/ROM must be considered when setting WCR3 and set to conform with the slowest of the two (normally EPROM/ROM).

2.6 RAM

The LCEVB-SH1's Emulation RAM is at U5 and U6, which contain a pair of 32k × 8, 62256-family static CMOS RAM organized for word-wide access. Figure 3.3 shows the memory map.

In either case, the RAM memory is located in area 2 of the SH1 memory space, nominally starting at location H'2000000. The configuration of U5 and U6 is 16 bits wide, so for proper access this RAM memory must be accessed starting at address H'A000000. When RAM is referenced at this address, the memory area select signal CS2, high byte strobe (HBS), and low byte strobe (LBS) signals are generated by the SH1 and are externally combined by OR gates U7A and U7B before being used as the RAM device select signal. In order for HBS and LBS signals to be generated, bit BAS of the SH1 bus control register (BCR) must be asserted.

The value of SH1 wait-state control register 3 (WCR3) bits A02LW1, A02LW0 controls the number of wait states automatically inserted for accesses to area 0 and area 2 by the SH1 bus state controller. Since EPROM/ROM memory is located in area 0, the access time requirements for both RAM and EPROM/ROM must be considered when setting WCR3 and set to conform with the slowest of the two, normally ROM. Since a minimum of one wait-state is used for external accesses, and this corresponds to an access time of 120ns at a CPU speed of 20MHz, it is likely to be most convenient to use 120ns RAM and EPROM/ROM.

2.7 Serial Interface

The LCEVB-SH1 supports two three-wire serial channels using the two identical SH1 SCI UART-type devices:

- SCI0
- SCI1

Of these, SCI1 is normally dedicated to use by Monitor Firmware for communication with a Host PC. SCI0 is available for User Target System development.

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 the two channels. U9 is a standard 16-pin MAX-232 device.

2.8 LED Driver

U3 (7400, NAND chip), is used to drive LED D1 (Green LED). Jumper J10 should be set to 2-3 connecting to SH1 port pin PA15. Alternately, jumper J10 set to 1-2 cause connection to SH1 port pin PB15.

Jumper setting (J10)	**Assignment**
1-2	Connecting to PB15 (default)
2-3	Connecting to PA15

Table 2-5 Jumper J10 setting

2.9 External User Interface

The External User Interface output most of the Processor (SH1) signals to User Target System. The arrangement is consistent 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.

Connector*	Signals
UCN1	• SH data lines (D0–D15) • SH Port B lines(PB14 & PB15) • SH address lines (A0-A5)
UCN2	• SH address lines (A6–A21) • CS0-CS3 • SH1 Port A lines (PA0-PA3)
UCN3	• SH Port A lines (PA4–PA15) • CK • WDTOVF • NMI
UCN4	• SH Port B lines (PB0-PB13) • SH Port C lines (PC0–PC7) • AVCC • AVREF • AVSS

Table 2-6 External User Interface pin-assignment

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-SH1. External circuits drawing >50mA at +5V 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 below:

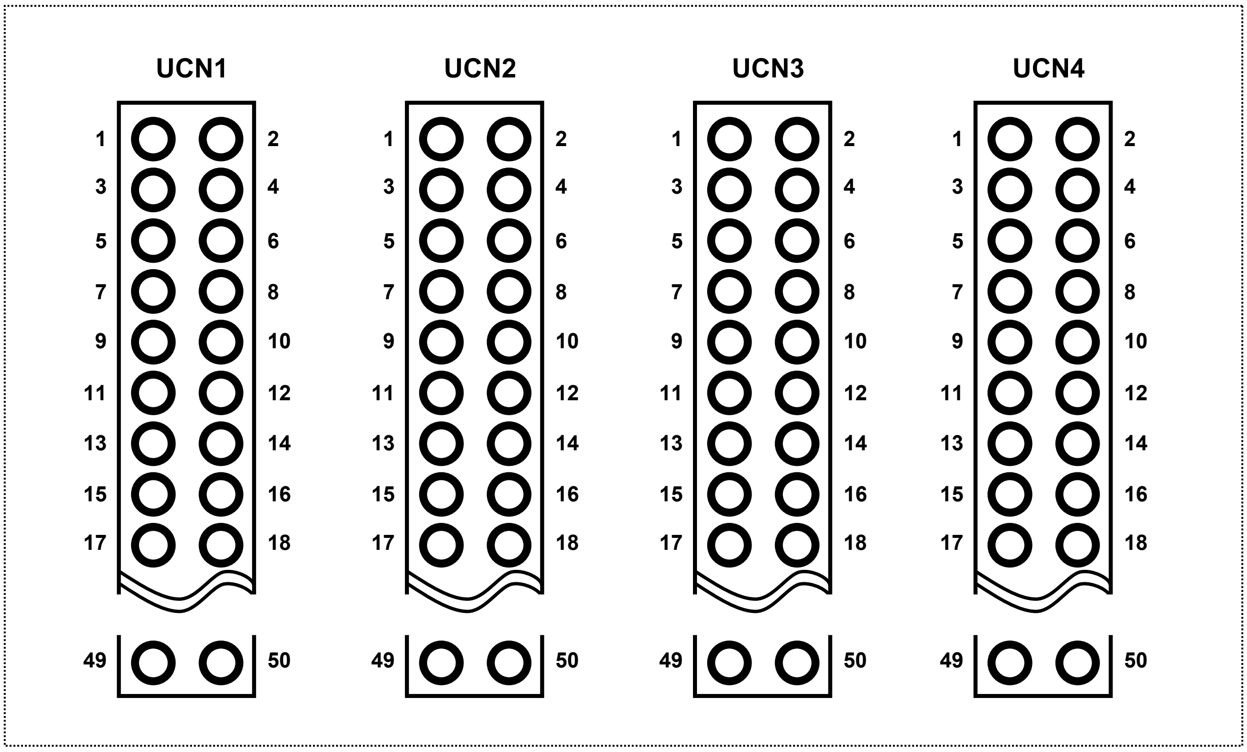


Figure 2-4 User External Interface Connector Configuration

Section 3. Board Options

The LCEVB-SH1 provides a number of user-settable optional configurations. All of these are chosen by jumper settings.

3.1 Jumpers

LCEVB-SH1 jumpers allow User to configure the board as required for evaluation. For simplicity, all jumpers are:

- three-pin header
  or
- two-pin header

In each case, the default jumper setting is pin 1 to pin 2 (1-2). For most LCEVB-SH1 uses, these settings need not be changed.

Jn	Use	Default (1-2)	Alternate (2-3)
J1	XTAL Selection	Resonating XTAL	TTL XTAL
J2	XTAL Selection	Resonating XTAL	No connection
J3	A_{VCC}	= digital V_{CC}	Set externally
J4	A_{VREF}	= digital V_{CC}	Set externally
J5	A_{VCC}	= digital V_{SS}	Set external
J6	NMI	Internal	External
J7	SH MD2	Mode 0	Set according to table 4.2
J8	SH MD1	Mode 0	Set according to table 4.2
J9	SH MD0	Mode 0	Set according to table 4.2
J10	User LED	PB15 drives LED	PA15 drives LED
J11	Power	Direct Power supply	Power-jet Power supply (DC adapter)
J12	Power	Unregulated Power Supply (power-jet power supply)	
JP1	Power	Regulated Power Supply (direct power supply)	
J15	TxD1	PB11 connected	Do not connect Jumper if • PB11 • PB10 • PB9 • PB8 are to be left open.
J16	RxD1	PB10 connected	
J17	TxD0	PB9 connected	
J18	RxD0	PB8 connected	

Table 3-1 Jumper Settings and Options

The following sections describe each jumper and its alternative settings.

3.1.1 Jumper J1 and J2 (Crystal Clock Source)

The LCEVB-SH1 comes with two types of clock source:

- AT-cut parallel resonant crystal
- TTL Can Crystal

Either clock source may be used. To make a selection, simply make the necessary connection on jumper J1 and 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).

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

As described in section 14 of the *SH7032 and SH7034 RISC Hardware Manual,* the port C bits of the SH1 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 SH1 analog subsystem. For demonstration purposes, this configuration may be sufficient. However, to demonstrate the full capabilities of the SH1 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.

The recommended noise suppression capacitors are provided on reference circuits as recommended by the hardware manual section 14.7.2.

If an external analog V_{CC} (AV_{CC}) is provided to the SH1 on Header 4 Pin 1, set J3 (2-3).

If an external analog reference voltage (V_{REF}) is provided to the SH on Header 4 Pin 2, set J3 (2-3).

If an external analog ground (AV_{SS}) is provided to the SH on Header 4 Pin 12, set J5 (2-3).

Leaving any of these jumpers open is not recommended.

3.1.3 Jumper J6 (NMI)

Default (1-2) Setting	: The SH NMI input is controlled by the set-reset flip-flop de-bounce circuitry implemented with AND gates U3A and U3B (schematic diagram).
Alternate (2-3) Setting	: The SH1 NMI input is controlled by an external signal (on-board pull-up is provided).
Open Setting	: Not recommended. The SH1 NMI signal should be driven in most conditions. Failure to do so may cause the board to operate erratically.

3.1.4 Jumpers J7, J8, and J9 (Setting Operating Mode)

As described in section 3 of the *SH7032 and SH7034 RISC Hardware Manual,* the operating mode of the SH1 processor 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 SH1 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.

Mode	J7	J8	J9	Description	Implementation
0	1-2	1-2	1-2	Memory area 0 is 8-bit	Default
1	1-2	1-2	2-3	Memory area 0 is 16-bit	Not supported
2	1-2	2-3	1-2	Memory area 0 is on-chip	SH7034 *only*
7	2-3	2-3	2-3	EPROM programming	Not supported

Table 3-2 Operating Mode Settings

3.1.5 Jumper J10 (LED Indicator Source)

LED D1 (Green LED) may be driven by SH1 port signal PA15 by setting J10 (2-3) or PB15 by setting J10 (1-2). The default source is PB15. If these two sources is allocated for other use, remove the jumper at J10.

3.1.6 Jumper J11 (DC Regulator Bypass)

Jumper J11 selects the use of on-board DC Regulator. If Unregulated Power input is used through Connector J12, Jumper J11 setting to 2-3 enables the regulator to regulate a usable voltage of 5VDC for LCEVB-SH1.

Either, if connector JP1 is used to used to provide direct power supply of regulated 5VDC to LCEVB-SH1, Jumper J11 setting should be at 1-2 (bypass the on-board regulator).

3.1.6.1 Connector J12 (Power-jet Connector)

General power-jet connector allowing unregulated power ($\geq$7.5V) to be supplied to LCEVB-SH1. Note that if this type of power supply is selected over the direct power supply (regulated) to JP1, J11 should be set to 2-3

3.1.6.2 Connector JP1 (Direct Power Connector)

Direct regulated Power supplying 5VDC to LCEVB-SH1. Note that J11 should be set to 1-2

3.1.7 Jumpers J15, J16, J17, and J18 (Serial Port Disconnects)

UART1 is dedicated by default to the Firmware Monitor. UART0 is unassigned, and usable by User Target System. 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 PB8, remove Jumper at J18 (1-2). To free PB9, remove Jumper at J17 (1-2).
To free PB10, remove Jumper at J16 (1-2). To free PB11, remove Jumper at J15 (1-2). This will normally disable serial communications between the LCEVB-SH1 and its host.

3.2 Serial Port Hard-wiring Options

As supplied, the LCEVB-SH1 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. It is not possible to support all configurations, but there is provision for Jumpers providing some additional signals. Active hardware control is not possible without additional hardware.

UCN1

Pin No.	Signal Name	Pin No.	Signal Name
1	PB14	2	PB15
3	GND	4	GND
5	GND	6	AD0
7	GND	8	AD1
9	GND	10	AD2
11	GND	12	AD3
13	GND	14	AD4
15	GND	16	AD5
17	GND	18	AD6
19	GND	20	AD7
21	GND	22	AD8
23	GND	24	AD9
25	GND	26	VCC
27	GND	28	AD10
29	GND	30	AD11
31	GND	32	AD12
33	GND	34	AD13
35	GND	36	AD14
37	GND	38	AD15
39	GND	40	A0
41	GND	42	A1
43	GND	44	A2
45	GND	46	A3
47	GND	48	A4
49	GND	50	A5

UCN3

Pin No.	Signal Name	Pin No.	Signal Name
1	GND	2	PA4
3	GND	4	PA5
5	GND	6	PA6
7	GND	8	PA7
9	GND	10	GND
11	GND	12	PA8
13	GND	14	PA9
15	GND	16	PA10
17	GND	18	PA11
19	GND	20	PA12
21	GND	22	PA13
23	GND	24	PA14
25	GND	26	PA15
27	GND	28	VCC
29	GND	30	CK
31	GND	32	-
33	VCC	34	-
35	UNMI	36	VCC
37	GND	38	WDTOVF
39	GND	40	-
41	GND	42	-
43	GND	44	-
45	GND	46	-
47	GND	48	-
49	GND	50	-

UCN2

Pin No.	Signal Name	Pin No.	Signal Name
1	GND	2	A6
3	GND	4	A7
5	GND	6	A8
7	GND	8	A9
9	GND	10	A10
11	GND	12	A11
13	GND	14	A12
15	GND	16	A13
17	GND	18	A14
19	GND	20	A15
21	GND	22	GND
23	GND	24	A16
25	GND	26	A17
27	VCC	28	A18
29	GND	30	A19
31	GND	32	A20
33	GND	34	A21
35	GND	36	CS0
37	GND	38	CS1
39	GND	40	CS2
41	GND	42	CS3
43	GND	44	PA0
45	GND	46	PA1
47	GND	48	PA2
49	GND	50	PA3

UCN4

Pin No.	Signal Name	Pin No.	Signal Name
1	No ADC	2	No ADC
3	GND	4	No ADC
5	GND	6	No ADC
7	GND	8	No ADC
9	GND	10	No ADC
11	GND	12	No ADC
13	GND	14	No ADC
15	GND	16	No ADC
17	GND	18	No ADC
19	GND	20	No ADC
21	GND	22	PB0
23	GND	24	PB1
25	VCC	26	PB2
27	GND	28	PB3
29	GND	30	PB4
31	GND	32	PB5
33	GND	34	PB6
35	GND	36	PB7
37	GND	38	GND
39	GND	40	PB8
41	GND	42	PB9
43	GND	44	PB10
45	GND	46	PB11
47	GND	48	PB12
49	GND	50	PB13

Table 2-7 SH7020/21 Connector Pinout

UCN1

Pin No.	Signal Name	Pin No.	Signal Name
1	PB14	2	PB15
3	GND	4	GND
5	GND	6	AD0
7	GND	8	AD1
9	GND	10	AD2
11	GND	12	AD3
13	GND	14	AD4
15	GND	16	AD5
17	GND	18	AD6
19	GND	20	AD7
21	GND	22	AD8
23	GND	24	AD9
25	GND	26	VCC
27	GND	28	AD10
29	GND	30	AD11
31	GND	32	AD12
33	GND	34	AD13
35	GND	36	AD14
37	GND	38	AD15
39	GND	40	A0
41	GND	42	A1
43	GND	44	A2
45	GND	46	A3
47	GND	48	A4
49	GND	50	A5

UCN3

Pin No.	Signal Name	Pin No.	Signal Name
1	GND	2	PA4
3	GND	4	PA5
5	GND	6	PA6
7	GND	8	PA7
9	GND	10	GND
11	GND	12	PA8
13	GND	14	PA9
15	GND	16	PA10
17	GND	18	PA11
19	GND	20	PA12
21	GND	22	PA13
23	GND	24	PA14
25	GND	26	PA15
27	GND	28	VCC
29	GND	30	CK
31	GND	32	-
33	VCC	34	-
35	UNMI	36	VCC
37	GND	38	WDTOVF
39	GND	40	-
41	GND	42	-
43	GND	44	-
45	GND	46	-
47	GND	48	-
49	GND	50	-

UCN2

Pin No.	Signal Name	Pin No.	Signal Name
1	GND	2	A6
3	GND	4	A7
5	GND	6	A8
7	GND	8	A9
9	GND	10	A10
11	GND	12	A11
13	GND	14	A12
15	GND	16	A13
17	GND	18	A14
19	GND	20	A15
21	GND	22	GND
23	GND	24	A16
25	GND	26	A17
27	VCC	28	A18
29	GND	30	A19
31	GND	32	A20
33	GND	34	A21
35	GND	36	CS0
37	GND	38	CS1
39	GND	40	CS2
41	GND	42	CS3
43	GND	44	PA0
45	GND	46	PA1
47	GND	48	PA2
49	GND	50	PA3

UCN4

Pin No.	Signal Name	Pin No.	Signal Name
1	AVCC	2	AVREF
3	GND	4	PC0
5	GND	6	PC1
7	GND	8	PC2
9	GND	10	PC3
11	GND	12	AVSS
13	GND	14	PC4
15	GND	16	PC5
17	GND	18	PC6
19	GND	20	PC7
21	GND	22	PB0
23	GND	24	PB1
25	VCC	26	PB2
27	GND	28	PB3
29	GND	30	PB4
31	GND	32	PB5
33	GND	34	PB6
35	GND	36	PB7
37	GND	38	GND
39	GND	40	PB8
41	GND	42	PB9
43	GND	44	PB10
45	GND	46	PB11
47	GND	48	PB12
49	GND	50	PB13

Table 2-7 SH7032/34 Connector Pinout

Section 4. Hardware Startup

4.1 Installing the LCEVB-SH1 Board

Installing the LCEVB-SH1 requires connecting the following:

- Serial communication cable to Host PC
- Power supply

4.2 Serial Communication Connection

Plug-in the Serial Communication Cable provided to LCEVB-SH1 and the Host PC Serial Communication Port (usually at the rear of the PC). LCEVB-SH1 System support the following Ports:

- COM1
- COM2
- COM3
- COM4

LCEVB-SH1 System will auto-detect the correct Port connection at HDI (Interface software) startup.

Ensure that Host PC is off when you are making the connection.

4.3 Power Supply Connection

Two type of Power supplies can be used on LCEVB-SH1, they are:

- Direct Power Supply
- Power-jet Power Supply

To select between Direct Power Supply or Power-jet Power Supply simply change the connection at Jumper J11. Jumper J11 (1-2) is the default setting at shipment (using direct power supply of 5VDC). The connection should not be left open.

Jumper setting (J11)	**Assignment**
1-2	Direct power supply of 5VDC, regulated (default)
2-3	Power-jet power supply of ≥7.5VDC, unregulated

Table 4-1 Jumper J11 setting

4.3.1 Direct Power Supply

The LCEVB-SH1 hardware uses a direct power supply of 5VDC (≈100mA) injecting into JP1 with J11 jumper set to 1-2 (bypass regulator).

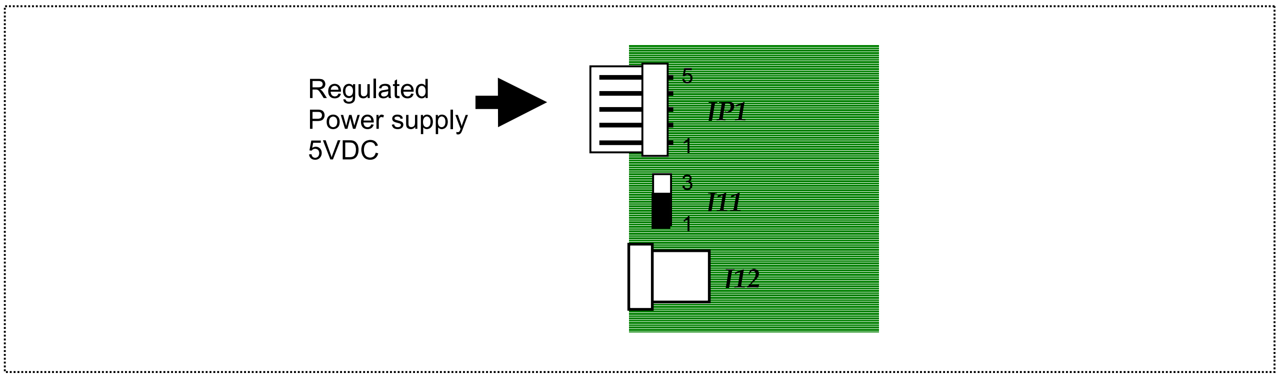


Figure 4-2 Direct Power supply to LCEVB-SH1

The pin assignment of JP1:

Pin	Assignment
1	+5VDC
2	NC
3	NC
4	NC
5	0V (GND)

Table 4-2 Pin assignment of JP1

4.3.2 Power-jet Power Supply

Alternatively, a power-jet supplying unregulated voltage (≥7.5VDC) to LCEVB-SH1 through J12. The power-jet configuration:

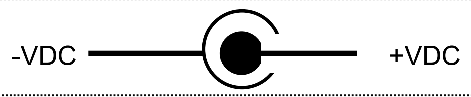


Figure 4-2 Power-jet configuration for LCEVB-SH1

The power-jet connection with the appropriate J11 setting of the jumper at 2-3 (using regulator):

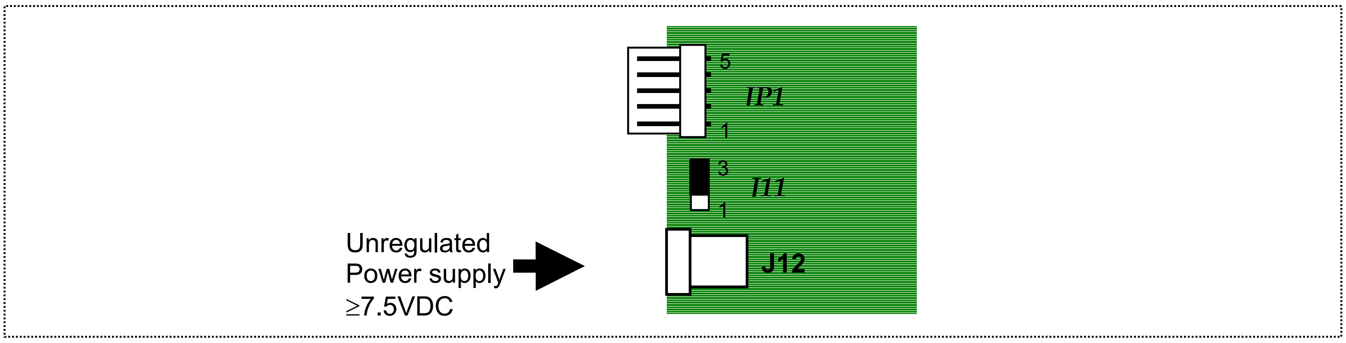


Figure 4-3 Power-jet Power supply for LCEVB-SH1

Since total power consumption can vary widely due to external connections, SH1 port state, and memory configuration, generally use a power supply capable of providing at least 500mA at +5VDC (regulated) ±5%.

Section 5. Software Startup

5.1 Host PC Computer Requirement

HDI is powerful yet easy to use MS-Window based Interface Program communicating with LCEVB-SH1 hardware system. For program development with the LCEVB-SH1 package, you will need a PC:

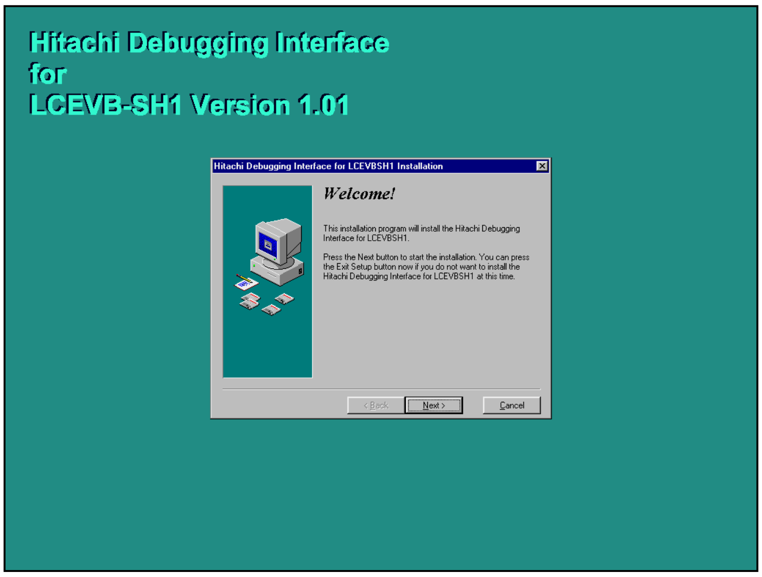
- Capable of hosting the Renesas's tools, editing files, and communicating with the evaluation board. The computer must be an i386 ,i486 or Pentium® standard PC running DOS 5.0 or higher with Windows 3.x or higher
- Memory of at least 8Mbytes is highly recommended
- CD-ROM drive is required since the supplementary tools and tutorials are distributed on CD-ROM
- Standard serial port must be available on host computer for communication with the LCEVB-SH1 system
- Harddisk is required (at least 6Mbyte of free space). The supplementary tools and tutorials files require a trivial amount of hard disk space
- Text editor capable of editing program source files without inserting non-printing characters in the file. An ASCII editor such as EDIT is acceptable.
- SuperH RISC engine C/C++ Compiler, Assembler, Optimizing Linkage Editor.
- Evaluation copy of these tools is available in the distribution CD-ROM.

5.2 HDI Installation

Install the HDI software from the CD-ROM is as follows:

- Startup Windows if it is not already running.
- Close all other applications that are running.
- Insert Installation CD-ROM into CD-ROM Drive
- Click (or Double-click) on the CD-ROM Drive to view the content of the CD-ROM
- Click (or Double-click) on the folder *HDI16* to view its content
- Execute *LSH1_101.EXE* by clicking (or Double-clicking).

The following Welcome! Screen will be displayed:


Hitachi Debugging Interface
for
LCEVB-SH1 Version 1.01

Hitachi Debugging Interface for LCEVBSH1 Installation

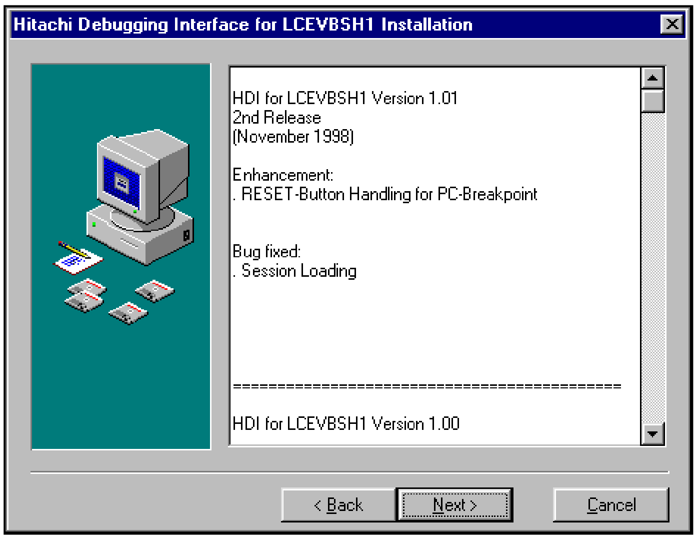
Welcome!

This installation program will install the Hitachi Debugging Interface for LCEVBSH1.

Press the Next button to start the installation. You can press the Exit Setup button now if you do not want to install the Hitachi Debugging Interface for LCEVBSH1 at this time.

Figure 5-1 Welcome Screen of HDI for LCEVB-SH1

- Click *Next* to proceed with the installation.
- Read the update information for any important information concerning the installation



HDI for LCEVBSH1 Version 1.01
2nd Release
(November 1998)

Enhancement:
. RESET-Button Handling for PC-Breakpoint

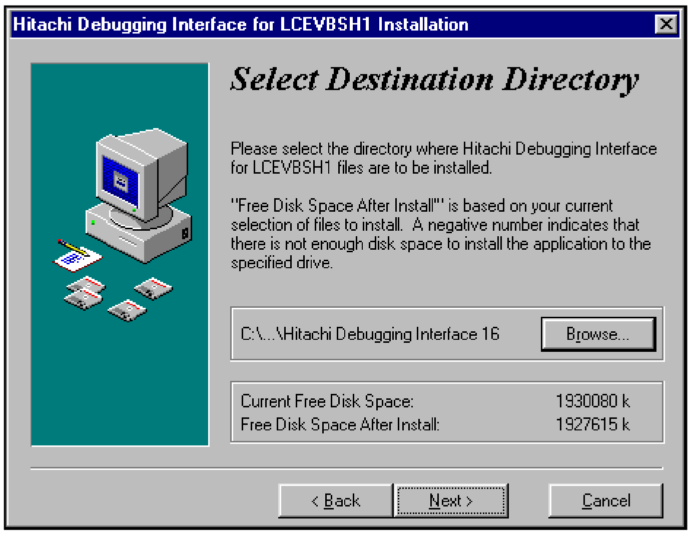
Bug fixed:
. Session Loading

==

HDI for LCEVBSH1 Version 1.00

Figure 5-2 Latest information on HDI for LCEVB-SH1

- Click `Next` to proceed
- Select which directory you wish to install HDI for LCEVB-SH1



Select Destination Directory

Please select the directory where Hitachi Debugging Interface for LCEVBSH1 files are to be installed.

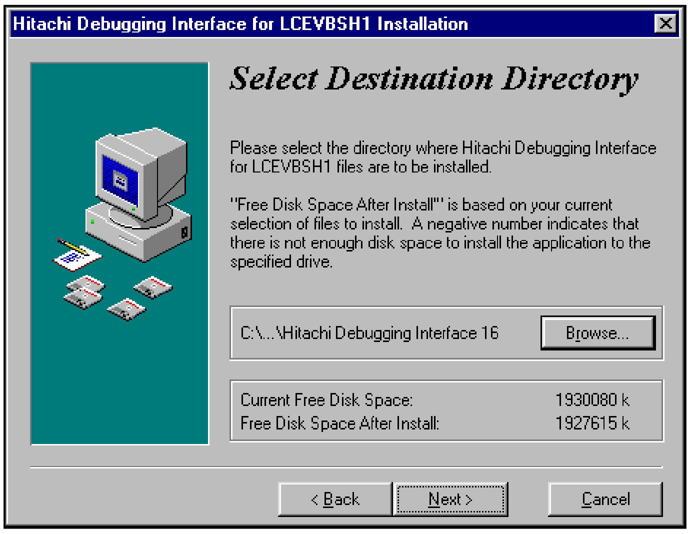
"Free Disk Space After Install"' is based on your current selection of files to install. A negative number indicates that there is not enough disk space to install the application to the specified drive.

C:\...\Hitachi Debugging Interface 16

Current Free Disk Space: 1930080 k
Free Disk Space After Install: 1927615 k

Figure 5-3 Selecting which Directory to install HDI for LCEVB-SH1

- Click *Browse* if you wish to change the default directory. The default directory is `C:\Program File\Hitachi Debugging Interface 16`, or specify an alternative directory and click `OK`.
- Click `Next` to proceed


Hitachi Debugging Interface for LCEVBSH1 Installation

Select Destination Directory

Please select the directory where Hitachi Debugging Interface for LCEVBSH1 files are to be installed.

"Free Disk Space After Install" is based on your current selection of files to install. A negative number indicates that there is not enough disk space to install the application to the specified drive.

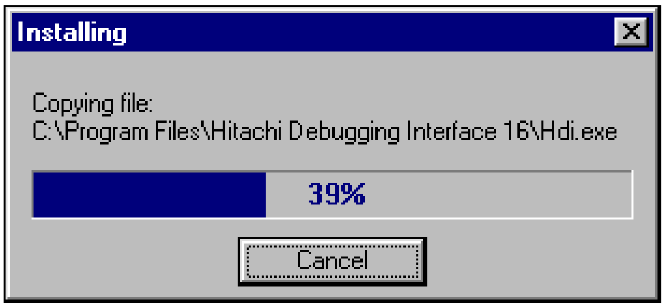
C:\...\Hitachi Debugging Interface 16

Current Free Disk Space:	1930080 k
Free Disk Space After Install:	1927615 k

Figure 5-4 Selecting which Group the HDI application should place its Icon

- The Program Group is where icons for HDI application will be. The default Program Group (ProgMan Group) is `Hitachi Debugging Interface 16`.
- Click `Next` to proceed
- Click *Next* in the Ready to Install Dialog-box to start installation

The installation will then copies the necessary HDI for LCEVB-SH1 files into the specified directory:


Installing

Copying file:
C:\Program Files\Hitachi Debugging Interface 16\Hdi.exe

39%

Figure 5-5 Installation Progress

Finally icons for HDI will be created into the Program Group specified earlier. The installer creates the following icons in the program group (*Start Menu\Programs*) you specified, by default HDI:

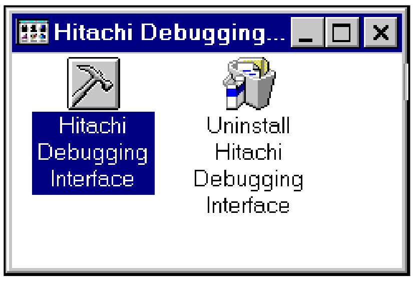

Hitachi Debugging...
Hitachi Debugging Interface
Uninstall Hitachi Debugging Interface

Figure 5-6 HDI Icons

These icons have the following functions:

• Hitachi Debugging Interface	: HDI for LCEVB-SH1
• Uninstall Hitachi Debugging Interface	: will remove HDI for LCEVB-SH1, and its associated files, if you need to uninstall it at any stage

5.3 Evaluation Compiler and Assembler Installation

Compiling/Assembling of SH Series Embedded codes need the following tools:

▪ Evaluation SuperH RISC engine C/C++ Compiler, Assembler, Optimizing Linkage Edition

Install these tools by executing *SETUP.EXE* in *EVAL* directory. A similar automated installation process will copy all the necessary files and setup the environment variables.

Note that User need to restart Host PC to complete this stage of installation.

Section 6. LCEVB-SH1 System Check

The next step is to run the HDI software to check that the LCEVB-SH1 System is working correctly. Follow the sequence below (ensure both the Serial Cable and Power Supply is properly connected up to LCEVB-SH1 before continuing):

- Switch on the LCEVB-SH1 and check that the red LED is illuminated.
- Select *Hitachi Debugging Interface* under the *Start Menu/Programs/Hitachi Debugging Interface* menu or Double-click the HDI icon:

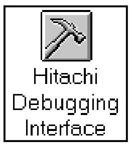
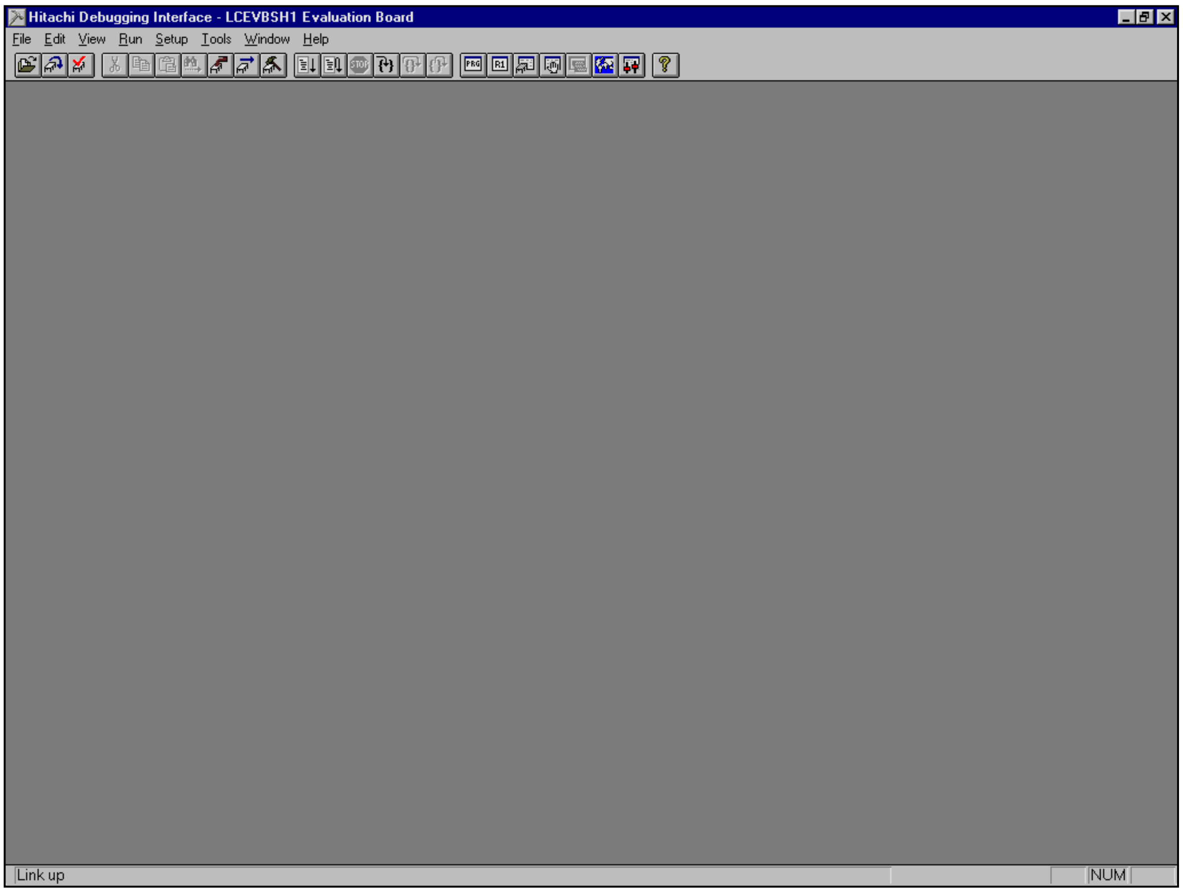


Figure 6-1 HDI Application Icon

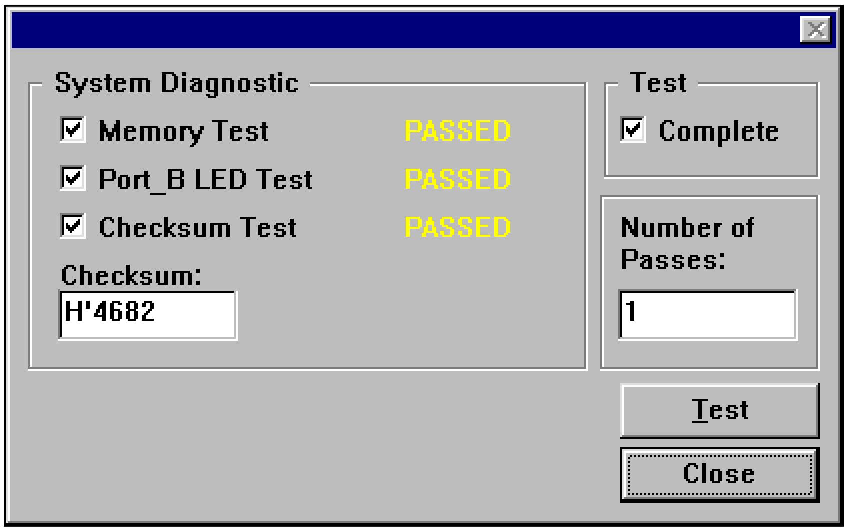
When everything is setup correctly the status bar will display *Link up* to indicate that everything is set up correctly, and the HDI screen will be displayed as shown below:


Hitachi Debugging Interface - LCEVBSH1 Evaluation Board
File Edit View Run Setup Tools Window Help
NUM

Link up

Figure 6-2 HDI startup Desktop

Perform the Diagnostic test by selecting *View/Diagnostic Window…* to verify the LCEVB-SH1 functionality:


System Diagnostic

☑ Memory Test PASSED

☑ Port_B LED Test PASSED

☑ Checksum Test PASSED

Checksum: H'4682

Test

☑ Complete

Number of Passes: 1

Figure 6-3 Diagnostic Window

The test performed above will cover certain features/functions of LCEVB-SH1:

1.	Memory Test	: Emulation Memory accessible
2.	Port_B LED Test	: Inter-active test with User observing the blinking of Green LED
3.	Checksum Test	: Ensure that the Firmware Monitor used is correct

Note that performing Diagnostic will cause the HDI to restart. Thus, never perform Diagnostic Test in-between Emulation process, as Emulation Memory content will be erased.

Section 7. Tutorial Session

7.1 Introduction

The following describes a sample debugging session, designed to introduce the main features of the LCEVB-SH1 Evaluation Board used in conjunction with the Hitachi debugging interface (HDI) software.

The tutorial is designed to run on the Emulation memory so that it can be used without connecting the LCEVB-SH1 to an external user system.

The tutorial is based on a simple C program.
Before reading this chapter:

- Setup the LCEVB-SH1 and verify that it is communicating correctly with HDI.
- Make sure you are familiar with the architecture and instruction set of SH1 before continuing. For more information refer to
- SH1/SH2 Series Programming Manual
- SH7034 or SH7021 Series Hardware Manual

7.1.1 Overview

This Tutorial program is an infinite loop that sort elements based on NAME in the alphabetical order, and AGE and ID in the ascending order. It is a simple C program to illustrate the capabilities of LCEVB-SH1 with simple C code.

7.1.2 Compiling/Assembling

The Tutorial is provided on the installation disk as the file tutorial.c. A workable version of the Tutorial should be compiled at User Host PC by executing *Tutorial.bat*, this needs the Evaluation SuperH RISC engine C/C++ Compiler, Assembler, Optimizing Linkage Edition (refer to section 5.3).

Note that the following files need customization:

- *Tutorial.bat*
- *Tutorial.sub*

Depending on the directory path where the Compiling/Assembling Tools are installed, the path of the respective files contains in these two files above need to be altered.

The final workable Tutorial should have *.ABS* as its file-extension (*Tutorial.abs*).

7.1.3 How the Tutorial Program Works

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

```
#include <machine.h>
#include "\CH38\INCLUDE\string.h"
```

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

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

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

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

int count;

void sort();
```

Now the main program.

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

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

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

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


                                            list[i] = list[k];
```

```
                                list[k] = worklist;
                        }
                        break;

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


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

    }
```

7.2 Execute HDI

To run the HDI select *Hitachi Debugging Interface* under the *Start Menu/Programs/Hitachi Debugging Interface* menu or double-click the Hitachi debugging interface icon:

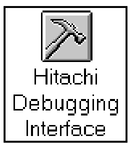


Figure 7-1 HDI Application Icon

7.2.1 Selecting the Target Platform

The HDI can be extended to support multiple target platforms, and if your system is set up for more than one platform you will first be prompted to choose a platform for the current session:

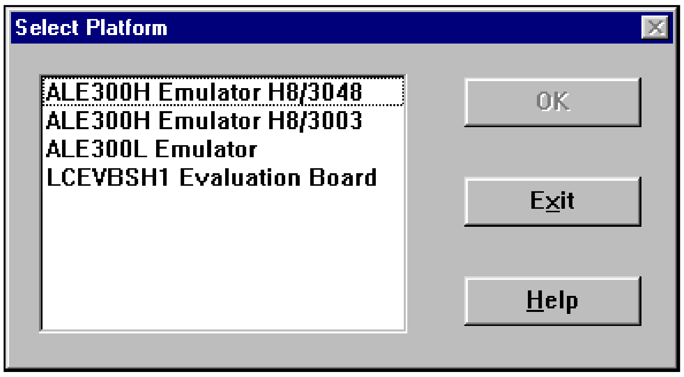


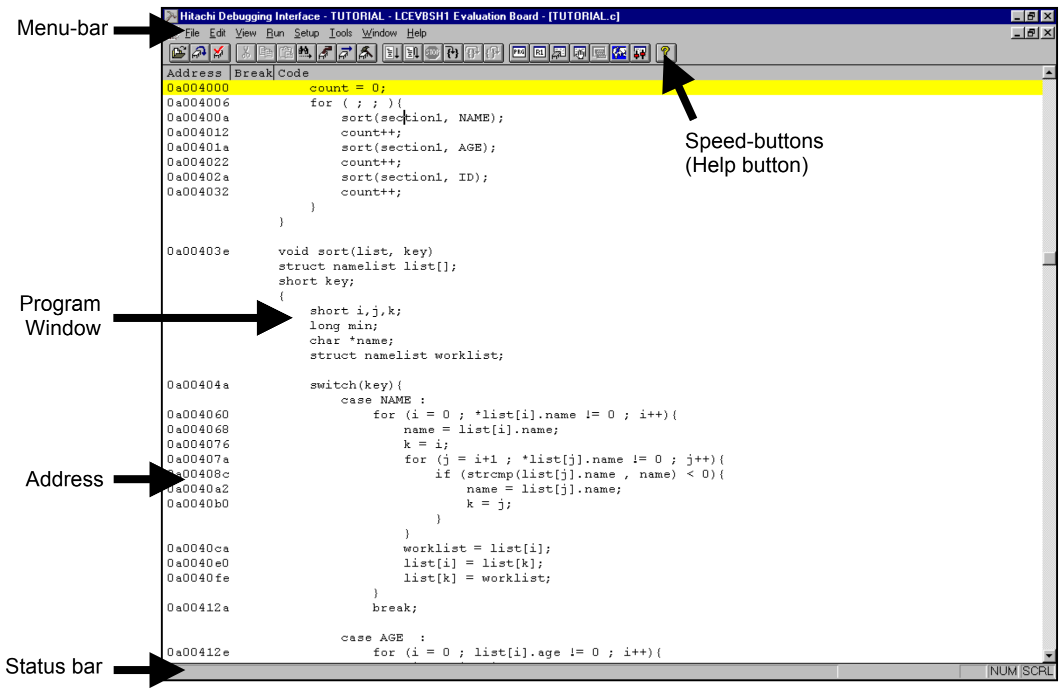
Figure 7-2 Select Platform

- For this tutorial select *LCEVBSH1 Evaluation Board* and click *OK* to continue.

Note that you can change the target platform at any time by choosing *Select Platform…* from the *Setup* menu.

Note that if you have only one platform installed this menu option will not be available.

When the emulator has been successfully set up the HDI Desktop window will be displayed, with the message Link up in the status bar.


Menu-bar
Program Window
Address
Status bar
Speed-buttons (Help button)
Hitachi Debugging Interface - TUTORIAL - LCEVBSH1 Evaluation Board - [TUTORIAL.c]
File Edit View Run Setup Tools Window Help
NUM SCRL

```
Address  Break Code
0a004000            count = 0;
0a004006            for ( ; ; ){
0a00400a                sort(section1, NAME);
0a004012                count++;
0a00401a                sort(section1, AGE);
0a004022                count++;
0a00402a                sort(section1, ID);
0a004032                count++;
                    }
                }

0a00403e        void sort(list, key)
                struct namelist list[];
                short key;
                {
                    short i,j,k;
                    long min;
                    char *name;
                    struct namelist worklist;

0a00404a            switch(key){
                        case NAME :
0a004060                    for (i = 0 ; *list[i].name != 0 ; i++){
0a004068                        name = list[i].name;
0a004076                        k = i;
0a00407a                        for (j = i+1 ; *list[j].name != 0 ; j++){
0a00408c                            if (strcmp(list[j].name , name) < 0){
0a0040a2                                name = list[j].name;
0a0040b0                                k = j;
                                }
                            }
0a0040ca                        worklist = list[i];
0a0040e0                        list[i] = list[k];
0a0040fe                        list[k] = worklist;
                        }
0a00412a                    break;

                        case AGE  :
0a00412e                    for (i = 0 ; list[i].age != 0 ; i++){
```

Figure 7-3 HDI Application Code Window

The key features of HDI are described in the following sections:

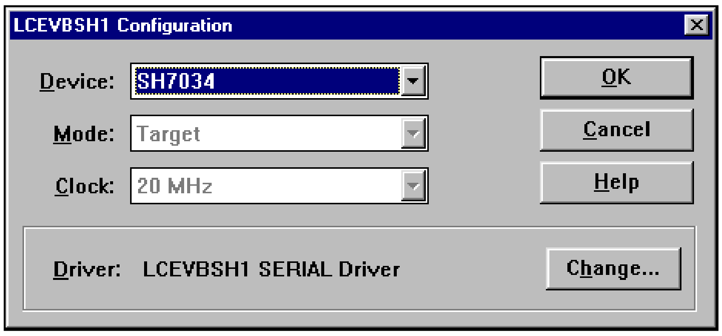
- **Menus** : Give you access to the HDI commands for setting up the SH1 Evaluation Board and using the HDI debugger.

- **Speed-buttons** : Provides convenient buttons as shortcuts for the most frequently used menu commands. The Help Button is the short-cut for *Help\Index*.
- **Program Window** : Displays the source of the program being debugged (Maximised Window in this case.
- **Address** : The address column show the respective absolute address of the C code.
- **Status Bar** : Displays the status of the LCVEB-SH1. For example, progress information about User Program downloads, Execution, etc.

7.3 Setup the Debugging Environment

Before downloading a program to the LCEVB-SH1, you need to set up the user system for your application. The Device type in LCEVB-SH1 Configuration Dialog-box needs to be setup. This leads to Device Memory Mapping selection for the respective device automatically. Take note that LCEVB-SH1 is a common tool for current SH1 Series. The configuration of this setup is for information during debugging.

To set up the target configuration choose *Configure Platform…* from the *Setup* menu.:


LCEVBSH1 Configuration

Device: SH7034

Mode: Target

Clock: 20 MHz

Driver: LCEVBSH1 SERIAL Driver

Figure 7-4 Target Configuration Dialog-box

- Setup the option as shown below.
- Click OK to change the target configuration.

Option	Value
Device	SH7034
Mode	Target

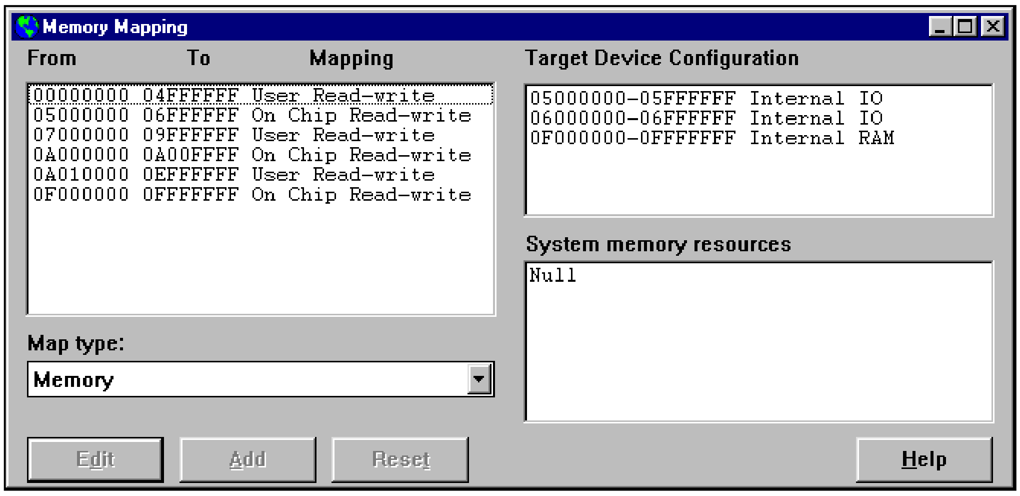
Table 7-1 Target Configuration option

7.4 Memory Mapping

The next step is to open the Memory Mapping Window, select *View* then *Memory Mapping Window*. Alternatively, click on the Mapping Speed-button:

Figure 7-5 Memory Mapping window Icon

The Memory Mapping window shown in the following figure is displayed:



```
00000000 04FFFFFF User Read-write
05000000 06FFFFFF On Chip Read-write
07000000 09FFFFFF User Read-write
0A000000 0A00FFFF On Chip Read-write
0A010000 0EFFFFFF User Read-write
0F000000 0FFFFFFF On Chip Read-write
```

```
05000000-05FFFFFF Internal IO
06000000-06FFFFFF Internal IO
0F000000-0FFFFFFF Internal RAM
```

```
Null
```

Figure 7-6 Memory Mapping Window

The *Target Device Configuration* shows the following:

Memory	Description
Reserved Area	MCU unused & reserved area
Internal IO	MCU on-chip Input / Output port registers
Internal RAM	MCU on-chip random access memory
Internal ROM	MCU on-chip read only memory

Table 7-2 Target Device Configuration Description

In the Memory Mapping Window, the following memory attribution description:

Access Type	Description
On-Chip Read-write	Emulation RAM memory.
On-Chip Read-only	Emulation ROM memory.
On-Chip Guarded	No access allowed.
User Read-write	External RAM memory.
User Read-only	External ROM memory.
User Guarded	No access allowed.

Table 7-3 Target Memory Attribute

7.5 User Target Program Downloading

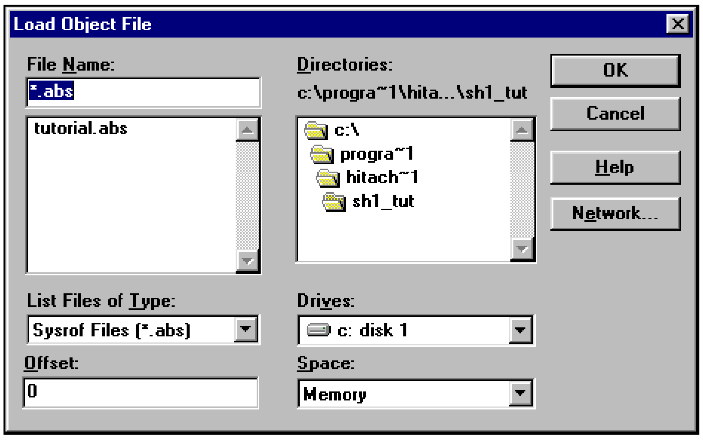
Once the LCEVB-SH1 is properly configured you can download the User Target Program (object program) you want to debug.

First load the SYSROF-format object file (file extension *.abs*), as follows:

- Choose *Load Program* from the *File* menu, or click the Load program speed-button in the toolbar.

Figure 7-7 File-open speed-button

- Select the file *tutorial.abs*, in the *\progra~1\hitach~1\sh1_tut* directory, and click OK.



Load Object File

File Name: *.abs

tutorial.abs

Directories: c:\progra~1\hita...\sh1_tut

c:\
progra~1
hitach~1
sh1_tut

List Files of Type: Sysrof Files (*.abs)

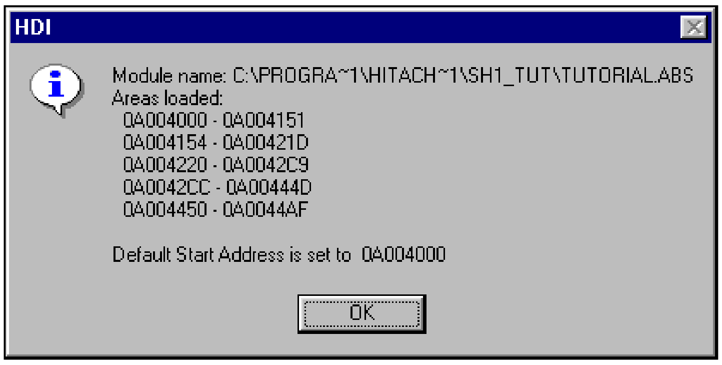
Drives: c: disk 1

Offset: 0

Space: Memory

Figure 7-8 File-open Dialog-box

When the file has been loaded the dialogue box shown in the following figure displays information about the memory areas that have been filled with the program code:



HDI

```
Module name: C:\PROGRA~1\HITACH~1\SH1_TUT\TUTORIAL.ABS
Areas loaded:
  0A004000 - 0A004151
  0A004154 - 0A00421D
  0A004220 - 0A0042C9
  0A0042CC - 0A00444D
  0A004450 - 0A0044AF

Default Start Address is set to  0A004000
```

Figure 7-9 User Target Program downloaded Information

Note that all the code lies within the on-chip ROM.

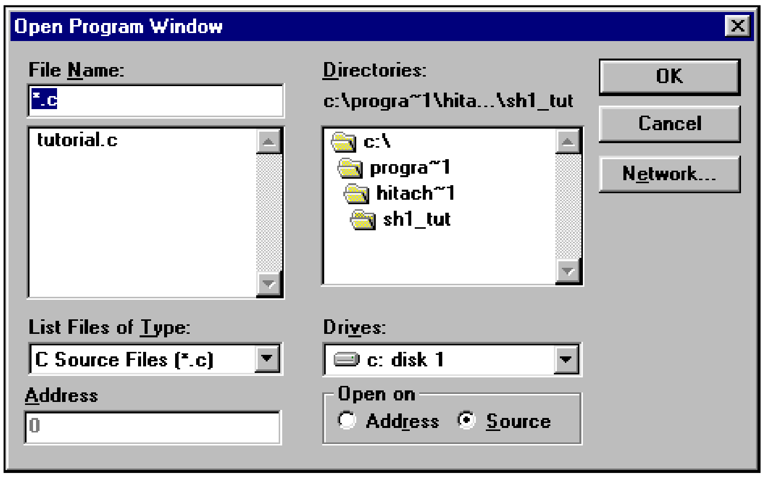
- Click *OK* to continue.

7.6 Displaying User Target Program code

HDI allows you to debug a program at source level, so that 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.

- Choose *Program Window…* from the *View* menu, or click the Program window button in the toolbar.

You will be prompted for the C source file corresponding to the object file you have loaded:



File Name: *.c

Directories: c:\progra~1\hita...\sh1_tut

tutorial.c

c:\
progra~1
hitach~1
sh1_tut

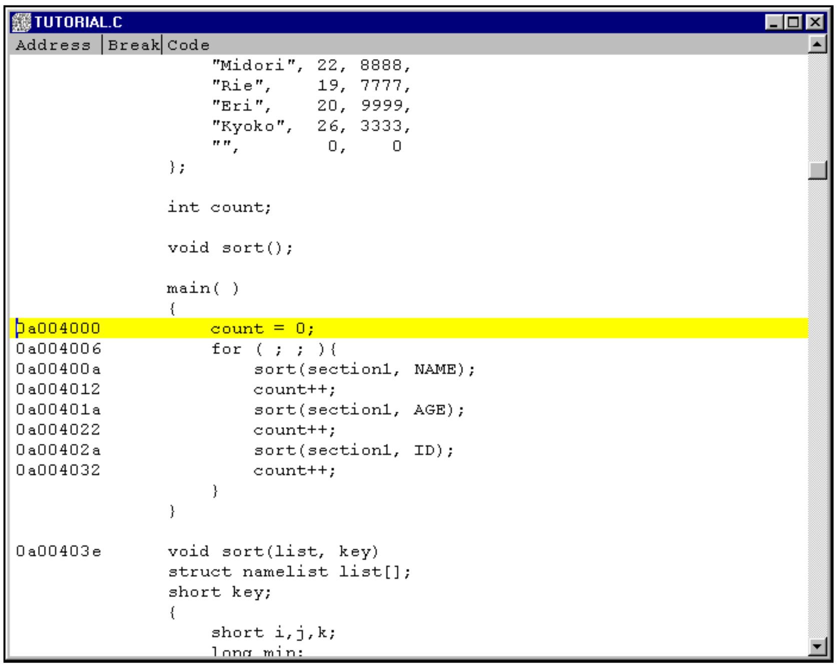
List Files of Type: C Source Files (*.c)

Drives: c: disk 1

Address: 0

Open on: ( ) Address [✓ Source]

- Select *tutorial.c* and click *OK* to display the program window:



```
                        "Midori",  22,  8888,
                        "Rie",     19,  7777,
                        "Eri",     20,  9999,
                        "Kyoko",   26,  3333,
                        "",         0,     0
                };

                int count;

                void sort();

                main( )
                {
0a004000                count = 0;
0a004006                for ( ; ; ){
0a00400a                    sort(section1, NAME);
0a004012                    count++;
0a00401a                    sort(section1, AGE);
0a004022                    count++;
0a00402a                    sort(section1, ID);
0a004032                    count++;
                        }
                }

0a00403e        void sort(list, key)
                struct namelist list[];
                short key;
                {
                        short i,j,k;
                        long min;
```

Figure 7-11 User Target Program Source Code window

- If necessary choose *Font* option from the *Customise* submenu on the *Setup* menu to choose a font and size suitable for your computer.

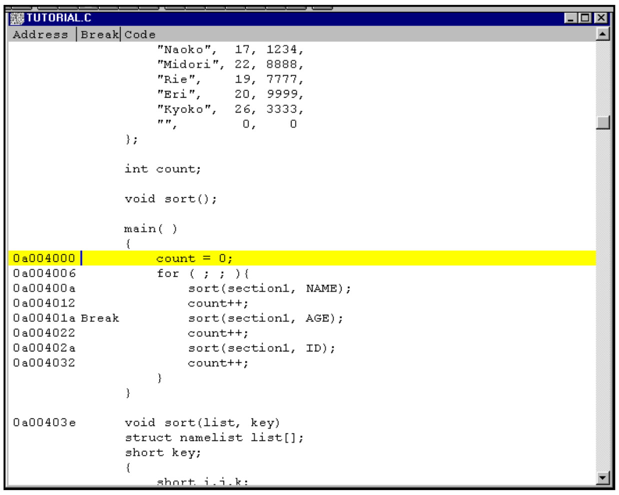
7.7 Using Breakpoint (PC Breakpoint)

The simplest debugging aid is the PC breakpoint (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.

7.7.1 Setting PC Breakpoint

The program window provides a very simple way of setting a PC breakpoint. For example, set a breakpoint at address H'0A00401A as follows:

- Double-click in the **Break** column on the line containing address H'0A00401A:



```
Address  Break Code
                   "Naoko",  17, 1234,
                   "Midori", 22, 8888,
                   "Rie",    19, 7777,
                   "Eri",    20, 9999,
                   "Kyoko",  26, 3333,
                   "",        0,    0
               };

               int count;

               void sort();

               main( )
               {
0a004000           count = 0;
0a004006           for ( ; ; ){
0a00400a               sort(section1, NAME);
0a004012               count++;
0a00401a Break         sort(section1, AGE);
0a004022               count++;
0a00402a               sort(section1, ID);
0a004032               count++;
                   }
               }

0a00403e       void sort(list, key)
               struct namelist list[];
               short key;
               {
                   short i,j,k;
```

Figure 7-12 Setting a PC Breakpoint

The word *Break* (at the Break column) will be displayed there to show that a program breakpoint is set at that address.

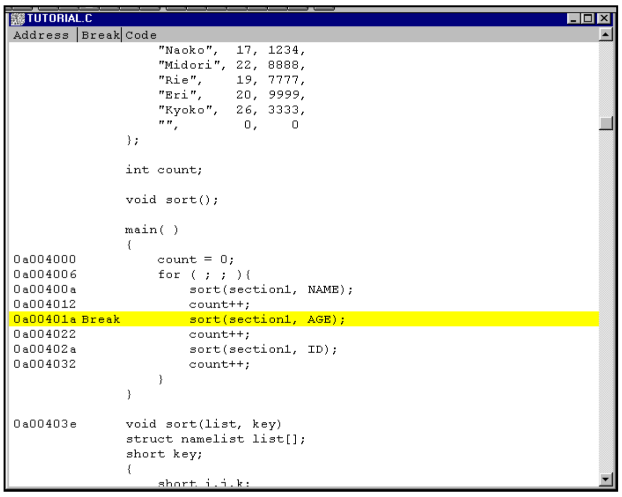
7.8 Executing User Target Program

To run the program from reset:

- Choose *Go Reset* from the *Run* menu, or click the *Go Reset* speed-button in the toolbar:

Figure 7-13 *Go Reset* speed-button

The program will be executed up to the breakpoint you inserted, and the statement will be highlighted in the program window to show that the program has halted



```
Address  Break Code
                       "Naoko",  17, 1234,
                       "Midori", 22, 8888,
                       "Rie",    19, 7777,
                       "Eri",    20, 9999,
                       "Kyoko",  26, 3333,
                       "",        0,    0
                   };

                   int count;

                   void sort();

                   main( )
                   {
0a004000               count = 0;
0a004006               for ( ; ; ){
0a00400a                   sort(section1, NAME);
0a004012                   count++;
0a00401a Break             sort(section1, AGE);
0a004022                   count++;
0a00402a                   sort(section1, ID);
0a004032                   count++;
                       }
                   }

0a00403e           void sort(list, key)
                   struct namelist list[];
                   short key;
                   {
                       short i,j,k;
```

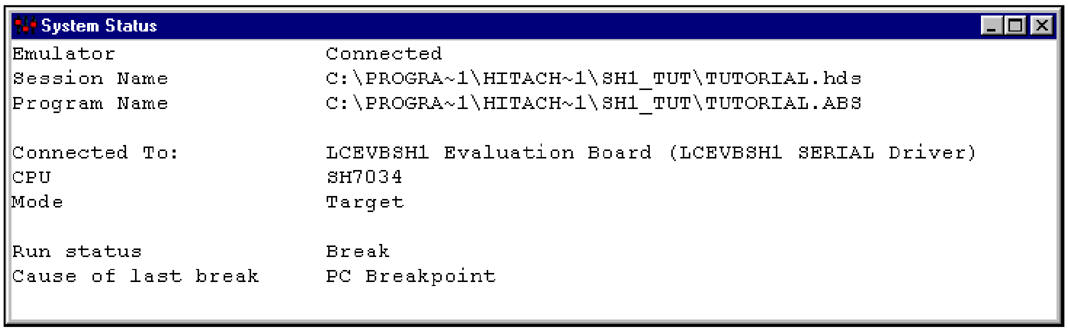
Figure 7-14 Program (PC) Break

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

You can also see the cause of the last break in the *System Status* window.

- Choose *Status Window* from the *View* menu, or click the Status window button in the toolbar:

Figure 7-15 Status speed-button



```
Emulator                 Connected
Session Name             C:\PROGRA~1\HITACH~1\SH1_TUT\TUTORIAL.hds
Program Name             C:\PROGRA~1\HITACH~1\SH1_TUT\TUTORIAL.ABS

Connected To:            LCEVBSH1 Evaluation Board (LCEVBSH1 SERIAL Driver)
CPU                      SH7034
Mode                     Target

Run status               Break
Cause of last break      PC Breakpoint
```

Figure 7-16 Status window

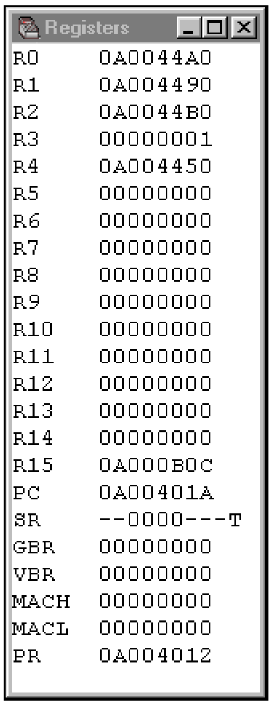
The *Cause of last break* line shows that the break was a Program (PC) Break.

7.9 Examining Registers

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

- Choose *Registers* window from the *View* menu, or click the Registers window speed-button in the toolbar:

Figure 7-17 Register speed-button



```
R0     0A0044A0
R1     0A004490
R2     0A0044B0
R3     00000001
R4     0A004450
R5     00000000
R6     00000000
R7     00000000
R8     00000000
R9     00000000
R10    00000000
R11    00000000
R12    00000000
R13    00000000
R14    00000000
R15    0A000B0C
PC     0A00401A
SR     --0000---T
GBR    00000000
VBR    00000000
MACH   00000000
MACL   00000000
PR     0A004012
```

Figure 7-18 Register window

As expected, the value of the program counter (PC), is the same as the highlighted statement, H'0A00401A. Note that during the execution of User Target Program, Emulation control is totally handed over to LCEVB-SH1 hardware. Thus, HDI has lose all control of the LCEVB-SH1 until it break (by PC breakpoint or by pressing RESET/NMI button) where control is once again back with HDI.

You can also change the MCU registers from the *Registers* window. For example, to change the value of the PC:

- Double-click PC in the *Register* window.

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

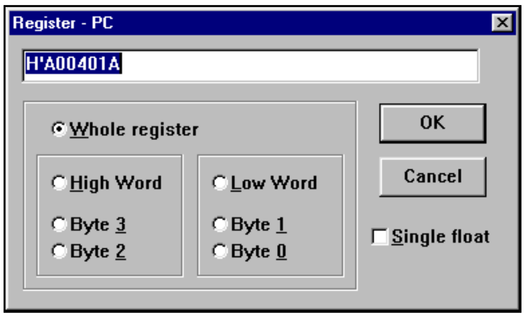


Figure 7-19 Register-PC Edit Dialog-box

- Edit the value to *H'0A004000*, the starting address of the User Target Program, and click *OK*.

The yellow-highlighted bar will move to the new address in the program window (*Tutorial.c*) to show current program counter value.

- Choose *Go* from the *Run* menu, or click the Go speed-button in the toolbar, to execute up to the breakpoint again.

Figure 7-20 Go speed-button

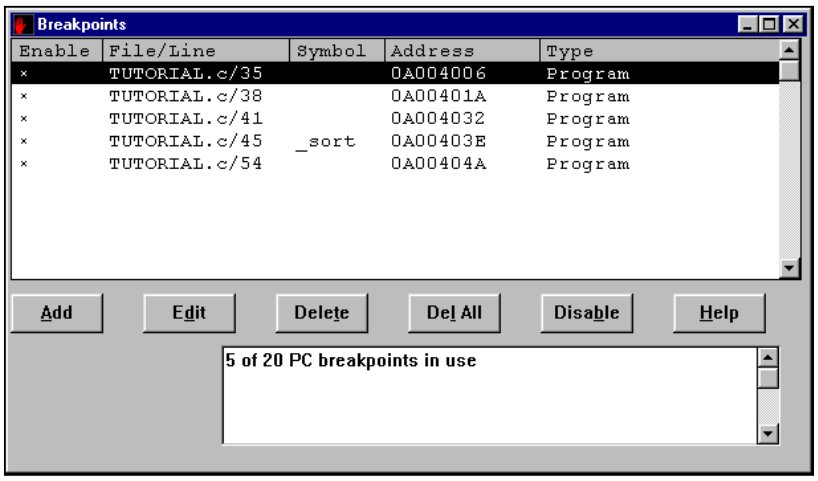
7.10 Reviewing PC Breakpoints

You can see a list of all PC breakpoints set in the program in the Breakpoint window.

- Choose *Breakpoint Window* from the *View* menu, or click the Breakpoint Window speed-button in the toolbar:

Figure 7-21 Breakpoint window speed-button

The Breakpoint window also allows you to enable and disable individual breakpoint, define new breakpoints, and delete breakpoints. Below shows the breakpoints set at several location besides address *H'0A00401A* previously set:

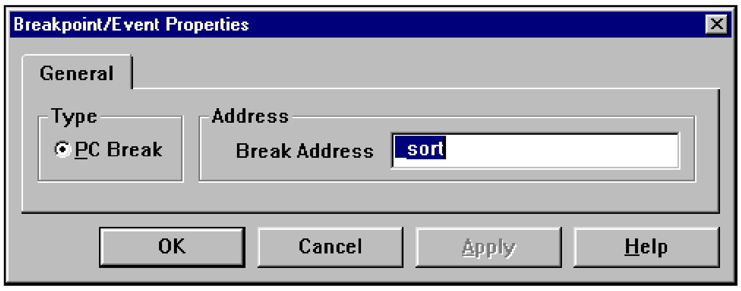


Enable	File/Line	Symbol	Address	Type
×	TUTORIAL.c/35		0A004006	Program
×	TUTORIAL.c/38		0A00401A	Program
×	TUTORIAL.c/41		0A004032	Program
×	TUTORIAL.c/45	_sort	0A00403E	Program
×	TUTORIAL.c/54		0A00404A	Program

```
5 of 20 PC breakpoints in use
```

Figure 7-22 Breakpoint window

Note that Symbol is actual address location. In another words, valid Labels/Symbols can be entered in the B *Break Address* entry:



Type: PC Break (selected)

Address — Break Address: sort

Figure 7-23 Breakpoint window Add Dialog-box

Before proceeding remove the breakpoint as follows:

- Highlight (move black-highlight bar) the breakpoint with address *H'0A00401A* and click *Delete*.
- Close the Breakpoint window.

7.11 User Memory Monitoring

You can monitor the behavior of User Target Program by:

- Examining content of the User memory
- Displaying variables' content used

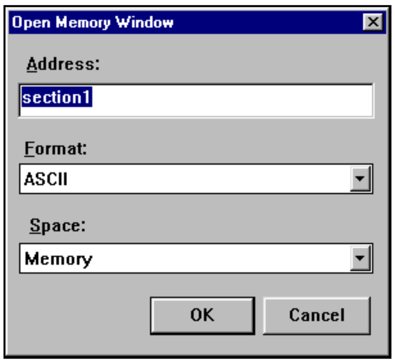
7.11.1 Viewing User Memory

You can view the contents of User Memory in the Memory window. For example, to view the memory corresponding to the array *section1* in ASCII:

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

Figure 7-24 Memory window speed-button

- Enter *section1* in the *Address* field, and set *Format* to ASCII.



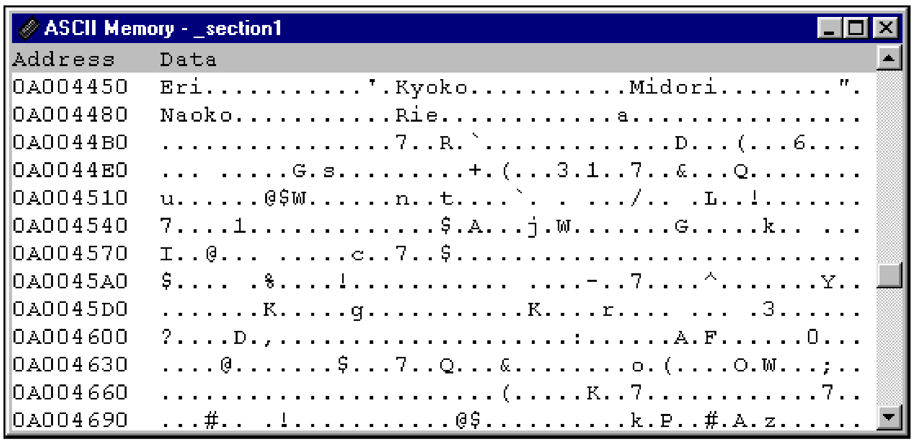
Address: section1

Format: ASCII

Space: Memory

Figure 7-25 Memory window Open Dialog-box

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



```
Address    Data
0A004450   Eri...........'.Kyoko...........Midori........".
0A004480   Naoko...........Rie.............a...............
0A0044B0   ................7..R.`.............D...(...6....
0A0044E0   ... .....G.s.........+.(...3.1..7..&...Q........
0A004510   u......@$W......n..t....`. . .../.. .L..!.......
0A004540   7....1.............$.A...j.W.......G.....k.. ...
0A004570   I..@... .....c..7..$............................
0A0045A0   $.... .%....!........... ....-..7....^........Y..
0A0045D0   .......K.....g...........K....r.... ... .3......
0A004600   ?....D.,.......................:......A.F......0...
0A004630   ....@.......$...7..Q...&.........o.(....O.W...;..
0A004660   ........................(.....K..7............7..
0A004690   ...#.. .!...........@$...........k.P..#.A.z......
```

Figure 7-26 Memory window in ASCII format start from address *section1*

- Leave the Memory window open so that you can monitor the contents of the array section1.

7.12 Watch Window

As you step through a program it is useful to be able to watch the Variables' content used change according. in your program, to verify that they change in the way that you expected. For example, set a watch on the structure variable *section1*, declared at the beginning of the program, using the following procedure:

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

```
            .
            .
            .
    main( )
    {
            count = 0;
            for ( ; ; )
            {
                    sort(section1, NAME);
                    count++;
            .
            .
            .
```

- Place the blinking cursor in the word *section1* in the program window.
- Click on the right mouse button in the program window to display a pop-up menu, and choose *Instant Watch…*:

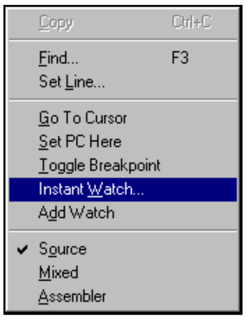
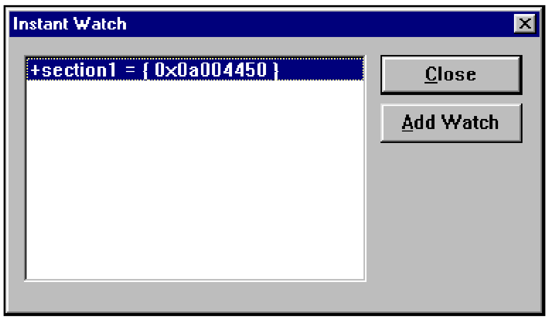


Figure 7-27 Program window Popup menu

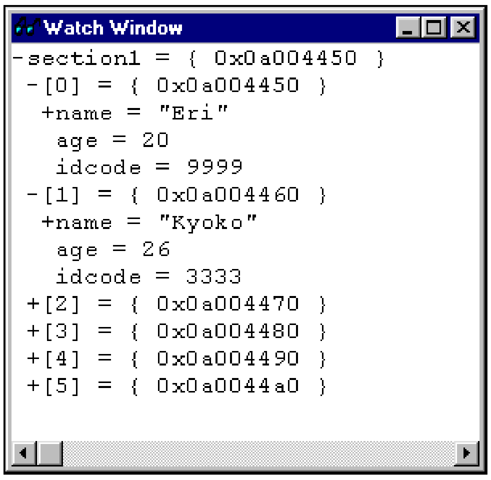
- The *Instant Watch* window will be displayed:



+section1 = { 0x0a004450 }

Figure 7-28 Instant Watch Dialog-box

- Click Add Watch to add the variable to the Watch window. Note, double-click on the Symbols (e.g. *section1*) with the '+' at the left of Symbol/Label will expand that Symbol/Label further if it has finer detail.



```
-section1 = { 0x0a004450 }
 -[0] = { 0x0a004450 }
  +name = "Eri"
   age = 20
   idcode = 9999
 -[1] = { 0x0a004460 }
  +name = "Kyoko"
   age = 26
   idcode = 3333
 +[2] = { 0x0a004470 }
 +[3] = { 0x0a004480 }
 +[4] = { 0x0a004490 }
 +[5] = { 0x0a0044a0 }
```

Figure 7-29 Watch window

You can also add a watch to the **Watch** window by specifying its name. Use this method to add a Watch on the variable count as follows:

- Click right mouse button in Watch window and choose *Add Watch…* from the pop-up menu.:

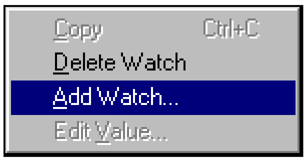
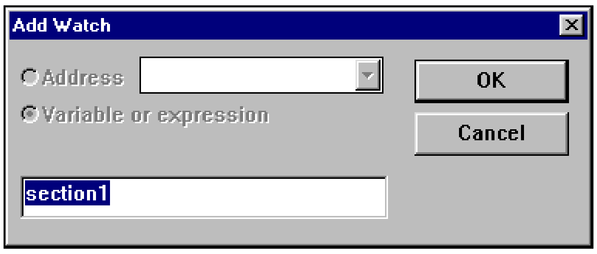


Figure 7-30 Watch window Popup menu

The **Add Watch** dialogue box appears:



section1

Figure 7-31 Watch window Add Watch Dialog-box

- Type the Symbol/Label to watch and click *OK* will show the similar display as shown in Figure 7-28.

7.12 Local Variable Watch

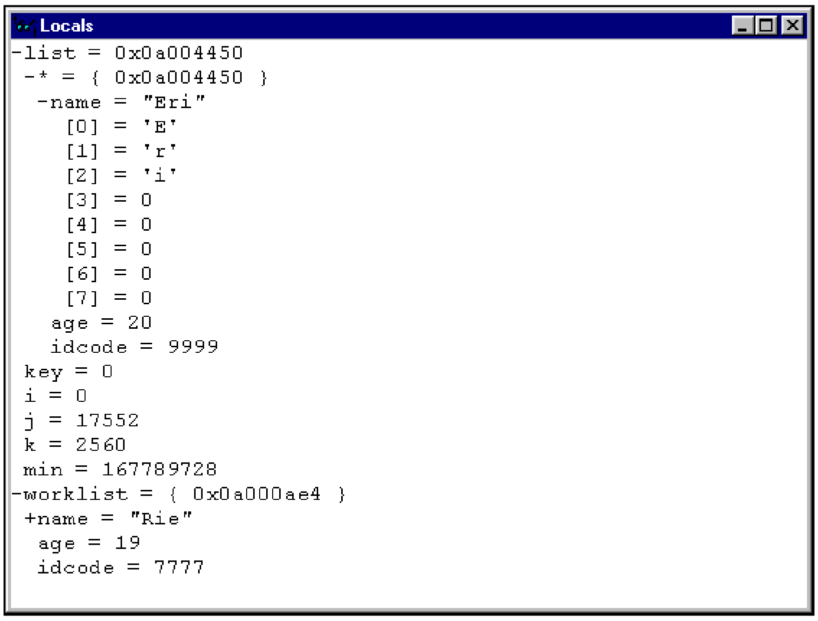
Local Variable window is treated different from Watch window. Content of Watch window is determinate by User, while content of Local Variable window will only appear once User Target Program execution reach function that has variables within the scope of the function.

You can watch variables local in a function using the Local Variables window. For example, we will examine the local variables in the function *sort*.

- Open the Locals window by choosing *Local Variable window* from the *View* menu.

Note again that the Locals window will be empty if the local variable declarations have not yet been executed.

The Local Variable window will now show the local variables with their values once execution has reach the function *sort*:



```
-list = 0x0a004450
 -* = { 0x0a004450 }
  -name = "Eri"
    [0] = 'E'
    [1] = 'r'
    [2] = 'i'
    [3] = 0
    [4] = 0
    [5] = 0
    [6] = 0
    [7] = 0
   age = 20
   idcode = 9999
 key = 0
 i = 0
 j = 17552
 k = 2560
 min = 167789728
-worklist = { 0x0a000ae4 }
 +name = "Rie"
  age = 19
  idcode = 7777
```

Figure 7-32 Local Variable window

- Double-click the '+' in front of the variable *worklist* in the Local Variable window to display the individual elements of the array *worklist*.

7.13 Stepping User Target Program

LCEVB-SH1 provides a range of options for single stepping through a program, executing an instruction or statement at a time. The stepping commands is listed below:

Command	Description
Step In	Executes every statement, including statements within functions.
Step…	Allows you to step repeatedly at a specified rate.

Table 7-4 Single Step Command

7.13.1 Single Step

Single step by selecting *Step In…* in *Run* menu or click on the Step-in speed-button at the Toolbar:

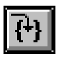


Figure 7-33 Step-in speed-button

- Observe that the PC (Register window) is incremented each time the Step-in speed-button is clicked. The PC always points to the current instruction to be executed by MCU. Note the yellow-highlight bar in the program window also indicate the current location of PC.

7.14 Save Session

Before exiting it is good practice to save your session, so that you can resume with the same ALE300L emulator and HDI configuration at your next debugging session.

- Choose **Save Session…** from the **File** menu.
- Choose **Exit from the File** menu to exit from HDI.

7.14 What Next?

This tutorial has introduced you to some of the key features of the ALE300L emulator, and their use in conjunction with the HDI. By combining the emulation tools provided in the ALE300L emulator you can perform extremely sophisticated debugging, allowing you to track down hardware and software problems very efficiently by precisely isolating and identifying the conditions under which they occur.

Section 8: Troubleshooting

Communication Problems	
1	Verify the communication channel
	• Is the Communication Port used by another device? • Right Communication Port (COM1/COM2) selected? • Right Communication Baud Rate (57,600bps) supported by Host PC
2	Power supply not switched on, or not connected, or connected loosely to the LCEVB-SH1. Check the power LED on the LCEVB-SH1.
3	The PC interface cable is not correctly connected between the PC interface board and the LCEVB-SH1.
5	Wrong PC interface/serial cable used? PC Interface /serial cable could be faulty? Verify that pins 2 and 3 of each end of cable are connected to each other respectively.
6	Is the target system drawing too much current? User Target System connected is drawing to much current.
No Power	
1	The fuse may have been blown due to mishandling such as shorting of VCC and Ground, or drawing of too much current from target system. Simply replace the 1A fuse located beside the main power switch. **Note:** Please investigates the cause before replacing the fuse.
Target system not working	
1	Check User voltage
	• Running at LCEVB-SH1 supply(5 Volt) Target may be drawing too much current from the LCEVB-SH1
2	Check the clock used to perform emulation
	• Only one of the on-board clock source can be used at any one time.

Appendix A: Frequently Asked Questions

This section contains a list of frequently asked questions about developing and evaluating programs using the LCEVB-SH1.

1. How do I write to the SH serial ports?
See the tutorial examples. Refer to the also Appendix C.

2. Why is RAM on the LCEVB-SH1 at such a high address?
The LCEVB-SH1 RAM starts at H'A000000. Since the monitor must reside at 0 (area 0) to provide for system boot-up, system RAM must be located somewhere else. Areas 0 and 2 share internal wait-state generation facilities, so area 2 is a logical place for RAM. Area 2 starts at location H'2000000, but to access it as word-wide, it must be addressed starting at H'A000000.

3. Can LCEVB-SH1 RAM be accessed at a different address? Why do I have trouble with the RAM at H'200000?
Conceptually, memory that is in SH area 2 is addressable at both address H'2000000 and H'A000000. The correct addressing considering the LCEVB-SH1's hardware memory configuration is starting at H'A000000.The LCEVB-SH1 RAM is word-wide. If you address it at H'2000000, you'll be telling the SH that it is byte-wide. In hardware terms, for purely 8-bit accesses, the SH signal LBS will never go low, so half the RAM memory on the board will not be used.

4. Why doesn't the monitor use SH on-board RAM?
On-board SH RAM is the fastest possible, being accessible in 32-bit chunks without wait states. Committing this RAM to the monitor might interfere with using this area for full-speed benchmarks.
Since monitor source is included with the LCEVB-SH1 kit, users are free to allocate onboard RAM for this use.

5. Why does the LCEVB-SH1 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.
- The monitor may work erratically if RAM is unavailable or only partly available, that is if one or both RAM ICs are not correctly plugged into the board.
- Make sure your power supply is more than sufficient for the needs of the LCEVB-SH1. 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. *When must I use HINT and when can I use, say, PROCOMM® with CMON?*
The only time you must use HINT is when you want to use the Save (“SV”) command. Most users won’t have any reason to upload an s-record representation of an LCEVB-SH1 memory region to a host computer, so this should be no problem. Otherwise, for all commands, you can use almost any terminal emulator.

9. *So what’s the problem with using Save (“SV”) and—if that’s a problem—how does Load work?*
The problem is specifying a filename on the host computer. Barring the use of a full-blown protocol like ZMODEM, there’s no standard way that CMON can inform the host of what file name to use for the data.
There is one somewhat awkward way of using SV with any terminal emulator, but this requires you to start up your terminal emulator’s ASCII capture before invoking SV in CMON. You’ll undoubtedly have to enter a file name.

Load works smoothly because you can invoke the “L” command in CMON, then start up your emulator’s ASCII download facility.

10. *Besides that, why use HINT?*
It would be very difficult to document setups for using CMON with every terminal emulator commonly used. By supplying an easy-to-use baseline terminal emulator, we can ask you “What happens when you use HINT” rather than trying to figure out your terminal emulator setup.

11. *Does an application program need to establish its own stack for proper operation on the LCEVB-SH1?*
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.

12. *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 SH 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).

13. *My benchmark shows that the SH 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-SH1 automatically inserts 4 wait states into area 0 and area 2 accesses. This will certainly make the SH run slower. You can adjust the value of the A02LW1 and A02LW0 bits in WCR3, consistent with the operating speed of the SH processor 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 B.1 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

Table B.1: Command Syntax Abbreviations

B.2 Commands Sorted Alphabetically

Available assembler instruction listing:

add #imm,Rn	cmp/ge Rm,Rn	jsr @Rn
add Rm,Rn	cmp/gt Rm,Rn	ldc Rn,GBR
addc Rm,Rn	cmp/hi Rm,Rn	ldc Rn,SR
addv Rm,Rn	cmp/hs Rm,Rn	ldc Rn,VBR
and #imm,R0	cmp/pl Rn	ldc.l @Rn+,GBR
and Rm,Rn	cmp/pz Rn	ldc.l @Rn+,SR
and.b #imm,@(R0,GBR)	cmp/str Rm,Rn	ldc.l @Rn+,VBR
bf disp8	div0s Rm,Rn	lds Rn,MACH
bra disp12	div0u	lds Rn,MACL
bsr disp12	div1 Rm,Rn	lds Rn,PR
bt disp8	exts.b Rm,Rn	lds.l @Rn+,MACH
clrmac	exts.w Rm,Rn	lds.l @Rn+,MACL
clrt	extu.b Rm,Rn	lds.l @Rn+,PR
cmp/eq #imm,R0	extu.w Rm,Rn	mac.w @Rm+,@Rn+
cmp/eq Rm,Rn	jmp @Rn	mov #imm,Rn
mov Rm,Rn	mov.w @Rm,Rn	shlr8 Rn
mov.b Rm,@(R0,Rn)	mov.w R0,@(disp,Rm)	sleep
mov.b Rm,@-Rn	mov.w R0,@(disp,GBR)	stc GBR,Rn
mov.b Rm,@Rn	mova @(disp,PC),R0	stc SR,Rn
mov.b @(disp,Rm),R0	movt Rn	stc VBR,Rn
mov.b @(disp,GBR),R0	muls Rm,Rn	stc.l GBR,@-Rn

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

B.3.3 Logical

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

B.3.4 Shift/Rotate

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

B.3.5 Branches

bf disp8	jmp @Rn
bt disp8	jsr @Rn
bra disp12	rts
bsr disp12	

B.3.6 System Control

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

Appendix C: Mini-project

This section contains the schematic for a mini DC motor project that can be constructed and connected with the LCEVB-SH1. The purpose of this project is to test the PWM function of the SH1 and also it's Input and output pin.

Also provided is the SH1 C program required to run it. This C program can be compiled using the SuperH RISC engine C/C++ Compiler, Assembler, Optimizing Linkage Editor. The Object file will then have to be converted to S-record using the H series Linker and the S record converter before it can be downloaded to the LCEVB-SH1. When invoking the Linker, please take note that user program can only start at H'A004000. Thus it is required to specified the start address of the program section. The C source program of the Motor-control can be found in project directory.

C.1 Bill of Material

Item	Quantity
• Push Button switch (normally open)	02
• DC Motor	01
• 7 segment Display (Common anode)	01
• Resistor 330Ω • 470Ω • 1.2KΩ • 3.9KΩ • 22KΩ	07 01 01 02 02
• Capacitor 0.1μF	02
• Transistor 2N2222A • BC131	01 01
• Diode 1N4001	01
• LED (RED) • (GREEN)	01 01
• 74HC04	01
• Buzzer	01

Table E.1: Bill of Material for Mini Project

C.2 Operation

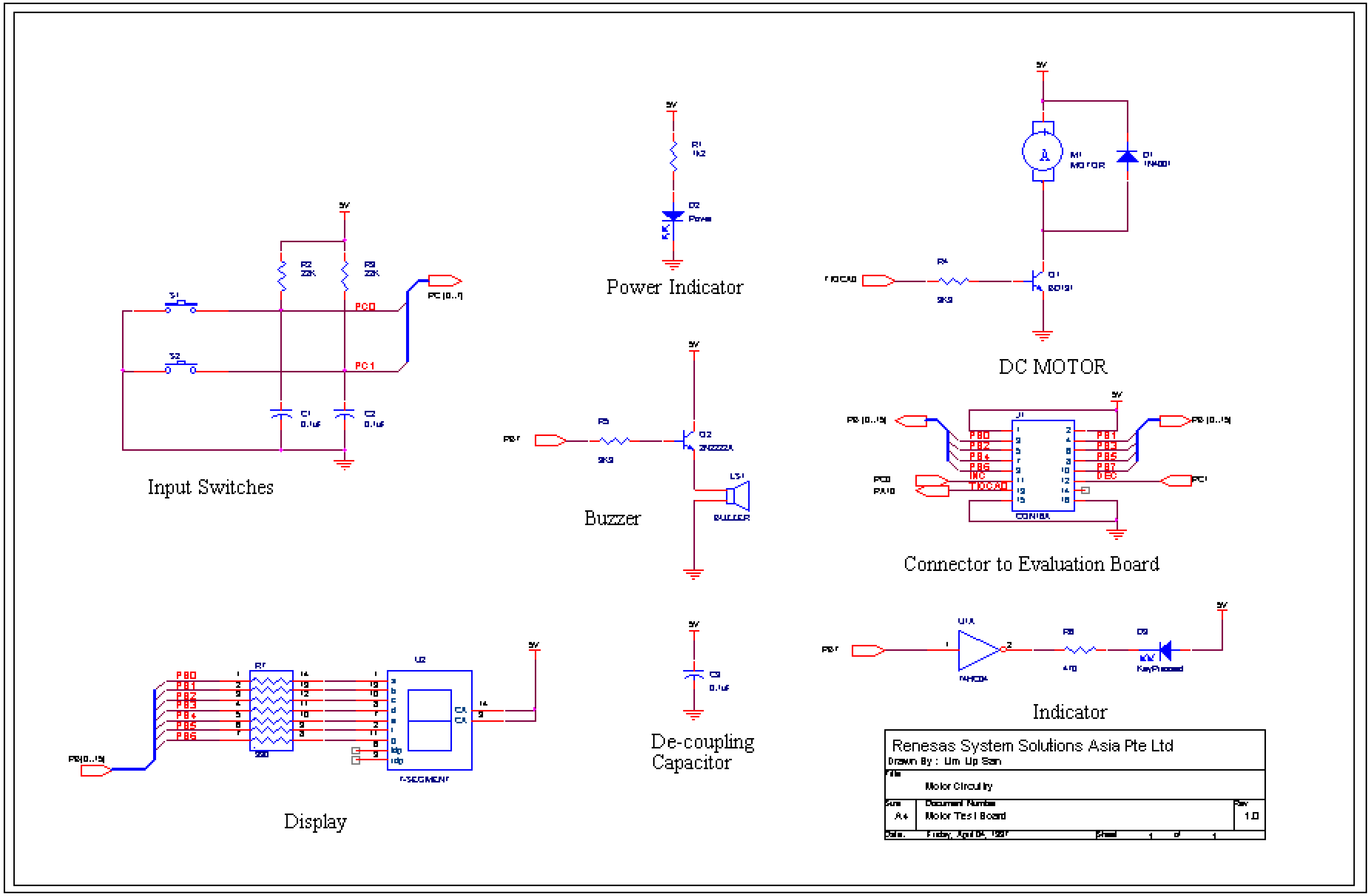
The program starts out by displaying 1 to 10 on the 7 segment display. After which depressing S10 will increment the 7 segment display by 1 to a maximum of 5 and depressing S11 will decrement the display by 1 to a minimum of 0. Respectively the duty cycle of the PWM driving the motor will increase or decrease by 1% starting from 4% to a maximum of 8% corresponding to the value displayed . Depressing both S10 and S11 causes motor to stop.

C.3 Software Listing

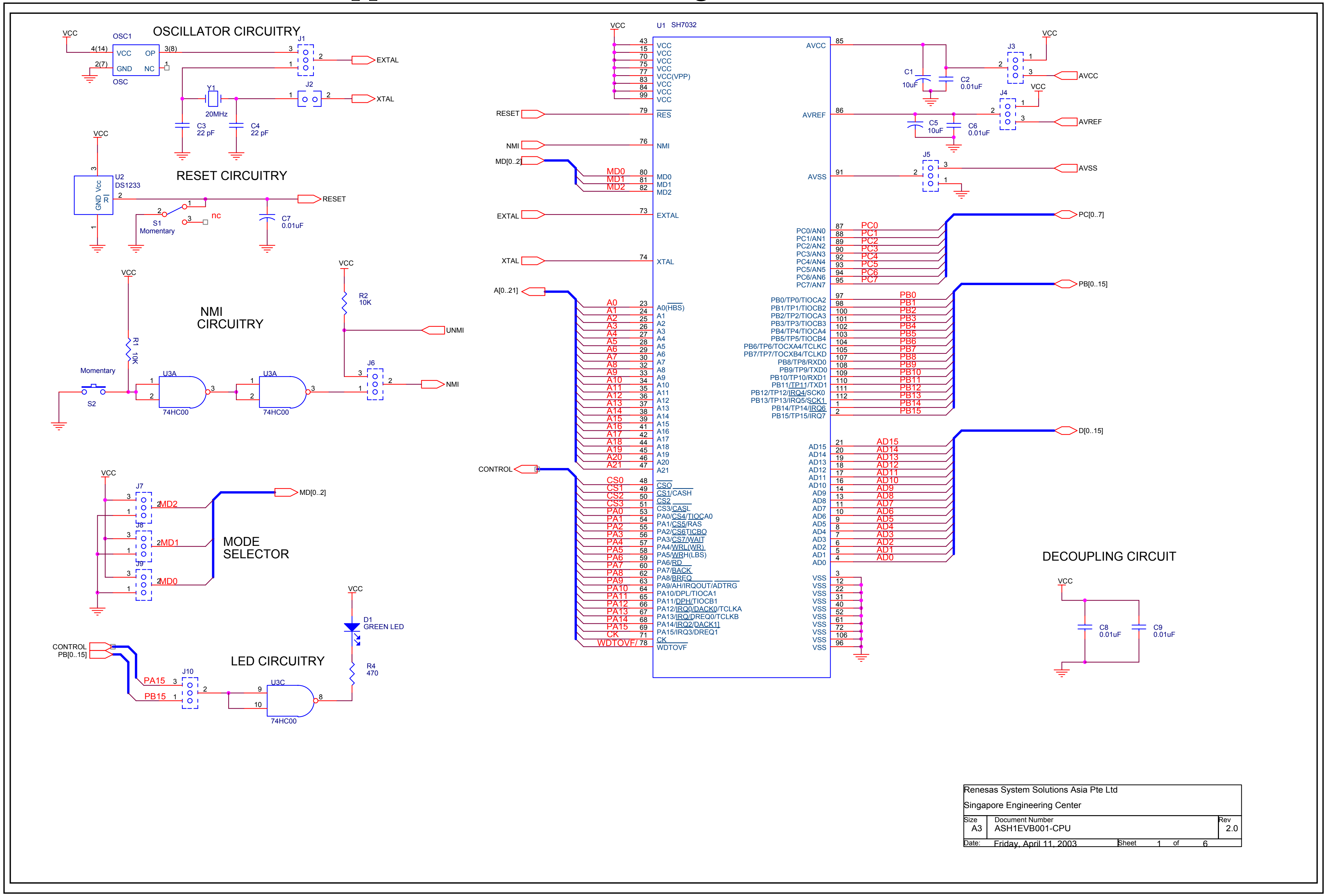
```
/*  -----------------------------------------------------------------
    Program Name    : motor.c
    Date            : 20 Mar 97
    ----------------------------------------------------------------- */

#include <machine.h>
#include <iosh7030.h>

/*  -----------------------------------------------------------------
                                Constants
    ----------------------------------------------------------------- */

#define delay_constant 80160            /* delay of 0.1 sec */

unsigned char display_table[11]={0x40,0x79,0x24,0x30,0x19,0x12,
                                 0x02,0x78,0x00,0x18,0x7F};

unsigned short pwm_table[11]={0,160,200,240,280,320};   /* 0,4,5,6,7,8 */

/*  -----------------------------------------------------------------
                        Global Variables
    ----------------------------------------------------------------- */

unsigned char motor_speed=0;            /*motor speed*/
unsigned char past_motor_speed=0;       /*previous motor speed*/

main()  /* main program */
{

        init_ports();
        init_timers();
        count();

        while (1)       /* infinite loop */
        {
                get_data();
                out_motor();
        }

}               /* End Main */


/* ----------------------------------------------------------------
        Routine : out_motor()
        Purpose : output to motor
        Note    : kick starts stationary motor
   -------------------------------------------------------------- */

out_motor()
{
        if ( (past_motor_speed == 0) && (motor_speed == 1) )
        {
                ITU_GRB1 = 600; /* kick-starts */
                delay20ms();
        }

        ITU_GRB1 = pwm_table[motor_speed];      /* o/p to motor */

}
```

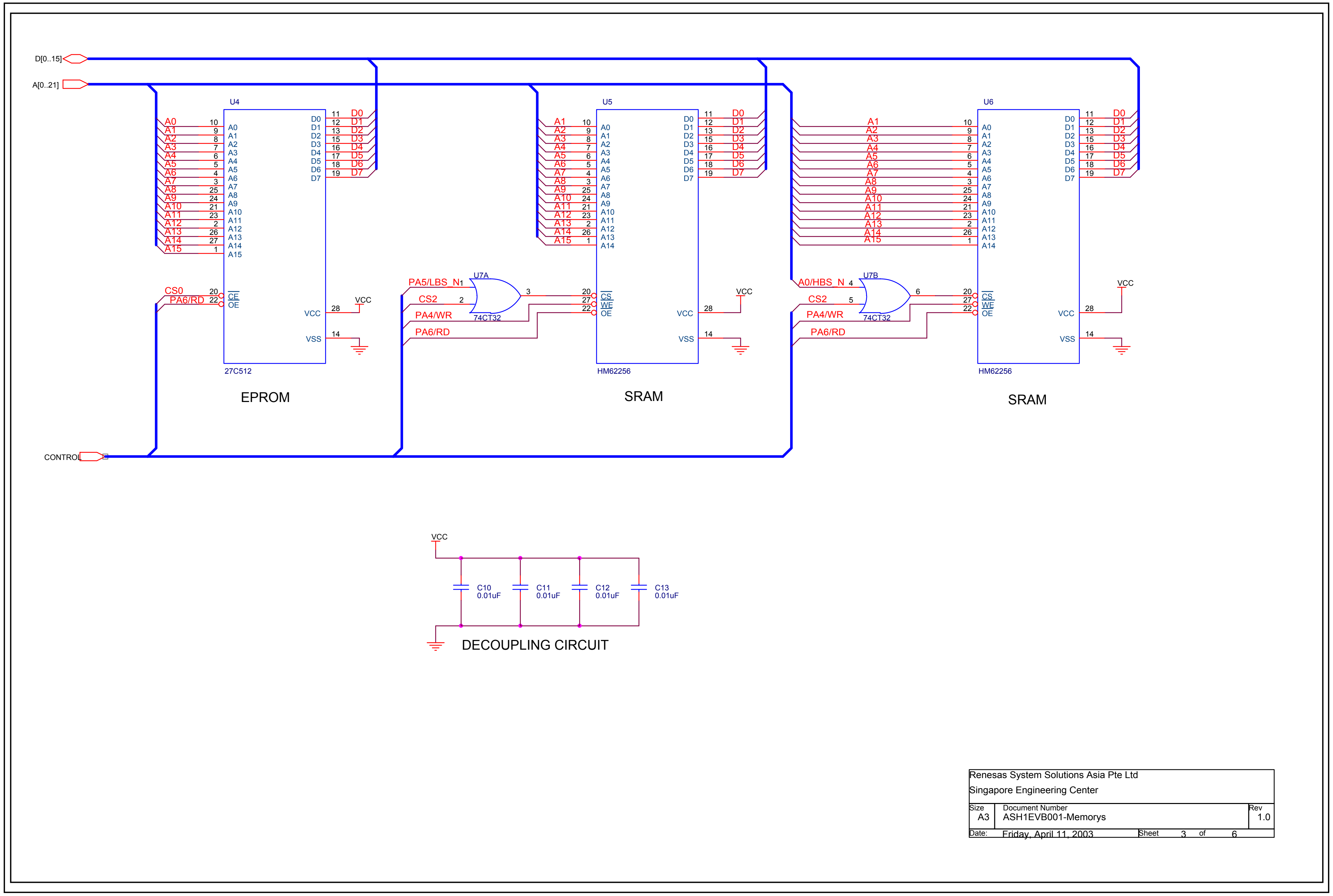
```
/* ----------------------------------------------------------------
        Routine : get_data()
        Purpose : read PC0 & PC1 using software debouncing
   --------------------------------------------------------------- */

get_data()
{
        unsigned char temp,temp1,temp2;
        unsigned char stable = 1;

        while(stable)
        {
                temp = (PCDR & 0x03);

                delay20ms();

                temp1 = (PCDR & 0x03);

                delay20ms();

                temp2 = (PCDR & 0x03);

                if ( (temp == temp1) && (temp1 == temp2) && (temp != 0x03) )
                {
                        stable = 0;
                        beep();
                }
        }

        past_motor_speed = motor_speed; /* update past_motor_speed */

        switch(temp)
        {
                case 0x00:      /* both switches are ON */

                        motor_speed = 0;
                        break;

                case 0x02:      /* PC0 is ON */

                        if (motor_speed >= 5)
                        {
                                motor_speed = 5;
                        }
                        else
                        {
                                motor_speed++;
                        }

                        break;

                case 0x01:      /* PC1 is ON */

                        if (motor_speed <= 1)
                        {
                                motor_speed = 0;
                        }
                        else
                        {
                                motor_speed--;
                        }

                        break;

                default:
                        motor_speed = 0;
                        break;

        }       /* End Switch */

        PBDR = ( (PBDR & 0xFF80) | display_table[motor_speed] );

        while ( temp == (PCDR & 0x03) ) /* check for switch to be released */
        {
                delay20ms();
        }

        PBDR = (PBDR  & 0xFF7F);        /* off buzzer */
}               /* End get_data() */


/* -----------------------------------------------------------------
        Routine : count()
        Purpose : display 0 to 9 at 0.2 second interval
   --------------------------------------------------------------- */

count()
{
        int     display_count;

        for (display_count = 0 ; display_count < 11 ; display_count++)
        {
                PBDR = display_table[display_count];
                d_xx(10);       /* delay = 10 x 20ms = 0.2 sec */
        }
}

/* -----------------------------------------------------------------
        Routine : beep()
        Purpose : set PB7 => on buzzer
   --------------------------------------------------------------- */

beep()
{
        PBDR = (PBDR | 0x0080); /* set PB7 */
}

/* -----------------------------------------------------------------
        Routine : delay20ms()
        Purpose : generate a software delay of 20ms
   --------------------------------------------------------------- */

delay20ms()
{
        int     dvar1;

        for (dvar1 = 0 ; dvar1 < 16000 ; dvar1 ++ );

}

/* -----------------------------------------------------------------
        Routine : d_xx(count)
        Purpose : generate a delay in multiples of 20ms using ITU_TCNT0
   --------------------------------------------------------------- */

d_xx(count)
int count;
{
        int delay_var;
        unsigned char temp;

        ITU_TCNT0 = 0;                  /* starts counting from 0 */
        ITU_TSTR = (ITU_TSTR | 0x01);   /* starts ITU_TCNT0 */

        for (delay_var = 0 ; delay_var <= count ; delay_var ++)
        {
                while ( (temp = (ITU_TSR0 & 0x01) ) == 0)
                /*check for IMFA = 1*/
                {
                }

                ITU_TSR0 = (ITU_TSR0 & 0xF8);   /* IMFA flag = 0 */
        }
        ITU_TSTR = (ITU_TSTR & 0xFE);   /* stop ITU_TCNT0 */

        return(0);
}

/* -----------------------------------------------------------------
        Routine : delay(count)
        Purpose : generate a software delay in multiples of 20ms
   -------------------------------------------------------------- */

delay(count)
int count;
{
 int delay_var,ddd;

        for (ddd=0 ; ddd != count ;ddd ++)
        {
                for (delay_var=0 ; delay_var<=delay_constant ; delay_var++);
        }

 return(0);
}

/* ----------------------------------------------------------------
        Routine : init_ports()
        Purpose : initialize Ports A and B
        Port Configuration :
        a. PA10 : TIOCA0
        b. PB0 <-> PB6 : output to drive 7-segment LED
   -------------------------------------------------------------- */

init_ports()
{
        PFC_PACR1 = 0x0020;   /* PA15 <-> PA8 except PA10: general purpose I/O */
        PFC_PACR2 = 0xBF55;   /* PB7  <-> PB0 : general purpose I/O */
        PFC_PAIOR = 0xFFFF;   /* PB15 <-> PB0 : all output */
        PFC_PBCR1 = 0x0000;   /* PB15 <-> PB8 : general purpose I/O */
        PFC_PBCR2 = 0x0000;   /* PB7  <-> PB0 : general purpose I/O */
        PFC_PBIOR = 0xFFFF;   /* PB15 <-> PB0 : all output */
}

/* ----------------------------------------------------------------
        Routine : init_timers()
        Purpose : initialize timers 0 & 1
   -------------------------------------------------------------- */

init_timers()
{
        ITU_TSNC = 0x00;        /* All channels to function independently */
        ITU_TMDR = 0x82;        /* TCNT0 : normal operation */
                                /* TCNT1 : PWM operation */

        ITU_GRA1 = 4000;        /* base = 4000 */
        ITU_GRB1 = 0;

        ITU_TCR0 = 0xA3;
        /*      ITU_TCNT0 is cleared by output compare match A
                Select internal clock /8
                1 period = 8/20 MHz = 400 ns */

        ITU_TIOR0 = 0x88;
        /*      Select output compare operation with pin output disabled */

        ITU_GRA0 = 50000;       /* 50000 x 400 ns = 20 ms */
        ITU_TCR1 = 0x23;        /* TCNT1 is cleared by GRA compare match */
        ITU_TSTR = (ITU_TSTR | 0x02);   /* starts ITU_TCNT1 */
}

exit()
{
        trapa(0);
}

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

Appendix D: Schematic Diagram of Mini-project

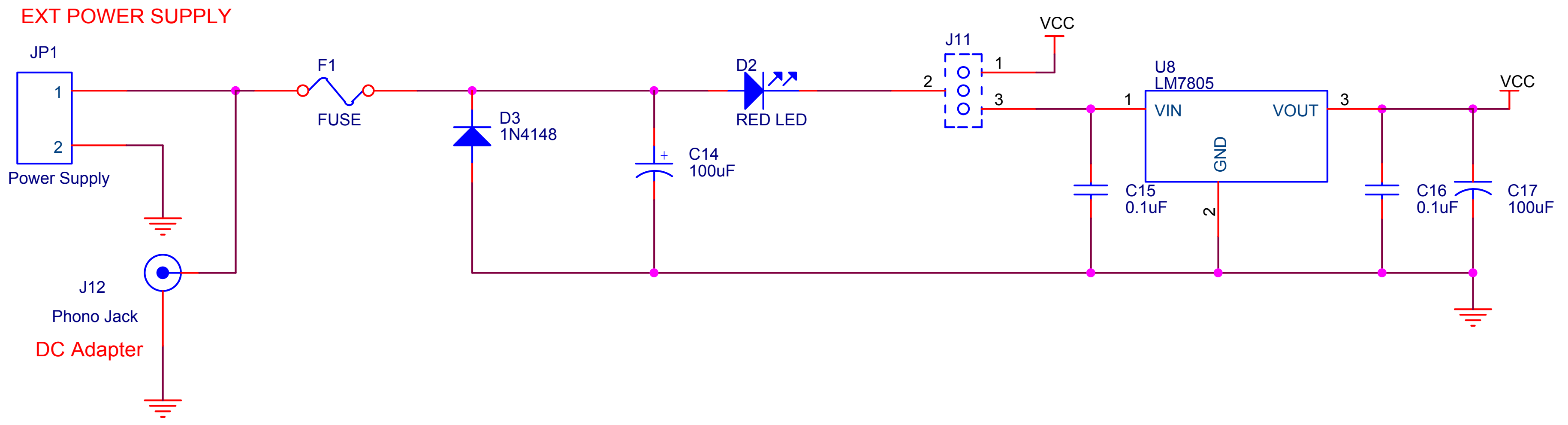


Appendix E: Schematic Diagram of LCEVB-SH1

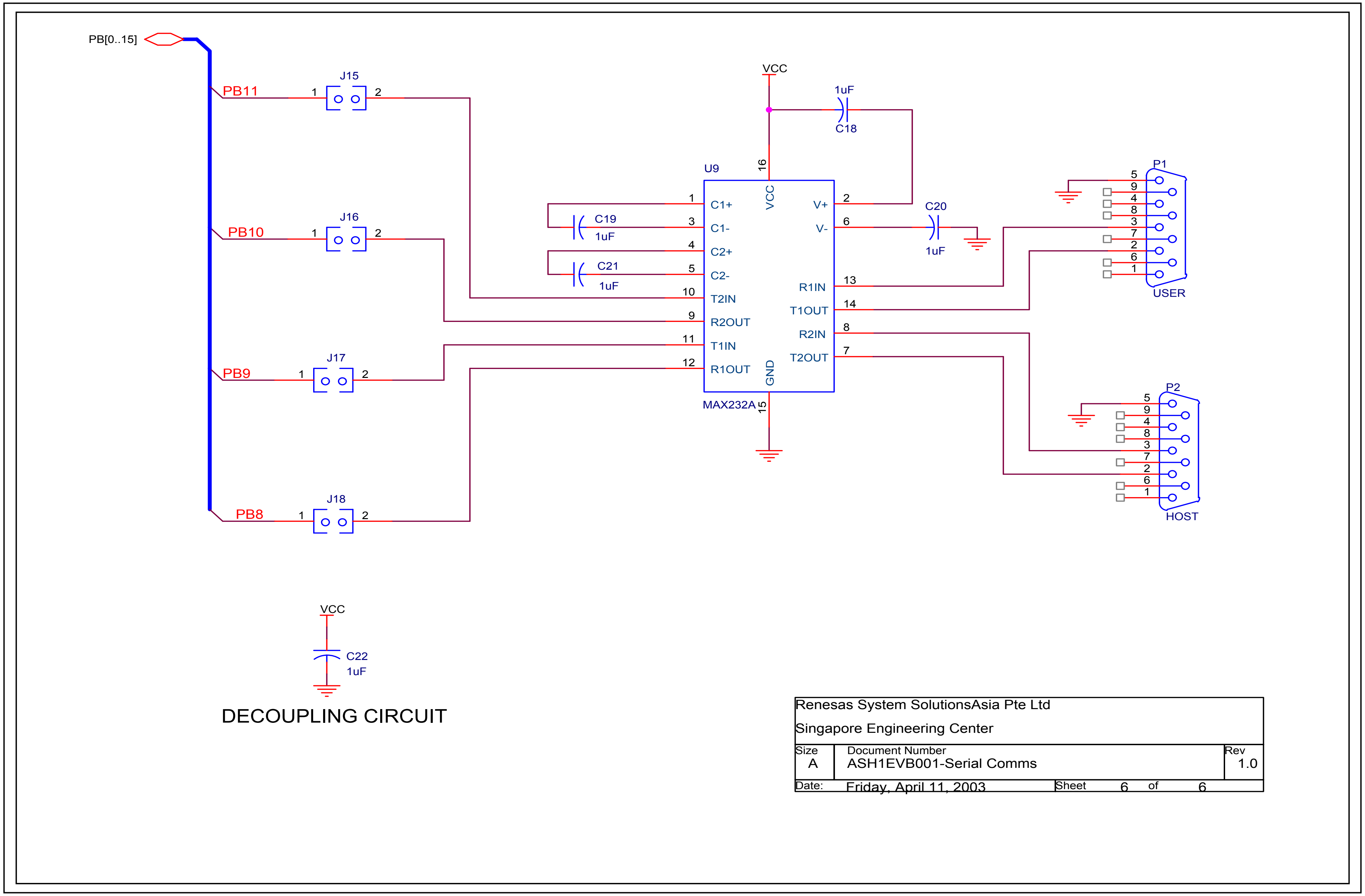




Renesas System Solutions Asia Pte Ltd Singapore Engineering Center		
Size A3	Document Number ASH1EVB001-Memorys	Rev 1.0
Date: Friday, April 11, 2003	Sheet 3 of 6	

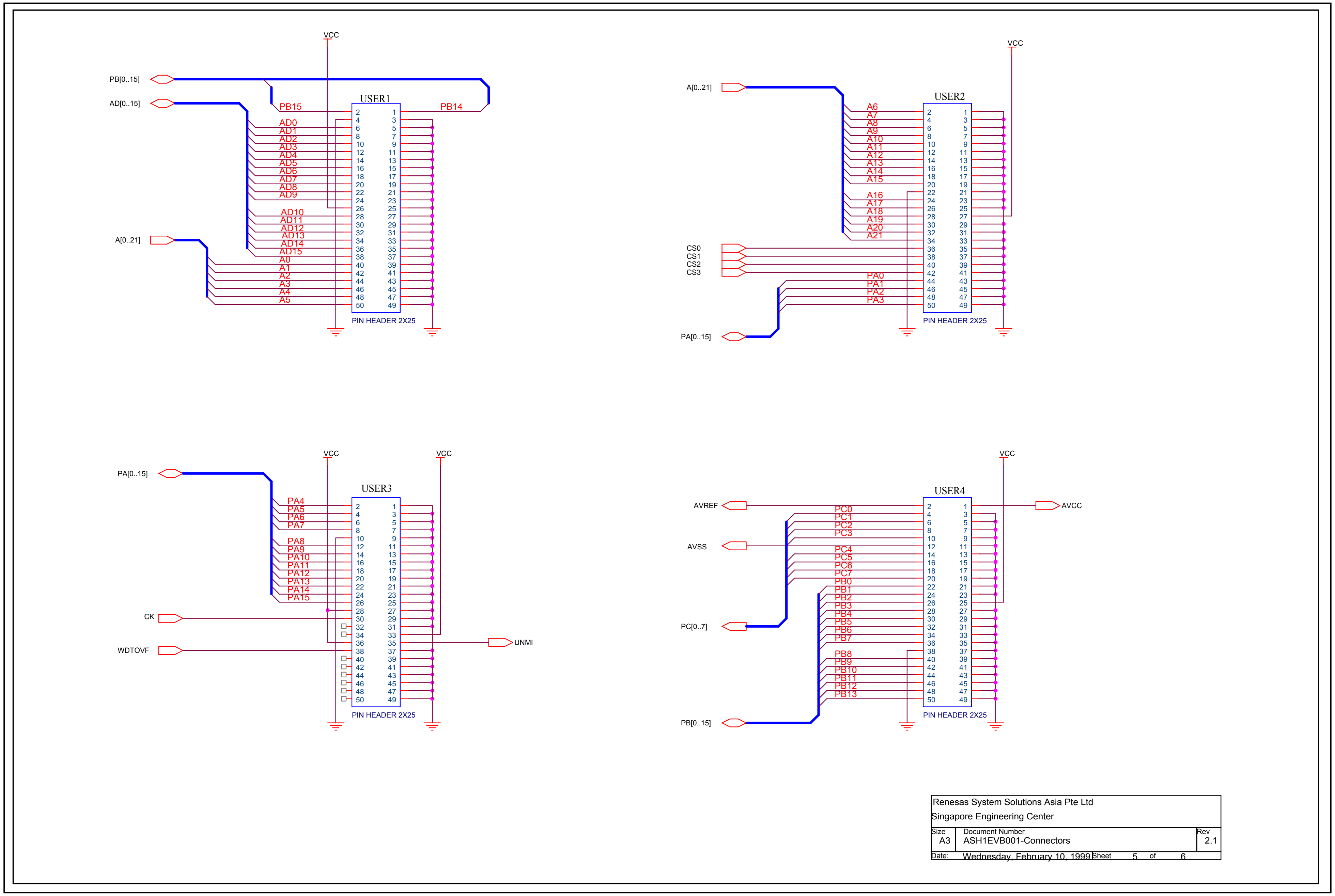


Renesas System Solutions Asia Pte Ltd Singapore Engineering Center		
Size A	Document Number ASH1EVB001- Power Supply	Rev 2.1
Date:	Friday, April 11, 2003	Sheet 4 of 6



DECOUPLING CIRCUIT

Renesas System SolutionsAsia Pte Ltd Singapore Engineering Center		
Size A	Document Number ASH1EVB001-Serial Comms	Rev 1.0
Date: Friday, April 11, 2003	Sheet 6 of 6	



Renesas System Solutions Asia Pte Ltd Singapore Engineering Center		
Size A3	Document Number ASH1EVB001-Connectors	Rev 2.1
Date: Wednesday, February 10, 1999	Sheet 5 of 6	

Renesas Technology (Asia Sales Offices)

URL: http://www.renesas.com

LCEVB-SH1

Renesas System Solutions Asia Pte. Ltd.