
EDK7145

USER MANUAL

FOR SH2/7145

ON-CHIP FLASH MICROCONTROLLER

Preface

Cautions

1. This document may be, wholly or partially, subject to change without notice.
2. All rights reserved. No one is permitted to reproduce or duplicate, in any form, a part or this entire document without Hitachi Micro Systems Europe Limited's written permission.

Trademarks

General

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

Specific

Microsoft, MS and MS-DOS are registered trademarks and Windows and Windows NT are trademarks of Microsoft Corporation.

Document Information

Product Code: D004130_11

Version: 2.0

Date: 13/11/2002

Copyright © Hitachi Micro Systems Europe Ltd. 1995-2002. All rights reserved.

Global: <http://www.hitachisemiconductor.com/>

Europe: <http://www.hmse.com>

1. TABLE OF CONTENTS

1. TABLE OF CONTENTS	3
2. START-UP INSTRUCTIONS	4
2.1. INSTALLING THE EVALUATION DEVELOPMENT KIT (EDK).....	4
2.2. SERIAL CONNECTION	4
2.3. POWER SUPPLY	4
3. EDK BOARD LAYOUT	5
3.1. EDK BLOCK DIAGRAM.....	5
4. EDK OPERATION	6
4.1. USER INTERFACE.....	6
4.2. SERIAL INTERFACE.....	6
4.3. SRAM.....	7
4.4. MEMORY MAP	8
4.5. SRAM ACCESS TIMING	8
4.6. LEADS.....	8
5. BOARD OPTIONS	9
5.1. JUMPER LINKS.....	9
5.2. USER MODE SETTINGS – CJ5.....	9
5.3. EDK OPTIONS – CJ4	10
5.4. SERIAL PORT SELECTION.....	10
5.5. FLASH PROGRAMMING HEADER	11
5.6. E10A HEADER	11
5.7. BOOT CONTROL	11
6. MICROCONTROLLER HEADER CONNECTIONS	13
6.1. HEADER J1	13
6.2. HEADER J2	14
7. CODE DEVELOPMENT	15
7.1. HMON	15
7.2. ADDITIONAL INFORMATION	19

2. START-UP INSTRUCTIONS

2.1. INSTALLING THE EVALUATION DEVELOPMENT KIT (EDK)

Please refer to the quick start guide provided for initial installation of the EDK.

A copy of the quick start guide and other information relating to this EDK at:

<http://www.hmse.com/products/support.htm>

Installing the EDK requires power and serial connection to a host computer.

2.2. SERIAL CONNECTION

The serial communications cable for connecting the EDK to a host computer is supplied. The serial cable has 1:1 connectivity.

Figure 2-1 shows how to connect the EDK to a PC or notebook computer equipped with a nine pin D connector.

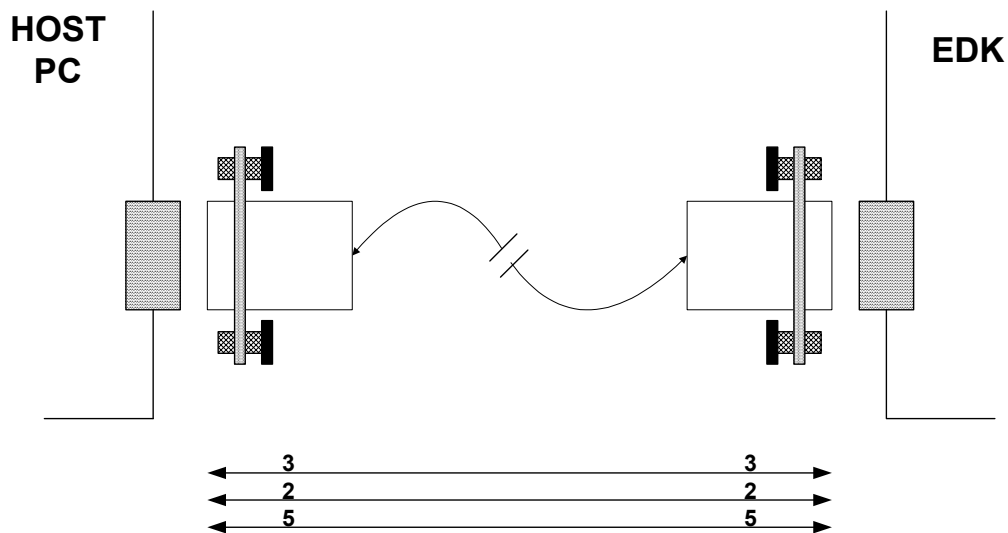


FIGURE 2-1: SERIAL CONNECTION TO PC/NOTEBOOK WITH DB-9 CONNECTOR (SUPPLIED)

2.3. POWER SUPPLY

The EDK hardware requires a power supply of +5V. Since total power consumption can vary widely due to external connections, port states, and memory configuration, use a power supply capable of providing at least 500mA at +5V DC \pm 5%.

The design is specified for evaluation of the microcontroller and so does not include circuitry for supply filtering/noise reduction, under voltage protection, over current protection or reversed polarity protection. Caution should be used when selecting and using a power supply.

The power connector on the EDK is a 2.5mm Barrel connector. The center pin is the positive connection.



FIGURE 2-2: POWER SUPPLY CONNECTION

Caution: Existing customers using E6000 products note that the polarity of this board is opposite to that for the E6000. Use of the E6000 power supply with this board will damage both board and power supply.

3. EDK BOARD LAYOUT

The diagram shows a general layout of the EDK board.

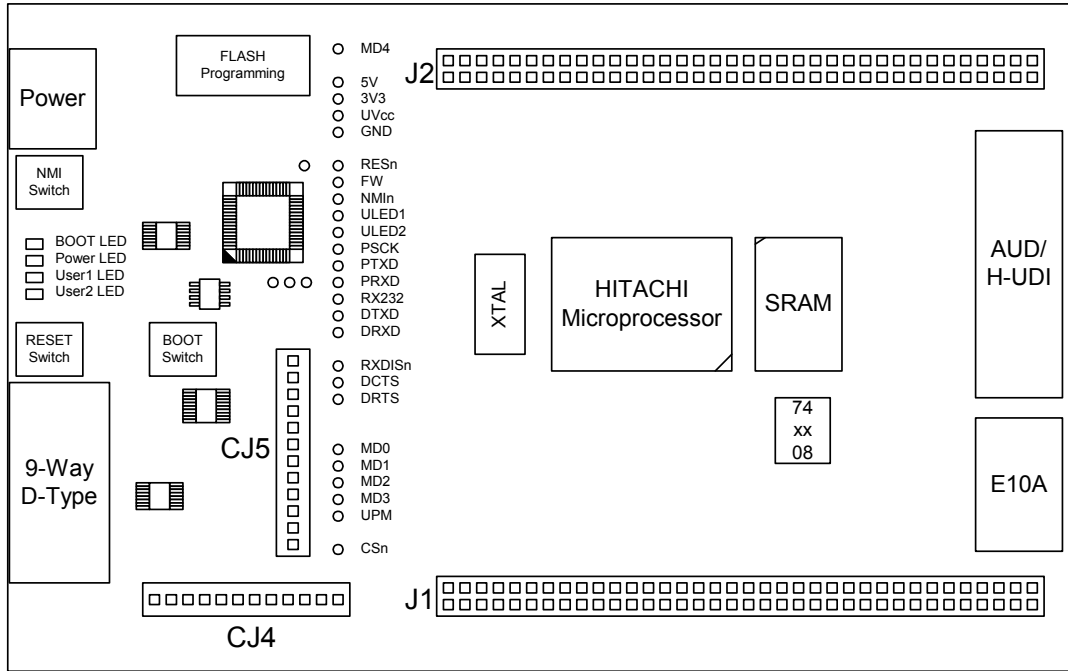


FIGURE 3-1: EDK BOARD LAYOUT

3.1. EDK BLOCK DIAGRAM

The diagram shows the connectivity of the components on the EDK board.

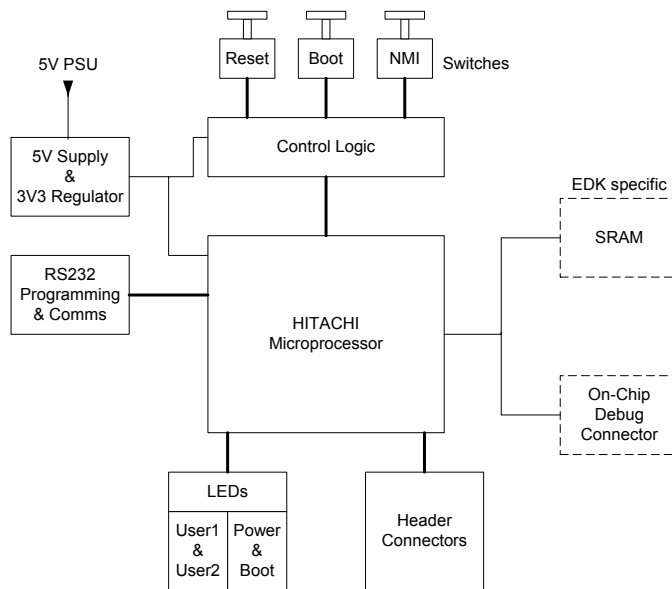


FIGURE 3-2: EDK BLOCK DIAGRAM

4. EDK OPERATION

4.1. USER INTERFACE

The EDK provides three buttons for influencing the operation of the board. The purpose of each button is clearly marked next to it. Refer to the board layout for positions (Figure 3.1)

1. Reset Switch

This button provides the microcontroller with a timed reset pulse of at least 250mS.

2. Boot Switch

This button toggles the operating mode of the microcontroller. A complete description of this function is given in section 5.7.

3. NMI Switch

This button provides a de-bounced signal to the microcontroller for each operation of the button. There is no minimum or maximum activation time for this button.

4.2. SERIAL INTERFACE

The serial interface on the EDK board has several functions. The serial port on the microcontroller directly supports three wire serial interfaces. Options are provided on the board for the user to write handshaking routines using standard port pins. Other board option links allow users to control the entry and exit from boot mode using the same handshaking signals. Refer to section 5.4 for details on setting serial interface options.

4.2.1. CONNECTOR PIN DEFINITIONS

The EDK RS232 interface conforms to Data Communication Equipment (DCE) format allowing the use of 1-1 cables when connected to Data Terminal Equipment (DTE) such as an IBM PC. The cable used to connect to the EDK will affect the available board options. A fully wired cable can allow handshaking between the microcontroller and the host PC, subject to setting the board options and the availability of suitable host software. Handshaking is not supported as standard on the microcontroller so for normal use a minimal three-wire cable can be used. The minimum connections are unshaded in the following table.

EDK DB9 Connector Pin	Signal	Host DB9 Connector Pin
1	No Connection	1
2	EDK Tx Host Rx	2
3	EDK Rx Host Tx	3
4	No Connection	4
5	Ground	5
6	No Connection	6
7	* EDK CTS Host RTS	7
8	* EDK RTS Host CTS	8
9	No Connection	9

TABLE 4-1: RS232 INTERFACE CONNECTIONS

* These are not connected on the EDK by default. See section 5.4 for more details.

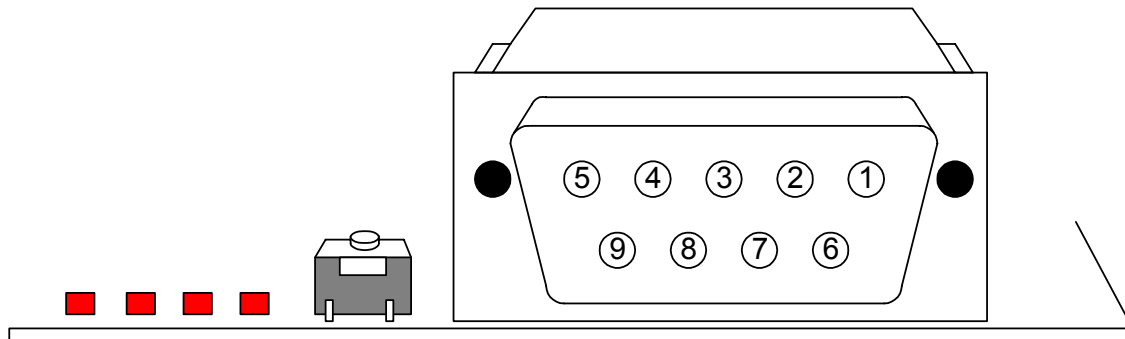


FIGURE 4-1: EDK SERIAL PORT PIN NUMBERING

4.2.2. CRYSTAL CHOICE

The operating crystal frequency has been chosen to support the fastest operation with the fastest serial operating speeds. The value of the crystal is 11.0592MHz.

The following table shows the baud rates and Baud Rate Register (BRR) setting required for each communication rate using the above default operating speed. It also confirms the resultant baud rate and the bit error rate that can be expected.

Baud Rate Register Settings for Serial Communication Rates												
SMR Setting:	0			1			2			3		
Comm. Baud	BRR setting	Actual Rate	ERR (%)	BRR setting	Actual Rate	ERR (%)	BRR setting	Actual Rate	ERR (%)	BRR setting	Actual Rate	ERR (%)
110	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	195	110	0.19	48	110	0.19
300	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	71	300	0.00	17	300	0.00
1200	Invalid	Invalid	Invalid	71	1200	0.00	17	1200	0.00	4	1080	-10.00
2400	Invalid	Invalid	Invalid	35	2400	0.00	8	2400	0.00	1	2700	12.50
4800	143	4800	0.00	17	4800	0.00	4	4320	-10.00	0	5400	12.50
9600	71	9600	0.00	8	9600	0.00	1	10800	12.50	Invalid	Invalid	Invalid
19200	35	19200	0.00	4	17280	-10.00	0	21600	12.50	Invalid	Invalid	Invalid
38400	17	38400	0.00	1	43200	12.50	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid
57600	11	57600	0.00	1	43200	-25.00	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid
115200	5	115200	0.00	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid
230400*	2	230400	0.00	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid
460800*	1	345600	-25.00	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid

TABLE 4-2 CRYSTAL FREQUENCIES FOR RS232 COMMUNICATION

* Note: The device used to convert the RS232 serial information to logic signals for the microcontroller is limited to 120kBaud. The rates above this level can only be utilised if the user provides direct logic level communications.

The default communication rate for the EDK is indicated by the shaded selection.

The user may replace the HC49/U surface mounted AT cut crystal with another of similar type within the operating frequency of the microcontroller device. Please refer to the hardware manual for the microcontroller for the valid operating range.

Alternatively the user may fit an oscillator module – or provide an external clock source. When providing an oscillator module or external source it is highly recommended that the load capacitors for the AT crystal are removed from the PCB. These are physically placed within the PCB outline of the oscillator module for easy location and to ensure they are removed when using this option.

When changing the crystal frequency the pre-loaded debugging monitor will not function. In this situation the user is responsible for providing code to evaluate the device away from the default operating speed.

4.2.3. REMOVABLE COMPONENT INFORMATION.

This information is provided to allow the replacement of components removed from the board as described in section 4.2.2.

Component	Cct. Ref	Value	Rating	Manufacturer
Load Resistor (X1)	R20	1MΩ	0805 1%	Welwyn WCR Series
Load capacitors (X1)	C13,C14	22pF	0603 10% 25V	0603 3 A 220 KAT

TABLE 4-3: REMOVABLE COMPONENT INFORMATION

Care must be taken not to damage the tracking around these components. Only use soldering equipment designed for surface mount assembly and rework.

4.3. SRAM

Positions have been supplied for customer fitting of an SRAM device and associated glue logic.

The SRAM device to be fitted to the board is a 4Mbit device allowing 256k x 16 operation.

Component	Cct. Ref	Part Number	Manufacturer
SRAM	U2	HM62W16255HLTT-12	Hitachi
Glue Logic	U3	SN74LVC08APWR	Texas Instruments

Table 4-4: SRAM and Glue Logic Component Information

The SRAM is mapped to area 0 via chip select 0 (port PA10), with a usable address range of H'00200000 – H'0027FFFF using address signals A1 – A18.

Glue logic provides the required SRAM control signals from the SH2/7145 micon.

4.4. MEMORY MAP

Table 4-5 illustrates the EDK memory map for mode 2.

Section End	Section Allocation
Section Start	
H' 00000000	On-chip ROM
H' 0003FFFF	
H' 00040000	Reserved Area
H' 001FFFFF	
H' 00200000	CS0 (512kB External RAM H'00200000 to H'0027FFFF)
H' 003FFFFF	
H' 00400000	External Address Space
H' 00FFFFFF	
H' 01000000	Reserved Area
H' FFFF7FFF	
H' FFFF8000	Internal IO Registers
H' FFFFBFFF	
H' FFFFC000	Reserved Area
H' FFFFDFFF	
H' FFFFE000	On-Chip RAM (8kB)
H' FFFFFFFF	

TABLE 4-5: MEMORY MAP (DEFAULT MODE 6)

4.5. SRAM ACCESS TIMING

External access timing is defined by several registers, allowing different types of devices to be addressed. The registers for the selection of wait states and signal extensions are given below with recommended values for the EDK.

Register	Recommended Setting for EDK	Function
PACRL1	0x1510	Enable CS0 (PA10), WRL (PA12), WRH (PA13), RD (PA14)
PBCR2	0x2005	Enable A15 (PB0), A16 (PB1), A17 (PB6)
PCCR	0xFFFFE	Enable A1 – A15 (PC1 – PC15)
PDCRL1	0xFFFF	Enable D0 – D15 (PD0 – PD15) – Register 1 of 2 for Data Bus
PDCRL2	0x0000	Enable D0 – D15 (PD0 – PD15) – Register 2 of 2 for Data Bus
(BIT)BCR1.A0SZ	1	CS1 accessed as word
(2 BITS)BCR2.IW0	1	1 Idle Cycle on CS1

TABLE 4-6: SRAM ACCESS CONTROL REGISTERS

Please refer to the hardware manual for the microcontroller for more information on these register settings.

4.6. LEDs

The EDK has four red LEDs. The function of each LED is marked on the silk screen of the PCB. Please refer to the board layout diagram for position information (Figure 3.1).

When the board is connected to a power source the Power (PWR) LED will illuminate. The Boot mode indication LED will illuminate when the microcontroller has been placed into Boot mode. Please see section 5.7 for more details of this function.

There are two LEDs dedicated for user control these are marked USR1 and USR2. Each LED will illuminate when the port pin is in a logical high state.

The user LEDs are connected to the following ports:

LED Identifier	Port Pin	Microcontroller Pin	Pin Functions on Port Pin
USR1	PE14	2	PE14/TIOC4C/DACK0
USR2	PE15	5	PE15/TIOC4D/DACK1/IRQOUTn

TABLE 4-7: LED PORT CONNECTIONS

5. BOARD OPTIONS

The EDK has a number of configuration settings set by jumpers CJ4 (A, B, C, D), CJ5 (A, B, C, D) and zero-ohm links. Common EDK functions can be set using the jumpers as described in sections 5.3 and 5.2. The additional zero-ohm links provide additional features that may be required to interface with other systems.

All the Jumper link settings are three pin options. There are four sets of options on each header.

The headers are numbered from 1 to 12 with pin 1 marked on the PCB by an arrow pointing to the pin. The diagram below shows the numbering of these jumper links and indicates jumpers fitted 1-2 for each three-pin jumper.

5.1. JUMPER LINKS

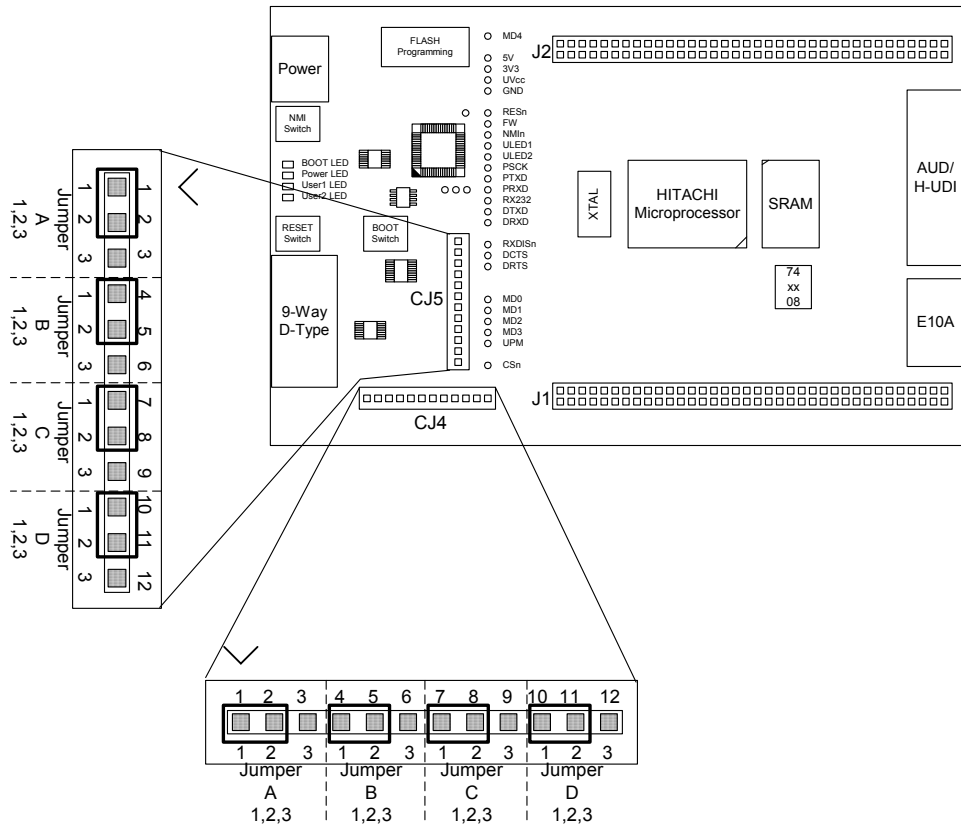


FIGURE 5-1: JUMPER CONFIGURATION

The following tables define each jumper and its settings.

5.2. USER MODE SETTINGS – CJ5

CJ5 is used to set the operating mode of the microcontroller.

These jumpers must be fitted at all times to ensure correct operation of the EDK.

Jumper	Function	Setting 1-2	Setting 2-3
CJ 5-A Default 2-3	User Mode Setting Bit 0	MD0 pulled High	MD0 pulled Low
CJ 5-B Default 1-2	User Mode Setting Bit 1	MD1 pulled High	MD1 pulled Low
CJ 5-C Default 1-2	User Mode Setting Bit 2	MD2 pulled High	MD2 pulled Low
CJ 5-D Default 1-2	User Mode Setting Bit 3	MD3 pulled High	MD3 pulled Low

TABLE 5-1: USER MODE: JUMPER SETTINGS (DEFAULT SETTINGS IN BOLD)

The default settings indicated in bold text place the microcontroller into operating Mode 3, i.e. on-chip ROM enabled, single chip mode and clock mode 3: System clock = x4, Peripheral clock = x2.

5.3. EDK OPTIONS – CJ4

The EDK options provide access to commonly used features of the EDK range.

These jumpers must be fitted at all times to ensure correct operation of the EDK.

Jumper	Function	Setting 1-2	Setting 2-3
CJ 4-A Default 2-3	Serial Receive Source	Disables the RS232 receive signal to enable the use of the Flash Programming Header	Enables the RS232 receive signal. The Flash Programming Header *1 must not be used in this state.
CJ 4-B Default 2-3	User Programming Mode	Disables the flash write hardware protection. The flash can be overwritten in User Mode.	Enables the flash write hardware protection. The flash cannot be overwritten in User Mode.
CJ 4-C Default 2-3	HUDI	Hitachi User Debugging Interface (E10A) Enabled.	Hitachi User Debugging Interface (E10A) Disabled.
CJ 4-D Default 1-2	CSn	SRAM enabled when SH/7145 CSn(0) signal is asserted	SRAM Disabled

TABLE 5-2: BOARD OPTION: JUMPER SETTINGS (DEFAULT SETTINGS IN BOLD)

*1 See section 5.5

The following table lists the connections to each jumper pin.

Pin	Net Name	Description
1	UVCC	Microcontroller Supply Voltage
2	RXDISn	Disable Flash Header functions. Pulled low. (Enables RX232)
3	No Connection	No Connection
4	UVCC	Microcontroller Supply Voltage
5	UPM	CPLD Controlled option to set Flash Write (FWP).
6	No Connection	No Connection
7	No Connection	No Connection
8	DBGMD	Debug Mode – Enables/Disables HUDI Interface
9	GDN	Ground
10	PA10	Microcontroller CSn(0) signal
11	CSn	SRAM CSn signal
12	UVCC	Microcontroller Supply Voltage

5.4. SERIAL PORT SELECTION

The programming serial port is connected to the RS232 connector by default. This allows direct programming of the EDK using the supplied software tools. A secondary serial port is available on the microcontroller and can be connected to the RS232 connector by changing some board option links. The additional port option allows the user to write messages or connect to other devices via the serial port while programming support is provided by the Flash programming header.

The following surface mount, zero-ohm link settings are fitted by default and connect the RS232 header to the programming serial port of the microcontroller.

Zero-ohm Link ID	Default	Function	Microcontroller Port Pin
CR20	Fitted	Transmit data from EDK	PA4
CR23	Fitted	Receive data to EDK	PA3
CR19	Not Fitted	Alternate Transmit data from EDK	PA1
CR22	Not Fitted	Alternate Receive data to EDK	PA0

TABLE 5-3: OPTION LINKS – DEFAULT SETTINGS

To enable the use of this alternate port the user must change the settings to those in the following table.

Zero-ohm Link ID	Default	Function	Microcontroller Port Pin
CR20	Not Fitted	Transmit data from EDK	PA4
CR23	Not Fitted	Receive data to EDK	PA3
CR19	Fitted	Alternate Transmit data from EDK	PA1
CR22	Fitted	Alternate Receive data to EDK	PA0

TABLE 5-4: OPTION LINKS – ALTERNATE SERIAL PORT

The user may implement a handshaking protocol on the EDK. This is not supported with the software tools supplied. To support this option two spare port pins have been allocated on the microcontroller. Using these port pins the CTS and RTS lines of the host serial interface can be controlled.

The user may also control the operation of the board via the same handshaking lines. This is not supported with the software tools supplied but may be written by the user. Using the CTS line the user may simulate pressing the boot button, see section:5.7. This will cause the EDK to swap into and out of Boot mode on each low-level activation of CTS. Feedback of the current mode is provided on the RTS line. A high level indicates boot mode and a low level indicates user mode.

The following settings are made by default, and ensure that there are no conflicts on unnecessary microcontroller pins.

Zero-ohm Link ID	Default	Function	Microcontroller Port Pin
CR12	Not Fitted	Mode State out from EDK	N/A (From CPLD*)
CR7	Not Fitted	Change Mode request to EDK	N/A (From CPLD*)
CR16	Not Fitted	Alternate RTS232 – Ready to send – from EDK	PA20
CR13	Not Fitted	Alternate CTS232 – Clear to send – to EDK	PB4

TABLE 5-5: OPTION LINKS – SERIAL PORT CONTROL

* See section 5.7

Note: These setting pairs are exclusive:
 If CR12 and CR7 are fitted; CR16 and CR13 must not be fitted.
 If CR16 and CR13 are fitted; CR12 and CR7 must not be fitted.

5.5. FLASH PROGRAMMING HEADER

The Flash Programming header is used with the Hitachi Flash Development Module (FDM). The FDM is a USB based programming tool for control and programming of Hitachi microcontrollers, available separately from Hitachi. This header provides direct access for the FDM to control the EDK microcontroller.

To utilise this header the user must make the following changes to the board configuration.

1. Disable the RX232 signal from the RS232 transceiver.
Jumper link CJ4-A is provided for this purpose. Please refer to section 5.3.
2. Disable User Program Mode using jumper CJ4-B. Please refer to section 5.3.

Caution: Do not operate the board with the user mode jumpers removed and the FDM disconnected as the microcontroller mode pins will float to an indeterminate state. This may damage the microcontroller device.

5.6. EXTERNAL DEBUG HEADER

The External debug header may be used with the Hitachi E10A Debugger or a third party debugger.

The E10A is an on-chip debug emulator available separately from Hitachi.

This header provides direct access for the debugger to control the EDK microcontroller.

To utilise this header the user must enable the E10A interface via jumper CJ4-C. Please refer to section 5.3.

5.7. BOOT CONTROL

The method for placing the microcontroller device in to Boot mode for reprogramming has been incorporated into a complex programmable logic device (CPLD). This is not necessary for most user designs but allows a measure of increased flexibility for the EDK designs. Mode transitions including boot mode transitions only require the reset to be held active while the mode settings are presented. On releasing reset the microcontroller will be in the required mode.

The logic design detects a power up event and provides a timed reset pulse to guarantee the reset of the device. At the end of the reset pulse the processor will be placed in user mode and any code in the device will execute.

During user mode the NMI button can be pressed at any time. This will provide a single de-bounced NMI interrupt to the device.

Pressing the boot button will cause the boot mode controller to reset the device and, during the reset period, present the required mode settings to start the device in boot mode. At the end of the reset period the boot mode settings will have been latched into the device which will then be ready to accept a boot mode connection via the RS232 interface or the flash programming header. Pressing the boot button during a normal reset will not cause the EDK to enter boot mode.

The boot mode settings are fixed at mode 0. The required mode settings are made using a tri-state capable buffer.

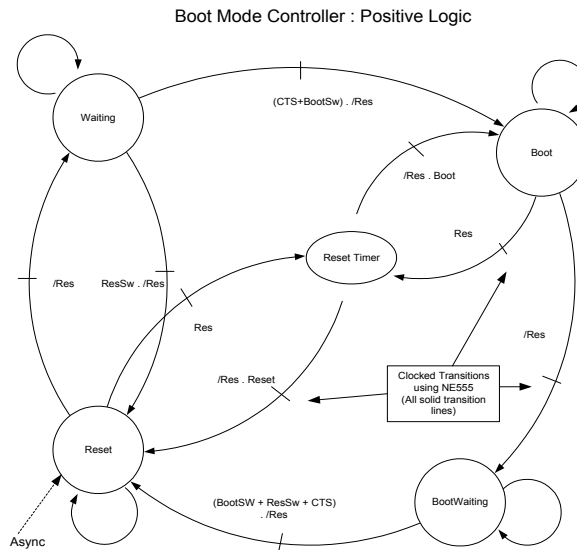
Note: The boot control device is programmed to support all possible EDK products.
 For this reason the reset pulse is over 500ms. Repetitive activation of either the Boot or Reset buttons will restart the reset timer and extend the reset period. Pressing the boot button within the 500mS period of a reset will not cause the board to enter boot mode.

5.7.1. CPLD CODE

The code is based upon a four state machine providing a guaranteed reset period that can be extended by holding the relevant control input in the active state. When released the timer will extend the reset for approximately 500mS.

The states are split into two functions, one for User mode and one for Boot mode. The first state of each is used to hold the reset line active. When the timer expires then the second state is used to hold the device in the selected mode and wait for an external control signal to either move back into the user reset state or into the boot reset state.

5.7.2. STATE DIAGRAM



6. MICROCONTROLLER HEADER CONNECTIONS

The following table lists the connections to each of the headers on the board.

6.1. HEADER J1

J1							
Pin No	Function	EDK Symbol	Device pin	Pin No	Function	EDK Symbol	Device pin
1	PD2/D2		18	2	PD3/D3		17
3	PD4/D4		16	4	VSS	GND	15
5	PD5/D5		14	6	VCC	UVCC	13
7	PD6/D6		12	8	PD7/D7		11
9	PD8/D8		10	10	PD9/D9		9
11	PD10/D10		8	12	VSS	GND	7
13	PD11/D11		6	14	VCC	UVCC	5
15	PD12/D12		4	16	PD13/D13		3
17	PD14/D14		2	18	PD15/D15		1
19	PD16/D16/IRQ0n/AU DATA0		144	20	VSS	GND	143
21	PD17/D17/IRQ1n/AU DATA1		142	22	PD18/D18/IRQ2n/AUD ATA2		141
23	PD19/D19/IRQ3n/AU DATA3		140	24	PD20/D20/IRQ4n/AUD RSTn		139
25	PD21/D21/IRQ5n/AU DMD		138	26	PD22/D22/IRQ6n/AUD CK		137
27	PD23/D23/IRQ7n/AU DSYNCn		136	28	VCC	UVCC	135
29	PD24/D24/DREQ0n		134	30	VSS	GND	133
31	PD25/D25/DREQ1n		132	32	PD26/D26/DACK0		131
33	PD27/D27/DACK1		130	34	PD28/D28/CS2n		129
35	PD29/D29/CS3n		128	36	VSS	GND	127
37	PA6/TCLKA/CS2n		126	38	PA7/TCLKB/CS3n		125
39	PA8/TCLKC/IRQ2n		124	40	PA9/TCLKD/IRQ3n		123
41	PA10/CS0n		122	42	PA11/CS1n		121
43	PA12/WRLn		120	44	PA13/WRHn		119
45	PD30/D30/IRQOUTn		118	46	PD31/D31/ADTRGn		117
47	WDTOVF _n		116	48	PA14/RD _n		115
49	DBGMD		114	50	PB9/IRQ7n/A21/ADTR Gn		113
51	VCC	UVCC	112	52	PB8/IRQ6/A20/WAIT _n		111
53	PB7/IRQ5n/A19/BRE Q _n		110	54	PB6/IRQ4n/A18/BACK n		109
55	PB5/IRQ3n/POE3 _n		108	56	ASEBRKAK _n		107
57	PB4/IRQ2n/POE2 _n	DCTS	106	58	PA18/BREQ _n /DRAK0		105
59	PB3/IRQ1n/POE1n/SDA0		104	60	PB2/IRQ0n/POE0n/SCL0		103
61	PA19/BACK _n /DRAK1		102	62	PA20	DRTS	101
63	VSS	GND	100	64	PB1/A17		99
65	VCC	UVCC	98	66	PB0/A16		97
67	PC15/A15		96	68	PC14/A14		95
69	PC13/A13		94	70	PC12/A12		93
71	PC11/A11		92	72	PC10/A10		91

6.2. HEADER J2

J2							
Pin No	Function	EDK Symbol	Device pin	Pin No	Function	EDK Symbol	Device pin
1	PD1/D1		19	2	PD0/D0		20
3	VSS	GND	21	4	XTAL	CON_XTAL	22
5	MD3		23	6	EXTAL	CON_EXTAL	24
7	MD2		25	8	NMI		26
9	FWP	FW	27	10	PA16/AUDSYNc _n	CON_PA16	28
11	PA17/WAIT _n		29	12	MD1		30
13	MD0		31	14	PLL _{VCC} (NO CONNECTION)		32
15	PLLCAP (NO CONNECTION)		33	16	PLL _{VSS} (NO CONNECTION)		34
17	PA15/CK		35	18	RES _n		36
19	PE0/TIOC0A/DREQ0 _n /AUDCK	CON_PE0	37	20	PE1/TIOC0B/DRAK0/AUDMD		38
21	PE2/TIOC0C/DREQ1 _n /AUDRST _n		39	22	VCC	UVCC	40
23	PE3/TIOC0D/DRAK1/AUDATA3	CON_PE3	41	24	PE4/TIOC1A/RXD3/AUDATA2	CON_PE4	42
25	PE5/TIOC1B/TXD3/AUDATA1	CON_PE5	43	26	PE6/TIOC2A/SCK3/AUDATA0	CON_PE6	44
27	VSS	GND	45	28	PF0/AN0		46
29	PF1/AN1		47	30	PF2/AN2		48
31	PF3/AN3		49	32	PF4/AN4		50
33	PF5/AN5		51	34	AVSS	CON_AVSS	52
35	PF6/AN6		53	36	PF7/AN7		54
37	AVREF	CON_AVREF	55	38	AVCC	CON_AVCC	56
39	VSS	GND	57	40	PA0/RXD0	DRXD	58
41	PA1/TXD0	DTXD	59	42	PA2/SCK0/DREQ0 _n /IRQ0 _n		60
43	PA3/RXD1	PRXD	61	44	PA4/TXD1	PTXD	62
45	VCC	UVCC	63	46	PA5/SCK1/DREQ1 _n /IRQ1 _n	PSCK	64
47	PE7/TIOC2B/RXD2		65	48	PE8/TIOC3A/SCK2/TMS		66
49	PE9/TIOC3B/TRST _n /SCK3		67	50	PE10/TIOC3C/TXD2/TDI	CON_PE10	68
51	VSS	GND	69	52	PE11/TIOC3D/TDO/RXD3	CON_PE11	70
53	PE12/TIOC4A/TCK/TXD3	CON_PE12	71	54	PE13/TIOC4B/MRES _n		72
55	PA23/WRHH _n		73	56	PE14/TIOC4C/DACK0	ULED1	74
57	PS22/WRHL _n		75	58	PA21		76
59	PE15/TIOC4D/DACK1/IRQOUT _n	ULED2	77	60	VSS	GND	78
61	PC0/A0		79	62	PC1/A1		80
63	PC2/A2		81	64	PC3/A3		82
65	PC4/A4		83	66	VCC	UVCC	84
67	PC5/A5		85	68	VSS	GND	86
69	PC6/A6		87	70	PC7/A7		88
71	PC8/A8		89	72	PC9/A9		90

7. CODE DEVELOPMENT

7.1. HMON

HMON is an on-chip debugger from HMSE. It allows code to be debugged in Flash and/or RAM on their target hardware using the MCU's debug capabilities. It consists of HMON components for HEW communicating with HMON debugger code, flash programming code and the developer's application code running on the MCU. Most Hitachi MCUs include some on-chip debugging functionality; this can comprise of software interrupt instructions (TRAP) and an address break peripheral unit. HEW debugging functionality combined with HMON code and Flash programming code enables the developer to run, step and set breakpoints in their application code as well as using other debugging functionality such as viewing memory and C/C++ source code. Please refer to section 7.1.7 for restrictions on the use of the UBC with HMON.

7.1.1. MODE SUPPORT

The HMON library is built to support Advanced Expanded Mode and Advanced Single Chip Mode only, modes 2 and 3, with the clock mode set to (System clock = x4), (Peripheral clock = x2). The Device supports Modes 0, 1, 2 and 3, however On-Chip ROM is active in modes 2 and 3.

7.1.2. BREAKPOINT SUPPORT

The monitor utilises the User Break Controller for code located in ROM, allowing a single breakpoint to be set in the code. Code located in RAM may have multiple breakpoints limited only by the size of the On-Chip RAM.

7.1.2.1. CODE LOCATED IN FLASH / ROM

Double clicking in the breakpoint column in the code sets the breakpoint. Adding a further breakpoint elsewhere in the code removes the previous one.

7.1.2.2. CODE LOCATED IN RAM

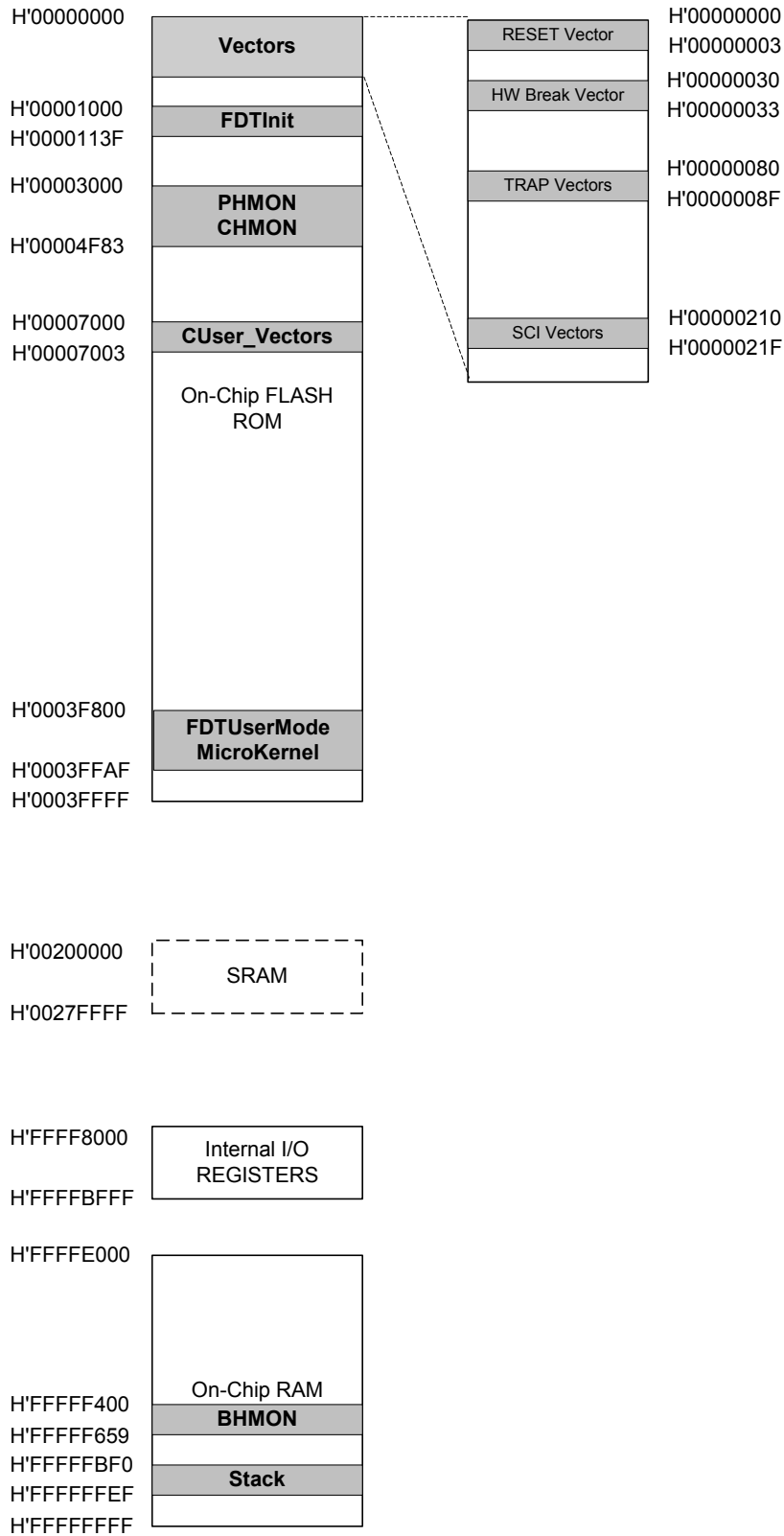
Double clicking in the breakpoint column in the code sets the breakpoint. Breakpoints will remain unless they are double clicked to remove them.

7.1.3. HMON CODE SIZE

HMON is built along with the debug code. Certain elements of the HMON code must remain at a fixed location in memory. The following table details the HMON components and their size and location in memory. For more information, refer to the map file when building code.

Section	Description	Start Location	Size (H'bytes)
RESET_VECTOR	HMON Reset Vector (Vector 0) Required for Startup of HMON	H' 00000000	4
HW_BREAK_VECTORS	HMON Break Controller (Vector 12) Required by HMON to create Breakpoints in ROM	H' 00000030	4
TRAP_VECTORS	Trap Vectors (Vector 32, 33, 34, 35) Required by HMON to create Trap Breakpoints in RAM	H' 00000080	10
SCI_VECTORS	HMON Serial Port Vectors (Vector 132, 133, 134) Used by HMON when EDK is configured to connect to the default serial port.	H' 00000210	C
PHMON	HMON Code	H' 00003000	1f84
CHMON	HMON Constant Data	H' 00004f84	140
BHMON	HMON Uninitialised data	H' fffff400	25A
FDTInit	FDT User Mode Kernel. This is at a fixed location and must not be moved. Should the kernel need to be moved it must be re-compiled.	H' 00001000	140
UserModeMicroKernel	UserModeMicroKernel. This is at a fixed location and must not be moved. Should the kernel need to be moved it must be re-compiled.	H' 0003f800	7B0
CUser_Vectors	Pointer used by HMON to point to the start of user code. This is at a fixed location and must not be moved for the Reset CPU, and Go Reset commands to function.	H' 00007000	4

7.1.4. MEMORY MAP



7.1.5. BAUD RATE SETTING

HMON has initially set to connect at 115200Baud. Should the user wish to change this, the value for the BRR in HMONserialconfiguser.c will need to be changed and the project re-built. Please refer to the HMON User Manual for further information.

7.1.6. INTERRUPT MASK SECTIONS

HMON has an interrupt priority of 14. The serial port has an interrupt priority of 7. Modules using interrupts should be set to lower than this value (14 or below), so that serial communications and debugging capability is maintained.

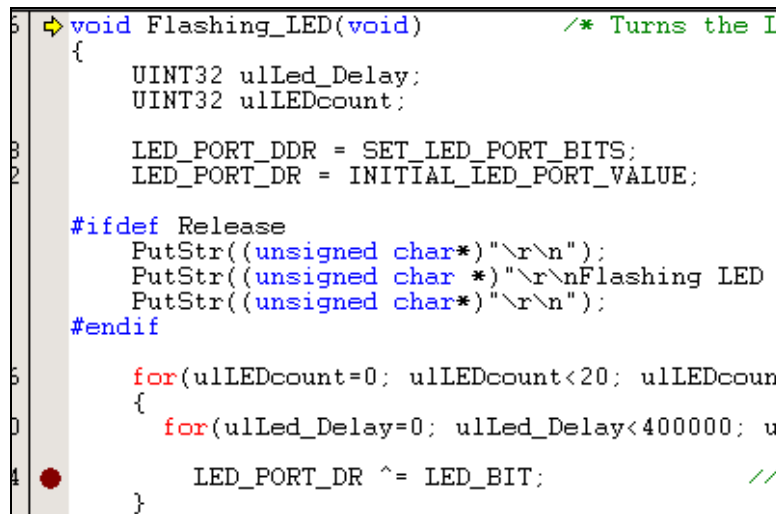
7.1.7. RESTRICTIONS WHEN STEPPING/RUNNING IN ROM WITH THE SH2 UBC:

For the SH/7145, HMON utilises the User Break Controller (UBC) to perform stepping of code. Breakpoints set in RAM use TRAPA software interrupts. Breakpoints set in ROM use the UBC. Due to the nature of the UBC, the functionality has some restrictions.

The UBC cannot be set to break on an instruction access after a non-delay branch instruction. This means that if we try and set a breakpoint directly inside a loop body constructed from BF(BRANCH IF FALSE)/BT(BRANCH IF TRUE), the break never occurs as the BSR will directly follow the BT/BF.

The problem is resolved by setting the data access break bit in addition to the instruction access, which causes the overrun fetch of the next opcode to cause the break to occur after the BT/BF. This has restrictions when stepping certain other types of constructs.

When setting a breakpoint in source code inside a loop body, execution may not stop at the expected source line. For example setting the breakpoint at the line shown below in Figure 7-1;



```
5 void Flashing_LED(void) /* Turns the LED on and off */
{
    UINT32 ulLed_Delay;
    UINT32 ulLEDcount;

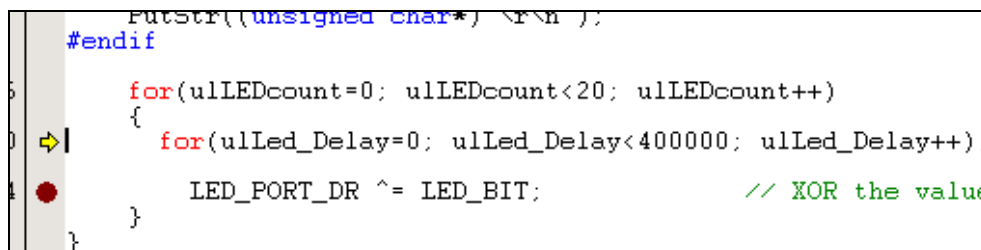
    LED_PORT_DDR = SET_LED_PORT_BITS;
    LED_PORT_DR = INITIAL_LED_PORT_VALUE;

#ifdef Release
    PutStr((unsigned char*)"r\n");
    PutStr((unsigned char*)"r\nFlashing LED\n");
    PutStr((unsigned char*)"r\n");
#endif

    for(ulLEDcount=0; ulLEDcount<20; ulLEDcount++)
    {
        for(ulLed_Delay=0; ulLed_Delay<400000; ulLed_Delay++)
        {
            LED_PORT_DR ^= LED_BIT; //
        }
    }
}
```

FIGURE 7-1: SOURCE CODE VIEW SHOWING BREAKPOINT

and choosing **Debug | Run** will produce the results shown below;



```
PutStr((unsigned char*)"r\n");
#endif
5     for(ulLEDcount=0; ulLEDcount<20; ulLEDcount++)
    {
0     for(ulLed_Delay=0; ulLed_Delay<400000; ulLed_Delay++);
4     LED_PORT_DR ^= LED_BIT; // XOR the value
    }
}
```

FIGURE 7-2: RESULT OF DEBUG | RUN TO BREAKPOINT

As illustrated in Figure 7-2, the execution has stopped at the source line preceding the breakpoint. Examination of the variable 'ulLed_Delay' reveals that the inner loop has only been executed once. Repeated calls to **Debug | Run** will

increment this inner loop counter by 1 each time but the source line with the breakpoint will never be reached, until the loop terminating condition is satisfied.

If we examine the actual opcodes being stepped, we can see the PC has stopped at the opcode after the non-delay branch instruction, as previously explained (i.e. the data access break bit is enabled).

```

Flashin 7FF8 ADD      #H'F8,R15
00002148 D347 MOV.L   @(H'011C:8,PC),R3
0000214A 6233 MOV     R3,R2
0000214C 9485 MOV.W   @(H'010A:8,PC),R4
0000214E 6023 MOV     R2,R0
00002150 8142 MOV.W   R0,@(H'04:4,R4)
00002152 E200 MOV     #H'00,R2
00002154 2421 MOV.W   R2,@R4
00002156 E500 MOV     #H'00,R5
00002158 6753 MOV     R5,R7
0000215A E514 MOV     #H'14,R5
0000215C 3752 CMP/HS  R5,R7
0000215E 8914 BT      @H'218A:8
00002160 E500 MOV     #H'00,R5
00002162 6253 MOV     R5,R2
00002164 D341 MOV.L   @(H'0104:8,PC),R3
00002166 6533 MOV     R3,R5
00002168 3252 CMP/HS  R5,R2
0000216A 8903 BT      @H'2174:8
  0000216C 7201 ADD     #H'01,R2
0000216E D63F MOV.L   @(H'00FC:8,PC),R6
00002170 3262 CMP/HS  R6,R2
00002172 8BFB BF      @H'216C:8
● 00002174 9471 MOV.W   @(H'00E2:8,PC),R4
00002176 6241 MOV.W   @R4,R2
00002178 622D EXTU.W  R2,R2
0000217A D33D MOV.L   @(H'00F4:8,PC),R3
0000217C 6523 MOV     R2,R5
0000217E 253A XOR     R3,R5
00002180 2451 MOV.W   R5,@R4
00002182 7701 ADD     #H'01,R7
00002184 E214 MOV     #H'14,R2
00002186 3722 CMP/HS  R2,R7
00002188 8BEA BF      @H'2160:8
0000218A 7F08 ADD     #H'08,R15
0000218C 000B RTS

```

FIGURE 7-3: DISASSEMBLY VIEW OF CODE

The solution to this problem is to bypass the BT/BF UBC issue and set the breakpoint at the actual opcode (as opposed to the source line consisting of the set of opcodes).

```

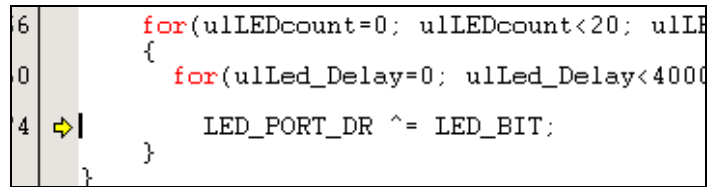
00002164 D341 MOV.L   @(H'0104:8,PC),R3
00002166 6533 MOV     R3,R5
00002168 3252 CMP/HS  R5,R2
0000216A 8903 BT      @H'2174:8
  0000216C 7201 ADD     #H'01,R2
0000216E D63F MOV.L   @(H'00FC:8,PC),R6
00002170 3262 CMP/HS  R6,R2
00002172 8BFB BF      @H'216C:8
00002174 9471 MOV.W   @(H'00E2:8,PC),R4
00002176 6241 MOV.W   @R4,R2
00002178 622D EXTU.W  R2,R2
0000217A D33D MOV.L   @(H'00F4:8,PC),R3
0000217C 6523 MOV     R2,R5
0000217E 253A XOR     R3,R5
● 00002180 2451 MOV.W   R5,@R4
00002182 7701 ADD     #H'01,R7
00002184 E214 MOV     #H'14,R2
00002186 3722 CMP/HS  R2,R7

```

FIGURE 7-4: BREAKPOINT SOLUTION IN DISASSEMBLY

In Figure 7-4 the breakpoint has been set at the opcode MOV.W R5, @R4 (the line that actually sets the LED bits into the appropriate register). If we perform a **Debug | Run**, we are no longer trying to break immediately after the non-delay branch,

so our code runs on until hitting the new breakpoint. This opcode still reflects the source line LED_PORT_DR ^= LED_BIT and this is illustrated in the source window, as shown below in Figure 7-5.



```
6      for(uLEDCount=0; uLEDCount<20; uLEDCount++)
0      {
4      for(uLEDDelay=0; uLEDDelay<4000; uLEDDelay++)
      LED_PORT_DR ^= LED_BIT;
      }
```

FIGURE 7-5: BREAKPOINT SOLUTION SOURCE CODE VIEW

The CPU performs 2 instruction-overrun fetches before getting the destination branch at a BT/BF, therefore providing that we are more than two opcodes away from the BT/BF along the execution path the break should occur as expected. In this example both MOV.W @(H'00E2:8, PC), R4, and MOV.W @R4, R2 are not suitable locations to place a breakpoint, however EXTU.W R2, R2 is.

7.2. ADDITIONAL INFORMATION

For details on how to use Hitachi Embedded Workshop (HEW), with HMON, refer to the HEW manual available on the CD or from the web site.

For information about the SH/7145 series microcontrollers refer to the SH/7144 SH/7145 *Series Hardware Manual*

For information about the SH/7145 assembly language, refer to the SH2 *Series Programming Manual*

Further information available for this product can be found on the HMSE web site at:

<http://www.hmse.com/products/support.htm>

General information on Hitachi Microcontrollers can be found at the following URLs.

Global: <http://www.hitachisemiconductor.com/>

Europe: <http://www.hmse.com>