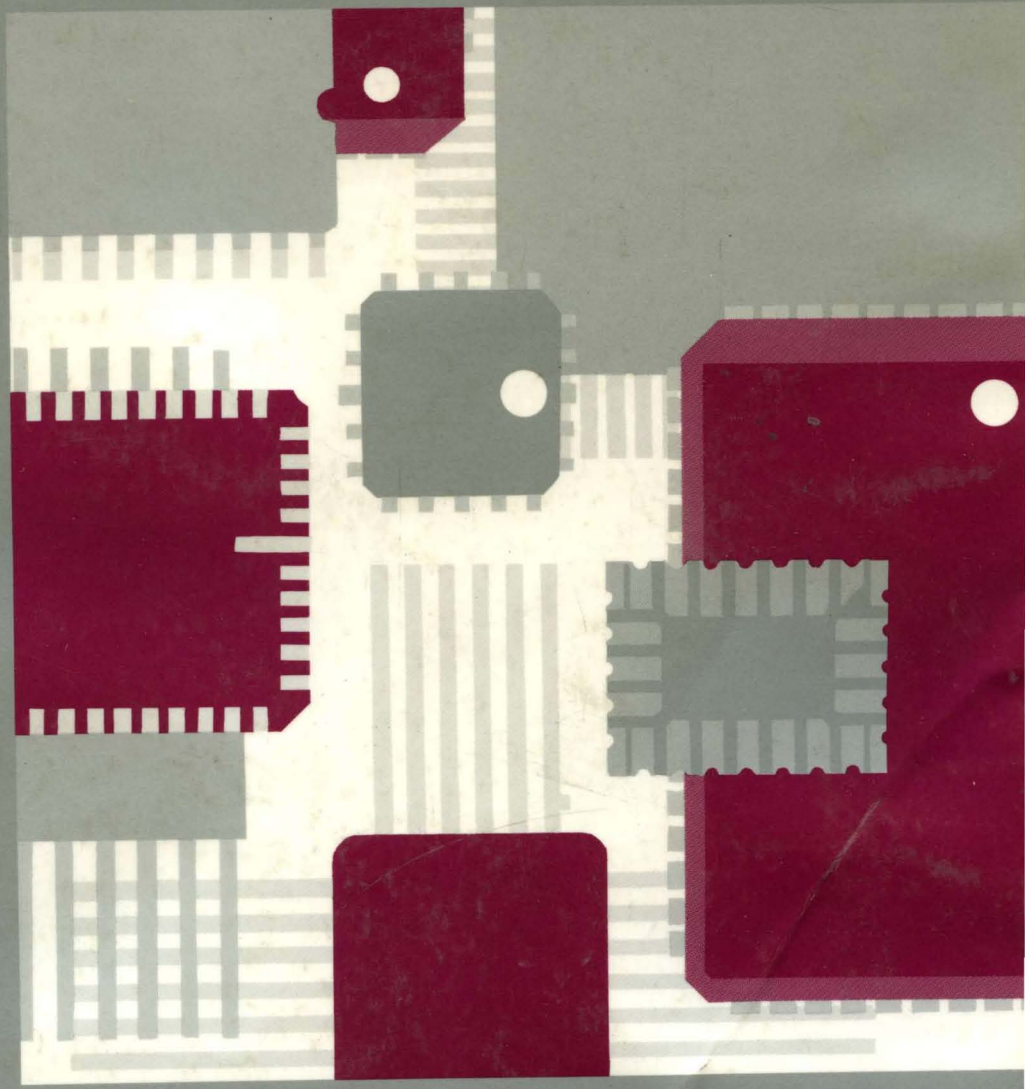
HITACHI®

APPLICATION HANDBOOK

- Application Notes
- Technical Notes
- Technical Briefs



APPLICATION HANDBOOK

- Application Notes
- Technical Notes
- Technical Briefs

HITACHI®

M00T021

August 1992 Printed in U.S.A.

Application Handbook Index

Application Notes, Technical Notes and Briefs

SECTIONS:

HITACHI®

HITACHI APPLICATION HANDBOOK

Table of Contents

HITACHI APPLICATION HANDBOOK

Table of Contents (cont'd)

HITACHI APPLICATION HANDBOOK

Table of Contents (cont'd)

Application Handbook

Application Notes, Technical Notes and Briefs

HITACHI®

Section 1

Introduction

HITACHI®

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

FOREWORD

In this manual you will find a collection of Application Notes, Technical Briefs, TechNotes, and Question & Answers written by Hitachi, Ltd. and Hitachi America, Ltd. This collection was compiled through the efforts of the Technical Marketing Group for Hitachi, Ltd., and the Application Engineering Group and the Field Applications Engineers for Hitachi America, Ltd.

The documents in this handbook included design ideas and examples, application tips and hints, and also product tutorials. They are designed to assist the user in further understanding the technical information already available on each of the products. At this time we would like to extend thanks to the individual contributors of Hitachi America, Ltd. for making this handbook possible.

Stan Ayers, Field Application Engineer
Tom Hampton, Manager, Application Engineering
Carol Jacobson, Senior Application Engineer
Amelia Lam, Application Engineer
Marnie Mar, Senior Field Application Engineer
Oomer Serang, Senior Field Application Engineer
Kash Yajnik, Senior Application Engineer
Paul Yiu, Associate Application Engineer

This title is not all inclusive, so please check with your nearest Hitachi representative for additional information.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Section 2

HD64180 Family

HITACHI®

HD641180X, HD643180X, HD647180X

Technical Q and A

Application Note

Preface

The HD643180X, a member of the HD64180 family, is a single-chip microcontroller incorporating 16 kbytes of mask ROM. The HD647180X is a ZTAT™ (zero turnaround time) microcontroller that incorporates 16 kbytes of programmable ROM instead of mask ROM. The HD641180X has no internal ROM. The HD641180X, HD643180X and HD647180X incorporate the following on a single chip:

- 512 bytes of RAM
- Memory management unit (MMU)
- DMA controller
- Timer
- Asynchronous serial communication interface (ASCI)
- Clock synchronous serial I/O port (CSI/O)
- Analog comparator
- Parallel I/O pins

Note: ZTAT™ is a trademark of Hitachi

How to Use This Technical Q&A Manual

This technical manual contains answers to questions that many users have asked regarding Hitachi microcontrollers. It is intended to supplement the explanations in the current data books and user's manuals. Thus, please use this manual together with the data books and user's manuals.

If any further questions arise as you use this manual and the products described, please do not hesitate to get in touch with your nearest Hitachi semiconductor sales office.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Contents

Bus Interface

Interrupt

I/O Port

Memory

Wait

Reset

Low Power Mode

Refresh

Clock Generator

ASE

Software

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-002B/E
Item	Logical to Physical Address Translation		

Q

Can the MMU base register (MMU common base register and MMU bank base register) be programmed so that common area 1 overlaps with the base area?

A

Yes, depending on the MMU base register programming, common area 1 and the bank area may overlap (figure 1).

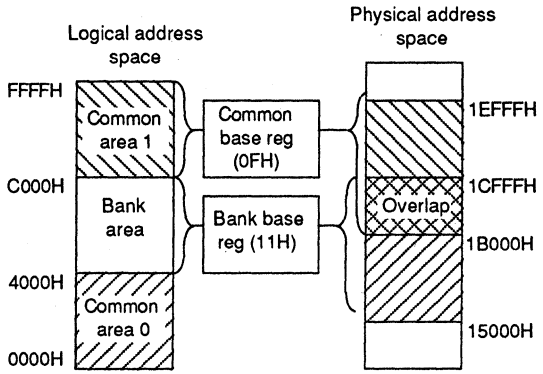


Figure 1 Overlapping Common Areas

Classification	
√	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-003A/E
Item	DE1 Bit in the DMA Status Register (DSTAT)		

Q

1. How long is DMA transfer disabled when the DE1 bit is set to 0?

2. How does DMA restart?

A

1. DMA transfer is disabled until DE1 is reset to 1. Write 0 to bit $\overline{\text{DWE1}}$ before performing any software write to DE1.

2. If memory ↔ memory DMA transfer is executed in burst mode, DMA transfer cannot be interrupted. It can only be interrupted in memory ↔ memory cycle steal mode, memory ↔ I/O, or memory ↔ memory-mapped I/O transfer mode.

   To restart DMA transfer, set DE1 to 1.

Classification	
	MMU
√	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-004B/E
Item	DME (DMA Master Enable) Bit in DMA Status Register		

Q

When $\overline{NMI}$ occurs, DME is reset to 0 and DMA operation is disabled, passing control to the CPU.

1. How is DMA operation timing halted?

2. How does DMA operation restart?

A

1. When $\overline{NMI}$ occurs, the CPU takes control after the current DMA cycle is completed (figure 1)

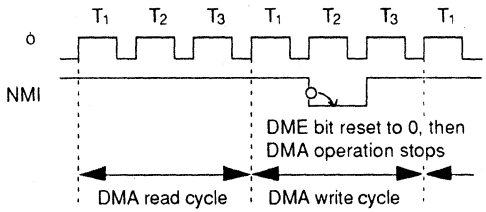


Figure 1 $\overline{NMI}$ Timing

2. To restart DMA operation, set DE bit (DE0 or DE1) to 1. (This operation sets DME to 1.) The following program restarts DMAC:

```
LD A, 80H
OUT0 (30H), A
```

Classification

	MMU
√	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

QA641-054A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-005A/E
Item	$\overline{\text{DWE}}$ Bit in DMA Status Register		

Q

What is the function of the $\overline{\text{DWE}}$ bit in the DMA status register?

A

The DE bit enables DMA operation for the internal DMAC, while the $\overline{\text{DWE}}$ bit enables a software write to the corresponding DE bit, for a specific channel operation.

Classification	
	MMU
√	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-006B/E
Item	Memory ↔ I/O Transfer (channel 0)		

Q

To enable $\overline{DREQ}_0$ input, both A_{17} and A_{16} of the I/O address must be set to 0.

Is the DMA requested by $\overline{DREQ}_0$ accepted if either A_{17} or A_{16} is set to 1?

A

No. If either A_{17} or A_{16} is set to 1, $\overline{DREQ}_0$ is disabled and the DMA request is not accepted.

To use $\overline{DREQ}_0$ input as DMA request, set the bank bit (A_{16}, A_{17}) according to tables 1 and 2.

Table 1 Source Address Register

SAR18	SAR17	SAR16	DMA Request
Don't care	0	0	$\overline{DREQ}_0$
Don't care	0	1	RDRF (ASCI ch0)
Don't care	1	0	RDRF (ASCI ch1)
Don't care	1	1	Reserved

Table 2 Destination Address Register

DAR18	DAR17	DAR16	DMA Request
Don't care	0	0	$\overline{DREQ}_0$
Don't care	0	1	TDRE (ASCI ch0)
Don't care	1	0	TDRE (ASCI ch1)
Don't care	1	1	Reserved

Classification

	MMU
√	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-007B/E
Item	Memory ↔ ASCI DMA Transfer		

Q

To execute memory ↔ ASCI DMA transfer, program DMA source/destination address register as follows:

1. Set bits A_0–A_7 to the address of the ASCI transmit or receive data register
2. Set bits A_8–A_{15} to 00H
3. Set bits A_{16}, A_{17} to 0, 1 or 1, 0

Can the memory ↔ ASCI DMA transfer be executed correctly if bits A_8–A_{15} in the DMA source/destination address register are not set to 00H?

A

No, if bits A_8–A_{15} in the DMA source/destination address register are not set to 00H, memory ↔ ASCI DMA transfer cannot be executed correctly.

For example, to execute ASCI (channel 0) RDR → memory DMA transfer, set bits A_0–A_7 to 08H, bits A_8–A_{15} to 00H, and bit A_{16}, A_{17} to 0, 1.

Classification	
	MMU
√	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment Setting bits A_8–A_{15} to anything other than 00H causes the internal DMAC to access another I/O address, not RDR. (DMA request from ASCI channel 0 is not reset).

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-008A/E
Item	Memory (specified in application program) ↔ I/O DMA Transfer		

Q

Is it possible to execute memory (specified in application program) ↔ I/O DMA transfer indepenently of the MMU base register?

A

No, to execute memory (specified in application program) ↔ I/O DMA transfer correctly, the physical source address must be defined as follows:

1. Software calculates the physical source address of the data area using the logical address and the base register
2. The calculated physical source address is loaded into the DMA source address register

If the physical address is known, it can be loaded into the DMA address register directly, but if the DMA transfer is executed within the logical memory area, block transfer instructions can be used.

Classification	
	MMU
√	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-009B/E
Item	Memory ↔ I/O (Z80SIO) DMA Transfer		

Q

When memory ↔ I/O (Z80SIO) DMA transfer is executed while $\overline{DREQ}$ is programmed for level sense, DMA transfer does not complete correctly.

Are there any restrictions on DMA operation? (RDY signal from Z80SIO is input to $\overline{DREQ}$ of HD64180.)

A

The Z80SIO RDY signal is negated during DMA write cycle to the peripheral LSI. Therefore, if the $\overline{DREQ}$ is programmed for level sensing, an additional DMA cycle starts since RDY is negated after $\overline{DREQ}$ signal sampling (figure 1)

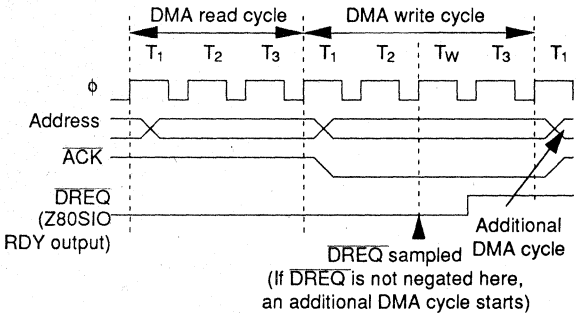


Figure 1 DMA Timing

Classification	
	MMU
√	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment Take one of three measures: 1) Program $\overline{DREQ}$ for edge sensitivity, 2) Insert a wait state during DMA write cycle to modify $\overline{RDY}$ response timing 3) Mask $\overline{DREQ}$ (RDY) signal by the $\overline{ACK}$ signal.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-040A/E
Item	DMAC Priority		

Q

Which has higher priority, ch 0 memory ↔ memory DMA transfer or ch 1 memory ↔ I/O DMA transfer?

Is it possible to execute ch 1 memory ↔ I/O DMA transfer before ch 0 memory ↔ memory DMA transfer?

A

DMA ch 0 has priority. However, when ch 1 DMA request is generated when ch 1 is enabled and ch 0 is disabled, and the ch 1 DMA is generated continuously, the ch 1 DMA transfer can be performed.

During ch 0 memory ↔ I/O DMA transfer, a ch 1 DMA request can be accepted if no more ch 0 DMA requests are generated.

During ch 0 memory ↔ memory DMA transfer, ch 1 DMA requests are ignored until ch 0 DMA transfer ends.

Classification	
	MMU
√	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-042A/E
Item	$\overline{\text{DACK}}$ Signal Generation		

Q

How can the $\overline{\text{DACK}}$ signal, which indicates DMA transfer completion, be generated during ch 0 memory ↔ external I/O DMA transfer?

A

When external I/O is accessed during a DMA cycle, the external I/O address is output through the address bus. At this time, the $\overline{\text{IOE}}$ signal and address output are decoded to generate a $\overline{\text{DACK-1}}$ signal (figure 1, ①).

When external I/O is accessed to initialize registers during a CPU cycle, $\overline{\text{DACK-1}}$ and ST signals are logical-ORed to generate $\overline{\text{DACK-2}}$ to distinguish a DMA from a CPU cycle (figure 1, ②).

Figures 1 and 2 (next page) show the $\overline{\text{DACK}}$ generation circuit and signal timing, respectively.

Classification	
	MMU
√	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

HD64180 App. Note

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-042A-2/E
Item	$\overline{\text{DACK}}$ Signal Generation		

A

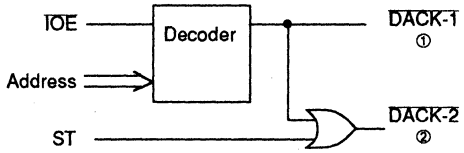


Figure 1 Circuit Example

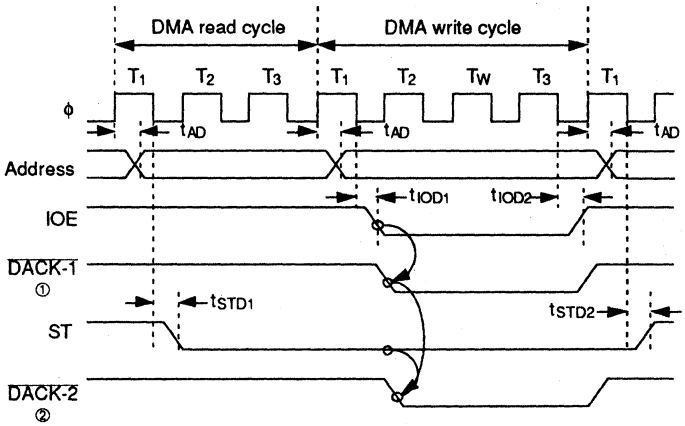


Figure 2 Timing Chart

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-065A/E
Item	DMA Transfer		

Q

How does the DMA transfer execute when BCR is set to 0000H?

A

When BCR is set to 0000H, 64 kbytes are transferred.

	Classification
	MMU
√	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-010A/E
Item	Asynchronous Serial Communication Interface (ASCI) Break Level Transfer		

Q

Is it possible to perform break level transfer with ASCI by software?

A

No, the (HD64180) ASCI cannot perform break level transfer through software.

However, break level can be transferred if an external circuit is connected to to the $\overline{RTS_0}$ pin and the user system port (figure 1).

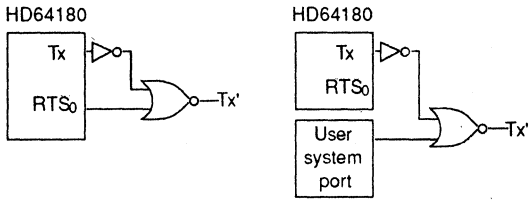


Figure 1 Break Level Transfer Circuit Example

If the $\overline{RTS_0}$ pin in ASCI control register A or port data register is set to 1, break level 0 can be transferred.

Classification	
	MMU
	DMAC
√	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-011A/E
Item	ASCI Baud Rate Calculation		

Q

How is ASCI baud rate calculated?

A

The following expression shows how to calculate ASCI baud rate:

$$\text{baud rate} = \frac{\text{system clock frequency}}{\text{(sampling rate) (PS bit) (divider ratio set by SS0–SS2)}}$$

Note: Sampling rate: 16 or 64
PS bit: 10 or 30
SS0–SS2: 1, 2, 4, 8, 16, 32, or 64

Classification	
	MMU
	DMAC
√	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-043A/E
Item	ASCI Halted by $\overline{CTS}$		

Q

What is the status of the ASCI after it is halted by negation of the $\overline{CTS}$ signal?

A

To continue ASCI transmission, write data into the ASCI transmit data register (TDR).

Normally, data can be programmed into the TDR either by ASCI transmit interrupt by the TDRE flag, or by TDRE polling.

However, if $\overline{CTS}$ is negated (high), the TDRE flag is masked and is alway read as 0. Data cannot be programmed into the TDR for the following reasons:

1. ASCI transmit interrupt by TDRE flag is disabled
2. Data programming routine is disabled, because the TDRE flag always reads 0 when polled

Therefore, ASCI is idle when $\overline{CTS}$ is negated (high).

Figure 1 shows timing for ASCI transmission control by $\overline{CTS}$.

Classification

	MMU
	DMAC
√	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-043A-2/E
Item	ASCI Halted by $\overline{CTS}$		

A

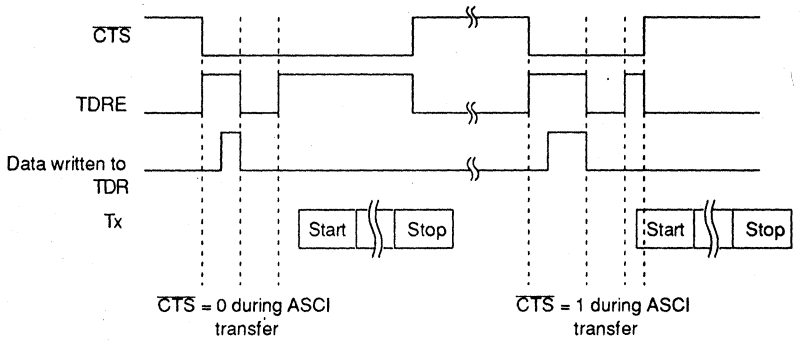


Figure 1 Timing of ASCI Transmission Control by $\overline{CTS}$

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-044A/E
Item	$\overline{DCD}$ Pin and DCD Flag		

Q

Is the DCD flag (ASCI status register ch 0, bit 2) reset when the $\overline{DCD}$ pin is asserted low?

A

No, the DCD flag is not reset unless the ASCI status register is read. This allows the $\overline{DCD}$ interrupt to be serviced correctly.

If the $\overline{DCD}$ pin and the DCD flag were reset simultaneously, the $\overline{DCD}$ interrupt request would always be cleared and could not be serviced when a higher priority interrupt occurred simultaneously with the $\overline{DCD}$ interrupt (figure 1). Figure 2 shows the actual function used by the HD647180X.

Classification	
	MMU
	DMAC
√	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-044A-2/E
Item	$\overline{DCD}$ Pin and DCD Flag		

A

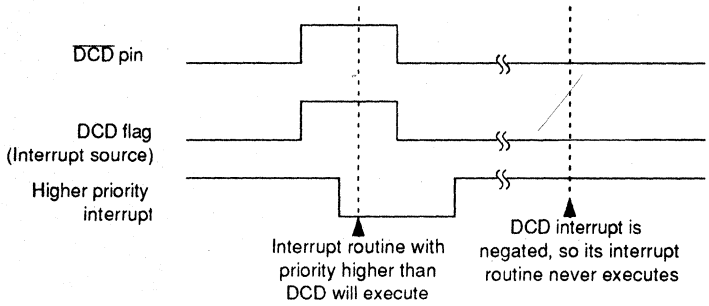


Figure 1 DCD Flag Synchronized with $\overline{DCD}$ Pin

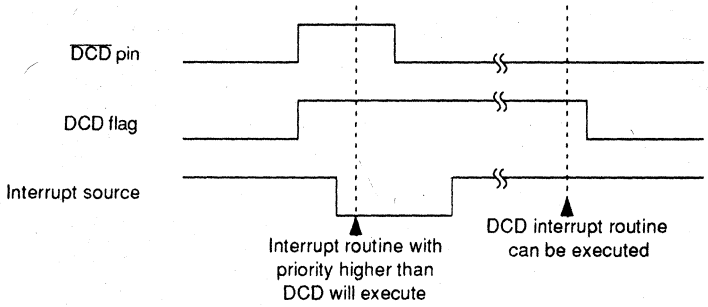


Figure 2 HD647180X DCD Function

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-045A/E
Item	External Clock Divide Ratio		

Q

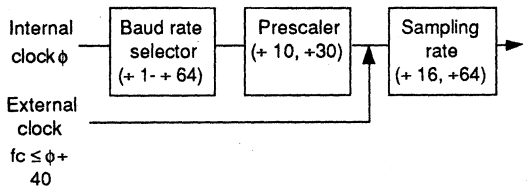
Can the internal baud rate generator divider circuit be used when the ASCI uses an external clock?

A

No, the divider cannot be used when the ASCI uses an external clock. Therefore, the external clock must be $\phi \div 40$. However, sampling rate can be controlled by the DR bit of the ASCI control register (table 1).

Table 1 Sampling Rate

DR Bit	Sampling Rate
0	÷ 16
1	÷ 64



Classification

	MMU
	DMAC
√	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-066A/E
Item	ASCI Data Sampling		

Q

Where on the baud rate clock does the CPU sample data?

A

The CPU samples ASCI data at the falling edge of the baud rate clock.

Classification	
	MMU
	DMAC
√	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-067A/E
Item	Restarting ASCI from I/O Stop Mode		

Q

How does ASCI restart from IOSTOP mode?

A

First, resetting the IOSTP bit in the I/O control register (ICR) causes I/O stop mode recovery.

Then setting the receive enable (RE) bit or transmitter enable (TE) bit of the ASCI control register restarts ASCI transmission.

Classification	
	MMU
	DMAC
√	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-068A/E
Item	TSR Status		

Q

When transmit enable (TE) is reset to 0, the transmitter is disabled.

Does this operation initialize the transmit shift register (TSR)?

A

No, resetting TE to 0 does not initialize the transmit shift register (TSR) (figure 1).

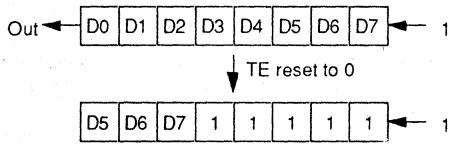


Figure 1 Transmit Shift Register

Classification	
	MMU
	DMAC
√	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-069A/E
Item	Transmit Interrupt Timing		

Q

When does the CPU acknowledge the transmit interrupt in the ASCI transmit sequence?

A

The CPU acknowledges the transmit interrupt when the start bit goes out to the TXA pin.

Classification	
	MMU
	DMAC
√	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-070A/E
Item	RE and External Serial Clock for the CSI/O		

Q

What is the relation between receive enable (RE) and the external serial clock (t_x in figure 1)?

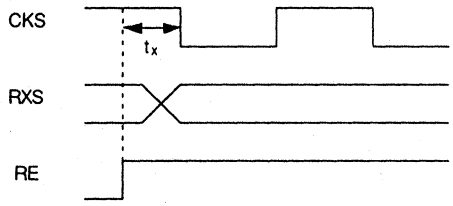


Figure 1 Receive Timing

A

Time t_x must be more than 5 system clocks (5ϕ). If it is less than 5ϕ, CSI/O receive operation cannot start correctly.

Classification	
	MMU
	DMAC
	ASCI
√	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-012B/E
Item	Timer Output		

Q

Does the timer (PRT) channel 0 provide a timer output function?

A

No, PRT channel 1 should be used for timer output.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
√	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-013A/E
Item	Timer (PRT) Count Down Using External Clock		

Q

Can the PRT count down using the external clock?

A

No. The PRT can count down using only the ϕ clock (divided by 20).

	Classification
	MMU
	DMAC
	ASCI
	CSI/O
√	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-071A/E
Item	Reload Timing		

Q

When the timer data register (TMDR) counts down to 0, it is automatically reloaded with the contents of the timer reload register (RLDR).

How long does reloading take?

A

TMDR is reloaded with the contents of RLDR after 20 ϕ cycles.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
√	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-072A/E
Item	TMDR Count Down		

Q

While the timer data register (TMDR) counts down, when timer count down enable (TDE) is reset to 0, what is the status of TMDR and the reload register (RLDR)?

A

The TMDR and RLDR remain the same when the counter stops.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
√	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-014B/E
Item	Bus State during Internal I/O Access		

Q

1. What is the bus status during internal I/O access?
2. What happens if external I/O is assigned to the same address as internal I/O?

A

1. Bus status during internal I/O access is as follows:
   - Data bus
     - — Read: High impedance state
     - — Write: Outputs data
   - Address bus
     - — Read/write: Outputs address
2. When an internal I/O address and an external I/O address overlap, bus status is as follows:
   - Data bus
     - — Read: Reads internal I/O; does not read external I/O
     - — Write: Outputs data to both internal and external I/O
   - Address bus
     - — Read/write: Outputs address

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
√	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-037A/E
Item	E Clock during Sleep Mode or Bus Release Mode		

Q

Is it possible to extend E clock pulse width by inserting wait states (T_W) during sleep mode or bus release mode?

A

No. Because $\overline{WAIT}$ input is ignored during sleep mode or bus release mode, the E clock cycle cannot be extended.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
√	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-038A/E
Item	E Clock Timing during DMA Cycles or Refresh Cycles		

Q

What is the E clock output timing during the DMA or refresh cycle?

A

DMA access memory or I/O duration of E clock output high is identical to the CPU. Table 1 shows output timing.

Table 1 Output Timing

Cycle	Timing
Memory R/W	T_2 rising (↑) to T_3 falling (↓)
I/O read	First T_W rising (↑) to T_3 falling (↓)
I/O write	First T_W rising (↑) to T_3 rising (↑)

During refresh cycles, E clock output is held low.

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	Timer
√	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-046A/E
Item	Data Sampling Timing during Memory Read		

Q

Does the CPU sample data at the rising edge of T_3 during opcode fetch cycles and at the falling edge of T_3 during operand and data read cycles?

A

Yes, the CPU samples the opcode on the data bus at the rising edge of T_3 while it samples operands and data at the falling edge of T_3 (table 1).

Table 1 Sampling Timing

CPU Cycle	Sampling Timing
Opcode fetch cycle	T_3 rising edge (↑)
Data and operand fetch cycle	T_3 falling edge (↓)

	Classification
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
√	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-015B/E
Item	Interrupt during MMU Operation		

Q

How will the MMU be affected if an interrupt occurs during its operation?

A

If an interrupt occurs during MMU operation, the interrupt vector is relocated according to the MMU base register programming. Therefore, the interrupt vector should be defined with reference to MMU base register programming (figure 1).

However, the interrupt vector can be located in common area 0, which is always located in the same logical address space.

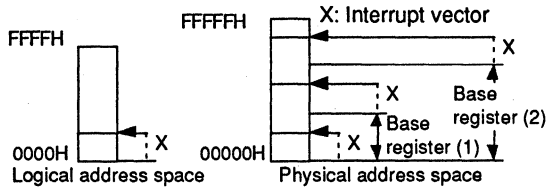


Figure 1 Interrupt Vector Generation during MMU Operation

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-016A/E
Item	Interrupt during DMA Operation		

Q

How will the DMAC be affected if an interrupt occurs during its operation?

A

1. If an $\overline{NMI}$ occurs, DMAC operation is disabled.
2. If an $\overline{INT}$ or an internal interrupt occurs during burst mode memory ↔ memory DMA operation, the interrupt is ignored.
3. If an $\overline{INT}$ or an internal interrupt occurs during cycle steal mode memory ↔ memory DMA operation, the interrupt is acknowledged and the interrupt sequence (CPU cycle) and DMAC read/write (DMAC cycle) are executed as in figure 1.

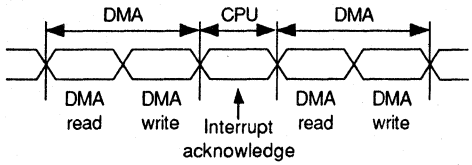


Figure 1 Interrupt during Cycle Steal Mode DMA

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-017A/E
Item	$\overline{INT}_0$ Mode 2		

Q

In Z80 $\overline{INT}_0$ mode 2, the LSB of the lower vector in the 16-bit vector address (A_0) is always 0.

In the HD647180X, is the LSB of the lower vector (D_0) automatically set to 0 in $\overline{INT}_0$ mode 2?

A

No, in Z80 $\overline{INT}_0$ mode 2, the LSB of the lower vector is not automatically set to 0. The Z80 data book explains that the LSB (A_0) must be set to 0.

In HD647180X $\overline{INT}_0$ mode 2, the LSB of the lower vector (D_0) must be set to 0 since $\overline{INT}_0$ mode 2 requires a 2-byte vector.

However, even if the LSB of the lower vector (D_0) is set to 1, the interrupt sequence is executed correctly.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-018A/E
Item	$\overline{NMI}$ during Interrupt Acknowledge Cycle		

Q

Is $\overline{NMI}$ acknowledged during the interrupt acknowledge cycle, such as for $\overline{INT}$?

A

Yes, one instruction (excluding EI and DI instructions) is executed after the $\overline{INT}$ acknowledge cycle, then the $\overline{NMI}$ acknowledge cycle starts.

The $\overline{NMI}$ response sequence during the $\overline{NMI}$ acknowledge cycle is the same.

	Classification
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-019A/E
Item	Interrupt after Reset		

Q

Is $\overline{NMI}$ or $\overline{INT}$ acknowledged immediately after reset?

A

No, for three cycles immediatley after reset (power-on reset cycle), $\overline{NMI}$ and $\overline{INT}$ are disabled.

After these three cycles, the first instruction is executed and interrupts are enabled. Figure 1 shows the timing.

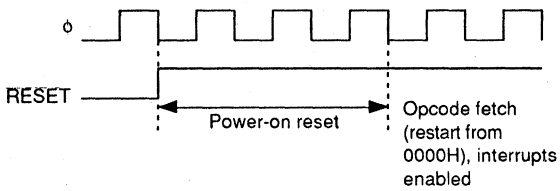


Figure 1 Power-On Reset Timing

Note: $\overline{NMI}$ is latched immediately after the power-on reset cycle.

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-020B/E
Item	Interrupt during Refresh Cycle		

Q

Is an interrupt ($\overline{NMI}$ or $\overline{INT}$) acknowledged during refresh cycles?

A

No, interrupts are ignored during refresh cycles. However, $\overline{NMI}$ is acknowledged immediately after refresh cycles during instruction execution because $\overline{NMI}$ is edge sensitive. Figure 1 shows $\overline{NMI}$ acknowledge timing after refresh cycle.

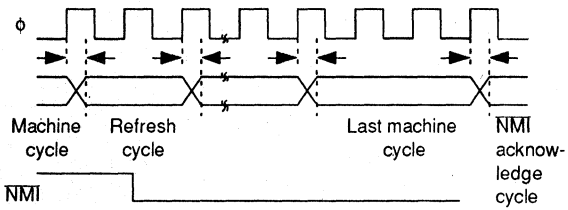


Figure 1 Refresh Timing

$\overline{INT}$ ($\overline{INT_0}$, $\overline{INT_1}$, and $\overline{INT_2}$) is ignored during refresh cycles. If $\overline{INT}$ remains active after the refresh cycle, it is acknowledged during the instruction just after refresh.

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X.Hardware Manual

Other Data

Reference Q&A

Comment

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-021B/E
Item	$\overline{NMI}$ Acknowledge		

Q

Is $\overline{NMI}$ acknowledged if it occurs during the timing sequence in figure 1?

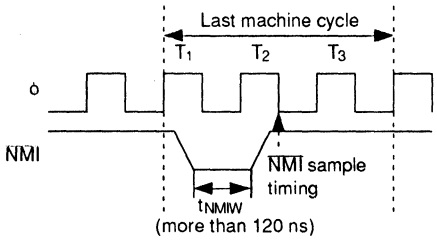


Figure 1 $\overline{NMI}$ Timing

A

Yes, if t_{NMIW} ($\overline{NMI}$ pulse width) is 120 ns or more, $\overline{NMI}$ is sampled and $\overline{NMI}$ acknowledge cycle begins after the last MC (machine cycle)

If the $\overline{NMI}$ is asserted low for t_{NMIW} or longer between 1 and 2 in figure 2, it will be sampled at the falling edge of the clock marked 2.

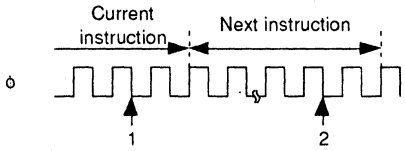


Figure 2 $\overline{NMI}$ Timing

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-047A/E
Item	Interrupt Acknowledge Timing after EI Instruction Execution		

Q

When is an interrupt acknowledged after an EI instruction?

A

Maskable interrupts ($\overline{INT}_0$, etc.) are acknowledged in the last machine cycle of any instruction cycle other than EI.

Note that no interrupts can be acknowledged during EI instruction execution. Therefore, if an interrupt occurs immediately before or during an EI instruction cycle, it is acknowledged after the end of the RETI instruction cycle following the EI instruction.

For example:

```
.
.
.        ← Interrupt request
EI
RETI     ← Interrupt request acknowledged during this
           instruction
(Interrupt acknowledge cycle)
```

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-048A/E
Item	Interrupt Sampling during Block Transfer Instruction Execution		

Q

Normally, the CPU samples interrupts at the falling edge of the ϕ clock pulse prior to state T_3 or T_i in the last machine cycle.

When does the CPU sample interrupts during a block transfer instruction which may require a hundred or more machine cycles?

A

The CPU samples interrupts at the falling edge of the ϕ clock pulse prior to state T_3 or T_i in the last machine cycle of each one byte transfer (figure 1).

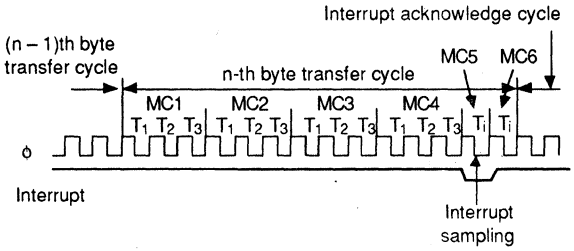


Figure 1 Interrupt during Block Transfer

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-049A/E
Item	Interrupt during SLP Instruction Cycle		

Q

What is the CPU status when an interrupt occurs during SLP instruction execution?

A

How the CPU operates when an interrupt occurs during SLP instruction execution depends on whether it is the HD647180X. The different responses are shown on the next page.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-049A-2/E
Item	Interrupt during SLP Instruction Cycle		

A

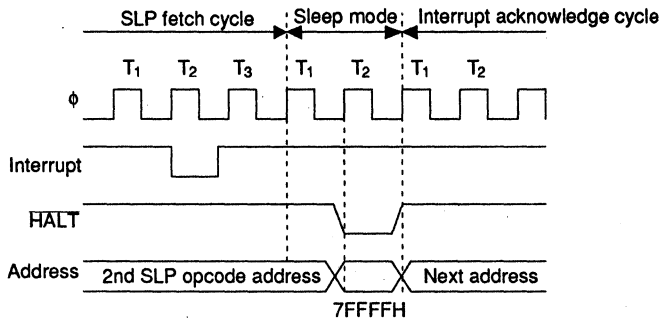


Figure 1 HD647180X SLP Timing

When an interrupt is sampled during an SLP instruction cycle, $\overline{\text{HALT}}$ is asserted low for 1 clock state, and the address bus outputs FFFFFH.

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-050A/E
Item	$\overline{INT}_0$ Mode 0		

Q

If the CALL instruction (three-byte instruction) is executed during $\overline{INT}_0$ mode 0 acknowledge cycle, the CPU cannot return from the interrupt service correctly. Why is this?

A

$\overline{INT}_0$ mode 0 operates as follows:

1. Stacks PC contents by an instruction, usually (one-byte) RST, fetched during $\overline{INT}_0$ mode 0 acknowledge cycle
2. Stops incrementing the PC during $\overline{INT}_0$ mode 0 acknowledge cycle
3. Executes instruction fetched from data bus during interrupt acknowledge cycle

However, if the (three-byte) CALL instruction, which requires three machine cycles to fetch including the operand, is executed during $\overline{INT}_0$ mode 0 acknowledge cycle, PC increment stops only during interrupt acknowledge cycle (one machine cycle) and is incremented by 2 during the rest of the CALL instruction (operand fetch). As a result, PC + 2 is stacked as the return address. Therefore, decrement the stacked PC value by 2 in software to return from the interrupt correctly.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-051A/E
Item	$\overline{NMI}$ Interrupt Sampling Timing		

Q

$\overline{NMI}$ is sampled at the falling edge of a ϕ clock state prior to T_3 or T_i in the last machine cycle of each instruction.

1. When is $\overline{NMI}$ sampled if the last machine cycle is an internal T_i cycle?
2. How about $\overline{INT}$?

A

Both $\overline{NMI}$ and $\overline{INT}$ are always sampled at the falling edge of a ϕ clock pulse prior to state T_3 or T_i of the last machine cycle. The $\overline{NMI}$ sampling is not affected by the number of internal T_i cycles (figure 1).

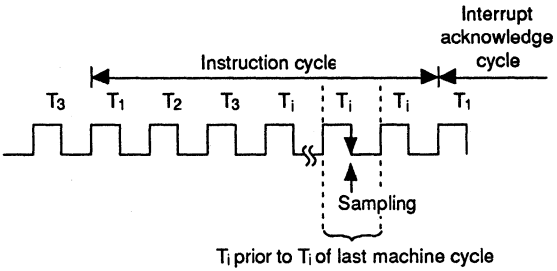


Figure 1 Interrupt Sample Timing during T_i

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-052A/E
Item	Status Bit during TRAP		

Q

1. What happens if an additional TRAP occurs before the INT/TRAP control register TRAP bit is cleared?
2. What is the status of the TRAP and UFO bits in this case?

A

1. An additional TRAP interrupt occurs.
2. The TRAP bit remains 1 since it can be cleared only by software.

   The UFO bit remains unchanged since it cannot be modified while the TRAP bit = 1.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment UFO bit: Indicates if TRAP occured in 2nd or 3rd opcode fetch cycle:
UFO = 0: TRAP occurred in 2nd opcode fetch cycle
UFO = 1: TRAP occurred in 3rd opcode fetch cycle

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-053A/E
Item	PC Stacking during TRAP		

Q

Why is the stacked PC value different for TRAP occurrence during second opcode fetch and during third opcode fetch?

A

Table 1 summarizes CPU operations when TRAP occurs during the second and third opcode fetches.

Table 1 CPU Operations during TRAP

Machine	TRAP during Second Opcode Fetch		TRAP during Third Opcode Fetch	
Cycle	Status	PC	Status	PC
1	TRAP occurrence	PC	TRAP occurrence	PC
2	Internal operation	PC	Memory read	PC
3	Stack	PC	Internal operation	*PC = PC + 1*
4	—	—	Stack	PC – 1

When the TRAP occurs in the second opcode, the CPU stacks the PC for the undefined opcode's location .

When it occurs in the third opcode, the CPU stacks PC – 1 for the undefined opcode's location .

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-054A/E
Item	$\overline{\text{NMI}}$ during DMA Transfer		

Q

What happens to DMAC after $\overline{\text{NMI}}$ assertion?

A

When $\overline{\text{NMI}}$ is asserted low during DMA transfer, the DMA transfer ends at the end of the current DMA cycle.

However, note that the $\overline{\text{NMI}}$ acknowledge cycle begins at different times, depending on the CPU status before DMA transfer (figures 1, 2, and 3). In addition, $\overline{\text{NMI}}$ is sampled twice to stop the DMA cycle and start the $\overline{\text{NMI}}$ acknowledge cycle.

DMAC operations can be restarted by writing to the corresponding channel's DE bit.

$\overline{\text{NMI}}$ acknowledge cycle timings are shown on the next pages.

	Classification
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-054A-2/E
Item	$\overline{NMI}$ during DMA Transfer		

A

1. When DMA transfer starts during instruction execution cycle

   a. When the DMA cycle starts during the instruction execution cycle, before the last machine cycle (T_1, T_2, and T_3) of instruction A, $\overline{NMI}$ is sampled at the falling edge of T_2 in the last machine cycle of A (figure 1).

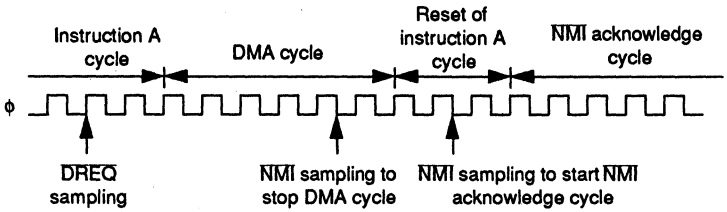


Figure 1 DMA Cycle Starting Before Last Machine Cycle

   b. When the DMA cycle starts during the instruction execution cycle, before the last internal cycle (T_i) of instruction A, $\overline{NMI}$ is sampled during the DMA cycle (figure 2).

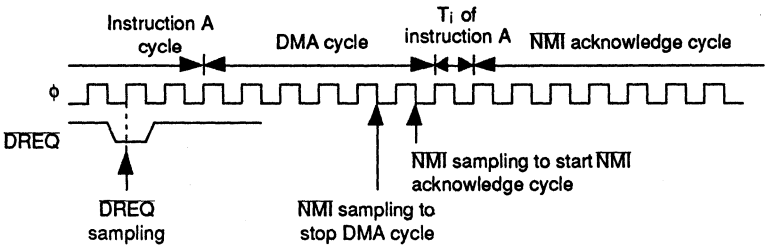


Figure 2 DMA Cycle Starting Before Last Internal Cycle

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-054A-3/E
Item	$\overline{NMI}$ during DMA Transfer		

A

2. When DMA transfer starts at the end of the instruction execution cycle, $\overline{NMI}$ is sampled at the next falling edge of T_2 or T_i of the last machine cycle of the next instruction, B (figure 3).

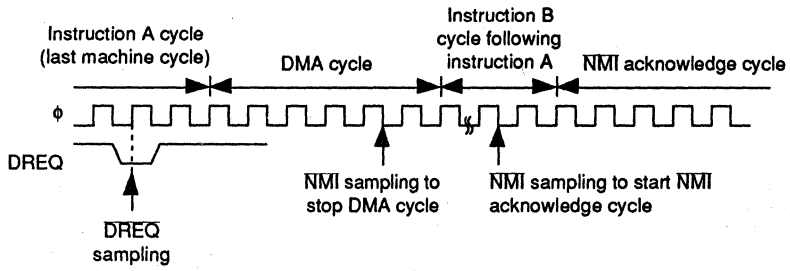


Figure 3 DMA Cycle Starting at End of Instruction Cycle

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-055A/E
Item	$\overline{DREQ_i}$ and $\overline{NMI}$		

Q

What happens to DMAC operation if $\overline{NMI}$ is asserted low while the DMAC operates under the control of the $\overline{DREQ_i}$ pin?

A

DMAC operation is suspended and $\overline{NMI}$ is sampled with the timing shown in figure 1.

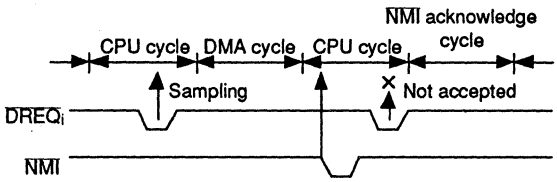


Figure 1 DMA Cycle Stopped by $\overline{NMI}$

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-055A-2/E
Item	$\overline{DREQ}_i$ and $\overline{NMI}$		

A

Note that if $\overline{DREQ}_i$ and $\overline{NMI}$ are asserted simultaneously, $\overline{NMI}$ sampling has priority (figure 2).

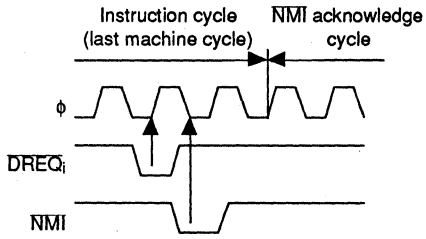


Figure 2 $\overline{DREQ}_i$ and $\overline{NMI}$ Conflict

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-056A/E
Item	Internal Interrupt Sampling Timing		

Q

External interrupts are sampled at the falling edge of state T_2 or T_i in the last machine cycle.

When are internal interrupts sampled?

A

During a DMA cycle, internal interrupts from sources like DMAC, ASCI, timer, and CSIO are not sampled.

They are sampled at the falling edge of state T_2 or T_i in the last machine cycle of the instruction cycle following the DMA cycle (figure 1).

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-056A-2/E
Item	Internal Interrupt Sampling Timing		

A

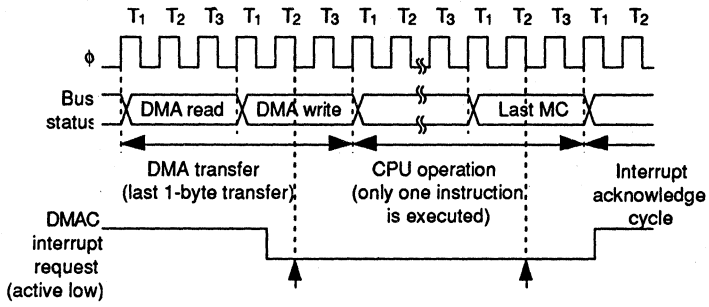


Figure 1 Internal Interrupt Sampling

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-057A/E
Item	$\overline{\text{INTA}}$ Signal Generation		

Q

The HD64180 can be interfaced to the 8259 to control I/O interrupts.

1. How can we generate an $\overline{\text{INTA}}$ signal to be input to the 8259 from the HD64180?
2. Are there any precautions?

A

1. Three $\overline{\text{INTA}}$ signal pulses must be input when the 8259 is used to control interrupts:

   a. One $\overline{\text{INTA}}$ pulse for opcode fetch

   b. Two $\overline{\text{INTA}}$ pulses for operand fetch

   The $\overline{\text{INTA}}$ pulse for opcode fetch can be produced by $\overline{\text{LIR}}$ and $\overline{\text{IOE}}$. The $\overline{\text{INTA}}$ pulse for operand fetch can be produced by $\overline{\text{RD}}$.

   This interface is the same as for Z80 and 8259.

   Figure 1 shows an example of an $\overline{\text{INTA}}$ signal generation circuit.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

QA641-050A

Comment

This circuit is for reference only. Check logic and timing carefully for your application.

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-057A-2/E
Item	$\overline{\text{INTA}}$ Signal Generation		
A			

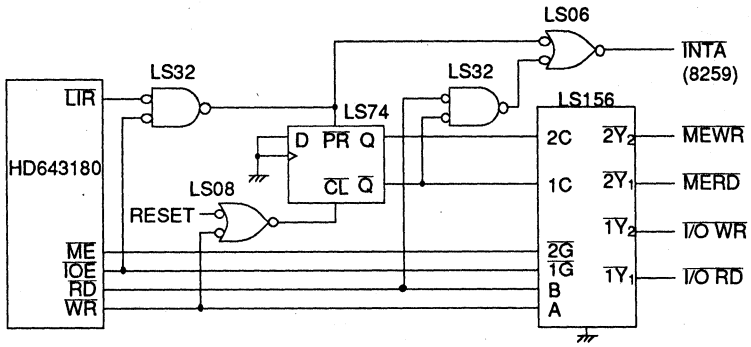


Figure 1 $\overline{\text{INTA}}$ Signal Generation Circuit Example

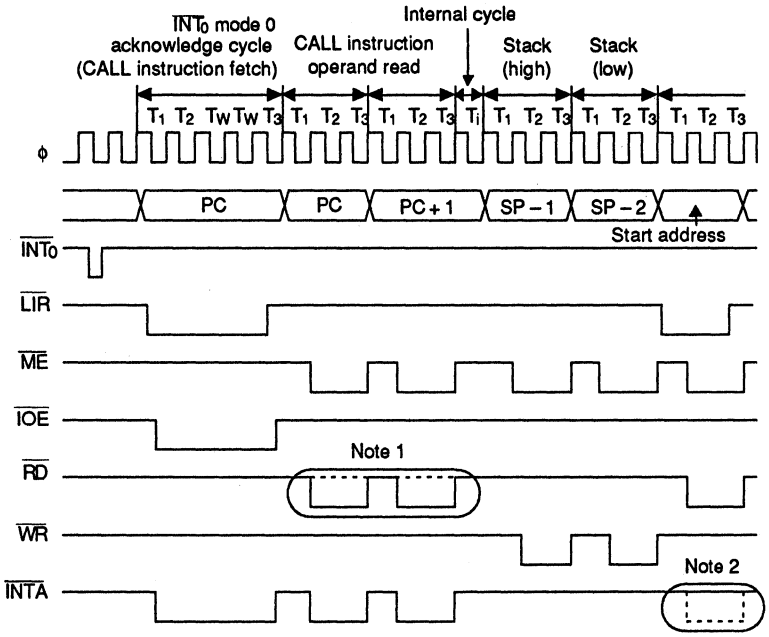
2. Precautions

- This circuit cannot be used when the DMAC is used in the system.
- When signal $\overline{\text{RD}}$ is used to generate the $\overline{\text{INTA}}$ signal for operand read (1b above) $\overline{\text{OE}}$ must be used to avoid data conflict between I/O and memory (note 1 in figure 2).
- In $\overline{\text{INT}}_0$ mode 0, if the RST instruction is executed during its acknowledge cycle, the PC is put on the stack. If a CALL instruction is executed, PC + 2 is put on the stack.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-057A-3/E
Item	$\overline{INTA}$ Signal Generation		

A



Notes: 1. $\overline{RD}$ input to memory pulled high to prevent bus conflict.
2. $\overline{RD}$ input to $\overline{INTA}$ pulled high to prevent 8259 malfunction.

Figure 2 $\overline{INTA}$ Timing

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-073A/E
Item	Interrupt Request during HALT Opcode Fetch		

Q

Can the CPU acknowledge the interrupt request during HALT opcode fetch?

A

Yes. When interrupt enable flag 1 is set to 1, the CPU will acknowledge the interrupt request.

	Classification
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
√	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-074A/E
Item	Sample Mode Programming Pins' Levels		

Q

1. When does the CPU sample the level on the mode programming (MP) pins?
2. Is it possible to change the CPU mode during operation?

A

1. The CPU samples the MP pins' levels at the rising edge of RESET.
2. No. It is impossible to change the CPU mode during CPU operation. (CPU does not check mode pins' levels.)

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
√	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-075A/E
Item	Change from Input to Output		

Q

What does the I/O port output just after it changes from input to output?

A

The I/O ports output depends on the contents of the output data register (ODR). If the data direction register (DDR) is set to 1 and the ODR is not set to output level data, the I/O port outputs undefined data. Therefore, put valid data in the ODR before setting the DDR to 1.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
√	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-076A/E
Item	I/O Port Status in Sleep Mode		

Q

What is the status of the I/O port during sleep mode?

A

The I/O port holds its output levels when the CPU enters sleep mode.

	Classification
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
√	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-077A/E
Item	Port G		

Q

Can port G be left open?

A

No. Connect port G to V_{CC} or GND through a resistance.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
√	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

Other Data

Reference Q&A

Comment

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-120A/E
Item	Notice at alternating Port A's function		

Q

How can we alternate Port A's function ?

A

Port A pins can also be used as ASCI channel 1 pins or DMA channel 1 pins. If you want to use Port A as ASCI ch 1 pins or DMA ch 1 pins, please program DDRA after DERA.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
√	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-078A/E
Item	RAM Relocate Area		

Q

Is it possible to locate the external memory to the RAM relocate area?

A

Yes, it is possible to locate the external memory in any physical address except for the internal RAM (or ROM) areas.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
√	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD643180X, HD647180X	Q&A No.	QA641-121A/E
Item	Inserting Wait State to Internal ROM		

Q

When internal ROM is accesed, are wait states inserted according to the programmed value in operating mode 2?

A

Yes, wait states are inserted for internal ROM.

	Classification
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
√	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-022B/E
Item	WAIT Insertion during Refresh Cycle		

Q

Can WAIT cycles be inserted during refresh cycle by activating the $\overline{WAIT}$ input (figure 1)?

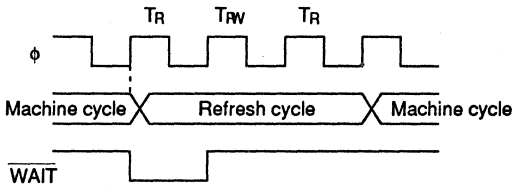


Figure 1 Wait dzuring Refresh

A

No, $\overline{WAIT}$ input is disabled during the refresh cycle. However, the refresh cycle can be programmed to two or three cycles by setting the REFW bit in the refresh control register accordingly (figure 2).

7	6	5	4	3	2	1	0
REFE	REFW					CYC1	CYC0

REFE: 0 = Refresh disabled, 1 = refresh enabled
REFW: 0 = No wait during refresh,
1 = 1 wait inserted during refresh
CYC1, CYC0: Refresh cycle interval

Figure 2 Refresh Control Register

Refresh cycles occur every 10 clock cycles and last for 3 clock cycles after RESET.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
√	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-023B/E
Item	WAIT Function at I/O Access		

Q

Is the WAIT state (T_W) always inserted during I/O access?

A

Yes, at least one wait state is inserted during external I/O access.

During on-chip I/O access, zero to four wait states are automatically generated, depending on the status of CPU and on-chip I/O (ASCI, CSI/O, PRT DATA register access). For internal I/O access, the value of the DMA/WAIT control register is ignored.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
√	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-122A/E
Item	Inserted Wait States		

Q

How many wait states are inserted after the reset start in the single-chip mode?

A

After reset, three wait states are inserted, depending on the initial value.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
√	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-024A/E
Item	Power-On Reset Sequence		

Q

How is the power-on reset sequence performed?

A

Figure 1 shows the power-on reset sequence.

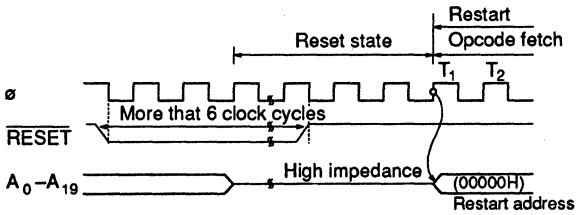


Figure 1 Power-On Reset Sequence

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
√	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

RESET pin should be low for more than 6 clock cycles.

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-058A/E
Item	Control Signal Status after Reset		

Q

What is the status of the control signals after each reset?

A

The $\overline{\text{RESET}}$ signal must be asserted for at least 6 states. Table 1 shows the status of each control signal.

Table 1 Control Signal Status

Control Signal	Status
Address bus	High impedance
Data bus	High impedance
Control signals ($\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{ME}}$, $\overline{\text{IOE}}$, ST, $\overline{\text{LIR}}$, HALT, $\overline{\text{BUSACK}}$, $\overline{\text{TEND}}$,)	High (1)
E	Low (0)
ϕ	Clock output

However, if $\overline{\text{RESET}}$ is not held low for at least 6 clock states at power-on reset, the state of these signals is undefined. For external reset, each signal remains unchanged until it is reset.

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
√	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

QA641-024A

Comment

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-025A/E
Item	Bus Status during Sleep Mode		

Q

What is the bus status when the SLP instruction is executed?

A

Table 1 shows the bus status.

Table 1 Bus Status

Signals	Status
Address bus	High (A_0–A_{19} = FFFFFH)
Data bus	High impedance
Control signals	Inactive

	Classification
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
√	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-026A/E
Item	Sleep Mode and System Stop Mode		

Q

What is the difference between sleep mode and system stop mode?

A

Table 1 shows the major differences.

Table 1 Sleep Mode and System Stop Mode

Mode	Function	Exit
Sleep	CPU stop	• Interrupt (internal/external) • Reset
System stop	CPU and internal I/O stop	• Interrupt (external) • Reset

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
√	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-027B/E
Item	Recovery from System Stop		

Q

What is the system status after recovery from system stop mode?

A

System stop mode is a combination of sleep and I/O stop modes.

The HD64180 exits system stop mode on detection of $\overline{NMI}$ or $\overline{INT}$ external interrupts only, except for RESET.

If interrupts are globally disabled (IEF1 = 0), instruction execution begins with the instruction following the SLP instruction.

If interrupts are globally enabled (IEF1 = 1), the appropriate normal interrupt response sequence executes.

However, I/O stop mode continues until the I/O stop bit is set to 0 after recovery from system stop mode.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
√	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-028B/E
Item	System Standby Function		

Q

Does the HD64180 have a system standby function (stop clock) to reduce power consumption?

A

No, clock stop function is not provided. Minimize the clock frequency to reduce power consumption. However, if you stop the clock completely, MPU operation and data in the registers are not guaranteed.

To minimize power consumption, we recommend:

1. Store all register information into battery backed-up RAM
2. Stop power supply to HD64180

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
√	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-059A/E
Item	Interrupt Sampling in Sleep Mode		

Q

1. Can an interrupt be accepted in sleep mode?
2. If so, when is sleep mode cancelled?

A

1. The CPU accepts interrupts at the falling edge of the ϕ clock pulse one pulse after it enters sleep mode (figure 1).
2. Sleep mode is cancelled one and a half ϕ clock pulses after an interrupt is accepted. The CPU status is recovered according to the IEF flag status:
   - IEF flag = 1: CPU begins an interrupt acknowledge cycle
   - IEF flag = 0: CPU begins an $\overline{NMI}$ acknowledge cycle or executes the instruction following the SLP instruction for maskable interrupts

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
√	Low Power Mode
	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-059A-2/E
Item	Interrupt Sampling in Sleep Mode		

A

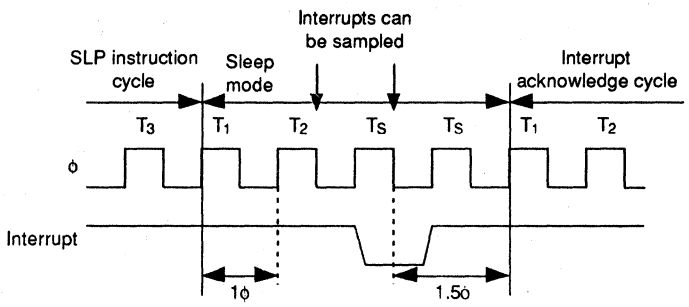


Figure 1 Timing of Interrupt during Sleep Mode

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-029A/E
Item	Dynamic RAM Refresh during DMA		

Q

Is DRAM refreshed during internal DMA operation?

A

Yes, refresh cycles are inserted during internal DMA cycles.

The refresh controller does not distinguish DMA cycles from CPU cycles.

Dynamic RAM refresh is performed at the end of the machine cycle during both CPU and DMA cycles. The interval and duration of the refresh cycle are programmable.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
√	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-030B/E
Item	Dynamic RAM Refresh		

Q

1. Is the HD647180X refresh controller different from the Z80 refresh controller?
2. What is the function of the R counter?

A

1. Yes, the refresh controller is different from the Z80 refresh controller. Refresh cycles are inserted or supressed by software. Also, the interval (10ϕ–80ϕ) and length (2ϕ–3ϕ) of the refresh cycle are programmable. The refresh address (8-bit address) is output at A_0–A_7 (figure 1).

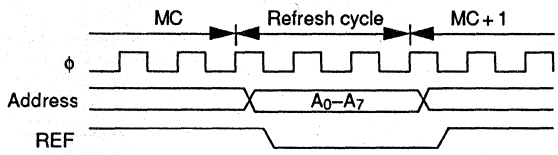


Figure 1 Refresh Example (refresh programmed to 3 cycles)

2. The R counter counts the number of CPU opcode fetches. It has no relation to dynamic RAM refresh.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
√	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

QA641-022B

Comment

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-060A/E
Item	Refresh Cycle Insertion		

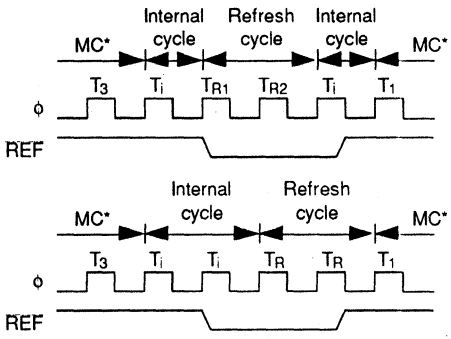
Q

Normally, a refresh cycle is inserted at the breakpoint of an instruction cycle (machine cycle).

Is it possible to insert a refresh cycle between consecutive internal machine cycles (T_i)?

A

Yes, a refresh cycle can be inserted between internal cycles, and between internal and machine cycles (figure 1).



MC*: Normal machine cycle (T_1, T_2, (T_W), and T_3)

Figure 1 Refresh Cycle Insertion Point

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
√	Refresh
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

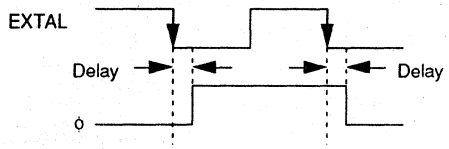
Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-061A/E
Item	EXTAL and ϕ		

Q

What is the relationship between EXTAL input and ϕ clock output when an external clock is input through EXTAL?

A

ϕ clock changes synchronously with the falling edge of EXTAL (figure 1).



Delay: 40 ns typ (reference only)

Figure 1 External Clock

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
√	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-062A/E
Item	ϕ Clock Output Frequency Error		

Q

Normally, ϕ clock output frequency is one half of the crystal oscillator frequency.

Why does ϕ clock output frequency equal the crystal frequency in our system?

A

How $\overline{\text{RESET}}$ and Tout 1 terminals are handled may effect ϕ clock output frequency. Therefore, take the following two types of measures:

1. Check that the reset circuit design asserts the $\overline{\text{RESET}}$ signal for at least six clock states.
2. Do not pull down Tout 1 (it is an output signal).

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
√	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-123A/E
Item	ϕ Pin Handling		

Q

If ϕ pin is not used, can it be connect capacitively to GND (figure 1)?

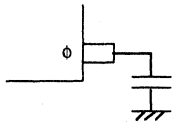


Figure 1 Unused ϕ Pin

A

Yes. The ϕ pin can be connected to GND through a maximum capacitance of 90 pF.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
√	Clock Generator
	ASE
	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-031A/E
Item	ASE Trace Function		

Q

How is ASE trace information displayed on the CRT?

A

After go or step command execution, the trace buffer pointer indicates the last trace data location. Specify display numbers with negative values until the pointer corresponds to the trace pointer. After moving the trace buffer pointer with the trace pointer command, you can specify the display number with a positive value (figure 1).

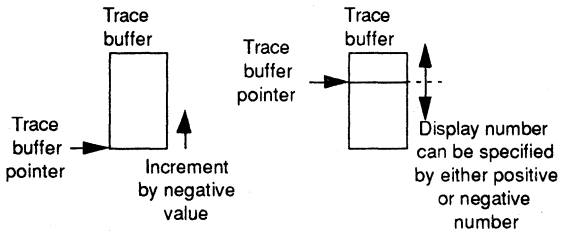


Figure 1 Displaying Trace Information

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
√	ASE
	Software
	Others

Application Manual

H180AS01 User's Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-032A/E
Item	Dynamic RAM Refresh of ASE		

Q

Dynamic RAM refresh depends on the refresh control register programming.

If the dynamic RAM refreshed during the wait state for command input when using ASE?

A

When ASE is used, dynamic RAM refresh is executed as follows, depending on refresh control register programming:

1. Refresh enable: REFE = 1

   If REFE bit is set to 1, dynamic RAM is refreshed while ASE is waiting for command input.

2. Refresh disable REFE = 0

   If REFE bit is set to 0, dynamic RAM is not refreshed while ASE is waiting for command input. (But refresh cycles are inserted during trace.)

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
√	ASE
	Software
	Others

Application Manual

H180AS01 User's Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-033A/E
Item	Difference between RET and RETI Instructions		

Q

What is the difference between the RET and the RETI instructions?

A

Both RET and RETI instructions return from a subroutine to the main program. Both instructions have identical functions.

However, RETI is normally used to return from an external interrupt ($\overline{INT}_0$, $\overline{INT}_1$, or $\overline{INT}_2$) service routine.

Since RETI is a two-byte instruction, peripheral devices know the completion of the current interrupt service routine during RETI execution, especially when using daisychain (Z80 peripheral).

However, for external interrupts, especially daisychained interrupts, the RET instruction is useful for identifying an internal interrupt service routine.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
√	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-034A/E
Item	LD A, R and LD R, A Instructions		

Q

Can the refresh address be read by executing an LD A, R or LD R, A instruction?

A

No, the refresh address cannot be read by executing the LD A, R or LD R, A instruction.

The HD64180 incorporates a dynamic RAM refresh controller. But the R counter indicates the number of CPU opcode fetch cycles and has no relation to dynamic RAM refresh.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
√	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-035A/E
Item	Processing Speed of SD200 Cross Assembler		

Q

What is the processing speed of the cross assembler for the SD200?

A

It is about 2.5 minutes per 1,000 steps (2.5 min/kstep) (floppy disk based) on the SD200 for S180XAS6F (version 1.0).

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
√	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-036A/E
Item	LDIR Instruction		

Q

What is the bus cycle status during LDIR instruction execution?

A

Fourteen instruction cycles are repeated. The last execution cycle (BC = 0) is twelve cycles. That is:

- BC ≠ 0: fourteen instruction execution cycles are repeated
- BC = 0: twelve instruction execution cycles are repeated

See figure 1.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
√	Software
	Others

Application Manual

HD641180X, HD643180X, HD647180X Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

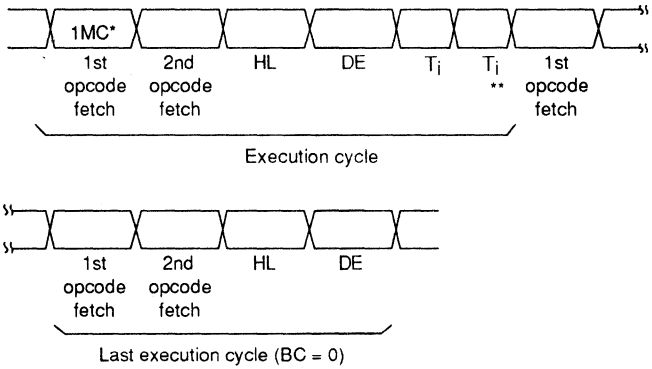
Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-036A-2/E
Item	LDIR Instruction		

A

LDIR (ED B ϕ)
Operation

$(HL)m \rightarrow (DE)m$
$BC_r - 1 \rightarrow BC_r$
$DE_r + 1 \rightarrow DE_r$
$HL_r + 1 \rightarrow HL_r$
Repeat until $BC_r = 0$

Bus cycle



*: 1 machine cycle
**: Internal operation machine cycle

Figure 1 LDIR Instruction Timing

SECTION 2

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-063A/E
Item	Extension Instructions (IN0, OUT0)		

Q

Are there any limitations when the IN0 or OUT0 instructions access external I/O?

A

Yes, the IN0 and OUT0 instructions can only access the lower 256 bytes of I/O space.

IN g, (c) and OUT (c), g instructions can access more than 256 bytes of I/O space (figure 1).

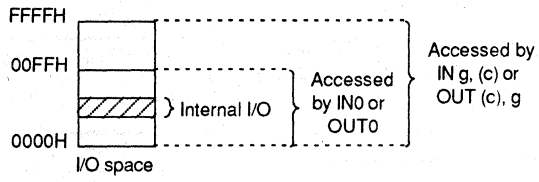


Figure 1 I/O Space Access

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
√	Software
	Others

Application Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD641180X, HD643180X, HD647180X	Q&A No.	QA641-064A/E
Item	DEC (INC) and DAA Instructions		

Q

Normally, the DAA instruction is executed to obtain BCD data after ADD or SUB instruction execution.

Does the DAA instruction adjust the result after DEC (INC) instructions?

A

No, the DAA instruction does not support BCD adjustment after DEC (INC) instructions.

DAA execution results depend on flag conditions. See table 1 for an example.

Table 1 DAA Example

Instruction	Acc	Flag N	Flag C	Flag H
(Initial value)	00	0	0	0
DEC A	FF	1	0	1
DAA	F9 (FF + FA)	1	1	1

Refer to the HD64180 user's manual for details.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	Timer
	Bus Interface
	Interrupt
	I/O Port
	Memory
	Wait
	Reset
	Low Power Mode
	Refresh
	Clock Generator
	ASE
√	Software
	Others

Application Manual

HD64180 Data Sheet

Other Data

Reference Q&A

Comment

SECTION 2

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD64180S NPU Network Processing Unit

Technical Q and A

Application Note

Preface

The HD64180S NPU (network processing unit) is a single-chip microcontroller that facilitates high-speed, low-cost processing of a variety of communication functions such as communication protocol and other user-specified functions.

The HD64180S mainly incorporates the following on a single chip:

- 8-bit CPU
- Multiprotocol serial communication interface (MSCI)
- Asynchronous serial communication interface/clock-synchronous serial I/O port (ASCI/CSIO)
- DMA controller

The HD64180S enables high-speed data transfer by taking over communication program processing from the host CPU.

This LSI, for example, can be used in an auxiliary communication system for computer-to-computer communication, or in the distributed control unit installed in an industrial robot.

In addition, the HD64180S can be easily applied to any type of existing communication system because it can interface with LSIs having conventional communication functions and can be controlled by conventional communication programs.

How to Use This Technical Q&A Manual

This technical manual contains answers to questions that many users have asked regarding Hitachi microcontrollers. It is intended to supplement the explanations in the current data books and user's manuals. Thus, please use this manual together with the data books and user's manuals.

If any further questions arise as you use this manual and the products described, please do not hesitate to get in touch with your nearest Hitachi semiconductor sales office.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Contents

Timer

Bus Interface

Interrupt

Wait

Reset

Low Power Mode

Refresh

Clock Generator

ASE

Software

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-002B/E
Item	Logical to Physical Address Translation		

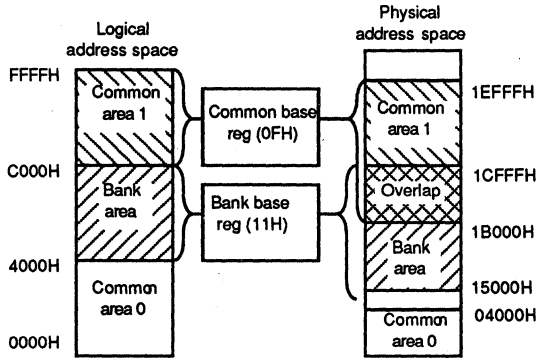
Q

Can the MMU base register (MMU common base register and MMU bank base register) be programmed so that common area 1 overlaps with the bank area?

A

Yes, depending on the MMU base register programming, common area 1 and the bank area may overlap (figure 1).

Figure 1 Overlapping Common Areas



Classification

√	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-004C/E
Item	DE (DMA Enable) Bit in DMA Status Register		

Q

When $\overline{NMI}$ occurs, DE is reset to 0 and DMA operation is disabled, passing control to the CPU.

1. How is DMA operation timing halted?

2. How does DMA operation restart?

A

1. When $\overline{NMI}$ occurs, the CPU takes control after the current DMA cycle is completed (figure 1).

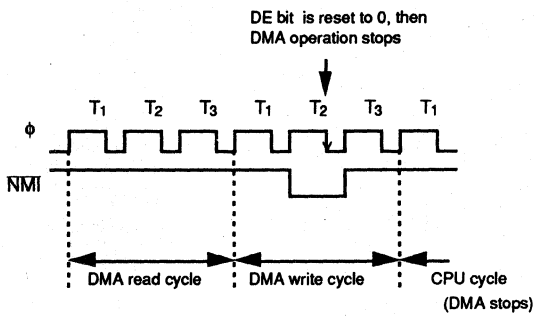


Figure 1 $\overline{NMI}$ Timing

2. To restart DMA operation, set DE bit to 1.

Classification

	MMU
	MSCI
	ASCI/CSIO
√	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-005A/E
Item	$\overline{\text{DWE}}$ Bit in DMA Status Register		

Q

What is the function of the $\overline{\text{DWE}}$ bit in the DMA status register?

A

The $\overline{\text{DWE}}$ bit, which enables write operation to the DE bit, is necessary to prevent the DE bit from being affected by the write operation to the other bits of the DMA status register. The following gives a more specific description.

Before writing to the EOT, EOM, BOF, or COF bit during DMA transfer, 1 must be written to the DE bit. However, if the DMA transfer ends immediately before writing to the DE bit, DMA transfer will undesirably resume when 1 is written to the DE bit. The $\overline{\text{DWE}}$ bit prevents this, disabling any unnecessary write operation to the DE bit.

Classification

	MMU
	MSCI
	ASCI/CSIO
√	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-008A/E
Item	Memory (specified in application program)-to-I/O DMA Transfer		

Q

Is it possible to execute memory (specified in application program)-to-I/O DMA transfer independently of the MMU base register?

A

No, to execute memory (specified in application program)-to-I/O DMA transfer correctly, one of the following two conditions must be met:

1. The physical source address is defined beforehand.

2. A common area contains the value of the base register that corresponds to the physical source address.

If the DMA transfer is executed using only the logical address, block transfer instructions can be used.

Classification	
	MMU
	MSCI
	ASCI/CSIO
√	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-009B/E
Item	Memory ↔ I/O (Z80 Peripheral) DMA Transfer		

Q

When memory ↔ I/O (Z80 peripheral) DMA transfer is executed while $\overline{DREQ}$ is programmed for level sensitivity, DMA transfer does not complete correctly.

Are there any restrictions on DMA operation? (RDY signal from Z80 peripheral is input to $\overline{DREQ}$ of HD64180S, with one inverter inserted to invert the active-high RDY signal.)

A

The Z80 peripheral RDY signal is negated during DMA write cycle to the peripheral LSI. Therefore, if $\overline{DREQ}$ is programmed for level sensitivity, an additional DMA cycle starts since RDY is negated after $\overline{DREQ}$ sampling (figure 1).

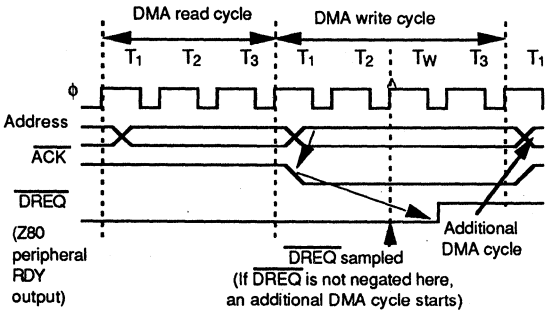


Figure 1 DMA Timing

Classification	
	MMU
	MSCI
	ASCI/CSIO
√	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment Take one of three measures: 1) Program $\overline{DREQ}$ for edge sensitivity, 2) Insert a wait state during DMA write cycle to modify RDY response timing, 3) Mask the $\overline{DREQ}$ (RDY) signal by the $\overline{ACK}$ signal.

SECTION 2

Type	HD64180S	Q&A No.	QA641-042B/E
Item	$\overline{\text{DACK}}$ Signal Generation		

Q

How can the $\overline{\text{DACK}}$ signal be generated during channel 0 memory ↔ external I/O DMA transfer?

A

When external I/O is accessed during a DMA cycle, the external I/O address is output through the address bus. At this time, the $\overline{\text{IOE}}$ signal and address output are decoded to generate a $\overline{\text{DACK-1}}$ signal (figure 1, ①).

When external I/O is accessed to initialize registers during a CPU cycle, $\overline{\text{DACK-1}}$ and ST signals are logical-ORed to generate $\overline{\text{DACK-2}}$ to distinguish a DMA from a CPU cycle (figure 1, ②).

Figures 1 and 2 (next page) show the $\overline{\text{DACK}}$ generation circuit and signal timing, respectively.

Classification	
	MMU
	MSCI
	ASCI/CSIO
√	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Application Note

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-042B-2/E
Item	$\overline{\text{DACK}}$ Signal Generation		
A			

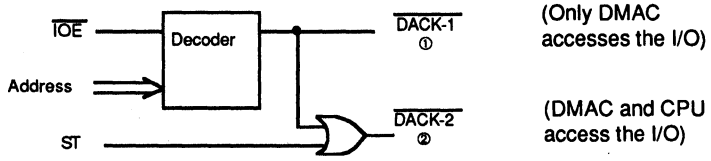


Figure 1 Circuit Example

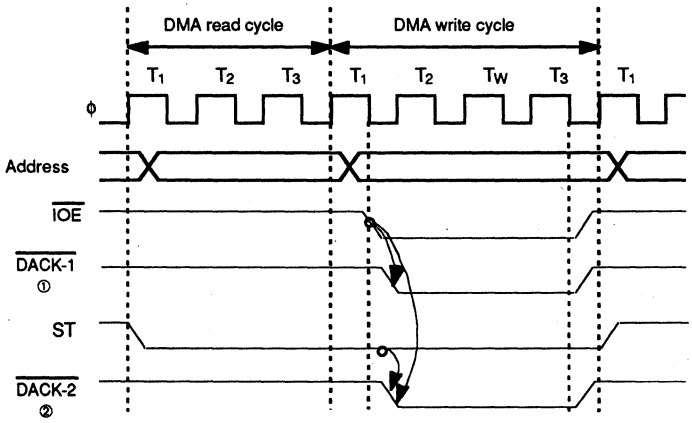


Figure 2 Timing Chart

SECTION 2

Type	HD64180S	Q&A No.	QA641-097A/E
Item	Ring Configuration during a Chained-Block Transfer		

Q

Is it possible to configure a ring, by using the function that allows a descriptor to specify n number of buffers during a DMAC chained-block transfer?

A

Yes. Load the starting address of the descriptor corresponding to the first buffer into the chain pointer corresponding to the n-th buffer. In this case, operations are the same as for the case without a ring.

Classification	
	MMU
	MSCI
	ASCI/CSIO
√	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-098A/E
Item	CRC Code Transmission during a Memory-to-MSCI DMA Transfer		

Q

How is the CRC code sent during a memory-to-MSCI DMA transfer?

A

During a chained-block transfer, the CRC is automatically sent when status bit 7 of the descriptor corresponding to the buffer containing the last data value is set to 1.

During a single-block transfer, set the CRCCC bit to1, and the CRC code will be automatically sent if an underrun error occurs at DMA transfer completion, setting the UDRN flag to 1.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
√	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-099A/E
Item	DMAC Error Handling		

Q

What will happen when an error occurs during a DMA transfer between memory and the MSCI?

A

When an error such as counter overflow occurs in the DMAC, the DMAC stops transfer.

When an error occurs in the MSCI, the DMAC does not stop the transfer, but indicates the error status in the descriptor.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
√	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-044B/E
Item	$\overline{DCD}$ Pin and CDCD Flag		

Q

Is the CDCD flag (MST1, ST1) reset when the $\overline{DCD}$ is asserted low?

A

No, the CDCD flag is not reset unless 1 is written to the bit position. This allows the $\overline{DCD}$ interrupt to be serviced correctly.

If the $\overline{DCD}$ pin and the CDCD flag were reset simultaneously, the $\overline{DCD}$ interrupt request would always be cleared and could not be serviced when a higher priority interrupt occurred simultaneously with the $\overline{DCD}$ interrupt (figure 1). Figure 2 shows the actual function used by the HD64180S.

Classification	
	MMU
√	MSCI
√	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No	QA641-044B-2/E
Item	$\overline{DCD}$ Pin and CDCD Flag		
A			

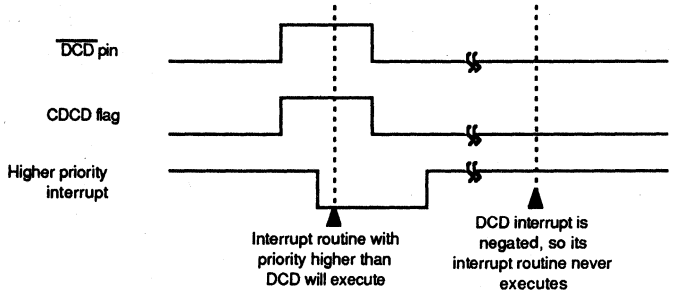


Figure 1 CDCD Flag Synchronized with $\overline{DCD}$ Pin

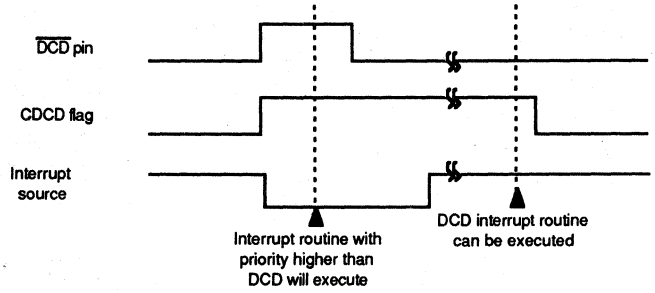


Figure 2 HD64180S CDCD Function

Type	HD64180S	Q&A No.	QA641-100A/E
Item	Signal Direction of TXCM Pin (1)		

Q

Does the TXCM pin input or output the receive clock signal specified as the transmit clock signal by the MSCI MTXS register?

A

The TXCM pin outputs the clock signal supplied from the RXCM pin.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

MTXS register: MSCI TX clock source register

SECTION 2

Type	HD64180S	Q&A No.	QA641-101A/E
Item	Signal Direction of TXCM Pin (2)		

Q

Does the TXCM pin function as an input or output when the TXCM line input is specified as the transmit clock by the MSCI MTXS register, and the auto-echo function (where the TXCM pin functions as an output) is selected by MMD2?

A

The TXCM pin functions as an output, with the MMD2 register setting (auto-echo or local loop-back) given higher priority. Similarly, MMD2 and MD2 register settings are given higher priority for pins TXCA and RXCA of ASCI/CSIO, and pin RXCM.

Classification	
	MMU
√	MSCI
√	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

MMD2: MSCI mode register 2

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-102A/E
Item	AC Characteristics of Manchester Codes		

Q

How are the AC characteristics of the Manchester codes defined? The bit boundaries are unclear.

A

When TXCM functions as an input pin, the time between the rising edge of the TXCM input and the intermediate transition point of the Manchester code is defined as t_{TDD1M}. The same time period is also defined as t_{TDD2M} when TXCM functions as an output.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-103A/E
Item	AC Characteristics of Receive FM Data		

Q

What is the timing relationship between the external clock signal and FM-coded data received?

A

The timing is shown in figure 1.

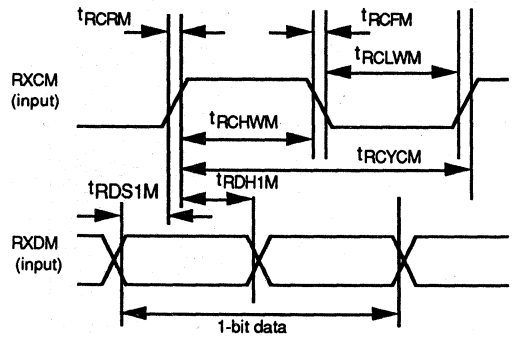


Figure 1 Receive Timing (RXCM = Input)

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-104A/E
Item	RXRDY Flag		

Q

Is it possible to clear the RXRDY flag (MST0) by any method other than the three given in the manual: hardware reset, channel reset command, and system stop mode?

A

The RXRDY flag is automatically cleared when the receive buffer is empty.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-105A/E
Item	CRC Error Handling		

Q

What must be done to prevent an interrupt if the CRC error interrupt bit is enabled, when the CRCE flag of MSCI status register 2 (MST2) is set to 0 or 1 according to the contents of the status FIFO during data reception?

A

The CRC interrupt (bit) usually must be masked. To determine when a CRC error has occurred, generate an interrupt at the end of the receive frame, and check bit 2 (CRCEF flag) of the MFST (MSCI frame status register) within the corresponding interrupt routine.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-106A/E
Item	Transmit Data Fixing		

Q

Is it possible to hold MSCI transmit data to either 0 or 1 for consecutive cycles, by software?

A

It is possible in the idle state, where 8-bit values in the idle pattern register (MIDL) are sent. The MIDL value should be set to 00H or FFH.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-107A/E
Item	Mark Output Modulation		

Q

Is mark output subject to modulation when FM codes are selected?

A

Yes. The 1 level is modulated for the corresponding code type before being output.

	Classification
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-108A/E
Item	Idle Pattern Transmission		

Q

In HDLC mode, does the idle pattern transmission state enter the flag transmission state after all eight bits of the idle pattern have been transmitted?

A

Yes.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-109A/E
Item	I/O Direction Setting of $\overline{\text{SYNC}}$ Line		

Q

How is the I/O direction of the $\overline{\text{SYNC}}$ line specified?

A

It is specified by the MMD0 register (002BH) as shown in table 1.

Table 1 $\overline{\text{SYNC}}$ Line Direction

MMD0 Bits 7	6	5	Protocol Mode	$\overline{\text{SYNC}}$ Direction
0	0	0	Asynchronous	Input
0	0	1	Byte synchronous mono-sync	Output
0	1	0	Byte synchronous bi-sync	Output
0	1	1	Byte synchronous external	Input
1	0	0	Bit synchronous HDLC	Output
1	0	1	Bit synchronous loop	Output
1	1	0	Reserved	Output
1	1	1	Reserved	Output

Classification

	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

MMD0: MSCI mode register 0

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-110A/E
Item	Synchronization Using $\overline{\text{SYNC}}$ Line		

Q

Is it possible to establish synchronization at the end of data similar to that at the beginning of data, by using the $\overline{\text{SYNC}}$ line in MSCI byte synchronous and external synchronous modes?

A

No. The HD64180S can establish synchronization only at the beginning of data.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-111A/E
Item	Phase Relationship between $\overline{SYNC}$ Line and Data		

Q

What is the phase relationship between the $\overline{SYNC}$ line and data?

A

The relationship is shown in figure 1. Note that sufficient setup time and hold time are necessary for $\overline{SYNC}$ and data in relation to the rising edge of the input clock pulse.

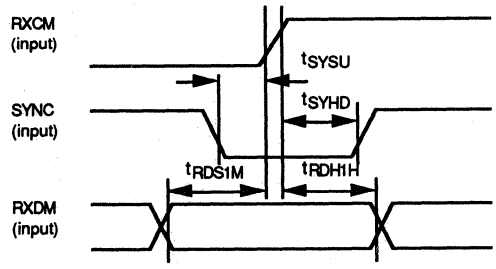


Figure 1 Phase Relationship between $\overline{SYNC}$ Line and Data

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-112A/E
Item	SYNC Codes for Byte Synchronous Mode		

Q

Must SYNC codes match during reception in MSCI byte synchronous and external synchronous modes?

A

No, it is not necessary, because synchronization is established using the SYNC line in byte synchronous and external synchronous modes. Any SYNC code can be transmitted.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-113A/E
Item	Data Insertion between Flags in Bit Synchronous Mode		

Q

Is it possible to insert data between a closing flag and an opening flag, in bit synchronous mode?

A

Yes. Delaying the data write operation to the transmit buffer allows the idle pattern register value to be inserted between a closing flag and an opening flag.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-114A/E
Item	Character Insertion in Bit Synchronous Loop Mode		

Q

Is it possible to insert a desired character during the idle state in Bit Synchronous loop mode?

A

Yes. The primary station transmits a character written to the idle pattern register. The secondary station transmits a character written to the idle pattern register when the GOP bit is set to 1. When the GOP bit is cleared to 0, the secondary station enters the retransmission idle state and retransmits the data, which was transmitted from the primary station, after a 1-bit delay time.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-115A/E
Item	Address Field Check Usage		

Q

In what kind of applications is single address 2 mode useful?

A

Single address 2 mode is very useful in the application shown in figure 1, where data is transmitted to a specific group of stations. This is achieved by specifying the group address as the high-order address, and each terminal address as the low-order address.

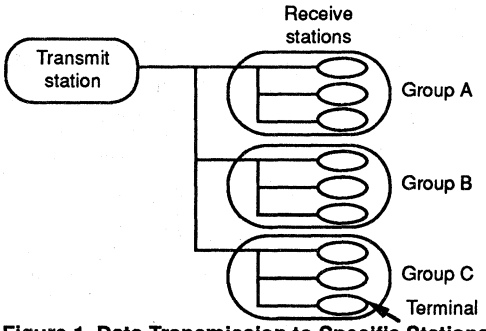


Figure 1 Data Transmission to Specific Stations

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-116A/E
Item	ADPLL Synchronization Pattern		

Q

Is it possible to establish synchronization using the Manchester-coded ADPLL synchronization pattern in which 0s and 1s are reversed?

A

Yes. Synchronization patterns AAH and 55H can both be used.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-117A/E
Item	Duty Cycle for ADPLL Operation		

Q

Does the ADPLL operate correctly using an operating clock whose duty cycle is smaller than 50%?

A

Yes. It operates correctly as long as t_{PLHWM} and t_{PLLWM} characteristics are satisfied, where t_{PLHWM} is the high-level pulse width of the ADPLL operating clock, and t_{PLLWM} is the low-level pulse width.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-118A/E
Item	ADPLL Operation in Local Loop-Back Mode		

Q

Is it possible to extract the clock element from the data in the receive shift register, in MSCI local loop-back mode?

A

No. Refer to figure 4-2, Block Diagram of the MSCI Receiver, on page 111 in the hardware manual.

Classification	
	MMU
√	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-013C/E
Item	Timer Counting Operation Using Input Clock		

Q

Can the timer count events by using the external clock?

A

Yes. The internal timer can count the inputs on the TIN0 and TIN1 pins by using the event-counting function with the external clock set to a frequency of $\phi/4$ or smaller.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
√	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-119A/E
Item	Compare-Match Flag		

Q

What happens if the CMF flag is not cleared after the interrupt is issued (if enabled) when the contents of the timer up-counter (TCNT) and timer constant register (TCONR) match, setting the CMF flag?

A

If the CMF flag is not cleared, the operation enters the same interrupt routine immediately after the preceding interrupt routine. Consequently, the CMF flag must be cleared within the (first) interrupt routine.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
√	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-014B/E
Item	Bus State during Internal I/O Access		

Q

1. What is the bus status during internal I/O access?

2. What happens when external I/O is assigned to the same address as internal I/O?

A

1. Bus status during internal I/O access is as follows:

   - Data bus
     - — Read: High impedance state
     - — Write: Outputs data
   - Address bus
     - — Read/write: Outputs address

2. When an internal I/O address and an external I/O address overlap, bus status is as follows:

   - Data bus
     - — Read: Reads internal I/O; does not read external I/O
     - — Write: Outputs data to both internal and external I/O
   - Address bus
     - — Read/write: Outputs address

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
√	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-046A/E
Item	Data Sampling Timing during Memory Read		

Q

Does the CPU sample data at the rising edge of T_3 during opcode fetch cycles and at the falling edge of T_3 during operand and data read cycles?

A

Yes, the CPU samples the operation code on the data bus at the rising edge of T_3 while it samples operands and data at the falling edge of T_3 (table 1).

Table 1 Sampling Timing

CPU Cycle	Sampling Timing
Operation code fetch cycle	T_3 rising edge (↑)
Data and operand fetch cycle	T_3 falling edge (↓)

Classification

	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
√	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-079A/E
Item	Address Bus Status during $\overline{INT_0}$ Acknowledge Cycle		

Q

What is on the address bus generated by the CPU during the $\overline{INT_0}$ acknowledge cycle when $\overline{IOE}$ is asserted low?

A

The CPU puts the PC value following the address of the last machine cycle on the address bus during the $\overline{INT_0}$ acknowledge cycle. Although $\overline{IOE}$ is asserted low in this case, it has no affect since $\overline{RD/WR}$ is negated.

	Classification
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
√	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-080A/E
Item	ST Output Timing in $\overline{INT_0}$ Mode 0		

Q

How is ST output during the $\overline{INT_0}$ mode 0 interrupt acknowledge cycle?

A

The timing is shown in figure 1.

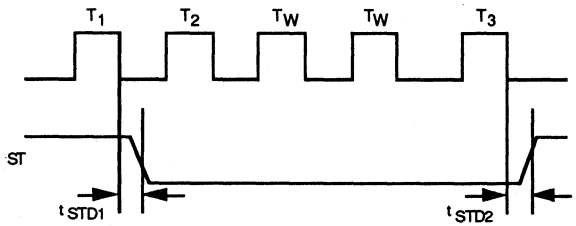


Figure 1 $\overline{INT_0}$ (Mode 0) Operation Timing

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
√	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-015B/E
Item	Interrupt during MMU Operation		

Q

How will the MMU be affected if an interrupt occurs during its operation?

A

If an interrupt occurs during MMU operation, the interrupt vector is relocated according to the MMU base register programming. Therefore, the interrupt vector should be defined with reference to the MMU base register programming (figure 1).

However, the interrupt vector can be located in common area 0 or in an area where banks are not switched. These areas are always located in the same logical address space.

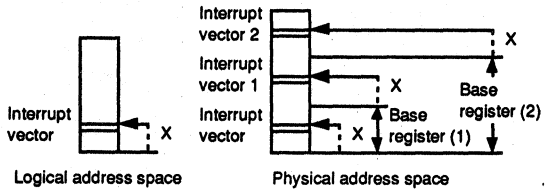


Figure 1 Interrupt Vector Generation during MMU Operation

Classification

	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-017A/E
Item	$\overline{INT_0}$ Mode 2		

Q

In Z80 $\overline{INT_0}$ mode 2, the LSB of the lower vector in the 16-bit vector address (A_0) is always 0.

In the HD64180S, is the LSB of the lower vector (D_0) automatically set to 0 in $\overline{INT_0}$ mode 2?

A

No, in Z80 $\overline{INT_0}$ mode 2, the LSB of the lower vector is not automatically set to 0. The Z80 data book explains that the LSB (A_0) must be set to 0.

In HD64180S $\overline{INT_0}$ mode 2, the LSB of the lower vector (D_0) must be set to 0 since $\overline{INT_0}$ mode 2 requires a 2-byte vector.

However, even if the LSB of the lower vector (D_0) is set to 1, the interrupt sequence is executed correctly.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-018A/E
Item	$\overline{NMI}$ during Interrupt Acknowledge Cycle		

Q

Is $\overline{NMI}$ acknowledged during the interrupt acknowledge cycle, such as for $\overline{INT}$?

A

Yes, one instruction (excluding EI and DI instructions) is executed after the $\overline{INT}$ acknowledge cycle, then the $\overline{NMI}$ acknowledge cycle.

The $\overline{NMI}$ response sequence during the $\overline{NMI}$ acknowledge cycle is the same.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-019A/E
Item	Interrupt after Reset		

Q

Is $\overline{NMI}$ or $\overline{INT}$ acknowledged immediately after reset?

A

No, for three cycles immediately after reset (power-on reset cycle), $\overline{NMI}$ and $\overline{INT}$ are disabled.

After these three cycles, the first instruction is executed and interrupts are enabled. Figure 1 shows the timing.

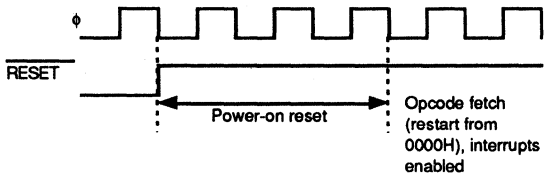


Figure 1 Power-On Reset Timing

Note: $\overline{NMI}$ is latched immediately after the power-on reset cycle.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-020B/E
Item	Interrupt during Refresh Cycle		

Q

Is an interrupt ($\overline{NMI}$ or $\overline{INT}$) acknowledged during refresh cycles?

A

No, interrupts are ignored during refresh cycles. However, $\overline{NMI}$ is acknowledged immideately after refresh cycles during instruction execution because $\overline{NMI}$ is edge sensitive. Figure 1 shows $\overline{NMI}$ acknowledge timing after refresh cycle.

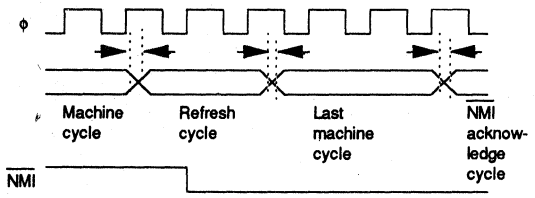


Figure 1 Refresh Timing

$\overline{INT}$ ($\overline{INT_0}$, $\overline{INT_1}$, and $\overline{INT_2}$) is ignored during refresh cycles. If $\overline{INT}$ remains active after the refresh cycle, it is acknowledged during the instruction just after refresh.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-021B/E
Item	$\overline{NMI}$ Acknowledge		

Q

Is $\overline{NMI}$ acknowledged if it occurs during the timing sequence in figure 1?

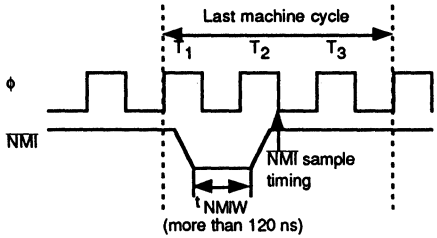


Figure 1 $\overline{NMI}$ Timing

A

Yes, if t_{NMIW} ($\overline{NMI}$ pulse width) is 120 ns or more, $\overline{NMI}$ is sampled and $\overline{NMI}$ acknowledge cycle begins after the last MC (machine cycle).

If the $\overline{NMI}$ is asserted low for tNMIW or longer between 1 and 2 in figure 2, it will be sampled at the falling edge of the clock marked 2.

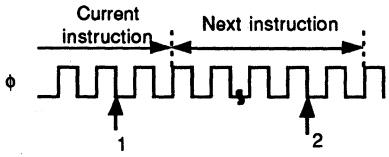


Figure 2 $\overline{NMI}$ Timing

Classification

	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-047A/E
Item	Interrupt Acknowledge Timing after EI Instruction Execution		

Q

When is an interrpt acknowledged after an EI instruction?

A

Maskable interrupts ($\overline{INT0}$, etc.) are acknowledged in the last machine cycle of any instruction cycle other than EI.

Note that no interrupts can be acknowledged during EI instruction execution. Therefore, if an interrupt occurs immediately before or during an EI instruction cycle, it is acknowledged after the end of the RETI instruction cycle following the EI instruction.

For example:

```
.
.
.        ← Interrupt request
EI
RETI     ← Interrupt request acknowledged during this
           instruction
(interrupt acknowledge cycle)
```

Classification

	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-048A/E
Item	Interrupt Sampling during Block Transfer Instruction Execution		

Q

Normally, the CPU samples interrupts at the falling edge of the ϕ clock pulse prior to state T_3 or T_i in the last machine cycle.

When does the CPU sample interrupts during a block transfer instruction which may require over hundreds of machine cycles?

A

The CPU samples interrupts at the falling edge of the ϕ clock pulse prior to state T_3 or T_i in the last machine cycle of each one byte transfer (14ϕ or 12ϕ) (figure 1).

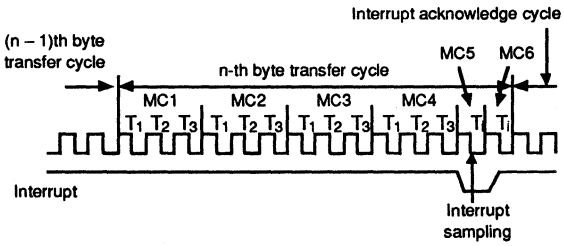


Figure 1 Interrupt during Block Transfer

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-049B/E
Item	Interrupt during SLP Instruction Cycle		

Q

What is the CPU status when an interrupt occurs during SLP instruction execution?

A

The CPU operates as shown in figure 1.

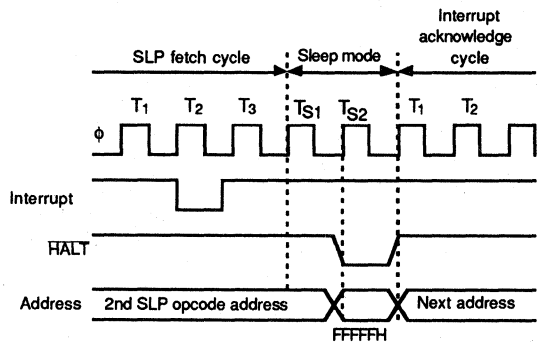


Figure 1 CPU Timing

When an interrupt is sampled during an SLP instruction cycle, $\overline{HALT}$ is asserted low for 1 clock state, and the address bus outputs FFFFFH.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-050A/E
Item	$\overline{INT_0}$ Mode 0		

Q

If the CALL instruction (three-byte instruction) is executed during $\overline{INT_0}$ mode 0 acknowledge cycle, the CPU cannot return from the interrupt service correctly. Why is this?

A

$\overline{INT_0}$ mode 0 operates as follows:

1. Stacks PC contents by an instruction, usually (one-byte) RST, fetched during $\overline{INT_0}$ mode 0 acknowledge cycle
2. Stops incrementing the PC during $\overline{INT_0}$ mode 0 acknowledge cycle
3. Executes instruction fetched from data bus during interrupt acknowledge cycle.

However, if the (three-byte) CALL instruction, which requires three machine cycles to fetch including the operand, is executed during $\overline{INT_0}$ mode 0 acknowledge cycle, PC increment stops only during interrupt acknowledge cycle (one machine cycle) and is incremented by 2 during the rest of the CALL instruction (operand fetch). As a result, PC + 2 is stacked as the return address. Therefore, decrement the stacked PC value by 2 in software to return from the interrupt correctly.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-051A/E
Item	$\overline{NMI}$ Interrupt Sampling Timing		

Q

$\overline{NMI}$ is sampled at the falling edge of a ϕ clock pulse prior to state T_3 or T_i in the last machine cycle of each instruction.

1. When is $\overline{NMI}$ sampled if the last machine cycle is an internal T_i cycle?

2. How about $\overline{INT}$?

A

Both $\overline{NMI}$ and $\overline{INT}$ are always sampled at the falling edge of the second last ϕ clock pulse of the last machine cycle. The $\overline{NMI}$ sampling is not affected by the number of internal T_i cycles (figure 1).

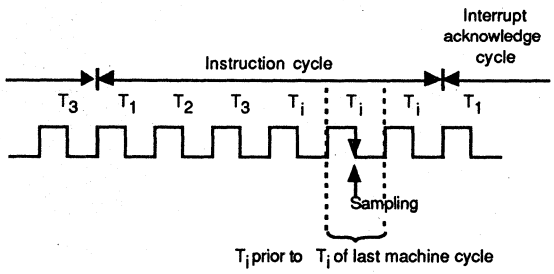


Figure 1 Interrupt Sample Timing during T_i

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-052A/E
Item	Status Bit during TRAP		

Q

1. What happens if an aditional TRAP occurs before the INT/TRAP control register TRAP bit is cleared?

2. What is the status of the TRAP and UFO bits in this case?

A

1. An additional TRAP interrupt occurs.

2. The TRAP bit remains 1 since it can be cleared only by software.

   The UFO bit remains unchanged since it cannot be modified while the TRAP bit = 1.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment UFO bit: Indicates if TRAP occurred in 2nd or 3rd opcode fetch cycle:
UFO bit = 0: TRAP occurred in 2nd opcode fetch cycle
UFO bit = 1: TRAP occurred in 3rd opcode fetch cycle

Type	HD64180S	Q&A No.	QA641-053B/E
Item	PC Stacking during TRAP		

Q

1. Why is the stacked PC value different for TRAP occurrence during second opcode fetch and during third opcode fetch?

2. How can the first opcode address be calculated?

A

1. Table 1 summarizes CPU operations when TRAP occurs during the second and third opcode fetches.

Table 1 CPU Operations during TRAP

TRAP during 2nd Opcode Fetch		TRAP during 3rd Opcode Fetch	
Bus Cycle	**PC**	**Bus Cycle**	**PC**
1st opcode fetch	PC – 1	1st opcode fetch	PC – 3
2nd opcode fetch	PC	2nd opcode fetch	PC – 2
Stack	PC	Operand read	PC – 1
—	—	3rd opcode fetch	PC
—	—	Memory read	PC
—	—	Stack	PC – 1

Classification

	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No	QA641-053B-2/E
Item	PC Stacking during TRAP		

A

Therefore, the first opcode address can be calculated from one of the following equations to execute the instruction disturbed by TRAP occurrence.

- TRAP during 2nd opcode fetch
  1st opcode address = stacked PC value – 1

- TRAP during 3rd opcode fetch
  1st opcode address = stacked PC value – 2

Type	HD64180S	Q&A No.	QA641-054B/E
Item	$\overline{\text{NMI}}$ during DMA Transfer		

Q

What happens to DMAC after $\overline{\text{NMI}}$ assertion?

A

When $\overline{\text{NMI}}$ is asserted low during DMA transfer, the DMA transfer ends at the end of the current DMA cycle.

However, note that the $\overline{\text{NMI}}$ acknowledge cycle begins at different times, depending on the CPU status before DMA transfer (figures 1, 2, and 3).

DMAC operations can be restarted by writing 1 to the corresponding channel's DE bit.

$\overline{\text{NMI}}$ acknowledge cycle timings are shown in the following pages.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No	QA641-054B-2/E
Item	$\overline{NMI}$ during DMA Transfer		
A			

1. When DMA transfer starts during instruction execution cycle

   a. When the DMA cycle starts during the instruction execution cycle, before the last machine cycle (T_1, T_2, and T_3) of instruction A, $\overline{NMI}$ is sampled at the falling edge of T_2 in the last machine cycle of A (figure 1).

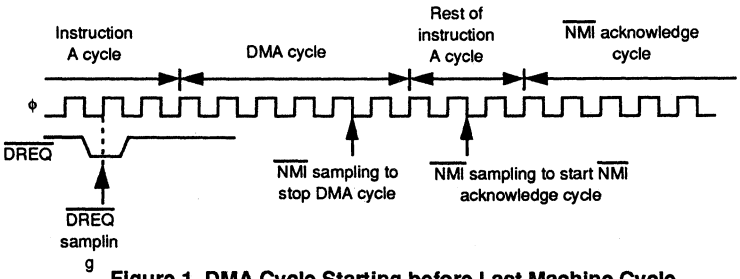


Figure 1 DMA Cycle Starting before Last Machine Cycle

   b. When the DMA cycle starts during the instruction execution cycle, before the last internal cycle (T_i) of instruction A, $\overline{NMI}$ is sampled during the DMA cycle (figure 2).

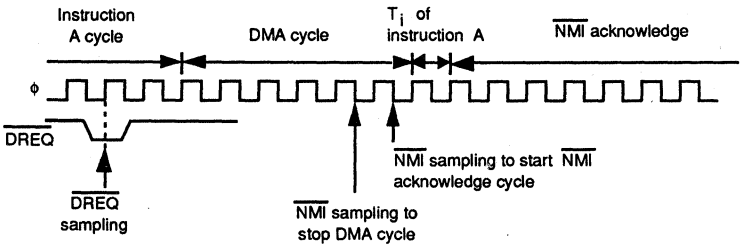


Figure 2 DMA Cycle Starting before Last Internal Cycle

Type	HD64180S	Q&A No	QA641-054B-3/E
Item	$\overline{NMI}$ during DMA Transfer		

A

2. When DMA transfer starts at the end of the instruction execution cycle, $\overline{NMI}$ is sampled at the next falling edge of T_2 or T_i of the last machine cycle of the next instruction, B (figure 3).

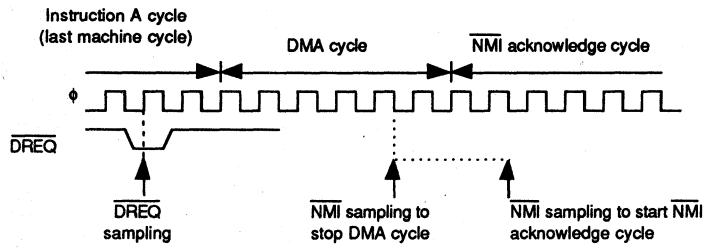


Figure 3 DMA Cycle Starting at the End of Instruction Cycle

As shown in figure 1 to figure 3, $\overline{NMI}$ has the following two functions:

1. To stop the DMA.
2. To execute the $\overline{NMI}$ routine.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-055A/E
Item	$\overline{\text{DREQ}}$i and $\overline{\text{NMI}}$		

Q

What happens to DMAC operation if $\overline{\text{NMI}}$ asserted low while the DMAC operates under the control of the $\overline{\text{DREQ}}$i pin?

A

DMAC operation is suspended and $\overline{\text{NMI}}$ is sampled with the timing shown in figure 1.

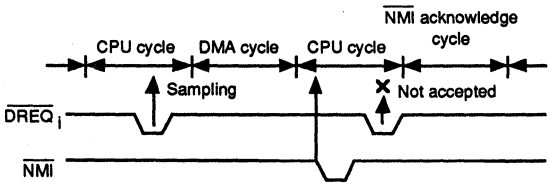


Figure 1 DMA Cycle Stopped by $\overline{\text{NMI}}$

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No	QA641-055A-2/E
Item	$\overline{DREQi}$ and $\overline{NMI}$		

A

Note that if $\overline{DREQi}$ and $\overline{NMI}$ are asserted low simultaneously, $\overline{NMI}$ sampling has priority (figure 2).

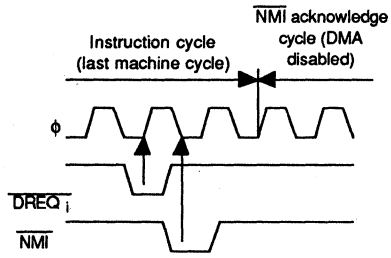


Figure 2 $\overline{DREQi}$ and $\overline{NMI}$ Conflict

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-056B/E
Item	Internal Interrupt Sampling Timing		

Q

External interrupts are sampled at the falling edge of state T_2 or T_i in the last machine cycle.

When are internal interrupts sampled?

A

They are sampled at the falling edge of state T_2 or T_i in the last machine cycle of the instruction cycle.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-057B/E
Item	$\overline{\text{INTA}}$ Signal Generation		

Q

The HD64180S can be interfaced to the 8259 to control I/O interrupts.

1. How can we generate an $\overline{\text{INTA}}$ signal to be input to the 8259 from the HD64180S?

2. Are there any precautions?

A

1. Three $\overline{\text{INTA}}$ signal pulses must be input from the HD64180S to the 8259 to control interrupts:

   a. One $\overline{\text{INTA}}$ pulse for opcode fetch

   b. Two $\overline{\text{INTA}}$ pulses for operand read

   The $\overline{\text{INTA}}$ pulse for opcode fetch can be produced by $\overline{\text{LIR}}$ and $\overline{\text{IOE}}$. The $\overline{\text{INTA}}$ pulse for operand read can be produced by $\overline{\text{RD}}$.

   This interface is the same as for Z80 and 8259.

   Figure 1 shows an example of an $\overline{\text{INTA}}$ signal generation circuit.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

This circuit is for reference only. Check logic and timing carefully for your application.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-057B-2/E
Item	$\overline{\text{INTA}}$ Signal Generation		
A			

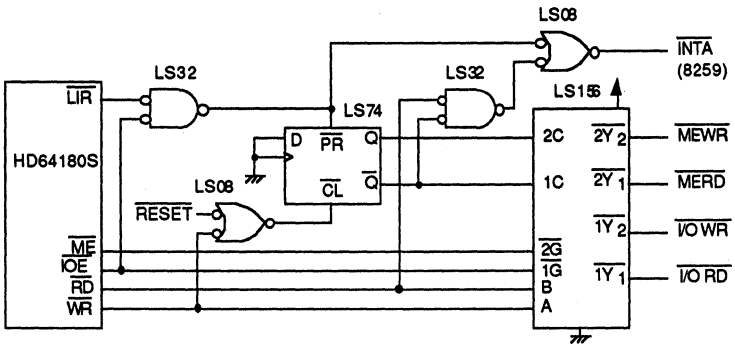


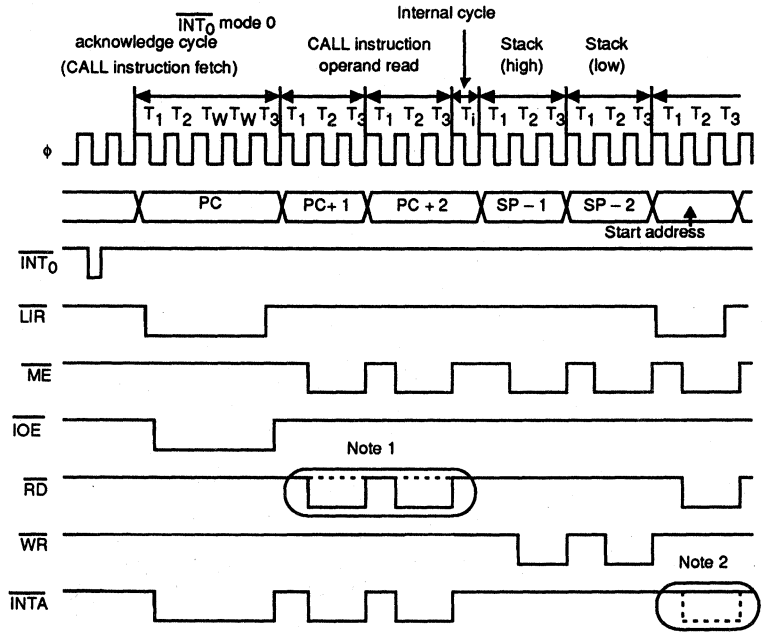
Figure 1 $\overline{\text{INTA}}$ Signal Generation Circuit Example

2. Precautions

- This circuit cannot be used when the internal or external DMAC is used in the system.
- When $\overline{\text{RD}}$ signal is used to generate the $\overline{\text{INTA}}$ signal for operand read (1b above) $\overline{\text{IOE}}$ must be used to avoid dataconflict between I/O and memory (note 1 in figure 2).
- In $\overline{\text{INT0}}$ mode 0, if the RST instruction (3-byte) is executed during its acknowledge cycle, the PC is put on the stack. If a CALL instruction is executed, PC + 2 is put on the stack.

SECTION 2

Type	HD64180S	Q&A No.	QA641-057B-3/E
Item	$\overline{INTA}$ Signal Generation		
A			



Notes: 1. $\overline{RD}$ input to memory pulled high to prevent bus conflict.
2. $\overline{RD}$ input to $\overline{INTA}$ pulled high to prevent 8259 malfunction.

Figure 2 $\overline{INTA}$ Timing

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-081A/E
Item	Interrupt Priority		

Q

1. How does the MPU accept interrupts that occurred simultaneously from many internal I/O operations?

2. What about internal interrupts that occurred simultaneously with $\overline{NMI}$ or $\overline{INT}$?

A

Internal interrupts are maintained, and interrupts are accepted in order of highest priority.

	Classification
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-082A/E
Item	Interrupt Request during HALT Opcode Fetch		

Q

Can the CPU acknowledge an interrupt request during HALT opcode fetch?

A

Yes. When interrupt enable flag 1 (IEF1) is set to 1, the CPU will acknowledge the interrupt request.

	Classification
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-083A/E
Item	ST Output Timing during Interrupt Acknowledge Cycle		

Q

What is the ST status during the $\overline{INT_0}$ acknowledge cycle in mode 0?

A

ST status is the same as in modes 1 and 2.

	Classification
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-023C/E
Item	Wait Function at I/O Access		

Q

Is a wait state (T_W) always inserted during I/O access?

A

Yes, the number of wait states specified by the software are inserted during external I/O access.

During on-chip I/O access, no wait state is inserted.

	Classification
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
√	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-024B/E
Item	Power-On Reset Sequence		

Q

How is the power-on reset sequence performed?

A

Figure 1 shows the power-on reset sequence.

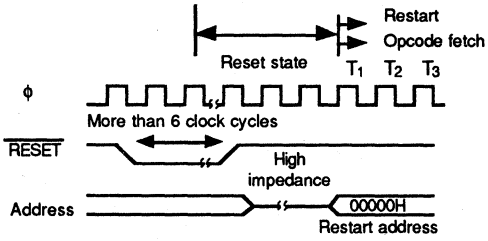


Figure 1 Power-On Reset Sequence

Classification

	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
√	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

RESET pin must be low for more than 6 clock cycles.

SECTION 2

Type	HD64180S	Q&A No.	QA641-058B/E
Item	Control Signal Status after Reset		

Q

What is the status of the control signals after each reset?

A

The $\overline{\text{RESET}}$ signal must be asserted for at least 6 states. Table 1 shows the status of each control signal.

Table 1 Control Signal Status

Control Signal	Status
Address bus	High impedance
Data bus	High impedance
Control signals ($\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{ME}}$, $\overline{\text{IOE}}$, ST, $\overline{\text{LIR}}$, $\overline{\text{HALT}}$, $\overline{\text{BUSACK}}$, $\overline{\text{TEND}}$)	High (1)
ϕ	Clock output

However, if $\overline{\text{RESET}}$ is not held low for at least 6 clock states at power-on reset, the state of these signals is undefined. For external reset, each signal remains unchanged until it is reset.

Classification

	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
√	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-026B/E
Item	Sleep Mode and System Stop Mode		

Q

What is the difference between sleep mode and system stop mode?

A

System stop mode is entered when the IOSTP bit is set to 1 and the sleep instruction is executed. Other major differences are as follows:

- Sleep mode
  CPU stopped; I/O not stopped

- System stop mode
  CPU and I/O stopped; clock generator and oscillator not stopped

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
√	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-028C/E
Item	System Standby Function		

Q

Does the HD64180S have a system standby function (stop clock) to reduce power consumption?

A

Yes. However, if the clock is completely stopped, MPU operation and data retention in the registers are not guaranteed.

To avoid these problems, we recommend that all register information be stored in a battery-powered RAM and then power supply to HD64180S be stopped.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
√	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-059B/E
Item	Interrupt Sampling in Sleep Mode		

Q

1. Can an interrupt be accepted in sleep mode?

2. If so, when is sleep mode canceled?

A

1. The CPU accepts interrupts at the falling edge of the ϕ clock pulse one pulse after it enters sleep mode (figure 1).

2. Sleep mode is cancelled two and a half ϕ clock pulses after an interrupt is accepted. The CPU status is recovered according to the IEF1 flag status:

   - IEF flag = 1: CPU begins an interrupt acknowledge cycle
   - IEF flag = 0: CPU begins an $\overline{NMI}$ acknowledge cycle or executes the instruction following the SLP instruction for maskable interrupts

   In this case, input the external interrupt signal so that a low level is sampled two or more consecutive times.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
√	Low Power Mode
	Reset
√	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No	QA641-059B-2/E
Item	Interrupt Sampling in Sleep Mode		
A			

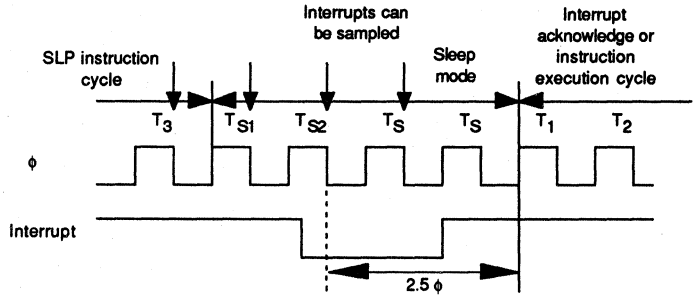


Figure 1 Timing of Interrupt during Sleep Mode

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-029A/E
Item	Dynamic RAM Refresh during DMA		

Q

Is DRAM refreshed during internal DMA operation?

A

Yes, refresh cycles are inserted during internal DMA cycles.

The refresh controller does not distinguish DMA cycles from CPU cycles.

Dynamic RAM refresh is performed at the end of the machine cycle during both CPU and DMA cycles. The interval and duration of the refresh cycles are programmable.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
√	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-030B/E
Item	Dynamic RAM Refresh (R Counter Function)		

Q

1. Is the HD64180S refresh controller different from the Z80 refresh controller?

2. What is the function of the R counter?

A

1. Yes, the refresh controller is different from the Z80 refresh controller. Refresh cycles are inserted or supressed by software. Also, the interval and length (2ϕ-9ϕ) of the refresh cycle are programmable. The refresh address (12-bit address) is output at A_0-A_{11} (figure 1).

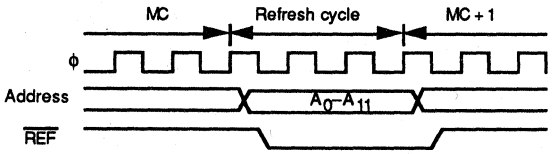


Figure 1 Refresh Example (refresh programmed to 3 cycles)

2. The R counter counts the number of CPU opcode fetches. It has no relation to dynamic RAM refresh.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
√	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-060A/E
Item	Refresh Cycle Insertion		

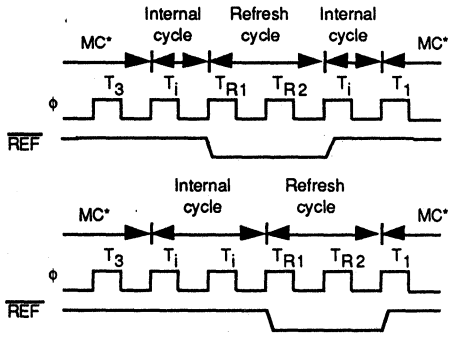
Q

Normally, a refresh cycle is inserted at the breakpoint of an instruction cycle (machine cycle).

Is it possible to insert a refresh cycle between consecutive internal machine cycles (T_i)?

A

Yes, a refresh cycle can be inserted between internal cycles, and between internal and machine cycles since an internal cycle is also considered as a machine cycle (figure 1).



MC*: Normal machine cycle (T_1, T_2, (T_W), and T_3)

Figure 1 Refresh Cycle Insertion Point

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
√	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-095B/E
Item	Bus Release Mode and Refresh		

Q

How is DRAM refreshed after $\overline{\text{BUSREQ}}$ occurs at ① in figure 1?

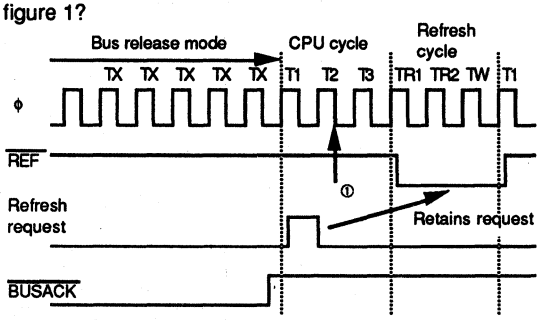


Figure 1 Bus Release Mode and Refresh

A

Only one refresh request is retained, and it is performed after the bus cycle following the CPU operation (one machine cycle) in the figure.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
√	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-061B/E
Item	EXTAL Input and ϕ Clock Output		

Q

What is the phase relationship between EXTAL input and ϕ clock output when an external clock is input through EXTAL?

A

ϕ clock changes synchronously with the falling edge of EXTAL (figure 1).

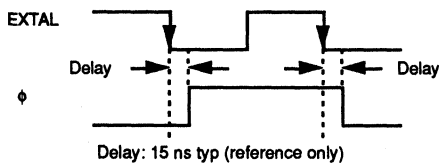


Figure 1 External Clock

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
√	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-062A/E
Item	ϕ Clock Output Frequency Error		

Q

Normally, ϕ clock output frequency is one half of the crystal oscillator frequency.

Why does ϕ clock output frequency equal the crystal frequency in our system?

A

How $\overline{\text{RESET}}$ and ST pins are handled may effect ϕ clock output frequency. Therefore, take the following two types of measures:

1. Check that the reset circuit design asserts the $\overline{\text{RESET}}$ signal for at least six clock states.

2. Do not pull down ST (it is an output signal).

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
√	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-096A/E
Item	ϕ Pin Handling		

Q

1. If ϕ pin is not used, can it be connected capacitively to GND (figure 1)?

2. How can radiated noise be prevented?

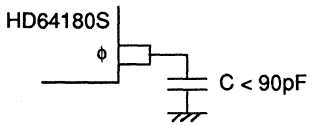


Figure 1 Unused ϕ Pin

A

1. Yes. The ϕ pin can be connected to GND through a maximum capacitance of 90pF.

2. Radiated noise can be reduced by shielding LSI.

Note that the ϕ pin should be opened normally so that it is an output signal.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
√	Clock Generator
	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

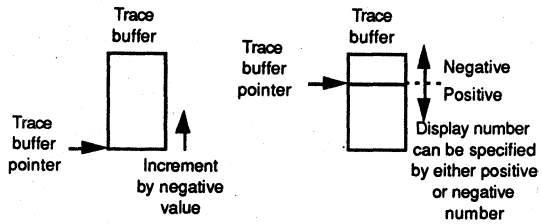
Type	HD64180S	Q&A No.	QA641-031A/E
Item	ASE Trace Function		

Q

How is ASE trace information displayed on the CRT?

A

After GO or STEP command execution, the trace buffer pointer indicates the last trace data location. Display numbers are indicated with negative values unless the pointer is moved with the LINE command. After that, the display number can be specified either with a positive or negative value (figure 1).



TRACE – n (CR) n: Displayed number

Figure 1 Displaying Trace Information

Classification

	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
√	ASE
	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-032A/E
Item	Dynamic RAM Refresh of ASE		

Q

Dynamic RAM refresh depends on the refresh control register programming.

Is the dynamic RAM refreshed during the wait state for command input when using ASE?

A

When ASE is used, dynamic RAM refresh is executed as follows, depending on refresh control register programming:

1. Refresh enable: REFE = 1

   If REFE bit is set to 1, dynamic RAM is refreshed and dynamic RAM contents remain intact, while ASE is waiting for command input.

2. Refresh enable: REFE = 0

   If REFE bit is set to 0, dynamic RAM is not refreshed while ASE is waiting for command input.

	Classification
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
√	ASE
	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-033A/E
Item	Difference between RET and RETI Instructions		

Q

What is the difference between the RET and RETI instructions?

A

Both RET and RETI instructions return from a subroutine to the main program. Both instructions have identical functions.

However, RETI is normally used to return from an external interrupt ($\overline{INT_0}$, $\overline{INT_1}$, or $\overline{INT_2}$) service routine.

RETI notifies peripheral devices of the completion of the current interrupt service routine.

Therefore, for internal interrupts, especially daisy-chained interrupts, the RET instruction is useful for identifying an internal interrupt service routine.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
√	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-034B/E
Item	LD A, R and LD R, A Instructions		

Q

Can the refresh address be read by executing an LD A, R or LD R, A instruction?

A

No, the refresh address cannot be read by executing the LD A, R or LD R, A instruction.

The HD64180S incorporates a dynamic RAM refresh controller. But the R counter indicates the number of CPU opcode fetch cycles and has no relation to dynamic RAM refresh.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
√	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-035A/E
Item	Processing Speed of SD200 Cross Assembler		

Q

What is the processing speed of the cross assembler for the SD200?

A

It is about 2.5 minutes per 1,000 steps (2.5 min./kstep) (floppy disk based) of the SD200 for S180XAS6F (version 1.0).

	Classification
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
√	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-036A/E
Item	LDIR Instruction		

Q

What is the bus cycle status during LDIR instruction execution?

A

Fourteen instruction cycles are repeated. The last execution cycle (BC = 0) is twelve instruction cycles. That is:

- BC ≠ 0: Fourteen instruction execution cycles are repeated
- BC = 0: Twelve instruction execution cycles are repeated

See figure 1.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
√	Software
	Others

Application Manual

HD64180S
Hardware Manual

Other Data

Reference Q&A

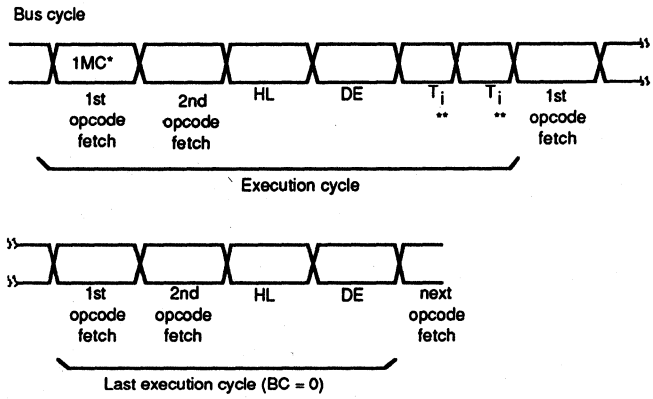
Comment

SECTION 2

Type	HD64180S	Q&A No	QA641-036A-2/E
Item	LDIR Instruction		
A			

LDIR (ED BO)
Operation

$(HL)m \rightarrow (DE)m$
$-1 \rightarrow BC_r$
$+1 \rightarrow DE_r$
$+1 \rightarrow$
Repeat until $BC_r = 0$



*: 1 machine cycle
**: Internal operation machine cycle

Figure 1 LDIR Instruction Timing

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180S	Q&A No.	QA641-063B/E
Item	Extension Instructions (IN0, OUT0)		

Q

Are there any limitations when accessing external I/O, by using the instructions whose higher address is 00H such as IN0 and OUT0?

A

Yes, the IN0 and OUT0 instructions can only access the lower 256 bytes of I/O space.

IN g, (c) and OUT (c), g instructions can access more than 256 bytes of I/O space (figure 1).

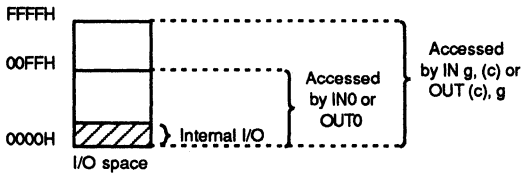


Figure 1 I/O Space Access

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
√	Software
	Others

Application Manual

HD64180S Hardware Manual

Other Data

Reference Q&A

Comment

SECTION 2

Type	HD64180S	Q&A No.	QA641-064B/E
Item	DEC (INC) and DAA Instructions		

Q

Normally, the DAA instruction is executed to obtain BCD data after ADD or SUB instruction execution.

Does the DAA instruction adjust the result after DEC (INC) instructions?

A

No, the DAA instruction does not support BCD adjustment after DEC (INC) instructions.

DAA execution results depend on flag conditions. See table 1 for an example.

Table 1 DAA Example

Instruction	Acc	Flag N	Flag C	Flag H
(Initial value)	00	0	0	0
DEC A	FF	1	0	1
DAA	F9 (FF + FA)	1	1	1

Refer to the HD64180S user's manual for details.

Classification	
	MMU
	MSCI
	ASCI/CSIO
	DMAC
	Timer
	Wait
	Refresh
	Chip Select
	Low Power Mode
	Reset
	Interrupt
	Bus Interface
	Clock Generator
	ASE
√	Software
	Others

Application Manual

HD64180S Programming Manual

Other Data

Reference Q&A

Comment

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD64180 8-Bit Microprocessor

Technical Q and A

Application Note

INTRODUCTION

The HD64180 is a high-performance 8-bit multichip microprocessor which has object code compatibility with the 80 families. System cost is reduced by incorporating such powerful components on-chip as CPU, MMU, DMAC, PRT, ASCI, and CSI/O. Internal Op-Code trap for error protection improves system reliability. As the HD64180 is fabricated by the advanced CMOS process technology, low power consumption and wide operational power supply voltage range are realized. The low power consumption modes(sleep and system stop) greatly reduce average power dissipation.

SECTION 2

HOW TO USE THIS MICROCOMPUTER
TECHNICAL Q&A MANUAL

This is a technical manual composed of answers to questions that many users have posed regarding Hitachi microcomputers in the recent months. This manual is intended to supplement the explanations in the current data books and user's manuals. Thus, please use this manual together with the user's manuals and the data books.

If any further questions arise as you use this manual and the products that described, please do not hesitate to get in touch with your nearest Hitachi semiconductor sales office.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

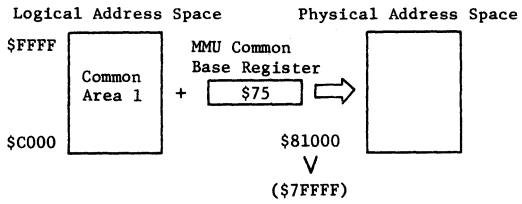
- Table of Contents -

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Item	MMU Operation		

Q

How will the MMU operate if MMU Base Registers (MMU Common Base Register and MMU Bank Base Register) are programmed to exceed 512k bytes of physical address space?

An example follows.



Classification

- ○ MMU
- DMAC
- ASCI
- CSI/O
- TIMER
- BUS INTERFACE
- INTERRUPT
- WAIT
- RESET
- LOW POWER MODE
- REFRESH
- CLOCK GENERATOR
- ASE
- SD
- SOFTWARE
- OTHERS

A

If physical address space exceeds 512k bytes, the carry bit is ignored and MMU accesses from physical address $00000.

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.

COMMENT

SECTION 2

No. QA641-002A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Item	Logical to Physical Address Translation		

Q

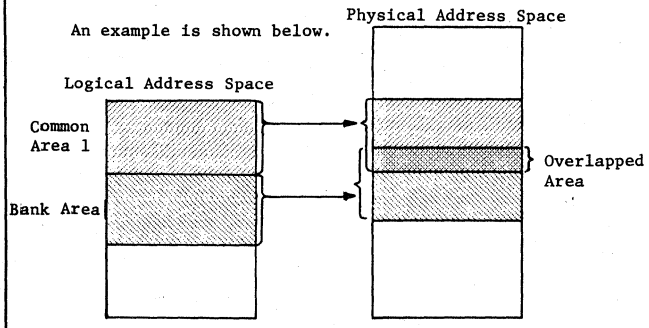
Can common area 1 overlap with bank area depending on the MMU Base Register's (MMU Common Base Register and MMU Bank Base Register) programming?

Classification	
○	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

Yes, common area 1 and bank area may overlap depending on the MMU Base Registers programming.

An example is shown below.



Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-003A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	DE1 bit in the DMA Status Register (DSTAT)		

Q

(1) How long is DMA transfer disabled when DE1 bit is set to 0?

(2) How does DMA restart?

Classification	
	MMU
O	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

(1) DMA transfer is disabled until DE1 bit is reset to 1. $\overline{DWE1}$ bit must be written with 0 when performing any software write to DE1.

(2) If memory ↔ memory DMA transfer is executed in burst mode, DMA transfer cannot be interrupted. It can only be interrupted in memory ↔ memory (cycle steal mode), memory ↔ I/O, or memory ↔ memory mapped I/O transfer mode.
To restart DMA transfer, DE1 must be set to 1.

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.:

COMMENT

SECTION 2

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	DME bit (DMA MASTER ENABLE bit) in DMA Status Register		

Q

When $\overline{NMI}$ occurs, DME bit is reset to 0 and DMA operation is disabled passing control to the CPU.

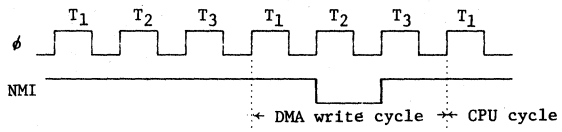
(1) How is DMA operation timing halted?

(2) How does DMA operation restart?

Classification	
	MMU
O	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

(1) When $\overline{NMI}$ occurs, the CPU is given control after the current DMA cycle is completed.
The following shows the timing



(2) To restart DMA operation, DE (DE0 or DE1) bit must be set to 1.

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-005A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	$\overline{\text{DWE}}$ bit in DMA Status Register		

Q

What is the function of $\overline{\text{DWE}}$ bit in DMA status register?

Classification	
	MMU
O	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

DE bit enables DMA operation of internal DMAC, while $\overline{\text{DWE}}$ bit enables a software write to its corresponding DE bit, for a specific channel operation.

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

SECTION 2

No. QA641-006A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Item	Memory ↔ I/O DMA Transfer		

Q

To enable external $\overline{DREQ_0}$ input, both A_{17} and A_{16} of I/O address must be set to 0.

Is DMA requested by $\overline{DREQ_0}$ acknowledged if either A_{17} or A_{16} is set to 1?

Classification

	MMU
O	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

No, if either A_{17} or A_{16} is set to 1, $\overline{DREQ_0}$ is disabled and the DMA request is not accepted.

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-007A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Item	Memory ↔ ASCI DMA Transfer		

Q

To execute memory ↔ ASCI DMA transfer, DMA source/destination address register should be programmed as follows.

(1) Bits A_0-A_7 must contain the address of the ASCI transmit or receive data register.

(2) Bits A_8-A_{15} must be set to 00H.

(3) Bits A_{16}-A_{17} must be set to "01" or "10".

Can the memory ↔ ASCI DMA transfer be executed correctly if bits A_8-A_{15} in DMA source/destination address register are not set to 00H?

	Classification
	MMU
O	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

No, if bits A_8-A_{15} in DMA source/destination address register are not set to 00H, Memory ↔ ASCI DMA transfer cannot be executed correctly.

For example, to execute ASCI (channel 0) RDR → memory DMA transfer, Bits A_0-A_7 must be set to 08H, bits A_8-A_{15} must be set to 00H and bits A_{16}-A_{17} must be set to "01" for correct ASCI (channel 1) RDR → memory DMA transfer.

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

Bits A_8-A_{15} set to other than 00H will cause internal DMAC to access other I/O address and not RDR.
(DMA request from ASCI channel 0 is not reset)

SECTION 2

No. QA641-008A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	Memory (specified in application program) ↔ I/O DMA Transfer		

Q

Is it possible to execute memory (specified in application program) → I/O DMA transfer independently of the MMU Base Register?

Classification	
	MMU
O	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

No, to execute memory (specified in application program) ↔ I/O DMA transfer correctly, physical source address must be defined as follows.

(1) Software calculates physical source address of data area using the logical source address and the Base Register.

(2) The calculated physical source address is loaded into the DMA source address Register.
When the physical address is known, it can be loaded into the DMA address register directly, but if DMA transfer is executed within the logical memory area, Block transfer instructions can be used.

Applicable Manual

Title

Other Data

Title

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	Memory ↔ I/O (Z80SIO) DMA Transfer		

Q

When memory ↔ I/O (Z80SIO) DMA transfer is executed while $\overline{DREQ_0}$ is programmed for level sense, DMA transfer does not complete correctly.

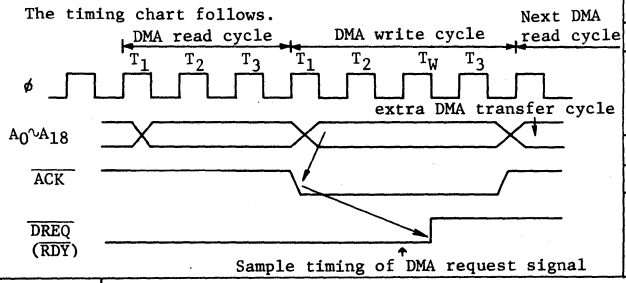
Are there any restrictions in DMA operation?
(RDY signal of Z80SIO is input to $\overline{DREQ_0}$ of HD64180)

Classification	
	MMU
O	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

When $\overline{DREQ_0}$ is programmed for level sense additional DMA cycle is executed since RDY signal (which is input to $\overline{DREQ_0}$) is set high after additional sampling of DMA request signal ($\overline{DREQ_0}$).

The timing chart follows.



Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

If $\overline{DREQ}$ is programmed for edge sense, DMAC will operate correctly.

SECTION 2

No. QA641-010A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐ Software ☐ Evaluation kit, Emulator ☐SD ☐SBC
Item	Break Level Transfer with Asynchronous Serial Communication Interface (ASCI)		

Q

Is it possible through software to perform break level transfer with ASCI?

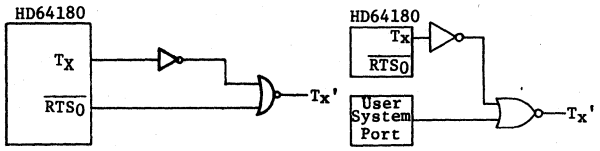
Classification	
	MMU
	DMAC
○	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

No, the (HD64180) ASCI cannot perform break level transfer through software.

However, break level can be transferred if an external circuit is connected to $\overline{RTS_0}$ pin and the user system port.

An example of a circuit for break level transfer follows.



If $\overline{RTS_0}$ bit in ASCI control register A or data register of port set to 1, break level "0" can be transferred.

Applicable Manual

Title

Other Data

Title

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-011A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit , Emulator ☐ SD ☐ SBC
Item	ASCI Baud Rate Calculation		

Q

How is ASCI baud rate calculated?

Classification	
	MMU
	DMAC
O	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

The following expression shows how to calculate ASCI baud rate.

baud rate =

$$\frac{\text{system clock (frequency)}}{\text{(sampling rate)}\cdot\text{(PS bit)}\cdot\text{(divide ratio set by SS0-SS2)}}$$

Note: Sampling rate : 16 or 64
PS bit : 10 or 30
SS0-SS2 : 1,2,4,8,16,32, or 64

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

SECTION 2

No. QA641-012A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit , Emulator ☐ SD ☐ SBC
Item	Timer Output		

Q

Does Timer (PRT) channel 0 provide a timer output function?

Classification

	MMU
	DMAC
	ASCI
	CSI/O
O	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

No, PRT channel 1 should be used for timer output. Note that TOUT pin is multiplexed with the A_{18} pin. Therefore, when accessing up to 512k bytes of physical memory address, A_{18}/TOUT pin always functions as A_{18}.

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.:

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-013A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Item	Timer (PRT) Counting Down Using External Clock		

Q

Can the PRT count down using the external clock?

Classification	
	MMU
	DMAC
	ASCI
	CSI/0
O	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

No.
The PRT can count down using only the ϕ clock (divided by 20).

Applicable Manual

Title

HD64180
Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

SECTION 2

No. QA641-014A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Item	Bus Status, during Internal I/O Access		

Q

What is the bus status during internal I/O access?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
O	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

Bus status during internal I/O access is as follows.

(1) Data bus (D_0 - D_7)

read cycle : high impedance
write cycle : data

(2) Address bus (A_0 - A_{18})

read/write cycle : address

Applicable Manual

Title

HD64180
Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-015A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit , Emulator ☐ SD ☐ SBC
Item	Interrupt during MMU Operation		

Q

How will MMU be affected if an interrupt occurs during its operation?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
O	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

If an interrupt occurs during MMU operation, the interrupt vector is relocated according to the MMU Base Register programming.

Therefore, the interrupt vector should be defined depending on the MMU Base Register programming.

However, it is possible for the interrupt vector to be located in Common Area 0 which is always located in logical address space.

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

SECTION 2

No. QA641-016A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐ Software ☐Evaluation kit , Emulator ☐SD ☐SBC
Item	Interrupt during DMA Operation		

Q

How will DMAC be affected if an interrupt occurs during its operation?

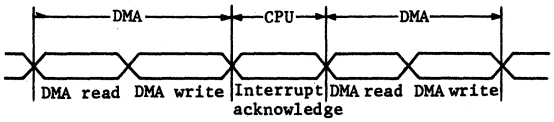
Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
O	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

(1) If $\overline{NMI}$ occurs, DMAC operation is disabled.

(2) If $\overline{INT}$ or an internal interrupt oecurs during memory ↔ memory DMA operation in burst mode, the interrupt is ignored.

(3) If $\overline{INT}$ or an internal interrupt occurs during memory ↔ memory DMA operation in cycle steal mode, the interrupt is acknowledged and the interrupt sequence (CPU cycle) and DMAC read/write (DMAC cycle) are executed as follows.



Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-017A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	$\overline{INT_0}$ Mode 2		

Q

In Z80 $\overline{INT_0}$ Mode 2, the LSB of lower vector in 16 bit vector address (A_0) is set to 0.

Is the LSB of lower vector (D_0) set to 0 in HD64180 $\overline{INT_0}$ Mode 2?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
O	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

No, in Z80 $\overline{INT_0}$ Mode 2, the LSB of lower vector is not automatically set to 0. The Z80 data book explains that the LSB (A_0) must be set to 0.

In HD64180 $\overline{INT_0}$ Mode 2, the LSB of lower vector (D_0) must be set to 0 since $\overline{INT_0}$ Mode 2 requires a 2-byte vector.

However, even if the LSB (D_0) is set to 1, the interrupt sequence is executed correctly.

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.

COMMENT

SECTION 2

No. QA641-018A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	$\overline{\text{NMI}}$ during Interrupt Acknowledge Cycle		

Q

Is $\overline{\text{NMI}}$ acknowledged during interrupt acknowledge cycle such as $\overline{\text{INT}}$?

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
O	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

Yes, one instruction (excluding EI and DI instructions) is executed after $\overline{\text{INT}}$ acknowledge cycle, then $\overline{\text{NMI}}$ acknowledge cycle starts.

$\overline{\text{NMI}}$ response sequence during $\overline{\text{NMI}}$ acknowledge cycle is the same as above.

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.:

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐ Software ☐ Evaluation kit, Emulator ☐SD ☐SBC
Item	Interrupt after RESET		

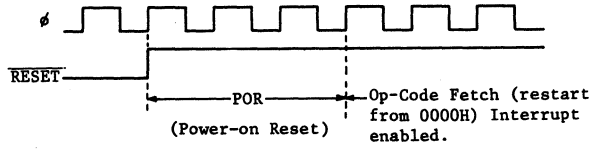
Q

Is $\overline{NMI}$ or $\overline{INT}$ acknowledged immediately after RESET?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
O	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

No, for three cycle immediately after RESET (Power-on Reset cycle), $\overline{NMI}$ and $\overline{INT}$ are disabled.
After these three cycles, the instruction is executed and interrupt is enabled. The timing is as follows.



Note : $\overline{NMI}$ can be latched immediately after POR cycle.

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.

COMMENT

SECTION 2

No. QA641-020A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	Interrupt during Refresh Cycles		

Q

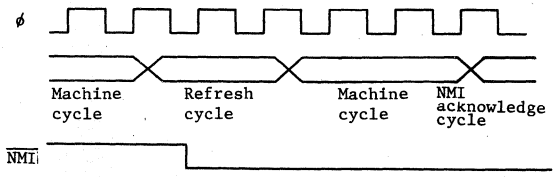
Is an interrupt ($\overline{NMI}$ or $\overline{INT}$) acknowledged during refresh cycles?

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
O	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

No, an interrupt is ignored during refresh cycles. However, $\overline{NMI}$ is acknowledged immediately after refresh cycles when executing instruction because $\overline{NMI}$ input is edge sensitive.
The $\overline{NMI}$ acknowledge timing after refresh cycle is as follows.



$\overline{INT}$ ($\overline{INT_0}$, $\overline{INT_1}$ and $\overline{INT_2}$) is ignored during refresh cycles. If $\overline{INT}$ remains active after refresh cycle it can be acknowledged during instruction executed just after reset.

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.

COMMENT

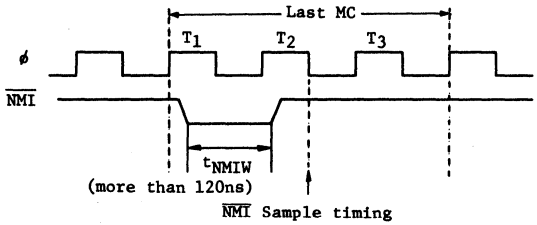
HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-021A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit , Emulator ☐ SD ☐ SBC
Item	$\overline{NMI}$ Acknowledge		

Q

Is $\overline{NMI}$ acknowledged if occuring during the following timing sequence?



MC: Machine Cycle

A

Yes, if t_{NMIW} ($\overline{NMI}$ pulse Width) is 120ns or more, $\overline{NMI}$ is sampled and $\overline{NMI}$ acknowledge cycle begins after last MC.

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
O	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.:

COMMENT

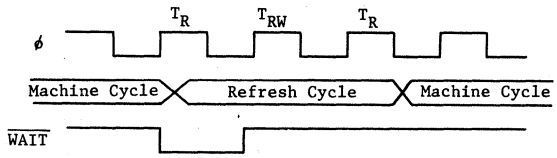
SECTION 2

No. QA641-022A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	WAIT Insertion during Refresh Cycle		

Q

Can WAIT cycle be inserted during refresh cycle by activating WAIT input?



	Classification
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
O	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

No, WAIT input is disabled during refresh cycle. However, the refresh cycle can be programmed to two or three cycles setting the REFW bit in the refresh control register accordingly.

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-023A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐ Software ☐Evaluation kit , Emulator ☐SD ☐SBC
Item	WAIT Function at I/O Access		

Q

Is wait state (T_W) always inserted when accessing I/O?

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
O	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

When accessing external I/O, a minimum of one wait state is inserted.
When accessing on-chip I/O, zero to four wait states are automatically generated depending on the status of CPU and on-chip I/O.
(ASCI, CSI/O, PRT DATA register access)

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.:

COMMENT

SECTION 2

No. QA641-024A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐ Software ☐Evaluation kit , Emulator ☐SD ☐SBC
Item	Power-on Reset Sequence		

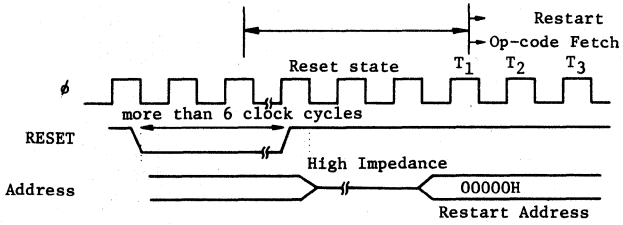
Q

How is the power-on reset sequence performed?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
O	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

The Power-on reset sequence is as follows.



Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

$\overline{\text{RESET}}$ pin should be held low for more than 6 clock cycles.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-025A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Item	Bus Status during Sleep Mode		

Q

What is the Bus status when the sleep instruction is excuted?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
O	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

The Bus status is as follows.

	Status
Address Bus	High (A_0 - A_{18} = 7FFFF)
Data Bus	3-state
Control signal	Inactive

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

SECTION 2

No. QA641-026A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit , Emulator ☐ SD ☐ SBC
Item	Sleep Mode and System Stop Mode		

Q

What is the difference between sleep mode and system stop mode?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
O	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

The major differences are as follows.

Mode	Sleep	System stop
function	CPU stop	CPU and internal I/O stop
exit	Interrupt (Internal/ External) Reset	Interrupt (External) Reset

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-027A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	Recovery from System Stop Mode		

Q

What is the system status after recovery from SYSTEM STOP mode?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
O	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

SYSTEM STOP mode is the combination of SLEEP and IO STOP modes.

The SYSTEM STOP mode is exited by detection of $\overline{NMI}$ or $\overline{INT}$ external interrupts only.

If interrupts are globally disabled (IEF1=0), instruction execution begins with the instruction following the SLEEP instruction.

If interrupts are globally enabled (IEF1=1), the appropriate normal interrupt response sequence is executed.

However, I/O STOP mode is maintained until the I/O STOP bit is set to 0 after recovery from SYSTEM STOP mode.

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

SECTION 2

No. QA641-028A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Item	System Standby Function		

Q

Does the HD64180 have a system standby function (stop clock) to reduce power consumption?

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
O	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

No, clock stop function is not provided. Minimize the clock frequency to reduce power consumption.

However, if clock is stopped completely, MPU operation and data in the registers are not guaranteed.

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.:

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-029A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	Dynamic RAM Refresh during DMA		

Q

Is DRAM refreshed during internal DMA operation?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
O	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

Yes, refresh cycle is inserted during internal DMA cycle.

Refresh controller does not distinguish DMA cycle from CPU cycle.

Dynamic RAM refresh is performed at the end of machine cycle during both CPU cycle and DMA cycle, and the internal and duration of the refresh cycle are programmable.

Applicable Manual

Title

Other Data

Title

Reference Q & A

No.

COMMENT

SECTION 2

No. QA641-030A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Item	Dynamic RAM Refresh		

Q

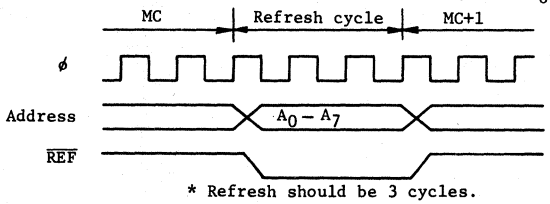
Is the refresh controller of the HD64180 different from that of the Z80?

What is the function of the R countor?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
O	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
	OTHERS

A

Yes, the refresh controller of the HD64180 differs from that of the Z80. Refresh cycle is inserted or suppressed by software. Also the internal and length of refresh cycles are programmable.
The refresh address (8-bit address) is output at A_0-A_7.



* Refresh should be 3 cycles.

The R Counter counts the number of CPU op-code fetch cycles and has no relation to dynamic RAM refresh.

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	Trace Function of ASE		

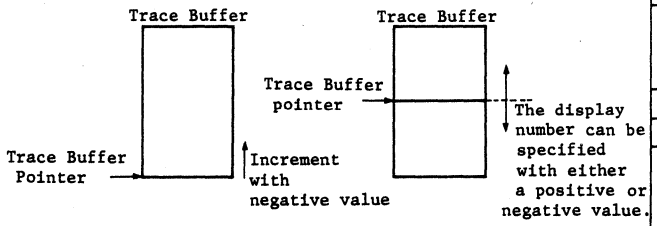
Q

How is the trace information of ASE displayed on the CRT?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
O	ASE
	SD
	SOFTWARE
	OTHERS

A

After excution of the Go or Step command, the trace buffer pointer indicates the last trace data.
Specify display number with negative value until the pointer corresponds with Trace Pointer.
After moving the Trace Buffer pointer with the Trace Pointer Command, it is possible to specify the display number with a positive value.



Applicable Manual

Title: H180AS01 User's Manual

Other Data

Title:

Reference Q & A

No.

COMMENT

SECTION 2

No. QA641-032A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	Dynamic RAM Refresh of ASE		

Q

Dynamic RAM is refreshed depending on the Refresh Control Register programming.

Is the dynamic RAM refreshed during the wait state for command input when using ASE?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
O	ASE
	SD
	SOFTWARE
	OTHERS

A

When using ASE, dynamic RAM refresh is executed as follows by programming the Refresh Control Register.

(1) Refresh enable: REFE=1
If REFE bit is set to 1, dynamic RAM is refreshed during waiting state for command input.

(2) Refresh disable: REFE=0
If REFE bit is set to 0, dynamic RAM is not refreshed during wait state command input.
(But refresh cycle is inserted during trace.)

Applicable Manual

Title

H180AS01
User's Manual

Other Data

Title

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-033A/E

Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit , Emulator ☐ SD ☐ SBC
Item	Difference between RET and RETI Instructions		

Q

What is the difference between the RET and RETI instruction?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
O	SOFTWARE
	OTHERS

A

Both the RET and RETI instructions are used to return to the main-program from a subroutine, and both instructions have identical functions.

However, RETI instruction is normally used to return from an external interrupt ($\overline{INT_0}$, $\overline{INT_1}$ or $\overline{INT_2}$) service routine.

Since RETI is a two-byte instruction, peripheral devices know the completion of the current interrupt service routine during RETI execution especially when using the daisy chain (Z80 Peripheral).

Also, when using an external interrupt, especially the daisy chain, RET instruction is useful in identifying an internal interrupt service routine.

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.:

COMMENT

SECTION 2

No. QA641-034A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐ Software ☐ Evaluation kit , Emulator ☐SD ☐SBC
Item	LD A, R/LD R, A Instructions		

Q

Can the refresh address be read by executing a LD A, R or LD R, A instruction?

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
O	SOFTWARE
	OTHERS

A

No, refresh address cannot be read by the LD A, R or LD R, A instruction.

The HD64180 incorporates a dynamic RAM refresh controller. But the R counter indicates the number of CPU op-code fetch cycles and has no relation to dynamic RAM refresh.

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-036A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	LDIR Instruction		

Q

What is the status of the bus cycle during LDIR instruction execution?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
O	SOFTWARE
	OTHERS

A

14 instruction execution cycles are repeated.

The last execution cycle (BC-0) is 12 cycles.

If BC ≠ 0, 14 instruction execution cycles are repeated.

If BC = 0, 12 instruction execution cycles are repeated.

(refer to the following page)

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

SECTION 2

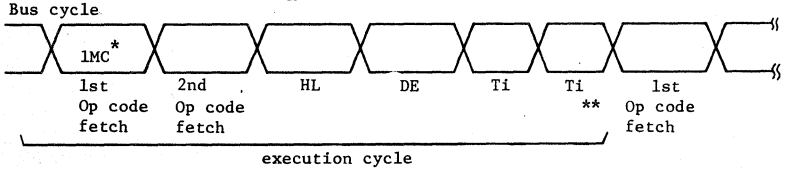
Type	HD64180	Div	☐ 4S ☐ 8S ■ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Item	LDIR Instruction		
Q			

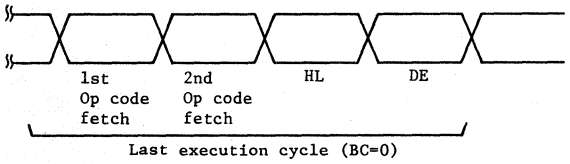
LDIR (ED Bø)

Operation

$(HL)m \rightarrow (DE)m$
$BC_R-1 \rightarrow BC_R$
$DE_R+1 \rightarrow DE_R$
$HL_R+1 \rightarrow HL_R$
Repeat until $BC_R=0$

Bus cycle





* 1 machine cycle

** machine cycle for internal operation

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-037A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	E Clock during Sleep Mode or Bus Release Mode		

Q

Is it possible to extend E clock pulse width by inserting wait status (T_W) during Sleep Mode or Bus Release Mode?

Classification

	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
O	OTHERS

A

No, since $\overline{WAIT}$ input is ignored during Sleep Mode or Bus Release Mode, E clock cycle cannot be extended.

Applicable Manual

Title: HD64180 Data Sheet

Other Data

Title:

Reference Q & A

No.:

COMMENT

SECTION 2

No. QA641-038A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐ Software ☐Evaluation kit, Emulator ☐SD ☐SBC

Item	E Clock Timing during DMA Cycles or Refresh Cycles

Q

What is the E clock output timing during the DMA or refresh cycle?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
O	OTHERS

A

DMA access memory or I/O duration of E clock output 'High' is identical to the CPU.

Thus, the output timing is as follows.

M R/W cycle	$T_2\uparrow \sim T_3\downarrow$
I/O read	1st $T_W\uparrow \sim T_3\downarrow$
I/O write	1st $T_W\uparrow \sim T_3\uparrow$

During refresh cycle clock output is held low.

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

No. QA641-039A/E

Type	HD64180	Div	☐4S ☐8S ■8M ☐16M ☐Software ☐Evaluation kit, Emulator ☐SD ☐SBC
Item	Internal I/O and External I/O Access		

Q

How is internal I/O accessed if an external I/O address conflicts with an internal I/O address?

Classification	
	MMU
	DMAC
	ASCI
	CSI/O
	TIMER
	BUS INTERFACE
	INTERRUPT
	WAIT
	RESET
	LOW POWER MODE
	REFRESH
	CLOCK GENERATOR
	ASE
	SD
	SOFTWARE
O	OTHERS

A

(1) Read case
Internal I/O is read.

(2) Write case
Both internal and external I/O are written with the same value.

Applicable Manual

Title

HD64180 Data Sheet

Other Data

Title

Reference Q & A

No.

COMMENT

SECTION 2

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD64180R/Z, HD647180X

Application Note

Working with Interrupts

Marnie Mar

Introduction

Hitachi's HD64180R/Z and HD647180X devices suit many controller applications that take advantage of the services of these devices' on-chip features. Such applications also require interrupt handling to respond to events occurring in the system.

This Application Note discusses details of the 64180 device's interrupt prioritization and handling capabilities. Included are examples of the use of the on-chip Asynchronous Serial Communications Interface transmit and receive interrupts.

This Application Note supplements information that is available in the 64180 family Hardware Manuals. For more information on interrupts and details on the registers mentioned, please refer to the Hardware Manual for the HD64180R/Z or the HD647180X.

Enabling Interrupts

A hardware reset disables all interrupts except NMI and TRAP, which are non-maskable. The following steps must be taken in order for the interrupt controller to accept and handle maskable interrupts:

1. When using INT0 Mode 2 or internal interrupts, load the Interrupt Vector Register (I) with the most significant eight bits of the 16-bit vector table address.

2. When using internal interrupts, load the Interrupt Vector Low Register (IL) to access to vector table.

3. For external interrupts, set the ITE2, ITE1 and ITE0 bits of the INT/Trap Control Register (ITC) to 1 to enable INT2, INT1 and INT0 respectively.

4. For internal interrupts, set the control register bits for the function to enable the required interrupt. For instance, enable the Transmit Interrupt for the Asynchronous Serial Communication Interface (ASCI) by setting bit 0 of the ASCI Status Register (STAT) to 1. Bit 0 is the Transmit Interrupt Enable (TIE) bit.

5. Finally, when the required vectors are initialized and interrupts enabled, the CPU must execute the Enable Interrupt (EI) command. This sets the IEF1 and IEF2 flags to 1. As described in the hardware manuals, IEF1 controls general enabling and disabling of maskable interrupts (enabled when IEF1 = 1). IEF2 manages the occurrence of NMI.

The 64180 maskable interrupts are enabled at two levels, a general enable using the IEF1 bits (controlled using the EI and DI instructions), and an interrupt specific enable in either the ITC or a on-chip control register. This combination of enabling allows the user to control when each interrupt can be accepted.

Interrupt Service Routines

All vectored interrupts require a user-written interrupt service routine (ISR) to execute the system response to the interrupt. This routine should perform the following steps:

1. Save all registers used by the ISR, or swap to the alternate register set using the EXX and EX AF,AF' instructions.

2. Perform interrupt handling tasks, polling status bits as necessary to determine the exact cause of the interrupt.

3. Restore registers saved or swapped.

4. Reenable interrupts using the EI command. When the CPU accepts an interrupt, the IEF1 flag is set to 0 preventing all other interrupts from occurring. The EI command sets this flag back to 1 to allow other interrupts to be accepted.

5. Return to the interrupted program using either RET or RETI. Both perform the same function, but RETI is a two byte instruction. Use RETI when interfacing to an external I/O device that decodes RETI to determine the end of an interrupt service.

Interrupt Prioritization

The 64180 interrupts are prioritized in order from highest to lowest as shown in the hardware manual for the particular 64180 device in use. When more than one interrupt occurs at a time, this prioritization determines which interrupt is accepted and serviced.

Once the CPU accepts an interrupt and interrupt service

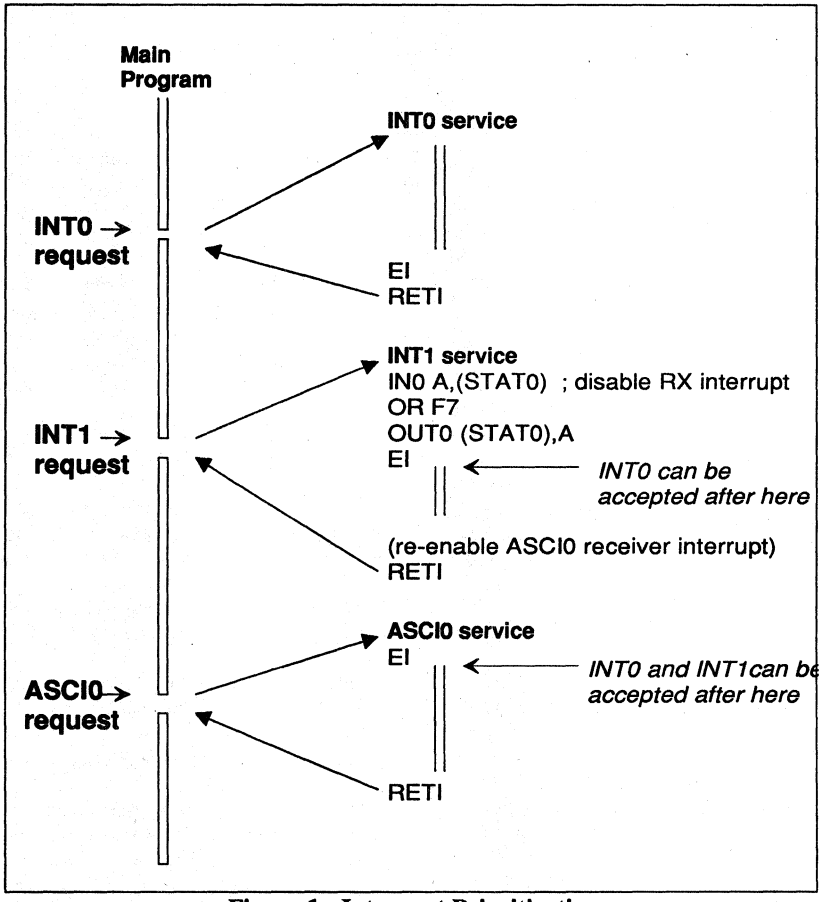


Figure 1 - Interrupt Prioritization

begins, no other interrupt can be accepted until the EI instruction is encountered in the executing interrupt service routine. When EI executes, the IEF1 flag is set, and any new or pending interrupt can be accepted.

Keep this in mind when setting up systems requiring nested interrupts, or higher priority interrupts that should preempt lower priority interrupt service. The EI command can be placed anywhere in an interrupt service routine, with the result of any enabled interrupt being acceptable following EI execution.

The 64180's interrupt prioritization allows the highest pending interrupt request to be serviced, but does not prevent a lower priority enabled interrupt from being accepted during a higher priority interrupt service routine once EI is executed. To prevent a lower priority interrupt from occuring in such a case, disable this interrupt at the control register level during the higher priority interrupt service routine.

Figure 1 shows an example of allowing higher priority requests to interrupt a lower priority interrupt service routine. In this example, INT0 is generated by an external source to signal an alarm condition that should trigger a response in most cases, but sometimes is masked. INT2 should be serviced without interruption, except when external circuitry triggers INT0. In addition, the system receives data through the ASCI under interrupt control.

Once the system has been initialized, only these three interrupts are enabled (INT0, INT1, and ASCI0). Since the INT1 service should not be interrupted by an ASCI0 interrupt, this routine should start by masking the ASCI0 receive interrupt by setting the RIE bit of STAT0 to 0 (this interrupt should be reenabled prior to returning). The next step of this routine, and the first step of the ASCI0 service routine would be to execute the EI command. This allows all other enabled interrupts to be executed.

Asynchronous Serial Communication Interface (ASCI) Interrupts

The 64180 ASCI channels can each generate an interrupt to a separate vector location. The ASCI channels can be programmed to generate an interrupt on either the receiver condition, the transmitter condition, or both.

An interrupt can be generated by the receiver condition when the receive data register is full (RDRF), or an overrun, parity or framing error occurs. An interrupt can be generated by the transmitter when the transmit data register is empty.

If both the receiver and transmitter interrupts are enabled for an ASCI channel, this channel's interrupt service routine must check to see which source caused the interrupt. Do this by polling the ASCI status register.

Interrupt driven serial data transfers can minimize CPU time while maximizing data transfer rates. When using interrupt driven data reception, data can be received into a buffer in memory via an interrupt service routine. This memory buffer can be accessed by the CPU when it is ready to read the data. This frees the CPU from polling the ASCI receiver, and allows the CPU to work on other tasks.

Interrupt driven data transmission can also minimize CPU involvement. Place data to be transmitted in a buffer in memory, and enable the transmit interrupt. Each time the transmit data register is empty, an interrupt service routine is executed. This routine takes a byte of data from the memory buffer and places it in the transmit data register. The service routine should disable the transmit interrupt when all the memory buffer data has transferred.

Whether to use interrupt driven serial I/O depends on the amount of data being transferred, and the relationship between the data transfer speed and the system operating speed. For slower system speeds and faster data transfer rates, the amount of time required to enter and process an interrupt service routine may offer little benefit over a polling scheme. Consider these parameters when choosing between interrupt driven or polled schemes.

Summary

The HD64180R/Z and HD647180X devices provide users with an interrupt controller with two levels of interrupt enabling, fixed prioritization, and flexibility that allows nested interrupts and preemption of lower priority interrupts by higher priority requests.

Both off-chip interrupts and interrupts generated by the on-chip peripheral functions can be handled. The ASCI interrupts, described in detail here, can be used to maximize data transfer rates, while minimizing CPU time for controlling these transfers.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD64180 Family

Application Note

Demo Board to PC Bus Shared Memory Interface

I. DESIGN NOTES:

The device used to provide shared memory is the Hitachi HD63310R Smart Dual Port RAM (SDPRAM). This device contains 1Kx8 of dual ported memory and 32 control/status registers. The devices is configured as two banks of 512 bytes of memory operating with a non-multiplexed address bus. Bits 0 and 1 of the Command Register (CMD) select whether the lower or upper bank is to be accessed by subsequent reads and writes to the device. The SDPRAM is mapped into the memory space of the PC and the '180 Board. The SDPRAM's control/status registers, including the semaphore registers are mapped into the I/O space of the '180 Board, and into the memory space of the PC. See Diagram 3 for this mapping. For more information on the operation of the SDPRAM, refer to the HD63310R data sheet.

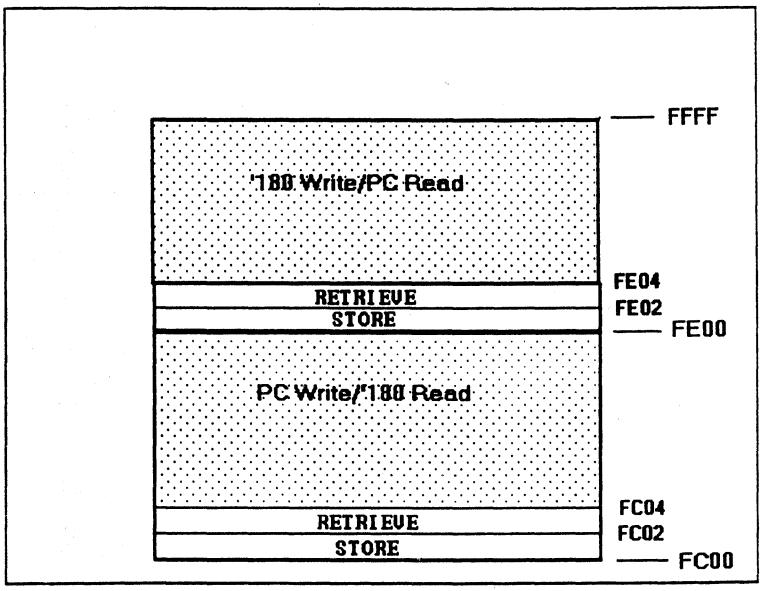


Diagram 1 - Shared Memory Divisions

The 1Kx8 of shared memory is divided into two equal portions. If we look at it like the '180 Monitor sees shared memory (occupying addresses FC00-FFFF) the first section starts at FC00 and ends at FDFF, and the second section starts at FE00 and ends at FFFF. Diagram 1 displays these divisions. The PC writes to the same section that the '180 Monitor reads from and the '180 writes to the PC's read section. Both portions are implemented as circular queues. When the last available memory location in a particular section is filled the next character is put at the start of the section (wrapping around to the beginning).

Two pointers for each section tell the requesting party where to put data or where to get data from. In both shared memory sections these pointers occupy the first four bytes of space available. These two pointers are called STORE and RETRIEVE. Diagram 1 (above) shows where these pointers are located in relation to the rest of shared memory. These pointers only contain the offset from the base of the shared memory section. The '180 always sees shared memory as FC00-FFFF but the PC may map this memory to one of several memory segments. To allow for this difference we only store the offset.

Memory semaphores are used explicitly by both the '180 Monitor and the '180 Client software. We use two of the eight semaphores available, each one controlling access to one portion of shared memory. The first semaphore controls access to shared memory FE00-FFFF and the second to memory locations FC00-FDFF.

Each time we access shared memory we follow the following steps:

1. Get the appropriate memory semaphore.
2. Access the appropriate pointer (STORE if you are writing a character, RETRIEVE if you are reading a character).
3. Add the contents of this pointer to the base address of the shared memory section you are accessing.
4. Perform the operation.

To demonstrate how this works let us assume that we want to write a character, 'X', from the PC to the '180. Let us also assume that our PC maps shared memory to C5C00-C5FFF. The layout of shared memory as seen by both the '180 and PC is shown in Diagram 2 (below).

The first step is to get the write semaphore. For the SDPRAM, this is done by writing a "1" to bit 1 of the Acquire Ownership Register (AQR). Once written, this bit is read back to determine if it is set. If it is set, the semaphore has been acquired. If not, continue to write "1" and read back until the bit reads back set.

Next we read the contents of memory location C5C00-C5C01 to get the 2 byte offset that represents the location where we need to write the character. Let us assume that that offset is C44.

We add that offset to the base address of the PC write section of shared memory, C5000 to get the complete address, C5C44. We store the character, 'X', in location C5C44 and increment the STORE pointer by 1 to obtain C45.

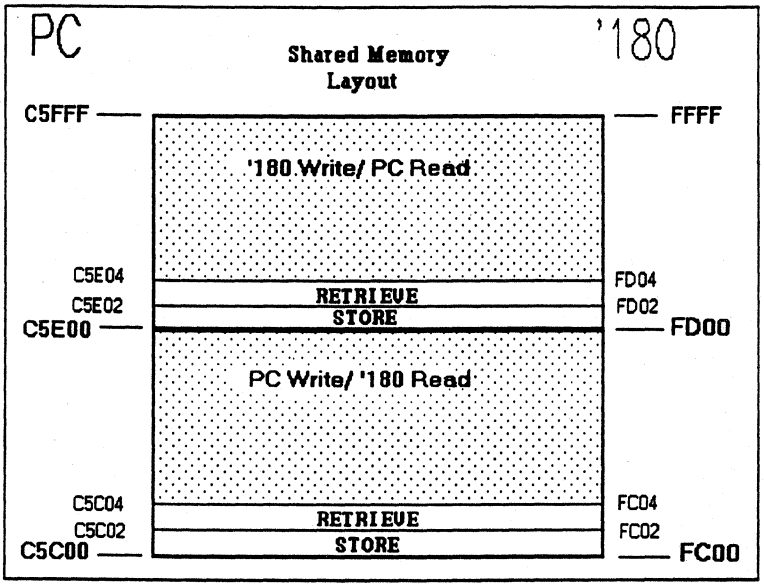


Diagram 2 - Shared Memory as seen by the '180 and the PC

Finally we write the new value for STORE, C45, to memory location C5C00-C5C01 and release the semaphore. To release the semaphore, bit one of the Release Available Register (REL) must be written with "1".

This is a very simplified example. We also need to be concerned with these factors:

-What if shared memory is full?
-What if we cannot get the memory semaphore?

What if shared memory is full?

First we need to know how to determine if shared memory is full. Since we already have pointers to the start and end of the circular queue we can compare the memory offsets that these pointers contain to determine if there is space available. In all cases the pointers are updated after the operation is complete, so we do our checking before the operation is done. If the start of the queue is equal to the end of the queue then the queue is empty. If the start of the queue is equal to one greater than the end of the queue then the queue is full.

If the queue is full we keep checking it until there is space available. The only time when this is a problem is when one side is writing to shared memory and the other side is also writing to shared memory (and thus not reading anything from shared memory). Eventually in this scenario shared memory would fill up and the sending side would stop and wait for available space. When the other side stops writing to shared memory it will read from shared memory, freeing up space.

There is only one specific instance when any type of deadlock can occur. If the '180 is sending a large amount of output to shared memory (such as a large dump) and is not reading any input into shared memory and the PC is also sending a large amount of output to shared memory and was no reading any input into shared memory, and this condition continues until both portions of shared memory are filled up then both sides will be waiting for free space (in a deadlock condition).

This instance is impossible since the only operation that can be run on the PC that sends a lot of data to SM without ever reading from shared memory is the File Transfer function, which requires the '180 to be in the 'load' state (always reading, never writing). So the way we handle the shared memory full condition works.

What if we cannot get the semaphore?
If the semaphore is not available we keep trying to get it until it is. The semaphore is obtained before each character is read or written and released when that operation is done, so the semaphore is not held for a long time. If we do have to wait it will not be noticeable.

II. IMPLEMENTATION NOTES

1. Automatic Interface Selection
When the '180 Monitor first starts up (or when a RESET is performed) the user is asked to press return to continue. At this point we determine which interface the <return> came from and make it the primary interface. All future output is only sent through this interface (before this point all output is sent to both interfaces).

To use the other interface the user may perform a RESET and press return on the terminal connected to the other interface.

2. Selecting the PC Memory Segment
Since there are several segments that the PC can use to map shared memory we have to find out which is being used. This is done at the hardware level and at the software level.

At the hardware level the DIP Switch SW3 must have positions 3 and 4 set to indicate the memory segment being used (see the User's Guide to the Shared Memory Interface). At the software level if the memory segment is not the default segment (C500) the user must enter the segment being used as a command line parameter when starting the NPU Client software on the PC.

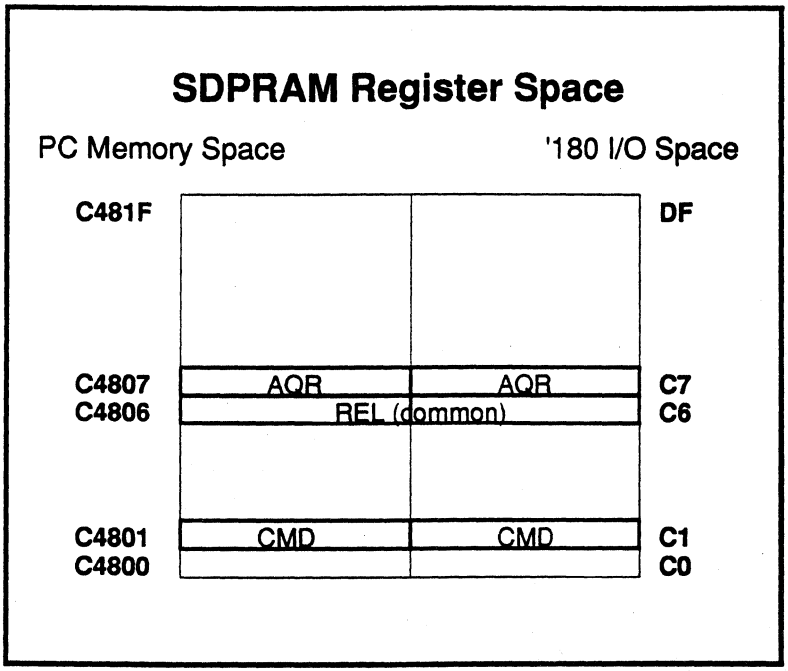


Diagram 3 - SDPRAM register space mapping for registers used by this application

All four possible mapping schemes are supported as long as both the hardware and software settings are correct.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD64180R,Z

Hardware Notes

180 Applications Board

Marnie Mar

The 64180 Applications Board (Aps Board) is a general purpose CPU board that can be used as the basis for systems designs using Hitachi or other manufacturer's peripheral devices. This document discusses the hardware features of this board, which are summarized below:

- HD64180R or Z operating at 9.216MHz

- SRAM (64K or 128Kword byte-wide device)

- EPROM (16Kbyte or larger device)

- HD63310 Smart Dual Port RAM

- PC bus interface with selectable address space

- RS-232 level interface from 64180 ASCI port to DB-9

- Memory decode logic for 5 additional memory spaces

- I/O decode logic for 6 additional I/O spaces

Logical Memory Map
The 64180's Memory Management Unit (MMU) is used to access a larger than 64Kbyte memory space. The memory space has been defined assuming that the MMU's logical space is assigned at reset, and not changed during the course of program operation. This logical space assignment is shown in Figure 1. Although some of the information to follow assumes that this logical space setup will be used, keep in mind that this setup is defined by software, and can be modified.

Physical Memory Map
The board's physical memory map is shown in Figure 2. The EPROM space is assigned to low memory, which allows it to be accessible as Common Area 0 at all times. The SDPRAM space is assigned to high memory, and is accessible through Common Area 1.

The SRAM, although it consists of a contiguous block of memory, is accessed in two separate areas of the physical memory. See the section on RAM Memory Map for more information.

The board's memory decode logic supports chip selects for five other blocks of memory. These select lines can be tied directly to user add-on hardware.

The SRAM and User Memory spaces are accessed through the MMU bank area. Portions of each section are accessed by programming the MMU's Bank Base Register (BBR) with the information that causes the desired area of memory to be addressed.

The areas indicated by cross-hatching should not be used for User memory space, since the decode logic allows these spaces to be overlayed by Aps Board memory spaces. For instance, if a 27256-type EPROM is used (8000h bytes), the

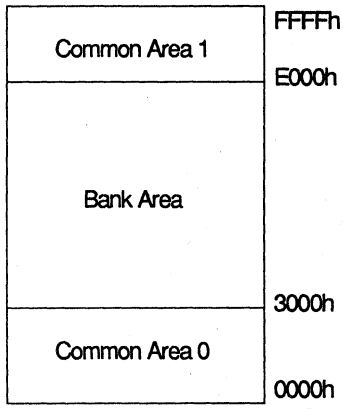


Figure 1 - Logical Memory Space

device will be activated for addresses 0000h-7FFFh, 8000h-FFFFh, 10000h-17FFFh, etc., since the EPROM is chip selected for all physical addresses in the range 00000h-2FFFFh, and the device responds address lines A0-A14.

Physical I/O Space
The Aps Board I/O space is shown in Figure 3. The SDPRAM control registers have been mapped into locations C0-FFh. HD64180R/Z on-chip I/O registers are mapped into locations 00-3Fh.

The locations identified as IO0, IO1, etc. are those that can be accessed using chip-select signals generated by I/O decode logic. The remaining I/O space is available for other system use.

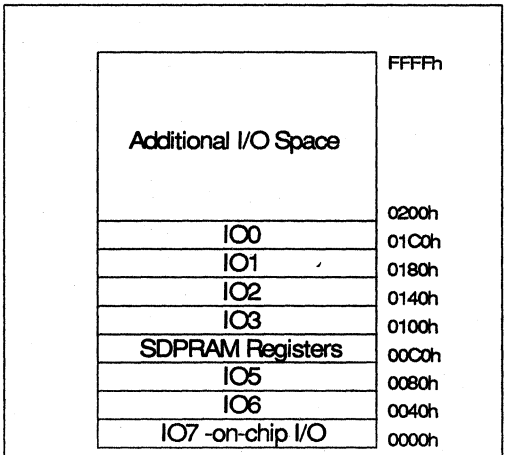


Figure 3 - System I/O Map

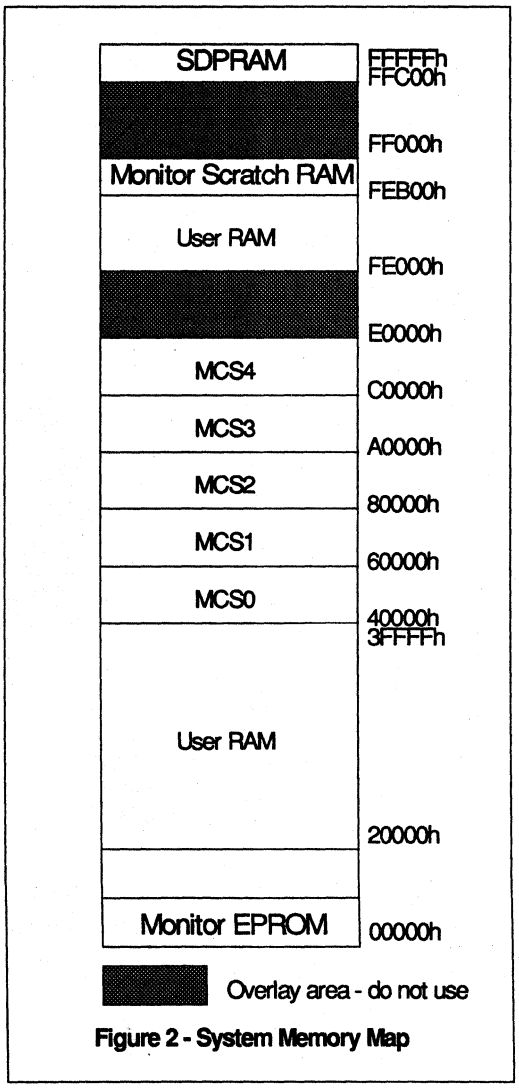


Figure 2 - System Memory Map

RAM Space
The RAM space is divided into two areas, one accessible through the bank area, and one accessible through Common Area 1. The bank area RAM would most likely be used for download of user code. The area accessed through Common 1 serves two purposes - one area (FEB00h-FEFFFh) is used by the monitor for scratchpad RAM, the remaining area (FE000h-FEAFFh) can be allocated by the user.

If 64Kbytes of RAM are used, physical locations 2E000h-2EFFFh are accessed through Common 1 for scratchpad and high user RAM.

Using the MMU set-up described earlier, physical RAM locations 3F000h-3FFFFh cannot be accessed by the processor, since this area overlays the logical memory space allocated to the SDPRAM. All other RAM areas can be accessed by loading the MMU Bank Base Register (BBR) with the appropriate address. For instance, to access physical locations 20000h-2AFFFh through the logical Bank Area located from 3000H-DFFFh, program the BBR with 1Dh.

SDPRAM Use
The HD63310 is a 1Kbyte Smart Dual Port RAM (SDPRAM) device which is used by the Aps Board for communication

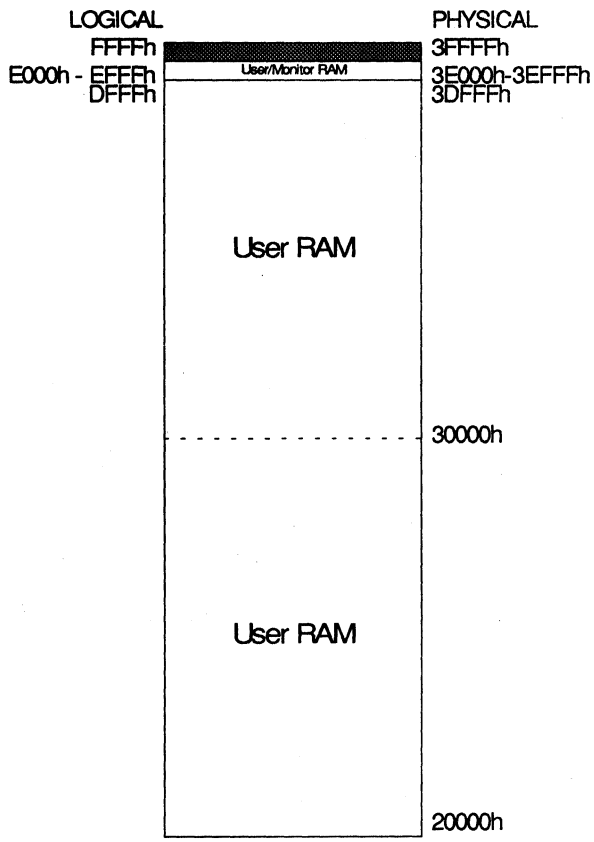


Figure 4 - SRAM Memory Map

with the host processor of a PC. The RAM is divided into two sections, one for transmissions of data from the Aps Board, and one for reception of data to the Aps Board. The first four bytes of both area are reserved for two 16-bit addresses. One address is a pointer to the next free location in the buffer, the other is a pointer to the next byte to be read from the buffer. These pointers are accessed and updated under program control by both the 64180 and the PC host CPU.

Control of access to each buffer is handled by a semaphore. Two bits of the 63310's Acquire Ownership register (AQR) are each assigned to one half of the DPRAM. The control programs obtain the semaphore before accessing the associated portion of DPRAM by writing a "1" to the appropriate bit of the AQR, and reading back until a "1" is read from that bit. When the access is complete, the program writes a one to the same bit position of the Release/Available Register (REL), which frees the semaphore.

Due to pin limitation, only 9 address lines (for 512Kbytes) are available to access the DPRAM in the non-multiplexed bus mode used. In order to indicate which half of the DPRAM these 9 address lines access, it is necessary to set bits in the 63310's Command Register.

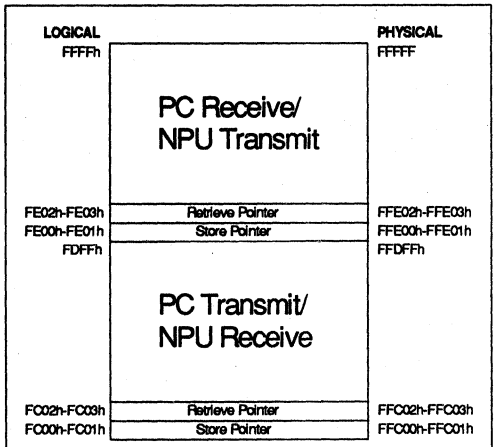


Figure 5 - Shared Memory (SDPRAM) Map

Memory Access Speed

The aps board has been designed to operate with minimum 1 wait state for all memory accesses. The SDPRAM is able to add wait states to bus cycles in the case of access contention. The SDPRAM's READY signal is used to generate an input to the WAIT line of the '180 and the PC bus to extend bus cycles as necessary.

The limiting factor on wait states is during access to the SDPRAM. In order cause the WAIT input on the '180 to add wait states to a cycle, the WAIT line must be pulled low in time to meet the setup required in T2. As the memory decode logic is currently designed with memory chip selects being determined based on the ME signal, it is impossible to pull the WAIT line low in time to increase the length of bus cycles when the SDPRAM is being accessed.

HD64180S (NPU)

Application Note

Using the NPU in AppleTalk Applications

Marnie Mar

OBJECTIVE

Hitachi's HD64180S NPU (Network Processing Unit) features an on-chip Multi-Protocol Serial Interface channel (MSCI) which handles asynchronous, byte-synchronous, and bit-synchronous protocols. This Application Note discusses the NPU's MSCI and its capability to support the requirements of transmitting and receiving data under AppleTalk's LocalTalk Link Access Protocol (LLAP).

AppleTalk's LLAP corresponds to the data-link layer of the ISO-OSI model. The physical data link used by LLAP is a shared link common to all nodes in the network. LLAP's main responsibilities are to:

- perform data transmission and reception
- provide link access control
- provide a way to address nodes

This application note addresses the NPU's hardware capabilities which handle the first two items. Address determination for nodes on the link is handled by high level language software routines, which are described in detail in *Inside AppleTalk*, available from Addison Wesley.

SCOPE

The NPU's features that aid in the implementation of AppleTalk's LLAP will be discussed here, along with details on how to use these features. Programs enabling the NPU to control these features and enact the LLAP protocol are also discussed. Program examples for receiving and transmitting frames using the LLAP protocol are given in the appendices.

This Application Note is based on routines that were written using the LLAP procedural model given in *Inside AppleTalk*. These routines were written in C and assembly language, and were tested by executing them on an NPU development board connected to an Apple Macintosh using the LocalTalk Connector Cable. LLAP Peek and Poke routines, available from Apple Developers Association, were run on the Macintosh to allow the Mac to send and receive individual LLAP frame dialogs.

A brief description of LLAP is given in Appendix A. For more information on AppleTalk or LLAP, please refer to *Inside AppleTalk*.[1]

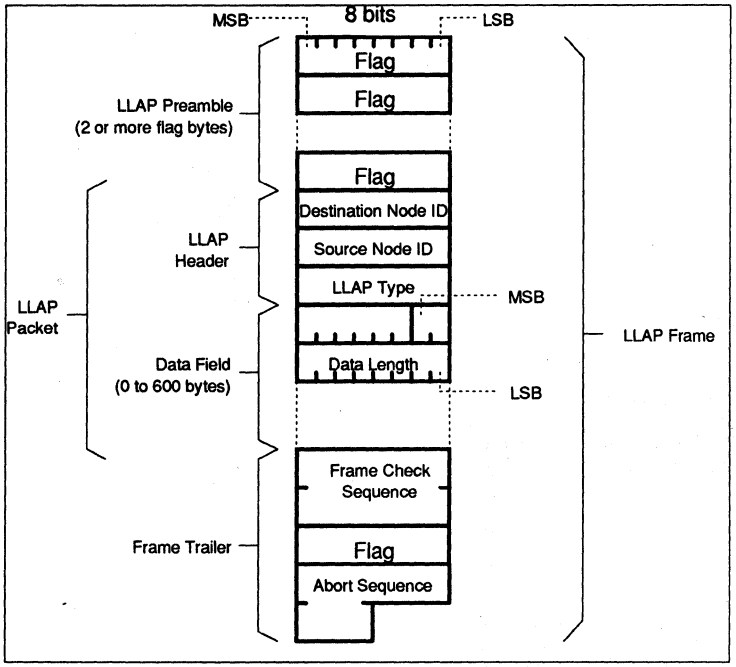


Figure 1 - LLAP Frame

A brief description of the NPU features as they relate to this application is given in Appendix B. For detailed information on the NPU, please refer to the HD64180S NPU Hardware Manual.[2]

NPU IMPLEMENTATION OF LLAP

The LLAP protocol requires that transmitted data be packed into frames, which are then unpacked by the receiver. The NPU's hardware has the capability of handling much of this packing and unpacking without additional software overhead. See Figure 1 for the LLAP frame format.

The LLAP protocol defines application programs that take care of acquiring the bus prior to transmit, transmitting frames, and receiving and processing frame data. The NPU's CPU core can execute these programs with the help of the on-chip MSCI, timers, and DMAC.

The NPU's MSCI contains a rich set of features that can be enabled and manipulated by programming bits in a set of control registers associated with this channel. These features are described below as they relate to NPU LLAP operation.

Bit Synchronous Operations

The MSCI channel can be programmed to operate in a bit-synchronous mode which supports HDLC-type frame transmissions. The features of this mode (optional Flag pattern output on idle, optional address checking on receive, CRC generation/checking support) directly support the requirements for LLAP frame communications.

FM0 encoding

The MSCI can be programmed to transmit and receive data with FM0 type coding (see Figure 2). Other encoding options are NRZ, NRZI, Manchester, and FM1 types.

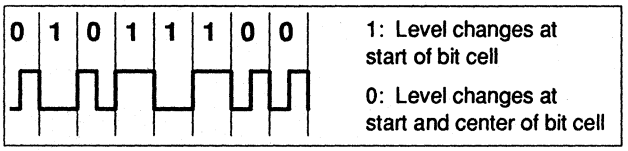


Figure 2 - FM0 encoding

To specify the encoding type desired, the user would program the NRZFM, CODE1, and CODE0 bits of MSCI Mode Register 2 (MMD2).

Idle/Flag Pattern

Whenever the MSCI transmitter is enabled but no data is available for transfer, it is said to be in the Idle mode. The MSCI supports the output of Flag bytes during idle transmission times if the Idle State Control (IDLC) bit of the MCTL is programmed to "1" (causing an idle pattern to be transmitted during idle states), and the idle pattern is written to the MSCI Idle Pattern (MIDL) register. For LLAP and many other bit-synchronous protocols, this pattern is defined to be 7Eh.

Error Checking

The MSCI has built in CRC generation and checking capabilities which directly supports the CRC-CCITT algorithm specified by LLAP. In order to enable CRC controller operation, the CRC Calculation Code (CRCCC) bit in MMD0 should be set high, the CRC1 bit (bit 1) should be set to 1 to select CRC-CCITT, and CRC0 (bit 0) should be set to 0 to indicate an initial value of all 0's.

Address Checking for Received Frames

Once the MSCI receiver has detected a Flag pattern, it is prepared to receive the LLAP frame that follows. The addressing scheme used by LLAP defines that the first non-flag byte received is a destination address byte, which indicates which node the data is intended. The NPU's Address Field Check capabilities can be used to eliminate the need to check the address of each incoming frame under program control. To cause address checking to occur, MMD1 should be programmed with 40h to cause Single-byte addresses to be checked. MSA0 should be programmed with the node's own address, as determined by the LLAP node initialization procedure. This will cause the MSCI to compare the first byte (destination address byte) of all incoming frames with the contents of MSA0 (node's own address) to determine if it should continue to receive the frame data.

If the address bytes match, the frame data will continue to be received into the MSCI receive buffer to be placed in memory under DMA or program control. If not, the MSCI will re-enter the Flag wait state.

The NPU recognizes destination bytes of FFh to indicate broadcast frames, and accepts the data of any frame with this destination byte.

ADPLL: Generation of Transmit and Receive Clocks

Since LLAP uses FMO data encoded with clock information, the receiver node must extract the clocking information from the incoming data in order to read this data. The NPU's Advanced Digital Phase Locked Loop circuit (ADPLL) handles this task.

Receive Clock Extraction

To use the ADPLL clock extraction function, an operating clock with frequency 8, 16 or 32 times the bit rate must be used. This operating clock can be generated by the baud rate generator, or can be input via the RXCM input line from and external clock.

The ADPLL operating clock's bit rate is specified in MMD2. MMD2 would be programmed as follows:

MMD2: C0h (1100 0000)

to select FM0 encoding, 8x ADPLL operating clock, and full duplex communications.

If the NPU system clock frequency is a multiple of the 230.4 Kbit per second bit rate, then the baud rate generator can be used to provide the ADPLL operating clock. Note, however, if the BRG is used to generate the transmit clock, it cannot be used for the ADPLL operating clock.

If the BRG is being used for the transmit clock, or if the NPU system clock frequency is not a multiple of 230.4 KHz, then an external clock such as a TTL clock oscillator must be used.

To specify that an external clock input on the MRXC pin provides the ADPLL operating clock, program:

MRXS: 70h (0111 0000)

To specify that the BRG generates the ADPLL operating clock, with NPU operating clock of 9.216 MHz, program:

MRXS: 60h (0110 0000)
MTMC: 05h (0000 0101)

This would cause the BRG to generate a clock with frequency of 9.216/5 = 1.8432MHz (which is equal to 230.4KHz * 8).

Once the ADPLL has been initialized to handle clock extraction, it must be enabled to search for data from which to extract the clock. This search state is entered by writing the Enter Search Mode command into the MSCI command register. The state of the ADPLL can be determined by reading the Search Mode (SRCH) bit of MST3. If this bit is set, the ADPLL has not detected any data on the line. If the bit is cleared, a transition on the receive data line has been detected. The Flag Detection (FLGD) bit of MST1 is set to 1 when a Flag pattern has been detected on the receive line, which indicates another node is preparing to transmit a frame of data.

Clocking Transmit Data

The NPU's on-chip baud rate generator can be used to generate the transmit data clock if the NPU system clock is a multiple of LLAP's 230.4 KBps transfer frequency. If a 9.216 MHz system clock is used (crystal frequency of 18.432 MHz), the NPU's BRG could be programmed as follows:

MTXS: 40h (0100 0000)
MTMC: 28h (0010 1000)

which would result in a baud clock of (9.216/28h)/1 or 230.4KHz.

However, if a 10MHz system clock is used, which is not a multiple of 230.4 KHz, an external clock operating at a frequency of 230.4 KHz must be placed in the system. In this case, the MSCI should be programmed:

MTXS: 00h

to indicate that a clock attached to the TXCM input pin should be used to provide the transmit clock. The MTMC does not need to be programmed, since the baud rate generator is not being used.

Once a transmit clock has been specified, the task of encoding the clock into FM0-type data can be taken care of by the MSCI. To cause this to occur, program the following:

MMD2: 110x xxxx

with the x-bits defined for the desired ADPLL and Channel Connection operation.

Bit stuffing/zero insertion

A LLAP receiver detects the end of a frame whenever a bit sequence matching a flag byte is received (a "0" bit, followed by 6 "1" bits). To prevent valid frame data from being detected as a flag sequence, the MSCI hardware will automatically insert a "0" bit whenever 5 "1" bits have been transferred in sequence in the data stream. This capability is referred to as zero-insertion or bit stuffing. The MSCI receiver hardware performs the reverse of this function, stripping out any "0" bit that follows a sequence of 5 "1" bits.

MSCI/DMA operation in bit-synchronous mode

To minimize the load on the processor, the MSCI and DMAC can operate with data read by the receiver being stored to memory under DMA control without processor intervention. When the MSCI determines that an incoming frame is addressed to itself, the data is placed in the MTRB, which triggers the DMAC to cause this data to be transferred to memory. The CPU is notified when a full frame of data has been received by enabling an interrupt when the End of Message (EOM) condition in the DMAC's DSR0 becomes true (by setting the EOME bit to "1" in DIR0).

MSCI Initialization Summary

The following summarizes the non-ADPLL MSCI registers that must be initialized to allow the NPU to transmit and receive frames under LLAP protocol:

- MSA0: program with own node address
- MIDL: program with $7F (idle pattern)
- MMD0: program with $86 for
  - Bit-sync, HDLC mode
  - CRC Calculation Enable
  - CRC-CCITT calculation, initial values 0s

- MMD1: program with $40 for
  Single address, 1 byte long
- MMD2: program with $C0 for
  FM0-type encoding
- MCTL: program with $31 for
  TXRDY true when transmit buffer empty
  transmits FCS and flag on Underrun
  transmits flag during idle
- MCMD: programmed with commands as required:
  TX enable to enable transmissions
  RX enable to enable reception

The ADPLL registers should be initialized as described above.

This initialization prepares the MSCI to transmit and receive LLAP frames. Transmission and reception are handled by application programs executed by the NPU's CPU which control MSCI operation. These programs may take advantage of the NPU's DMA controller, which can be initialized to transfer frame data between the MSCI and memory without the need for CPU intervention.

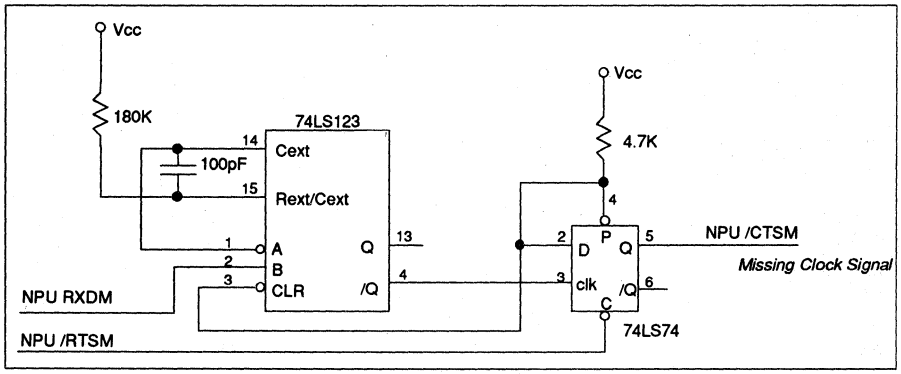


Figure 3 - Missing Clock Detect Circuit

ACCESSING THE SHARED LINK

The LLAP physical media consists of twisted-pair cable that carries both transmit and receive data between nodes. Since transmit and receive data from each node share the same transmission media, a node's transmitter must be disabled whenever the node is receiving data or listening for data on the bus. On the NPU Development Board, the RS-422 device that drives the LLAP link must be disabled along with the MSCI transmitter whenever the node is not sending data to the link.

Since LLAP is a shared-link protocol, any node on the link must not transmit data unless it has made an attempt to determine that no other nodes are transmitting. At all times that a node is not transmitting, it must be listening (receiver enabled). This would allow the node to determine that the link is free, or to determine if there is data to be received.

Checking the link for other transmitters

LLAP uses two methods to determine if the link is free. One method is by detecting that the link is not inactive, referred to as a Carrier Sense condition. The second method is by detecting a Missing Clock condition. Prior to sending an RTS frame, LLAP nodes transmit a synchonization pulse which is defined as a transition on the link followed by an idle period greater than 2-bit times. This transition is detected by all receivers as a clock, but when the idle period is occurs, receivers conclude that they have lost the clock, or detected a missing clock. This missing clock detection is a fast way of determining that a sender is using the shared link.

Carrier Sense

This condition corresponds to the Flag Detect (FLGD) status bit of the MSCI Status Register 1 (MST1). This bit is set when a flag pattern has been detected on the link and synchronization with an external transmitter has been established. It is cleared when the receiver is reset after a frame has been received.

Missing Clock Generation

A missing clock can be generated by enabling then disabling the transmit hardware line driver. A software timing loop ensures that one bit time passes between enable and disable. Receivers on the link detect this signal as an attempt to transmit by another node.

Missing Clock Detection

The detection of the Missing Clock signal is not supported by the NPU hardware. However, it can be supported using a simple external circuit. An example circuit based on a one-shot causes an indication to the MSCI that a missing clock has occurred. This signal is input on the RTS line of the MSCI (the modem control lines are not used by LLAP). Resetting the missing clock indication is done using the CTS line of the MSCI. This circuit is shown in Figure 3.

RECEIVING LLAP FRAMES

The LLAP protocol defines that all nodes are listeners (receiver circuitry of these nodes is enabled) except when a node has data to transmit and has determined that the link is free. To accomplish this, the MSCI receiver is enabled at initialization, and is only disabled when the node is transmitting data.

Following the reception of a frame, the receiver must be prepared to receive subsequent frames. An Rx Reset command should be issued to the MSCI CMD register to clear the receive buffer and status register values. The Enter Search Mode command should also be issued to cause the ADPLL to search for the next frame of data.

Once properly initialized, the MSCI can receive broadcast frames and frames whose addresses match the MSA0 value. If the frame is accepted, all bytes of data following the Destination Address byte are received and placed into the receive FIFO to be read from the MTRB (refer to LLAP frame information in Appendix A). An NPU program should be written to read these bytes from the MTRB (either directly or under DMA control), and process the frame data.

If DMA control is used to receive data, processing does not need to being until the entire frame has been received. This End Of fraMe (EOM) condition can be determined either by using the EOM status of the DMAC (bit 6 of the DSR) to cause an interrupt, or by CPU polling.

Search Mode

The receiver detects the start of the frame when a Flag byte is detected. Subsequent Flag bytes are ignored.

Address Checking

The first non-Flag byte received is taken to be the Destination Address byte. If this byte contains $FF, this frame is accepted by the NPU hardware as a broadcast frame. Otherwise, if this byte matches the value programmed in the MSA0 register, then this frame is accepted as a frame addressed to the node. In all other cases, the receiver returns to searching for a Flag byte.

DETECTING ERRORS IN RECEIVED FRAMES

LLAP specifies a number of error conditions that should be detected and acted upon in the course of receiving frames of data. These conditions include overrun, bad frame CRC, bad frame type and bad frame size (data greater than 600 bytes).

CRC checking

The MSCI receiver determines that the end of a frame has been received when a flag byte is detected in the receive data stream. When this occurs, the hardware CRC calculation is automatically compared with the CRC information received in the frame. If the values do not match, an error is detected. Error information is reflected in the CRCE bit of MST2 when the byte preceeding the CRC information (the last data byte of the frame) is read from the MTRB. LLAP routines poll for this error condition following the receipt of a frame.

Overrun

An overrun error can be automatically detected by the hardware. Detection of this error can occur by polling the OVRNF bit of the MSCI Frame Status Register (MFST). This register stores the status of the last frame received in the bit-synchronous mode, and is reset by when the receiver is reset following the receipt of a frame.

Bad frame size

LLAP defines that frames should contain 0 to 600 bytes of data. Frames larger than 600 bytes should cause a bad frame size error. The NPU can detect this condition in software by counting the number of bytes in a received frame.

Bad frame type

LLAP defines frame type field values of $00 to $80 to signify data frames, type $FF for broadcast frames, and types $81, $82, $84, and $85 for ENQ, ACK, RTS, and CTS frames, respectively. Any other value in the frame type field should generate an error. The NPU routine that handles receiving of data frames reads this type field, and returns an error indication if a non-valid value is found.

TRANSMITTING FRAMES

Generating LLAP Frames For Transmission

In order to transmit frames under LLAP protocol, the NPU must be initialized to place the information of the LLAP packet into the MTRB under program or DMA control. Once this preparation is done, the NPU's transmitter should be enabled.

Flag Preamble

LLAP defines that at least two Flag bytes preceed the destination address byte when a frame is transmitted on the link.

Once the MSCI transmitter is enabled, the MSCI will transmit the idle pattern ($7E, as programmed in the MIDL register) until transmit data is written to in the MTRB. The NPU timers or a timeout loop can be used to wait two byte times (69.4 uSeconds at 230.4Kbps) prior to placing data in the transmit buffer. This causes two Flag bytes to be output prior to the first byte of the LLAP packet.

Frame data transfer

Once this wait time has elapsed, the NPU should continue to place LLAP packet information bytes in the MTRB. Prior to sending the last data byte of the frame, an End-of-message

command should be issued to the MCMD. This informs the MSCI that the next byte transferred to the MTRB is the last character of the frame. Under Chained-block transfer DMA operation, EOM information is automatically passed to the MSCI by the DMAC.

CRC - generation
LLAP defines that an error checking byte be appended to transmitted frames to allow receivers to determine if data packets were correctly received. The EOM indicates the last byte of the frame being transmitted. Following this byte, the two generated CRC bytes are transmitted.

Flag Trailer
Following the transmission of the Frame Check Sequence (FCS), a flag byte is automatically transmitted.

Abort Trailer
LLAP protocol expects the Flag byte in the frame trailer to be followed by an abort sequence consisting of 12 to 18 one-bits. Once the transmission of a frame (including CRC bytes and trailing flag) has been completed, the NPU should disable the MSCI transmitter. The transmit drivers should be left enabled for approximately two byte-times to cause the abort sequence of 12 to 18 "1"'s to be output to the link. After this time, the transmit drivers should be disabled, which frees up the bus.

Inter-Frame and Inter-Dialog Gap Timing
The RTSframe - CTSframe - Dataframe interchange between two nodes on the network is referred to by LLAP as a dialog. LLAP defines that the maximum time lapse or gap between these frames must be less than 200 microseconds (uS). If greater than 200 uS elapses between frames, it is assumed that a collision occurred, and the transmitter defers and attempts to reinitiate the dialog at a later time.

The gap between dialogs on the link is defined to be at least 400 uS. If a transmitter detects that the line is idle for greater than this amount of time, it can assume that no dialog is occurring on the link. The node will wait some additional delay time, and if the link remians inactive, it can transmit a RTSframe in an attempt to take over the link for a dialog.

These gap times can be determined by the on-chip timers. With an operating clock of 9.216 MHz, elapsed times of 200 and 400 uS can be counted out using the timer base clock of 9.216 / 8, or 1.152 MHz, divided by 8 (resulting in a count frequency of 144 KHz). 200 uS would elapse after a count of 29, and 400 uS would elapse after a count of 58.

A transmitter waiting to access the link would perform the following steps:

1. Initialize the TCONR for the minimum IDG time plus some additional time, as determined by the LLAP algorithm
2. Initialize the TCNT to 0 and enable count
3. Poll the CMF bit of the TCSR, and poll the link
4. If activity is found on the link, wait for the link to become idle and go back to Step 1.

If count match occurs and no activity on the link is detected, disable count and continue preparations for transmitting a frame.

A transmitter waiting for a response to an RTSframe would perform the following steps:

1. Initialize the TCONR for the maximum IFG time
2. Initialize the TCNT to 0 and enable counting
3. Poll the CMF bit of the TCSR, and poll the link
4. If activity is found on the link, receive it and determine if it is the CTSframe and proceed

If count match occurs, disable count and assume a collision occurred

PROGRAMMING AND TESTING LLAP ROUTINES

Description of Routines and Testing
Inside AppleTalk contains a procedural model of LLAP written in pseudo-code. In order to demonstrate the NPU's ability to control a LLAP node, some of these routines were coded in C and assembler for the NPU, and executed using an development board designed around the NPU processor.

Hardware
The NPU Development Board is a Hitachi product which allows users to evaluate NPU software in a native environment. There is a small amount of prototype area on the board which allows users to also evaluate hardware interfaces to the NPU. The missing clock circuit mentioned above was built in this prototype area.

The board consists of an NPU, an optional ROM debug monitor, RAM and ROM space for code development, an off-chip UART for performing communications without using the on-chip channels, and driver and receiver circuitry to interface the serial channels to connectors going off-board.

RS-422 drivers/receivers
The MSCI port of the NPU can be interfaced using RS-232 or RS-422/485 signals. An on-board hardware switch selects which interface is active. Since LLAP requires RS-422 type communications, this feature is selected. The RS-422 transmit driver is controllable by writing to a hardware register in the board's I/O space. By writing to this register, an application

can turn on this driver to output information to the link, and can turn off the driver when the node is finished transmitting on the link. Since LLAP defines a shared link, only one transmitter can be active on the link at any time.

AppleTalk Cabling

The RS-422 signals that drive transmit data from and send receive data to the MSCI are carried off-board by a multi-pin header. These pins must be connected to an AppleTalk Local-Talk Connector kit. A simple way to do this is to build a cable from the pin header to a DB-9 type female connector. This DB-9 can directly interface to the 9-pin plug of a LocalTalk connector kit. See Figure 4 for the pin-out for this cable.

Software/development tools

The software for this Application Note was generated using a 64180 C Compiler and Cross Assembler. Generated code was executed using the Hitachi 64180 ASE emulator interfaced to the NPU Development Board as a target system. The code examples were executed out of zero wait state ASE emulation memory.

	LocalTalk DB-9 Connector Pin		NPU Board Header J485 Pin	
RXD+	8	———	6	+ pin RXDM receiver
TXD+	4	———	4	+ pin TXDM driver
RXD-	9	———	24	- pin RXDM receiver
TXD-	5	———	22	- pin TXDM driver

Figure 4 - LocalTalk Connections

Code execution speed requirements

At the start, all routines for implementing LLAP using the NPU were written in C. After some testing, however, it was determined that the C code generated was too slow to meet the requirement of IFG time of 200 uS. This IFG time must be met between the time a node finishes sending an RTS frame, and the time the addressed node begins sending an answering CTS frame. Because of this time requirement, the routine which handles receiving frames is modified from the specifications shown in Inside AppleTalk. In addition, portions of this routine were written in assembler to speed up execution.

These modifications removed some of the levels of subroutines by eliminating some of the code modularity. Instead of calling many subroutines that have specific functions, these functions were incorporated in "straight-line" code. This eliminated the time-consuming subroutine entry/exit code generated by the C compiler. Also, by accessing hardware registers using in-line assembly code rather than relying on C language subroutines, more execution time was cut.

DMA usage

The example routines used to implement the AppleTalk LLAP use the NPU's Chained Block Transfer mode of DMA operation. These routines use memory buffers to hold data to be transmitted and data that is received. These buffers are identified to the DMAC in a chain of descriptors (one descriptor for each buffer), which each consist of 10 bytes of information. This information includes a pointer to the memory buffer, the size of the buffer, status of the buffer, and a pointer to the next descriptor.

Transmitting frames under DMA control

In order to transmit a packet of data over the link, packet data must be placed in a LLAP frame. This frame must also be preceeded by a LLAP RTS frame in a dialog in order to establish the proper transmission protocol.

To cause this sequence to occur under DMA control, the buffer pointed to by the first transmit descriptor should hold the RTS frame information. The next descriptor in the chain should point to the buffer which contains the data to be transmitted. The application program which fills this buffer should provide the appropriate LLAP header information (destination ID, source ID and LLAP type bytes), followed by the data length (2 bytes) and the data to be transmitted (up to 600 bytes). The MSCI will automatically transmit Flags prior to the frame, and the Frame check sequence following the frame.

DMAC Channel 1 (which is internally connected to the MSCI transmitter) should be initialized for chained block transfer mode, single frame. This will cause a single frame to be transmitted each time the DMA channel is enabled. Once the DMAC channel and the descriptor chain have been initialized, the LLAP handling program would enable DMA channel 1, which causes the RTS frame to be sent. The LLAP program then polls for receipt of a CTS frame. If one is received, the

DMAC is reenabled to transmit the data frame. If the CTS frame is not received within the inter-frame gap time, a collision is assumed to have occurred, and the program prepares to resend the RTS frame and the data frame after a time interval determined by the LLAP algorithm.

Receiving frames under DMA control

For receiving data from the link, a chain of descriptors pointing to a set of buffers in memory would be initialized at system start-up. These buffers would be arranged in a circular fashion to allow them to be reused as receive data is read out of them by the application program.

In the case of receiving data, the DMAC channel 0 is initialized to operate in chained-block mode, with multi-frame operation. Once initialized in this manner, DMA channel 0 (which is internally connected to the MSCI receiver) is always prepared to receive data frames.

The number and size of receive data buffers are parameters that must be chosen carefully. If too few buffers are specified, it is possible that an overrun condition can occur with respect to the DMA receive buffers, and received data would be lost. If too many buffers are specified, system memory may need to be larger than is really required.

DMA buffer sizing

The size of buffers is also important for efficient operation. Unlike with transmission, where buffer sizes can be specified according the size of frame to be transmitted, receive frame sizes are only known once the frame has been received. If the buffers are allocated equal to the largest possible frame, some space will usually be wasted. If buffers are too small, then more time than necessary will be taken up by buffer switching, where the DMAC must go out and read the next buffer descriptor to find out where to place incoming data.

If memory space is not a problem, receive buffers can be specified to be as long as the longest possible frame. For LLAP, this would be 605 bytes. In this case, buffer switching would never have to occur in the middle of a received frame.

If a more efficient memory scheme is required, then set up the receive buffers to be the size of the average frame. Then frames less than half the size of the maximum would not waste as much space. However, frames longer than this would require a buffer switch partway through the frame to continue storing the data to a new frame.

End of frame conditions

Through use of the status byte of the descriptor, the DMAC keeps track of whether the buffer pointed to by the descriptor contains the end of a frame. For transmissions, the LLAP program should write the descriptor with a status byte indicating whether the buffer associated with the descriptor contains the end of frame. For receptions, the MSCI indicates to the DMAC if the end of frame has been received (indicated by the receipt of a Flag byte), and the DMAC then writes this information to the descriptor associated with the receive buffer that contains the end of the frame.

Initializing the DMAC

To initialize the DMA controller to be used in chained-block transfer mode with the MSCI for LLAP, the following registers should be written with the given values:

CPB (SARB): $00
locates Chain Pointer Base with 0000 as the four high order bits of the 20-bit address

CDAL0: $00 CDAL1: $00
CDAH0: $04 CDAH1: $03
locates Chain Pointer at $400 for transmit and $300 for receive initially, which are the addresses of the first descriptors in each chain

EDAL0: $40 EDAL1: $70
EDAH0: $40 EDAH1: $30
identifies the end of the Chain of descriptors

DMRA1: $98 (1001 1000) for transmit
DMRA0: $96 (1001 0110) for receive
DIR0: $40 (0100 0000) for receive
DIR1: $00 for transmit
causes interrupt to occur when end of frame received

PCR: $00
Channel 0 (receive) has priority over channel 1

To complete initialization of the DMAC, the descriptors in the descriptor chain must be initialized with chain pointer and buffer pointer addresses. For transmit descriptors, buffer size and status should also be written to the descriptors. For receive, status and size will be written by the DMAC once the buffer data has been received.

For receiving data through the MSCI, the DMAC only needs to be initialized once. Since channel 0 is initialized for multi-frame mode operation, DMA transmission over this channel never terminates. Transmission is only active, however, when the MSCI receiver receives data inputs.

For transmitting data through the MSCI, the DMAC is initial-

ized and enabled prior to the first frame transfer. Since channel 1 is initialized for single-frame mode operation, DMAC transfer terminates when the end of a frame has been transferred. DMA channel 0 must be reenabled for each frame transfer by placing the frame data in the buffer pointed to by the current descriptor, and writing $72 (to clear any flags and to enable DMA) to DSR0.

Hardware interface routines

Inside AppleTalk provides LLAP Access Control Algorithms in Appendix B, which defines a specification for AppleTalk LLAP implementations. Within these specifications are references to hardware-specific routines that must be available to LLAP procedures.

These hardware interfaces are declared as functions and procedures that interact with the hardware, in most cases by reading or writing a status bit or pattern into a hardware register. These declarations, and the NPU MSCI features used to implement the associated functions and procedures are listed in Table 1.

Transmit and Receive Routines

Routines written to transmit and receive frame data deviate from the proecedural model in that they use the NPU's chained-block transfer DMA capabilities. Instead of transmitting and receiving bytes under program control, the DMAC sends or receives a frame at a time without processor intervention. Other deviations from the model are in the area of handling timing delays (software timing loops or the on-chip timers are used to cause timing delays instead of a system RealTimer function).

Bit and Byte Timing

Since bit and byte times are relatively short compared to the resolution of the on-chip timers, timing of these shorter intervals can be handled using software delays.

For instance, to generate the 1-bit time period that the transmit driver should be enabled to generate part of the Missing Clock signal, the following code section can be used:

```
;       enable transmit for 1 byte time (flag)

        LD      A,4             ;counter for loop
        LD      B,TXDen
        OUT0    (NPUreg),B      ;enable transmit drive
        LD      B,TXen
        OUT0    (MCMD),B        ;enable transmit
TLOOP3:
        SUB     1               ; 6 states
        JR      NZ,TLOOP3       ; 6/8 states
```

This loop consists of ((3*12) + (1*14)) = 50 clock states. At 10 MHz, executing this loop would result in a 5 uSec delay, which is approximately the one bit time (4.34 uS at 230.4KBps) required.

Testing LLAP routines

Once the LLAP routines were written, they were tested for their ability to transmit and receive data over the LocalTalk Link by connecting the NPU board to a MacIntosh using a LocalTalk Connector kit. Routines provided by Apple Developers Association allow the Mac to scan the link for frames being transmitted, and to send data frames prefaced by an RTS frame. Using these routines, and those written for the NPU, frames of data can be exchanged between the Mac and the NPU Demonstration Board.

Summary

The Hitachi HD64180S Network Processing Unit's feature set is usefule in implementing AppleTalk's LocalTalk Link Access Protocol. The device's Multi-protocol Serial Communications Interface operating in bit-synchronous mode automatically performs much of the frame handling required to implement this protocol. The on-chip DMA controller assists in transferring frame data between the MSCI and memory without need for direct program control.

A procedural model for the software required to implement LLAP control programs are available in Inside AppleTalk. This procedural model is easily converted to C language, and compiled to NPU object code using cross-software tools.

Once programs are converted to C, they can be carefully examined for areas where timing constraints may cause problems, such as in the area of meeting inter-frame gap times during LLAP dialogs. These areas can be examined for execution speed, and can be recoded as straight-line code or in assembly code if necessary.

References

1. SIDHU, S.S., ANDREWS, R.F., and OPPENHEIMER, A.B.: **Inside AppleTalk**, Reading, MA: Addison Wesley, 1989.

2. **HD64180S NPU Hardware Manual**, #U16. Available from Hitachi Sales Offices, Sales Representatives, and Distributors.

NPU Implementation of LLAP Hardware Requirements	
CarrierSense	indicates that the hardware is sensing a frame on the link; FLGD (bit 4) of MST1
RcvDataAvail	indicates that a data byte is available; RXRDY (bit 0) of MST0
rxDATA	identifies the next data byte available (MTRB register)
EndOfFrame	indicates that a valid closing flag has been detected; EOM (bit 7) of MST2
CRCok	indicates that the received frame's FCS is correct (when EndOfFrame is true); complement of CRCE (bit 2) of MST2
OverRun	indicates that the code did not keep up with data reception; OVRN (bit 3) of MST2
MissingClock	indicates that the hardware has detected a missing transition on the link; /CTS (bit 3) of MST3 (due to user added circuit)
setAddress	sets the hardware to receive frames whose destination address matches MyAddress; ADDRS1, ADDRS0 (bits 7,6) of MMD1
enableTxDrivers enableRxDrivers	control the operation of the RS-422 drivers; RTS (bit 0) of MCTL
enableTx disableTx	control the operation of the data transmitter by means of issuing commands to the command register
txFLAG	causes the automatic transmission of a flag at frame opening when Tx Enable is set; however, code must delay long enough to cause the extra flag; the trailing flag is generated automatically at frame end as part of the Tx Underrun processing (relys on the NPU's transmission of flag pattern during idle state operation, IDLC bit 4 of MCTL)
txDATA	causes transmission of a data byte (MTRB register)
txFCS	causes the automatic transmission of the FCS by letting Tx Underrun occur; UDRNC (bit 5) of MCTL set to 1 causes FCS and flag transmission following underrun
txONEs	causes 12-18 one-bits (1's) to be sent by disabling the NPU transmitter while leaving the RS-422 drivers on and delaying
resetRX enableRX disableRx	control the receiver by means of the Rx Enable command (resetRX should also flush receiver FIFO)
resetMissingClock	causes the MissingClock indication to be cleared by a Reset Missing Clock command; RTS (bit 0) of CTL (ASCI port RTS line)

Table 1 - LLAP Hardware Interface Routines for NPU

Appendix A - A BRIEF LLAP OVERVIEW

LocalTalk link

LLAP uses RS-422 signalling for transmission and reception over the LocalTalk link. This method provides differential, balanced voltage signalling over a maximum of 300 meters. The link consists of twisted pair cable shared by all nodes.

Data is encoded on the link using a self-clocking technique known as FM-0. In this technique, clocking information is encoded in the transmitted data stream. Each bit cell in the serial data stream contains a transition at each end that provides timing information known as one bit-time. Zeros are encoded by adding a transition midway through the bit-time, as shown in Figure 2.

Link access

The LLAP protocol uses a CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) scheme for controlling access to the shared link. This algorithm attempts to prevent more than one transmitter from using the link at a time. The protocol's handshaking method ensures that if a collision between transmitters occurs, one or both will back off and retry at a later time.

The collision avoidance scheme requires two items of information from the system hardware - an indication that a frame has been detected as being transmitted by another node on the link, and an indication that a missing clock has been detected.

Because LLAP is a shared link protocol, transmitters of nodes on the link are disabled until it is determined that no other node is transmitting on the bus. Once a node has determined the link is free, further collision avoidance takes place in the form of a handshaking method in which a node sends a Request To Send (RTS) frame addressed to the intended receiving node. If a Clear To Send (CTS) frame is received by the sender within the Inter-Frame Gap (IFG) time, the sender assumes that the link has been made available to itself, and sends the data information. If the CTS frame is not received withing the allotted time, it is assumed that a collision occurred, and the sender defers for awhile before attempting to regain control of the bus.

The collision avoidance procedure described here can be summarized in the following steps:

1. Check if a frame transmission in occurring on the bus
   If so, delay until maximum delay reached or frame ends
   If maximum delay reached, reset receiver (assume error)
2. Reset Missing Clock signal
3. Wait for the minimum IDG time to pass or until frame transmission sensed
4. If frame transmission sensed, start over from 1.
5. Wait some additional time (determined by LLAP algorithm)
6. If frame transmission sensed or Missing Clock detected, then
   increment count of defers
   wait for some time (determined by LLAP algorithm)
   if excess defers, give up with error message
7. If no link activity detected
   send RTS frame
   if CTS frame not received, then assume collision, and start from 1.
   else send data frame

Frames

An AppleTalk Local Link Access Protocol (LLAP) frame is a High-level Data Link Control (HDLC)-type frame that consists of a LLAP packet encapsulated with a preamble and a trailer, as shown in Figure1.

LLAP defines that a frame transmission consists of a frame preamble of at least two flag bytes, followed by the LLAP packet, which is followed by a flag byte and a string of "1" bits. Flag bytes are defined by the protocol to contain the bit pattern 7Fh (0111 1110b).

The LLAP packet consists of the LLAP header and an optional data field. The LLAP header contains three bytes: the Destination (receiver) node ID, the Source (transmitter) node ID, and the LLAP frame type. Each node on the AppleTalk network is assigned an exclusive node ID. This ID is used by each node to determine if a frame on the network is intended for that node, by comparing the Destination node ID of the frame with it's own node ID. No node is assigned the ID of $FF. When a frame with Destination ID of $FF appears on the network, all nodes assume a broadcast is being made, and all nodes accept and process such a frame.

Frame types range from $00 to $FF. Values in the range $80 to $FF identify LLAP control packets. Such packets do not contain a data field. Frame types in the range of $00 to $7F are used for LLAP data packets, with the type field specifying the LLAP type of the client to whom the data should be delivered.

If the frame contains a data field, this field follows the byte containing the LLAP type field. This data information starts with a 10-bit value (in the two least significant bits of the first byte combined with the eight bits of data in the second byte of the field) indicating the size of the data in bytes, followed by the bytes of data themselves. The data field can contain up to 602 bytes (2 bytes for byte count, and 600 bytes of data).

In the frame trailer, the Frame Check Sequence (FCS) is a cyclic-redundancy check calculation based on the CRC-CCITT (Cyclic-Redundancy Check - Consultative Committee on International Telephone & Telegraph) algorithm. The flag byte contains the same pattern ($7F) used to signal the start of the frame. The abort sequence consists of the 12 to 18 one-bits (1's) which signal the end of frame.

LLAP Timing

LLAP requires that clock information be included in transmitted data, and therefore must be extracted from received data for proper reception. LLAP is defined to transmit data at 230.4 KBps.

A node's interaction with the rest of the link depends heavily on timing. LLAP specifies a maximum time gap between frames in a dialog between two nodes and a minimum time gap between dialogs. A transmitter, once it has sensed the link to be idle, must wait for the minimum IDG plus some random amount of time before attempting to send an RTS frame. The node must be capable of keeping track of this gap time. The NPU timers can be programmed to handle these timing requirements.

NPU handling of LLAP also requires keeping track of shorter time periods, such as for outputting a Missing Clock signal, for delaying to allow a flag byte to transmit prior to placing data in the MSCI transmit buffer, and for delaying to allow the abort sequence (12 - 18 one bits) to be output.

These shorter time periods can be generated using software code delays.

Appendix B - AN OVERVIEW OF THE NPU

CPU Core and MMU

The NPU is a member of Hitachi's 64180 family of high-integration microcontroller devices. These devices are all based on the same CPU core which is capable of executing object code that is upward compatible with the Z80. The family members also have access to an on-chip Memory Management Unit (MMU) which allows the devices access to a 1 MByte addressing space, a Multi-protocol Serial Interface channel and an asynchronous Serial Channel for handling communications tasks, two channels of DMA capability, and two eight-bit timers. In addition to these features, the chip also provides an interrupt controller for on and off-chip interrupts, a DRAM refresh controller, programmable chip select outputs, and low-power operation modes.

The MMU's ability to access a full 1 MByte of memory gives the CPU a large code and data space to work with, which is beneficial in supporting the LLAP protocol.

MSCI

The Multi-protocol Serial Communications Interface channel provides features which allow this device to control asynchronous, byte-synchronous and bit-synchronous communications. Many of the features required for the Open Systems Interconnect (OSI) model level 2 functionality are provided by this channel.

An on-chip baud rate generator can generate the clocks for transmitting and receiving data through this channel. A built-in Advanced Digital Phase Locked Loop circuit can be used to extact clock information from received data, or to reduce the noise component in an input receive clock or recieved data.

ADPLL

Associated with the MSCI is an on-chip Advanced Digital Phase Locked Loop (ADPLL). This feature allows the MSCI to extact clock information encoded in received data, and/or to perform noise suppression on received data or a receive clock.

DMAC

The chips DMA capabilities consist of both general purpose single and dual address DMA transfers, along with a special chained-block transfer mode that is available for use with the MSCI operating in bit-synchronous communications mode. This capability allows the MSCI to perform transmit and receive of serial data without CPU intervention.

The NPU's MSCI and DMAC allow fast transfers of bit-synchronous serial data with minimal processor intervention. The DMAC is equipped with a chained-block transfer mode which allows the user to pre-program a table in memory that is automatically read by the DMAC to determine locations and sizes of blocks of memory that are to transmitted or received. The DMAC is internally connected to the MSCI, so that MSCI status (such as receive buffer full or transmit buffer empty) triggers the DMAC operations to move data between memory and the MSCI Transmit/Receive Buffer (MTRB).

These features minimize CPU processing time required to transmit and receive data over the link. Since the data movement between the MSCI and memory is handled by DMA control, the processor can take care of other system functions.

APPENDIX C - ROUTINE DESCRIPTIONS AND CODE EXAMPLES

This appendix contains descriptions of the routines written to implement LLAP data reception on the NPU Evaluation Board, and source code listings for these routines. The source files for these routines are available in DOS file format on the Hitachi Application Engineering bulletin board. Contact the Field Application Engineer in your local Hitachi Sales Office for information on accessing the bulletin board.

The routines described below werelinked together and tested in a program which would receive dialogs from a Macintosh running the AppleTalk Pokeutility. This utility sends a dialog by sending an RTS frame addressed to the node controlled by the NPU, reading a CTS frame from this node, then sending a data frame. The Poke utility returns an error message if the dialog does not occur correctly.

transmitFrame

This routine depends heavily on the hardware features of the MSCI for transmitting a LLAP frame with the proper framing (preceeded by two flags, followed by an abort sequence). Since the NPU's MSCI is designed to send flags until data is placed in the transmit buffer, the hardware is unable to meet the LLAP requirement of two flags automatically. Instead, this must be handled using a software timing routine, which delays until two flags have been output before placing data in the transmit buffer.

This routine assumes that all information making up a LLAP frame has been placed in a buffer in memory, and the DMA controller has been initialized to access this buffer and transmit its data to the MSCI. This routine then enables the DMAC channel by writing to the enable bit of the DMAC channel control register. Once the DMA channel has transmitted all of the frame data, the MSCI automatically transmits the CRC bytes, eliminating the need for a routine (txFCS) to do this. The transmit of a trailing flag byte is also automatic, eliminating the need for the last txFLAG call in TransmitFrame.

In order to generate an abort sequence of 12 to 18 "1" bits on the data link, the MSCI's transmitter must be disabled by software, which causes the transmit data line to go high. The RS-485 transmit driver remains enabled for an interval of 12 to 18 bit times, which causes "1" bits to be sent over the link. After this interval, the transmit driver is disabled, allowing another transmitter to use the link.

Since much of this activity depends on writing to hardware registers and measuring short timing intervals that can be implemented using code loops, a section of this routine was written as an assembler routine which is called by transmitFrame.

ReceiveLinkMgmt

ReceiveLinkMgmt is actually an interrupt service routine that is executed in response to the DMAC channel 0 EOM (End Of Message) interrupt. This interrupt occurs when DMAC channel 0 has detected that the MSCI has received a full frame of data. The MSCI notifies the DMAC of this condition once it detects a flag pattern in the received data stream.

Upon entry to this routine, the MSCI receiver is reset to allow reception of the next frame, and the Enter Search Mode command is executed to allow the MSCI receiver to synchronize with the next incoming frame.

Once these steps have been taken, the routine calls receiveFrame(), which will return the frame type information. This type information is used to inform an application program is data is available to be processed.

receiveFrame

This routine is called by receiveLinkMgmt to determine the type of frame being received, and to respond accordingly. The routine could also be called by TransmitLinkMgmt in order to accept the CTSframe that would be sent by another node in response to an RTSframe sent under TransmitLinkMgmt control.

To determine if this is the case, this routine checks the variable fCTSexpected. If this variable is set non-zero, which corresponds to "true", then an RTSframe has been sent by the transmitter, and a CTSframe is expected. This routine then waits for the EndOfFrame condition (which occurs when the end of the frame has been detected), or until an inter-frame gap timeout occurs, whichever comes first.

If fCTSexpected is false, then the routine can assume that is was called by the ReceiveLinkMgmt interrupt service routine. In this case, receiveFrame then determines the type of frame that was received, and acts accordingly. If an RTSframe was received, the receiveFrame() routine takes care of sending the CTSframe reply. This section of the routine was coded in assembler to ensure that the reply is sent within the interframe gap time required by LLAP.

The node's response to an enquiry frame (ENQ) would also need to be sent within the interframe gap time. Although the sample routine shown does not provide for this, the node's handling of sending the acknowledge frame (ACKframe) could be done in the same manner as for the CTSframe.

For modularization an ease of understanding, the NPU LLAP program examples were written to follow as closely as possible to the procedural model shown in Inside AppleTalk.

However, after testing this code, it was determined that the nested levels of routines added overhead to the code that prevented the NPU from meeting the interframe gap requirements of LLAP. To solve this problem, parts of the routines were converted to straight-line code, with in-line assembly code used to eliminate the overhead added by the C compiler.

More efficient, but less understandable code for the NPU could be generated by combining the functionality of ReceiveLinkMgmt() and receiveFrame() into a single interrupt service routine. This routine could be written in C, with some in-line assembly code included to speed up critical code portions.

Additional LLAP routines

TransmitLinkMgmt was written for the NPU in C language, and is presented in the Appendix as an example, although it was not tested.

TransmitLinkMgmt implements the CSMA/CA algorithm used by AppleTalk's LLAP. Hardware status is read by this routine to determine if the node is sensing data on the link by examining the MSCI receiver status. If the link is determined to be free, the transmitFrame routine is called. This routine controls a transmit dialog which consists of sending an RTSframe, receiving a CTSframe, and then sending a Data frame.

The NPU version of this routine differs from the procedural model in that Inter Dialog Gap (IDG) timing is performed using the NPU timers rather than using a real-time clock value in the system ("RealTime"). Also, the function "Random" is simulated in the NPU system by reading the "R" counter register of the NPU (see Miscellaneous Routines on page 38). This register contains an incrementing counter that when read at random intervals, will return a random value.

Two routines, timerInit() and timeout(), were added to initialize an NPU timer, and then to poll the timer status to determine if timeout occurred.

InitializeLLAP, AcquireAddress, TransmitPacket, ReceivePacket depend mainly on lower level routines that interface with the NPU hardware. The procedural models for these routines can be translated to C and used with the lower level hardware-dependent routines.

Appendix C - NPU Initialization Routine

```
; ********************************************************
;       _NPUINIT : initialization of NPU for operation LLAP
;                        protocol communications
;               Initializes MSCI for operation
;
; ********************************************************
;
; Station Address Setting - user defined

STAADDR:  EQU      044H   ; set for network use
;
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;       npu io address assignments 10-11-88 06:23:00
;
ICR:      EQU      000H    ; INTERRUPT CONTROL REG
CBR:      EQU      001H    ; MMU COMMON BASE REG
BBR:      EQU      002H    ; MMU BANK BASE REG
CBAR:     EQU      003H    ; MMU COMMON/BANK AREA REG
OMCR:     EQU      004H    ; OPERATION MODE CTRL REG
IOCR:     EQU      005H    ; I/O CTRL REG
WCRL:     equ      0ah
WCRM:     equ      0bh
WCRH:     equ      0ch
IOWCR:    equ      0dh
INTWR:    equ      0eh
RWCR:     equ      0fh
RCR:      equ      18h
IER1:     EQU      013h     ; INTERRUPT ENABLE REG
;
MTRB:     EQU      020H    ; MSCI TX/RX BUFFER REG
MST0:     EQU      021H    ; MSCI STATUS REG 0
MST1:     EQU      022H    ; MSCI STATUS REG 1
MST2:     EQU      023H    ; MSCI STATUS REG 2
MST3:     EQU      024H    ; MSCI STATUS REG 3
MFST:     EQU      025H    ; MSCI FRAME STATUS REG
MIE0:     EQU      026H    ; MSCI INTERRUPT ENABLE REG 0
MIE1:     EQU      027H    ; MSCI INTERRUPT ENABLE REG 1
;
MIE2:     EQU      028H    ; MSCI INTERRUPT ENABLE REG 2
MFIE:     EQU      029H    ; MSCI FRAME INTERRUPT ENABLE REG
MCMD:     EQU      02AH    ; MSCI COMMAND REG
MMD0:     EQU      02BH    ; MSCI MODE REG 0
MMD1:     EQU      02CH    ; MSCI MODE REG 1
MMD2:     EQU      02DH    ; MSCI MODE REG 2
MCTL:     EQU      02EH    ; MSCI CONTROL REG
MSA0:     EQU      02FH    ; MSCI SYNCHRONOUS ADDRESS REG 0
;
MSA1:     EQU      030H    ; MSCI SYNCHRONOUS ADDRESS REG 1
MIDL:     EQU      031H    ; MSCI IDLE PATTERN REG
MTMC:     EQU      032H    ; MSCI TIME CONSTANT REG
MRXS:     EQU      033H    ; MSCI RX CLOCK SOURCE REG
MTXS:     EQU      034H    ; MSCI TX CLOCK SOURCE REG
;
DAR0L:    EQU      058H    ; DESTINATION ADDRESS REG CH 0 LOW
BAR0L:    EQU      058H    ; BUFFER ADDRESS REG CH 0 LOW
DAR0H:    EQU      059H    ; DESTINATION ADDRESS REG CH 0 HI
BAR0H:    EQU      059H    ; BUFFER ADDRESS REG CH 0 HI
DAR0B:    EQU      05AH    ; DESTINATION ADDRESS REG CH 0 BANK
BAR0B:    EQU      05AH    ; BUFFER ADDRESS REG CH 0 BANK
SAR0L:    EQU      05BH    ; SOURCE ADDRESS REG CH 0 LOW
DRWR0L:   EQU      05BH    ; DESCRIPTOR READ/WRITE REG CH 0 LOW
SAR0H:    EQU      05CH    ; SOURCE ADDRESS REG CH 0 HI
DRWR0H:   EQU      05CH    ; DESCRIPTOR READ/WRITE REG CH 0 HI
SAR0B:    EQU      05DH    ; SOURCE ADDRESS REG CH 0 BANK
CPB0:     EQU      05DH    ; CHAIN POINTER BASE CH 0
CDA0L:    EQU      05EH    ; ACCESS DESCRIPTOR ADDRESS REG CH 0 LOW
CDA0H:    EQU      05FH    ; ACCESS DESCRIPTOR ADDRESS REG CH 0 HI
;
EDA0L:    EQU      060H    ; ERROR DESCRIPTOR ADDRESS REG CH 0 LOW
EDA0H:    EQU      061H    ; ERROR DESCRIPTOR ADDRESS REG CH 0 HI
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Appendix C - NPU Initialization Routine (continued)

```
BUFL0L:  EQU     062H    ; RX BUFFER LENGTH CH 0 LOW
BUFL0H:  EQU     063H    ; RX BUFFER LENGTH CH 0 HI
BCR0L:   EQU     064H    ; BYTE COUNT REG CH 0 LOW
BCR0H:   EQU     065H    ; BYTE COUNT REG CH 0 HI
;
DSR0:    EQU     068H    ; DMA STATUS REG CH 0
DMRA0:   EQU     069H    ; DMA MODE REG A CH 0
DMRB0:   EQU     06AH    ; DMA MODE REG B CH 0
ICNT0:   EQU     06BH    ; FRAME END INTERRUPT COUNTER CH 0
DIR0:    EQU     06CH    ; DMA INTERRUPT ENABLE REG CH 0
DCR0:    EQU     06DH    ; DMA COMMAND REG CH 0
;
DAR1L:   EQU     070H    ; DESTINATION ADDRESS REG CH 1 LOW
BAR1L:   EQU     070H    ; BUFFER ADDRESS REG CH 1 LOW
DAR1H:   EQU     071H    ; DESTINATION ADDRESS REG CH 1 HI
BAR1H:   EQU     071H    ; BUFFER ADDRESS REG CH 1 HI
DAR1B:   EQU     072H    ; DESTINATION ADDRESS REG CH 1 BANK
BAR1B:   EQU     072H    ; BUFFER ADDRESS REG CH 1 BANK
SAR1L:   EQU     073H    ; SOURCE ADDRESS REG CH 1 LOW
DRWR1L:  EQU     073H    ; DESCRIPTOR READ/WRITE REG CH 1 LOW
SAR1H:   EQU     074H    ; SOURCE ADDRESS REG CH 1 HI
DRWR1H:  EQU     074H    ; DESCRIPTOR READ/WRITE REG CH 1 HI
SAR1B:   EQU     075H    ; SOURCE ADDRESS REG CH 1 BANK
CPB1:    EQU     075H    ; CHAIN POINTER BASE CH 1
CDA1L:   EQU     076H    ; ACCESS DESCRIPTOR ADDRESS REG CH 1 LOW
CDA1H:   EQU     077H    ; ACCESS DESCRIPTOR ADDRESS REG CH 1 HI
;
EDA1L:   EQU     078H    ; ERROR DESCRIPTOR ADDRESS REG CH 1 LOW
EDA1H:   EQU     079H    ; ERROR DESCRIPTOR ADDRESS REG CH 1 HI
BUFL1L:  EQU     07AH    ; RX BUFFER LENGTH CH 1 LOW
BUFL1H:  EQU     07BH    ; RX BUFFER LENGTH CH 1 HI
BCR1L:   EQU     07CH    ; BYTE COUNT REG CH 1 LOW
BCR1H:   EQU     07DH    ; BYTE COUNT REG CH 1 HI
;
DSR1:    EQU     080H    ; DMA STATUS REG CH 1
DMRA1:   EQU     081H    ; DMA MODE REG A CH 1
DMRB1:   EQU     082H    ; DMA MODE REG B CH 1
ICNT1:   EQU     083H    ; FRAME END INTERRUPT COUNTER CH 1
DIR1:    EQU     084H    ; DMA INTERRUPT ENABLE REG CH 1
DCR1:    EQU     085H    ; DMA COMMAND REG CH 1
;
;
;*********************************************************
;        ABSOLUTE SECTION INITIALIZATION
;*********************************************************
                 extern ReceiveLinkMgmt,receive
                 global rxDescript,CTSframe
                 aseg
                 org      0000h

RESET:   defw     _NPUINIT

;        locate interrupt table at IL==00 (0000h)
;        /INT1 overlaps reset vector, since not used

                 org      16h              ; DMIB0
DMIB:    defw     ReceiveLinkMgmt

;*****************************************************
;  RECEIEVE DESCRIPTOR SET UP
;*****************************************************
; RX BUFFER 1
rxDescript:
        ORG      100H
        DEFW     110H      ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 110H
        DEFW     300H      ; LOWER 16 BIT OF RX BUFFER POINTER = 300H
        DEFB     00         ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 00
        DEFB     0          ; RESERVED
        DEFW     0          ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
        DEFB     0          ; STATUS ===> TO BE WRITTEN BY DMA
```

SECTION 2

Appendix C - NPU Initialization Routine (continued)

```
        DEFB    0           ; RESERVED
; RX BUFFER 2
        ORG     110H
        DEFW    120H        ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 120H
        DEFW    320H        ; LOWER 16 BIT OF RX BUFFER POINTER = 320H
        DEFB    00          ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 00
        DEFB    0           ; RESERVED
        DEFW    0           ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
        DEFB    0           ; STATUS ===> TO BE WRITTEN BY DMA
        DEFB    0           ; RESERVED
; RX BUFFER 3
        ORG     120H
        DEFW    130H        ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 130H
        DEFW    340H        ; LOWER 16 BIT OF RX BUFFER POINTER = 340H
        DEFB    00          ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 00
        DEFB    0           ; RESERVED
        DEFW    0           ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
        DEFB    0           ; STATUS ===> TO BE WRITTEN BY DMA
        DEFB    0           ; RESERVED
; RX BUFFER 4
        ORG     130H
        DEFW    140H        ; STARTING ADDRESS OF NEXT DESCRIPTOR = 140H
        DEFW    360H        ; LOWER 16 BIT OF RX BUFFER POINTER = 360H
        DEFB    00          ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 00
        DEFB    0           ; RESERVED
        DEFW    0           ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
        DEFB    0           ; STATUS ===> TO BE WRITTEN BY DMA
        DEFB    0           ; RESERVED
; RX BUFFER 5
        ORG     140H
        DEFW    150H        ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 150H
        DEFW    380H        ; LOWER 16 BIT OF RX BUFFER POINTER = 380H
        DEFB    00          ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 00
        DEFB    0           ; RESERVED
        DEFW    0           ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
        DEFB    0           ; STATUS ===> TO BE WRITTEN BY DMA
        DEFB    0           ; RESERVED
; RX BUFFER 6
        ORG     150H
        DEFW    160H        ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 160H
        DEFW    3A0H        ; LOWER 16 BIT OF RX BUFFER POINTER = 3A0H
        DEFB    00          ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 00
        DEFB    0           ; RESERVED
        DEFW    0           ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
        DEFB    0           ; STATUS ===> TO BE WRITTEN BY DMA
        DEFB    0           ; RESERVED
; RX BUFFER 7
        ORG     160H
        DEFW    170H        ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 170H
        DEFW    3C0H        ; LOWER 16 BIT OF RX BUFFER POINTER = 3C0H
        DEFB    00          ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 00
        DEFB    0           ; RESERVED
        DEFW    0           ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
        DEFB    0           ; STATUS ===> TO BE WRITTEN BY DMA
        DEFB    0           ; RESERVED
; RX BUFFER 8
        ORG     170H
        DEFW    100H        ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 100H
        DEFW    3E0H        ; LOWER 16 BIT OF RX BUFFER POINTER = 3E0H
        DEFB    00          ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 00
        DEFB    0           ; RESERVED
        DEFW    0           ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
        DEFB    0           ; STATUS ===> TO BE WRITTEN BY DMA
        DEFB    0           ; RESERVED
;
;*****************************************************
;  CTSframe initialization
;*****************************************************
CTSframe:
                DEFB            0               ;reserved for dest addr
                DEFB            STAADDR ;source addr is self
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Appendix C - NPU Initialization Routine (continued)

```
                DEFB            85h             ;CTSframe type
;
;*****************************************************
;  TRANSMIT DESCRIPTOR SET UP
;*****************************************************
; TX BUFFER 1

        ORG     4000H

TXDESCR::
        DEFW    4000H       ; STARTING ADDRRESS OF NEXT
TXDESCBUF::
        DEFW    0000H       ; LOWER 16 BIT OF TX BUFFER POINTER = TBD
        DEFB     00          ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 00
        DEFB     0           ; RESERVED
TXDESCT::
        DEFW     0           ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN TX Prog
        DEFB     81H         ; STATUS EOM,EOT INCLUDED
        DEFB     0           ; RESERVED

;*****************************************************
;        MSCI initialization
;*****************************************************

                cseg
_NPUINIT::

;   DISABLE WAIT STATE/REFRESH

        LD      A,00h
        OUT0    (WCRL),A
        OUT0    (WCRM),A
        OUT0    (WCRH),A
        OUT0    (IOWCR),A
        OUT0    (INTWR),A
        OUT0    (RWCR),A
        OUT0    (RCR),A
        OUT0    (0F4h),A        ;disable transmit driver
;
;  TRANSMITTER AND RECEIEVER SET UP
;
        LD   A,21H
        OUT0 (MCMD),A    ; CHANNEL RESET
        LD   A,87H
        OUT0 (MMD0),A    ; BIT-SYNC HDLC,AUTO ENABLE=0,
                                        ; CRC-CCITT=1 INITIALLY
        LD   A,40H
        OUT0 (MMD1),A    ; Single ADDRESS CHECKED
        LD   A,0C0H
        OUT0 (MMD2),A    ; FULL DUPLEX, FM0 CODE,
                                        ; x8 ADPLL CLK
        LD   A,91H
        OUT0 (MCTL),A    ; TxRDY ON NOT FULL, -RTSM=HIGH
                         ; ABORT ON IDLE & UNDERRUN
        LD   A,70H
        OUT0 (MRXS),A    ; RXCM FROM BRG (ADPLL OP. CLK)
        LD   A,00H
        OUT0 (MTXS),A    ; TXCM FROM ?
        LD   A,05H
        OUT0 (MTMC),A    ; SET TMC=5  -> BAUD RATE
                                        ; for appletalk testing

        LD   A,000H
        OUT0 (MIE0),A    ; TXINT AND RXINT DISABLED
;       LD   A,83H
        OUT0 (MIE1),A    ; UNDERRUN,ABORT,IDLE DETECTION
                                        ; INTERRUPT disABLED
        OUT0 (MFIE),A       ;
        LD   A,STAADDR
        OUT0 (MSA0),A    ; SET SECONDARY STATION ADDRESS
```

SECTION 2

Appendix C - NPU Initialization Routine (continued)

```
        LD   A,7EH
        OUT0 (MIDL),A     ; SET FLAG PATTERN = 01111110 AS
                                          ; IDLE PATTERN

;*****************************************************
;  DMA CHANNEL 0 SET UP    (RECEIVER)
;*****************************************************
        LD    A,96H                ; DMA MODE REGISTER A
        OUT0  (DMRA0),A            ; MSCI,CBSA,
                                   ; MSCI->MEMORY,MULTI FRAME
        LD    A,0
        OUT0  (CPB0),A             ; 4 HIGHER BIT OF THE
                                   ; 20-BIT DESCRIPTOR ADDR

        LD    A,70H                ;
        OUT0  (EDA0L),A            ; 170H
        LD    A,01H                ; STARTING ADDR (LOW-ORDER
                                   ; 16 BITS) OF THE
        OUT0  (EDA0H),A            ; RX DESCRIPTOR 8

        LD    A,00H                ;
        OUT0  (CDA0L),A            ; 100H
        LD    A,01H                ; STARTING ADDR (LOW-ORDER
                                   ; 16 BITS) OF THE
        OUT0  (CDA0H),A            ; FIRST RX DESCRIPTOR

        LD    A,5dh
        OUT0  (BUFL0L),A
        LD    A,02h
        OUT0  (BUFL0H),A           ; ALLOWING 600 BYTES IN
                                   ; EACH RX BUFFER
        LD    A,40H
        OUT0  (DIR0),A             ; EOM INTERRUPT ENABLED

;*****************************************************
;  DMA CHANNEL 1 SET UP    (TRANSMITTER)
;*****************************************************
        LD    A,098H               ; DMA MODE REGISTER A
        OUT0  (DMRA1),A            ; MSCI,CHAINED,
                                   ; MEMORY->MSCI,SINGLE FRAME
        LD    A,0
        OUT0  (CPB1),A             ; 4 HIGHER BIT OF THE
                                   ; 20-BIT DESCRIPTOR ADDR

        LD    A,20H                ;
        OUT0  (EDA1L),A            ; 4020H
        LD    A,40H                ; STARTING ADDR (LOW-ORDER
                                   ; 16 BITS) OF THE
        OUT0  (EDA1H),A            ; DESCRIPTOR NEXT TO THE
                                   ; LAST TX BUFFER

        LD    A,00H                ;
        OUT0  (CDA1L),A            ; 4000H
        LD    A,40H                ; STARTING ADDR (LOW-ORDER
                                   ; 16 BITS) OF THE
        OUT0  (CDA1H),A            ; FIRST DESCRIPTOR OF THE
                                   ; FIRST TX BUFFER

        LD    A,00H
        OUT0  (DIR1),A             ; EOT INTERRUPT DisABLED
;
;       Enable DMIB0 interrupts
;
        LD      A,08h              ;DMIB0
        OUT0    (IER1),A

        EI                         ;enable flags

        JP      receive            ; receive frames
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Appendix C - receive routine

```
/*************************************************************/
/* This file processes the packet data received using        */
/* ReceivePacket                                             */
/* Calls:outport() - compiler routine to output to an I/O port */
/* Enables DMA controller for receiving frames               */
/* Manages circular receive descriptors                      */
/*************************************************************/

#include "ltdefs.h"

char octet, anAddress, aLAPtype;
char aDataField[maxFrameSize];
char MyAddress, fCTSexpected, fAdrValid, fAdrInUse;
char memory[605];
int rcvStatus, i;
int outgoingLength, incomingLength;

char dstParam,srcParam,typeParam,dataParam;
int dataLength;
struct structFrame ACKframe,*incomingPacket;
struct Descriptor *rxDescrPtr; /* global storage address of current descriptor */

receive()

{  char dstParam,srcParam,typeParam,dataParam;
   int dataLength,descrNumber;
   char *dataFrame;
   char EDAvalLo,EDAvalHi;
   unsigned int EDAaddr;

/* simulate that address has been checked out on network */
   fAdrValid = TRUE;

/* receiver does not expect CTS until RTS is received */
   fCTSexpected = FALSE;

/* for this example, a node address is selected
   in actual cases, network inquiries would be used
   to select an check for an unused node number          */

   MyAddress = 0x44;

   /* enable DMA ch 0 operation for receiving frames */
   outport(DSR0,Enable);
   /* enable MSCI receiver */
   outport(MCMD,RXenable);
   /* place receiver in Search Mode */
   outport(MCMD,EnterSearch);
   descrNumber = 0;
   /* zero out rcvStatus counter */
   rcvStatus = 0;
```

Appendix C - receive routine (continued)

```
  do
  {     rxDescrPtr = &rxDescript[descrNumber];
        while (rcvStatus == 0);
        dataFrame = (*rxDescrPtr).BufferPtr;
        dataFrame = dataFrame+5; /* skip address info */
        incomingLength = ((*rxDescrPtr).DataLength - 5);
        /* move frame data to memory location accessed by
                application */
        if(incomingLength > 3)
        {       for(i=0;i<incomingLength;i++)
                {       memory[i] = *dataFrame++;
                        txasci(memory[i]);
                }
        }
        descrNumber++;
        if(descrNumber == 8)
        {       descrNumber = 0;
                EDAaddr = &rxDescript[7];
                EDAvalLo = EDAaddr; /* recast pointer */
                EDAvalHi = EDAaddr >> 8;
                outport(EDAL,EDAvalLo);
                outport(EDAH,EDAvalHi);
        }

        /* check to see if rx buffers can be reused */

        else if (descrNumber == 1) /* circular buffer */
        {       EDAaddr = &rxDescript[0];
                EDAvalLo = EDAaddr; /* recast pointer */
                EDAvalHi = EDAaddr >> 8;
                outport(EDAL,EDAvalLo);
                outport(EDAH,EDAvalHi);
        }

    /* rcvStatus is incremented by receiveLinkMgmt() and
    receiveFrame() when a data frame is received */
        rcvStatus = rcvStatus - 1;

  } while (1>0);
}

txasci(item)        /* transmit a character to the ASCI port */
char item;

{  int status;

   while((status=IBIT(ST1,6)) == 0);
   outport(TRB,item);
}
```

Appendix C - LLAP Definitions

```
/*********************************************/
/*      ltdefs.h

        This file contains definitions and declarations
   that are used by all LocalTalk C routines          */
/*********************************************/

#define minFrameSize    3
#define maxFrameSize    605
#define maxDataSize     600
#define bitTime         4.34
#define byteTime        39.0
#define minIDGtime      400.0
#define IDGslottime     100.0
#define maxIFGtime      200.0
#define maxDefers       32
#define maxCollsns      32
#define lapENQ          0x81
#define lapACK          0x82
#define lapRTS          0x84
#define lapCTS          0x85
#define hdlcFLAG        0x7E
#define wksTries        20
#define MTRB            0x0020
#define MST1            0x0022
#define MST0            0x0021
#define MST2            0x0023
#define MCMD            0x002A
#define MCTL            0x002E
#define MSA0            0x002F
#define RXRDYmask       0x0001
#define SYNCDmask       0x0010
#define CRCmask         0x0004
#define OVRNmask        0x0008
#define EOMmask         0x0080
#define RXreset         0x0011
#define RXenable        0x0012
#define RXdisable       0x0013
#define TXenable        0x0002
#define EnterSearch     0x0031
#define EDAL            0x60
#define EDAH            0x61
#define CDAL            0x5E
#define CDAH            0x5F
#define DSR0            0x68
#define DIR0            0x6C
#define DSR1            0x80
#define IER1            0x13
#define DMACenable      0x02
#define receiveOK       0x0F0
#define Receiving       0x0E0
#define nullReceive     0x0E1
#define EOFrame         0x40
#define DMIB0E          0x08
#define Enable          0x02
#define EOMreset        0x41
#define NPUreg          0xF4
#define TRUE            0xFF
#define FALSE           0x00
#define OverrunError    0x20
#define badframeSize    0x21
#define badframeCRC     0x22
#define noFrame         0x23
#define lapRTSframe     0x24
#define lapCTSframe     0x25
#define lapACKframe     0x26
#define lapDATAframe    0x27
#define lapENQframe     0x28
#define badframeType    0x29
#define UnderrunError   0x2A
#define frameError      0x2B

#define transmitOK      0x30
#define excessDefers    0x31
#define excessCollsns   0x32
#define dupAddress      0x33

struct structFrame {
   char destAddr;
   char  srcAddr;
   char lapType;
   char dataField[maxFrameSize];
   };

struct rawFrame {
   char rawdataField[maxFrameSize];
   };

struct Descriptor {
   unsigned int ChainPtr;
   long int BufferPtr;
   unsigned int DataLength;
   char status;
   char reserved;
   int spacer1;          /* reserve 6 bytes to place */
   int spacer2;          /* descriptors on even boundary */
   int spacer3;
   };
```

SECTION 2

Appendix C - LLAP Definitions (continued)

```
extern char octet, anAddress, aLAPtype;
extern char aDataField[maxFrameSize];
extern char MyAddress, fCTSexpected, fAdrValid, fAdrInUse;
extern int Backoff, deferCount, collsnCount,i;
extern char deferHistory,collsnHistory;
extern int outgoingLength, incomingLength;
extern struct structFrame outgoingPacket, *incomingPacket;
extern struct structFrame ACKframe,CTSframe;
extern struct Descriptor rxDescript[8],*rxDescrPtr;
extern char memory[605];

extern char dstParam,srcParam,typeParam,dataParam;
extern int dataLength;
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Appendix C - ReceiveLinkMgmt Routine

```
/*************************************************************/
/*                                                           */
/* ReceiveLinkMgmt function: called by EOMF Interrupt        */
/*                                                           */
/* This routine processes each frame as it is received       */
/* Routine calls ReceiveFrame                                */
/* Sets global rcvStatus when a dataframe is received        */
/*                                                           */
/*************************************************************/

#include "ltdefs.h"

ReceiveLinkMgmt()

{  char status;
   extern int rcvStatus;

#entry
; assembly entry code
   push af
   push hl
   push de
   push bc
   push ix
   push iy
   exx
   ex    af,af'
   push af
   push hl
   push de
   push bc
#endasm

   /* when interrupt is received, full frame has been
     received, so reset MSCI receiver and Enter Search
      Mode to prepare to receive next frame */

   resetRx();
   EnterSearchMode();
+
   status = Receiving;
   while (status == Receiving)
   {     switch(receiveFrame())
         {        case badframeCRC:
                  case badframeType:
                  case UnderrunError:
                  case OverrunError:        status = frameError;
                  break;
```

Appendix C - ReceiveLinkMgmt Routine (continued)

```
                case lapENQframe:       {
                        if(fAdrValid)
                        {       ACKframe.destAddr=(*incomingPacket).srcAddr;
                                ACKframe.srcAddr=MyAddress;
                                ACKframe.lapType=lapENQ;
                                transmitFrame(&ACKframe,3);
                                status = nullReceive;
                        }
                        else
                        {       fAdrInUse = TRUE;
                                status = nullReceive;
                        }
                }
                break;

                case lapRTSframe:       {
                        if (!fAdrValid)
                        {       CTSframe.destAddr=(*incomingPacket).srcAddr;
                                CTSframe.srcAddr=MyAddress;
                                CTSframe.lapType=lapCTS;
                                transmitFrame(&CTSframe,3);
                        }
                        else  */
                        {       fAdrInUse = TRUE;
                                status = nullReceive;
                        }
                        else
                                status = nullReceive;
                }
                break;

                case lapDATAframe:      {
                        if (fAdrValid)
                        {       status = receiveOK;
                        }
                        else
                        {       fAdrInUse = TRUE;
                                status = nullReceive;
                        }
                }
                break;

                case noFrame:   status = nullReceive;
        }

    }
    rcvStatus = rcvStatus + 1;
    outport(DSR0,EOMreset);  /* reset EOM interrupt */
    return;
#asm
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Appendix C - ReceiveLinkMgmt Routine (continued)

```
aexit.:  pop   bc
         pop   de
   pop   hl
   pop   af
   ex    af,af'
   exx
   pop   iy
   pop   ix
   pop   bc
   pop   de
   pop   hl
   pop   af
   ei
   reti
```

Appendix C - ReceiveFrame Routine

```
/*************************************************************/
/*                                                           */
/* receiveFrame()                                            */
/* This routine receives a LLAP frame                        */
/* In-line assembly section added to speed up response       */
/* to RTS frames - must respond within inter-frame gap       */
/* time                                                      */
/*                                                           */
/* Calls:timerInit(), timeout(), CarrierSense(),             */
/*                EndOfFrame()
/*                ei(),di() - library routines to enable and
/*                            disable interrupts
/*                inport() - routine to read a byte from an
/*                           i/o port address
/*                                                           */

#include "ltdefs.h"
#define SLOCLK          0x12

char receiveFrame()

{ char error;
  char RcvFrame;
  struct structFrame *structPacket; /* struct. access */
  struct Descriptor *rxDescrPtr;    /*pointer to Descr*/
  char temp1,temp2;

/* set up packet pointers to point to same memory area */

  temp1 = inport(CDAL);
  temp2 = inport(CDAH);
  rxDescrPtr = (temp2 << 8) + temp1;
  incomingPacket=structPacket= (*rxDescrPtr).BufferPtr;
  error = FALSE;

/* Check if called by interrupt */

  if(fCTSexpected) /* waiting for txframe response */
  {     di();     /* disable receive interrupts */
        timerInit(maxIFGtime,SLOCLK);
        while(!timeout() && !CarrierSense());
        if(!CarrierSense()) {
                ei();
                return(noFrame);
        }
        else while(!EndOfFrame());
                /* check for frame receipt */
  }
```

Appendix C - ReceiveFrame Routine (continued)

```
/* Check on validity of the frame */

if (error == FALSE)
{  if (fAdrValid)
   {     if ((temp1=(*structPacket).lapType) >= 0x80)
         {        switch(temp1)
                  {        case lapRTS :
                           CTSframe.destAddr=(*structPacket).srcAddr;

/*********************************************************/
/*                                                       */
/*    TFCODE.asm                                         */
/*                                                       */
/*    Code segment to transfer a frame
/*       (for CTS response to RTS)                          */
/*    In-line assembly code                                 */
/*********************************************************/

#asm
mstat0:  equ      021H    ; MSCI Status Reg
mcomd:   EQU      02AH     ; MSCI COMMAND REG
;
Dstat1:  EQU      080H    ; DMA STATUS REG CH 1

txdrv:   EQU      0F4H    ; NPU board control register
TXDen:   EQU      018H    ; enables tx drivers on npu bd
TXDdis:  EQU      008H     ; disables tx drivers on npu bd
TXen:    EQU      002H     ; enables tx of MSCI
DMACen:  EQU      002H    ; enables DMAC channel

      extern      TXDESCBUF,TXDESCT

; assembly entry code
  push      af
  push      bc
  push      hl

  ld        a,12h
  out0      (2Ah),a           ;disable receiver
  ld        hl,CTSframe       ;load address of structure (ptr)
  ld        (TXDESCBUF),hl    ;place this in tx descriptor
  ld        a,03h             ;load count of bytes to transfer
  ld        (TXDESCT),a       ;place this in tx descriptor

;     enable transmit drivers for 1.5 bit times
tftime:

      LD    A,5               ;prepare to count 1.5 bit times
      LD    B,TXDen
      OUT0  (txdrv),B         ;enable transmit drivers
```

Appendix C - ReceiveFrame Routine (continued)

```
TLOOP1:     SUB   1                     ;decrement A for timing
            JR    NZ,TLOOP1             ;loop until timeout
;
;     disable transmit drivers for 1.5 bit times
            LD    A,5
            LD    B,TXDdis
            OUT0  (txdrv),B             ;disable transmit drive
TLOOP2:     SUB   1                     ;decrement A for timing
            JR    NZ,TLOOP2             ;loop until timeout
;
;     enable transmit for 1 byte time (flag)

            LD    A,28
            LD    B,TXDen
            OUT0  (txdrv),B             ;enable transmit drive
            LD    B,TXen
            OUT0  (mcomd),B             ;enable transmit
TLOOP3:     SUB   1
            JR    NZ,TLOOP3             ;loop until timeout

;     enable DMAC to load TX buffer

            LD    B,DMACen
            OUT0  (Dstat1),B            ;enable DMAC channel 1
;
ENDLP:      IN0   A,(mstat0)
            AND   02h
            JR    Z,ENDLP               ;wait for end of frame

; wait for all chars and CRC to transmit

            LD    A,0A0h                ;prepare for abort string
TLOOP4:     SUB   1                     ;12 bit times
            JR    NZ,TLOOP4             ;loop until timeout

;    disable transmitter
            LD    B,03h
            OUT0  (mcomd),B

; wait for NULLs to transmit as Abort sequence

            LD    A,43h
TLOOP5:     SUB   1                     ;12 bit times
            JR    NZ,TLOOP5             ;loop until timeout

; disable transmit drivers

            LD    B,TXDdis
            OUT0  (txdrv),B             ;disable transmit drive
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Appendix C - ReceiveFrame Routine (continued)

```
; re enable receiver, and enter search mode

        LD      B,12h
        out0    (2ah),b
        LD      B,31h
        out0    (2ah),b

        pop     hl
        pop     bc
        pop     af

#endasm
                        RcvFrame = lapRTSframe;
                                        break;
                        case lapENQ : RcvFrame = lapENQframe;
                                        break;
                        case lapACK :   /* Note ACK portion should be handled as RTS is to meet IFG requirements */
                                        RcvFrame = lapACKframe;
                                        fAdrInUse = TRUE;
                                        break;
                                                        case lapCTS :
                                        if(fCTSexpected)
                                                RcvFrame=lapCTSframe;
                                        else
                                        {       fAdrInUse = TRUE;
                                                RcvFrame = badframeType;
                                        }
                                        break;
                        default : RcvFrame = badframeType;
                }
        }
        else
        RcvFrame = lapDATAframe;
    }
    else if ((*structPacket).srcAddr != 0xFF)
        {       fAdrInUse = TRUE;
                RcvFrame = noFrame;
        }
}
else RcvFrame = noFrame;
return(RcvFrame);
}
```

Appendix C - TransmitFrame Routine (continued)

```
/*********************************************************/
/*                                                       */
/* transmitFrame - transmits a single LLAP frame         */
/*                                                       */
/* Based on procedural model in Inside AppleTalk         */
/* Calls: tftime(), disableRx(),ResetMissingClock()      */
/*                                                       */
/*********************************************************/

#include "ltdefs.h"

extern unsigned int TXDESCR;    /* address of tx descriptors*/
extern tftime(),disableRx(),ResetMissingClock();

transmitFrame(strucptr,framesize)

struct structFrame *strucptr;
int framesize;

{ struct Descriptor txDescript,*txDescrPtr;
  int mode;

  /* Disable Receiving, since link is shared */
  disableRx();

  /* Initialize TX Descriptor */
  txDescrPtr = &TXDESCR;
  (*txDescrPtr).BufferPtr = strucptr;
  (*txDescrPtr).DataLength = framesize;

  tftime();
  /* generate synchronizing pulse */
  /* Start frame transmission */
  /* allow one flag byte to transmit */
  /* enable DMAC to start transfer of data */

  ResetMissingClock();
  enableRx();
}
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Appendix C - Timing related routines for transmitting frames

```
;  ******************************************************
;
; tftime - transmit frame timing related routines
;
; called by TransmitFrame
;
; generates missing clock signal by enable and disable
;     of transmit driver
; enables transmitter for one byte time to allow two
;     flags to transmit
; enables DMAC channel 1 to load MSCI with transmit data
; reprograms MSCI to output MARK in idle state
; waits for end of transmission
; at end of frame, allows approx. 12 bit times of mark
;     for LLAP Abort sequence requirement
;
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;        npu io address assignments 10-11-88 06:23:00
;
MCMD:    EQU     02AH    ; MSCI COMMAND REG
MCTL:    EQU     02EH    ; MSCI CONTROL REG
MIDL:    EQU     031H    ; MSCI IDLE PATTERN REG
MST2:    EQU     023H   ; MSCI Status Register 2
;
DSR1:    EQU     080H    ; DMA STATUS REG CH 1
DMRA1:   EQU     081H    ; DMA MODE REG A CH 1
DMRB1:   EQU     082H    ; DMA MODE REG B CH 1
DIR1:    EQU     084H    ; DMA INTERRUPT ENABLE REG CH 1
DCR1:    EQU     085H    ; DMA COMMAND REG CH 1
;
NPUreg:  EQU     0F4H   ; NPU board control register
TXDen:   EQU     018H   ; enables tx drivers on npu bd
TXDdis:  EQU     008H    ; disables tx drivers on npu bd
TXen:    EQU     002H    ; enables tx of MSCI
DMACen:  EQU     002H   ; enables DMAC channel

;*****************************************************
      public tftime
      cseg

;     enable transmit drivers for 1.5 bit times
tftime:

            LD    A,5          ;prepare to count 1.5 bit times
            LD    B,TXDen
            OUT0  (NPUreg),B   ;enable transmit drivers
TLOOP1:     SUB   1            ;decrement A for timing
            JR    NZ,TLOOP1    ;loop until timeout
;
```

Appendix C - Timing related routines for transmitting frames (continued)

```
;     disable transmit drivers for 1.5 bit times
            LD    A,5
            LD    B,TXDdis
            OUT0  (NPUreg),B        ;disable transmit drive
TLOOP2:     SUB   1                 ;decrement A for timing
            JR    NZ,TLOOP2         ;loop until timeout
;
;     enable transmit for 1 byte time (flag)

            LD    A,28
            LD    B,TXDen
            OUT0  (NPUreg),B        ;enable transmit drive
            LD    B,TXen
            OUT0  (MCMD),B          ;enable transmit
TLOOP3:     SUB   1
            JR    NZ,TLOOP3         ;loop until timeout

;     enable DMAC to load TX buffer

            LD    B,DMACen
            OUT0  (DSR1),B          ;enable DMAC channel 1
;
ENDLP:      IN0   A,(MST2)
            AND   02h
            JR    Z,ENDLP           ;wait for end of frame

; wait for all chars and CRC to transmit

            LD    A,0A0h            ;prepare for abort string
TLOOP4:     SUB   1                 ;12 bit times
            JR    NZ,TLOOP4         ;loop until timeout

;     disable transmitter
            LD    B,03h
            OUT0  (MCMD),B

; wait for NULLs to transmit as Abort sequence

            LD    A,43h
TLOOP5:     SUB   1                 ;12 bit times
            JR    NZ,TLOOP5         ;loop until timeout

; disable transmit drivers

            LD    B,TXDdis
            OUT0  (NPUreg),B        ;disable transmit drive

; return to transmitFrame routine

            RET
```

Appendix C - NPU Hardware Interface Routines

```
/*********************************************************/
/*                                                       */
/* This file contains global variable declarations       */
/* and NPU hardware interface routines                   */
/* Filename: NPUhwrt.c                                   */
/*                                                       */
/*********************************************************/

#include "ltdefs.h"

int CarrierSense()
/* Reads MST1 and returns FLGD bit: (p. 145)
   1: Flag detected
   0: No flag detected                    */
{
   int MSCIStatReg1, status;

   MSCIStatReg1 = inport(MST1);
   status = (MSCIStatReg1 & SYNCDmask);
   return(status);
}

int RcvDataAvail()        /* Also referred to as RxCharAvail */
/* Reads MST0 and returns RxRDY bit:  (p. 141)
   1: data in rx buffer
   0: no data in rx buffer            */
{
   int MSCIStatReg0, status;

   MSCIStatReg0 = inport(MST0);
   status = (MSCIStatReg0 & RXRDYmask);
   return(status);
}

int OverRun()
/* Reads MST2 register and returns OVRN bit:       (p. 150)
   1: Overrun error detected
   0: No overrun error detected                    */
{
   int MSCIStatReg2, status;

   MSCIStatReg2 = inport(MST2);
   status = (MSCIStatReg2 & OVRNmask);
   return(status);
}
```

Appendix C - NPU Hardware Interface Routines (continued)

```
int EndOfFrame()
/* Reads MST2 register and returns EOM bit:        (p. 150)
   1: End of receive frame detected
   0: Receive frame end not detected          */
{
   int MSCIStatReg2, status;

   MSCIStatReg2 = inport(MST2);
   status = (MSCIStatReg2 & EOMmask);
   return(status);
}

int CRCok()
/* Reads MST2 register and checks CRC bit:         (p. 150)
   1: CRC error detected, return 0
   0: no CRC error detected, return 1              */
{
   int MSCIStatReg2, status;

   MSCIStatReg2 = inport(MST2);
   status = (MSCIStatReg2 & CRCmask);
   if (status)
         return(FALSE);
   else return(TRUE);
}

void resetRx()
/* Resets and reenables receive by issuing commands
  to the MCMD register  (p. 137)                   */

{
   outport(MCMD,RXreset);
   outport(MCMD,RXenable);
}
void enableRx()
/* reenables receive by issuing commands
  to the MCMD register  (p. 137)                   */

{
   outport(MCMD,RXenable);
}

void disableRx()
/* Disables receive by issuing commands to the MCMD reg. */
{
   outport(MCMD,RXdisable);
}
```

Appendix C - NPU Hardware Interface Routines (continued)

```
void EnterSearchMode()
/* Issues Enter Search Mode command to the MCMD register */

{  outport(MCMD,EnterSearch);
}

char rxDATA()

/*returns a character of data from the MSCI receive buffer*/

{  char data;
   data = inport(MTRB);
   return(data);
}

void enableTx()
/* enables transmit by issuing commands to the MCMD register        (p. 137)                    */

{
   outport(MCMD,TXenable);
}

void enableTxDrivers()
/* enables Tx drivers by writing to hardware register on NPU
   board (p. 18 of Dev Board UM)  */

{  char data;
   data = inport(NPUreg);
   data = data | 0x10;
   outport(NPUreg,data);
}
void disableTxDrivers()
/* disables Tx drivers by writing to hardware register on
 NPU    board (p. 18 of Dev Board UM)  */

{  char data;
   data = inport(NPUreg);
   data = data & 0x0EF;
   outport(NPUreg,data);
}
```

Appendix C - NPU Hardware Interface Routines (continued)

```
char endofTX()
/* polls DMA ch 1 status register 1 (DSR1) to check for End
   of Frame condition  (p. 576)  */

{  char data;

   data = inport(DSR1);
   data = data & 0x40;
   return(data);
}

void enableDMAC1()
/* enables DMAC to MSCI transfer for transmit data */

{  outport(DSR1,DMACenable);
}
```

Appendix C - Miscellaneous Routines

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

/* Miscellaneous Functions */

#define FALSE        0x00
#define CMF          7
#define TCSR0        0x52
#define TCONR0       0x51
#define TCNT0        0x50
#define TMRenable    0x12
#define IDGslottime  100.0
#define MST3         0x24
#define MCTL         0x2e

int rval;

int bitCount(bitVector)
int bitVector;

{ int   sum;

   sum = BIT(&bitVector,0);          /* BIT is a function which returns a bit value(specified by 2nd parameter) */
   sum = sum + BIT(&bitVector,1);    /*      from a bit vector (address specified as 1st parameter */
   sum = sum + BIT(&bitVector,2);
   sum = sum + BIT(&bitVector,3);
   sum = sum + BIT(&bitVector,4);
   sum = sum + BIT(&bitVector,5);
   sum = sum + BIT(&bitVector,6);
   sum = sum + BIT(&bitVector,7);
   return(sum);
}

int min(val1, val2)        /* returns the minimum of two values */
int val1,val2;

{ if (val1 < val2)
        return(val1);
   else return(val2);
}

int max(val1,val2)         /* returns the maximum of two values */
nt val1,val2;

{ if (val1 > val2)
        return(val1);
   else return(val2);
}
```

Appendix C - Miscellaneous Routines (continued)

```
float random (maxval)           /* returns a simulated random value based on the value in the NPU R register */
int maxval;

{  extern int rval;
   float randval,floatslot;

#asm
   ld    a,r
   ld    (rval),a
#endasm
   randval=maxval*((float)rval/(float)127);
   randval=(randval*IDGslottime)/776;

return (randval);
}

void timerInit(seed,enable)     /* initializes the NPU timer channel 0 */
char seed,enable;

{  outport(TCONR0,seed);
   outport(TCSR0,enable);
}

char timeout()                  /* polls the NPU timer channel 0 for timeout condition */
{  int status;                  /* disables timer when timeout detected */

   status = inport(TCSR0);
   status = BIT(&status,CMF);
   if(status)
   {     inport(TCNT0); /* clear CMF */
         outport(TCSR0,0x00);
   }     /* disable counter */
   return(status);
}

void ResetMissingClock()        /* generate negative pulse on /RTSM output line */
                                /* causes reset of external missing clock circuitry */
{  outport(MCTL,0x80);
   outport(MCTL,0x81);
}

char MissingClock()             /* reads state of output of missing clock circuitry */
/* if /CTSM high, then detected */
{
   return(IBIT(MST3,0x03));
}
```

HD64180S (NPU)

Application Note

Chained Block Transfer DMA

Marnie Mar, Jun Tsong, Tom Yu

Introduction

Hitachi's HD64180S Network Processing Unit (NPU) combines the 64180 8-bit CPU core with a set of on-chip peripherals which provide the user with high-integration communications control capability. The on-chip peripherals include a Multi-protocol Serial Communications Interface (MSCI) which can support Asynchronous, Byte Synchronous and Bit Synchronous communications protocols. To assist in handling data transmitted and received by the MSCI, the NPU also includes a two-channel Direct Memory Access Controller (DMAC).

The DMA capability provided on-chip includes standard DMA functions such as single and dual address transfers of data using external or auto (program generated) requests. In addition, when using Bit Synchronous protocols such as SDLC with the MSCI, the DMAC is capable of transmitting

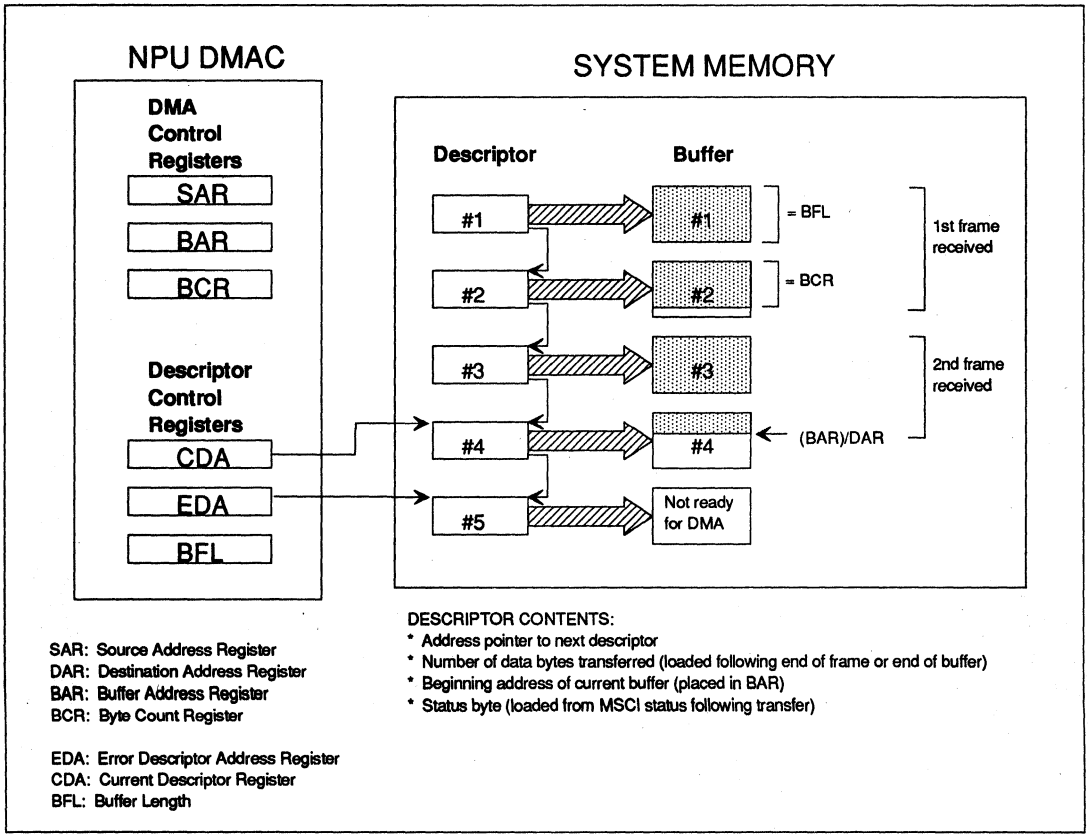


Figure 1 - Receive Data storage using MSCI and DMAC

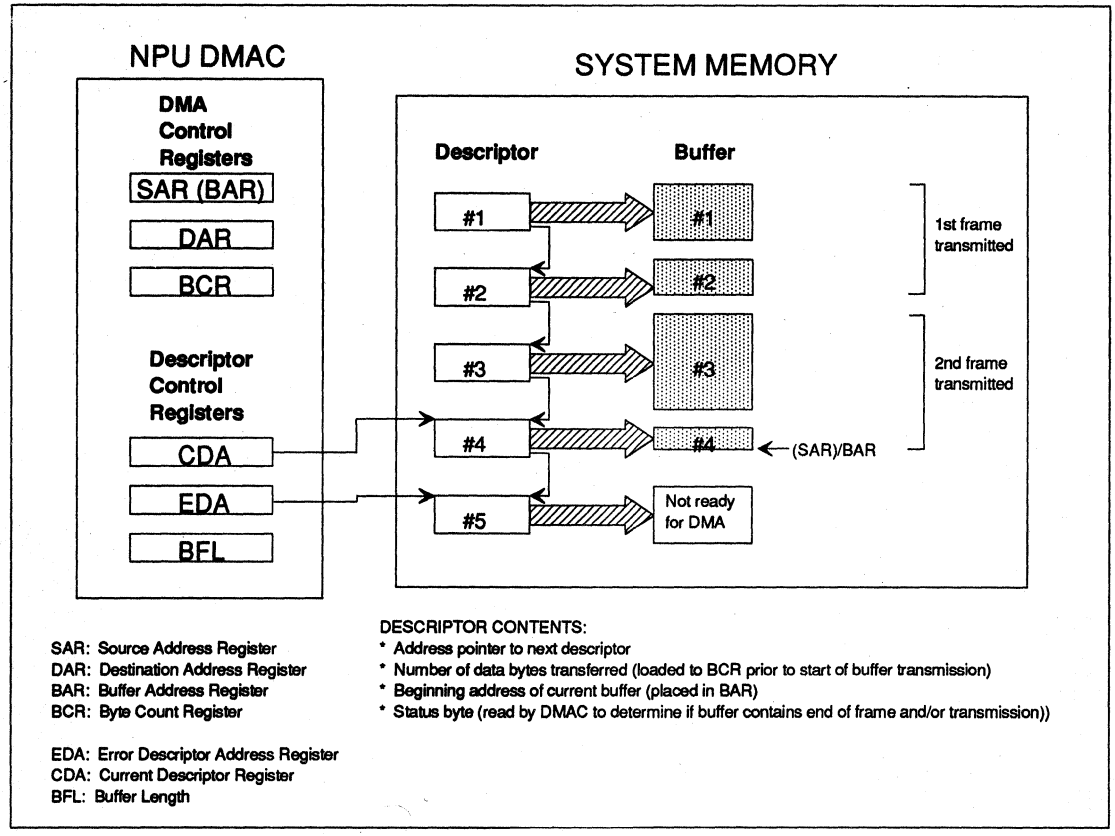


Figure 2 - Accessing transmit data usind MSCI and DMAC

data from and receiving data into a series of blocks, or buffers, in memory. This capability is provided by an internal connection between the DMAC channel 0 and the MSCI receiver, and channel 1 and the MSCI transmitter.

These memory buffers can be located anywhere in the 1 MByte memory space accessible by the NPU. Once initialization of the DMAC and a table in RAM defining the memory buffers occurs, data transfer and buffer management proceeds without further CPU intervention.

This Application Note discusses the details of this DMAC capability, refered to as the Chained-block Transfer Mode. A simple example program using the MSCI to transfer data using this mode is also discussed, and the program code is included as an example.

NPU's Buffer Management Scheme

The NPU can dynamically allocate individual buffers in various locations in system memory as the source or destination for DMA transfers. These buffers are identified when the user initializes a table of information in memory which contains an entry for each buffer. Each entry, called a descriptor, contains attributes such as start of buffer location, link pointer to the address of the next buffer's descriptor, buffer size, and buffer status. Using this descriptor table, the DMA channel can autonomously handle Bit Synchronous protocol data transfers between the MSCI and the memory buffers.

The size of buffers used depends on the application. Bit-

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

oriented protocols generally operate by transferring groups of data framed by control information, which are referred to as frames or packets of data. The size of buffers can be separately specified for transmit and receive data.

For transmissions, data to be sent in a frame may be located in one or more locations in memory, and therefore the transmit buffer(s) would be shorter than or equal in length to the data frame/packet. Data in several buffers can be contiguously sent to make up a frame. The status information held in each transmit buffer descriptordescriptor indicates to the DMAC if the buffer contains the end of a frame.

The DMAC can be programmed to transmit one frame of data at a time, or to transmit multiple frames to assist in high speed data transfer.

If the DMAC channel is initialized for multiple frame transmissions, the status information in the transmit buffer descriptor includes a bit to indicate if the buffer identified by the descriptor contains the end of data to be transmitted.

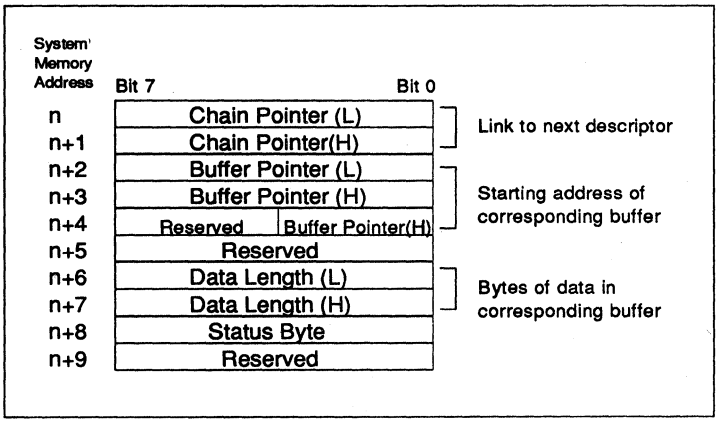


Figure 3 - Buffer Descriptor for Chained Block Transfer DMA

For reception, frames are received and stored in one or more buffers. A pre-determined maximum buffer size is used to allocate memory for these buffers. If the incoming data frame is longer than this maximum size specified, the frame will be broken up into separate buffers. The MSCI signals the DMAC when the last byte of a frame has been transferred using an internal signal line. By specifying a maximum buffer size equal to the size of the average frame, less memory space will be wasted by smaller frames or portions of frames which do not fill the buffer size allocated.

If multi-frame mode of operation has been specified, the DMAC will perform buffer switching following the receipt of the End of Frame signal. Data for the next frame will be received into a new buffer.

Advantages of the Linked-Chaining Mode

The NPU's linked chaining capabilities provide advantages over conventional DMA handling of serial data. Some of the advantages of this scheme are listed below:

- *High performance in both serial data transmission rates and system throughput* — the Chained-block Transfer Mode allows the DMAC to continuously load data received by the MSCI into memory without incurring CPU overhead to switch buffers. Coupled with the MSCI's multi-block transfer capability, the DMAC operating in this mode can move back-to-back frames of data located in different parts of memory to the MSCI's transmitter, also without CPU assistance. System processing time for received data is enhanced by the MSCI's three-byte FIFO for both receiver and transmitter.

- *Efficient utilization of system memory* — In a system without smart buffer management, consecutive memory blocks need to be reserved for each of the frames or packets of data to be received. The size of each memory block would need to be a long as the maximum frame/packet size (e.g. 264 bytes in an ISDN LSPD frame, 603 bytes in a LLAP packet for AppleTalk).

With the capabilities of the NPU, multiple buffers can be chained together to hold the data contained in one received frame. Although receive data buffer sizes must be fixed (for memory allocation), the size selected can be optimized by specifying the average frame size.

- *Efficient transmission of data stored in multiple locations* — In the case of data transmission, frame data may be scattered in system memory space in blocks of differing size. Provided with link information in the descriptors, the NPU's DMA can autonomously transfer such a frame of data to the MSCI's transmitter. Without this linking capability, the data

Receive Buffer Descriptor	Transmit Buffer Descriptor
Buffer's size is pre-programmed in BFL register	Buffer size is variable
Data length field is written with the BCR contents by the DMAC during buffer switching. If switching occurs in the middle of a frame, length field will equal BFL register value.	Data length field is programmed by the user During buffer switching, data length information iis written to the DMAC's BCR.
Status field is written by theDMAC during each buffer switch	CPU programs the Status field value
Status field is read by the user program prior to processing the receive buffer data	During buffer switching, DMAC reads the status field to determine if actions should be taken upon completion of transmission. If EOM bit is set, DMA notifies MSCI of end of frame. In multi-frame mode, if the EOT bit is set, the DMAC stops following data transfer. Otherwise, if data is available (CDA not equal to EDA), transmission continues.

Table 1 - Differences between transmit and receive buffer descriptors

blocks would have to be combined in memory prior to transmission. With the DMAC's multiple frame transmission capability, multiple frames of this type can be transferred to the MSCI's transmitter continuously by DMA without CPU intervention.

- *Low bus bandwidth overhead* — Without buffer management capability, the DMAC would have to poll to determine if data was available for transmission, or an interrupt would have to be generated and serviced in order to signal the DMAC that data was ready to be transferred.

The NPU's buffer management capability allows the CPU to dynamically update the Error Descriptor Address (EDA) register which indicates to the DMAC the address of the last descriptor in the linked list. The buffer referenced by this last descriptor is not available to the DMAC. Each time the DMAC finishes with a buffer and proceeds to the attributes of the next buffer in the linked descriptor list, the EDA is automatically compared with the address of the new descriptor. If the addresses match, then DMA transfer ends. If the addresses do not match, the DMAC begins transfer to or from this buffer. No bus activity is required to make this determination.

- *Low overhead in buffer switching* — the DMAC automatically performs housekeeping tasks during buffer switching. No programmed intervention is required.

DMAC Control Registers

The following registers are used by the DMAC to implement the Chained Block Transfer Mode.

CPB: Chain Pointer Base specifies the four highest order bits of the descriptor's address. Written only by the CPU.

CDA: Current Descriptor Address specifies the lower 16 bits of the current descriptor's 20 bit address. CPB provides the four highest bits. The CDA is initially programmed by the user, and is updated by the DMAC during buffer switching.

EDA: Error Descriptor Address indicates the lower 16 bits of a descriptor's starting address. The descriptor identified in this register is the entry in the linked list following the last descriptor which is valid for DMAC use. The user can dynamically reprogram the EDA during MSCI - DMAC

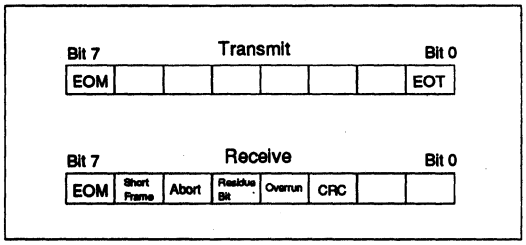


Figure 4 - DMA descriptor Status Bytes

operation which allows reuse of transmit or receive buffers.

BCR: Byte Count Register is a 16-bit down counter. For transmitting, the BCR indicates the number of remaining data bytes to be transferred in the transmit buffer current being accessed. For receiving, the BCR represents the number of unused bytes remaining in the receive buffer. The initial value of BCR is obtained from the current descriptor. The BCR is updated by the DMAC and cannot be written by the user.

BAR: Buffer Address Register specifies the address of data being transferred from or to the buffer. The initial value of BAR is obtained from the current descriptor. This register is updated by the DMAC, and cannot be written by the user.

BFL: Receive Buffer Length specifies the fixed size of receive buffers. All receive buffers have the same size determined by the value programmed in the BFL. During buffer switching, the BFL is copied to the BCR. BFL is programmed by the user.

DMA Buffer Descriptor Format

Each buffer to be used by the DMAC for data transfer must be identified using a descriptor, which has a format as shown in Figure 3. Each descriptor contains 10 bytes of data. All descriptor information must be contained in the 64Kbyte space defined by the CPB (Chain Pointer Base), so up to 6,553 descriptors may be defined. The components of a descriptor are described below:

Chain Pointer: Two bytes providing link information. The 20-bit address of the next descriptor in the linked list is formed by combining these bits with the information in the CPB (Chain Pointer Base) as the four most significant bits.
Buffer Pointer: Twenty bits which point to the start address of the buffer corresponding to this descriptor

Data Length: The amount of data stored in the buffer. For transmits, this information is initialized by the user. For receives, this information is written by the DMAC when the buffer is filled, and is determined by the BFL (Receive Buffer Length) and the BCR (Byte Count Register) value when reception into the buffer has completed.

Status Byte: Status information on the buffer. The status byte differs for transmit and receive descriptors, as shown in Figure 4.

For transmit buffer descriptors, the status byte's EOM and EOT bits are read by the DMAC during buffer switching. EOM = 1 notifies the DMAC that the corresponding buffer contains the end of a frame. The EOT bit is used only when the MSCI is in multi-frame transmit mode. EOT = 1 indicates that the corresponding buffer contains the end of data to be transferred, and DMAC operation ends when this buffer has been transmitted.

For receive buffer descriptors, the status byte is directly copied from the MSCI's Frame Status register (MFST) upon the DMAC's receipt of an EOM condition. When the DMAC receives the end of frame signal internally from the MSCI, the DMAC will request the MSCI to dump the contents of its status register during buffer switching. If the buffer switching occurs in the middle of receiving a frame, the DMAC writes 00h to the descriptor's status byte.

Constructing a ring of descriptors for re-using buffers

Since buffer descriptors are organized in a linked list, it is possible to form a ring by programming the address of the first entry into the link field of the last entry. This would allow the DMAC to automatically access the list of descriptors and their associated buffers in a ring configuration. This would allow re-use of buffers in memory.

Because the DMAC compares the CDA and EDA in order to determine when all buffers available to the DMAC have been accessed, there will always be at least one descriptor that references a buffer that is unavailable to the DMAC. There are two ways of dealing with this descriptor and buffer:

1. This descriptor can be set up as a dummy. The last real descriptor in the link would point to this dummy descriptor, and the EDA would be programmed with the address of this descriptor. When the DMAC reaches this dummy entry in the linked list, the CDA would equal the EDA, and the DMA channel would stop.

2. The EDA can be programmed to the address of this last descriptor at the start, then when the buffer associated with the

first descriptor has been accessed the EDA can be reprogrammed to the address of the first descriptor. The link pointer of the last descriptor should point to the address of the first descriptor. The CPU could poll the DMAC's CDA register to determine when buffer 1 transfer has completed (CDA will contain the address of buffer 2's descriptor). Using this method of reprogramming the EDA allows buffers to be used in a ring configuration.

An example using this ring configuration of receive buffers is shown in Figure 5. In this example, eight memory buffers are allocated for receiving data from the MSCI under DMA control.

A program using this ring configuration of receive buffers was written to demonstrate the initialization of the NPU required to handle this type of receive operation. The code for this program is included in Appendix A.

To simplify the hardware required to execute this program, the NPU MSCI's local loopback mode was selected to cause data placed in the MSCI's transmit buffer to be looped back into the receive buffer. The data to be transmitted was initialized as a single frame contained in two separate buffers in memory. The transmit descriptor table was configured as a linked list containing the two entries for the buffers plus a dummy descriptor.

The program initializes the MSCI for bit oriented communications, then uses chained block transfer mode of the DMAC to transmit a frame of data. This transmitted data is looped back into the MSCI receiver, where it is received using chained block transfer mode DMA. As each of the receive buffers are filled, the contents of the buffer is moved under program control to a new location in memory.

This program was tested using the Hitachi 64180S ASE (Adaptive System Evaluator) Emulator. Correct operation of the program was determined by examining memory upon completion of the program to show that the transmitted data had been received and stored into memory. Examination of the DMAC registers also showed that DMA transfers had completed sucessfully.

The sequence of events that occur in this program is described below:

Initialization by user program:

- Transmit data buffers initialized
- Transmit descriptor table for DMA initialized
- Receive descriptor table for DMA initialized
- MSCI initialized for bit oriented protocol, single frame transfer, local loopback mode
- DMAC channel 0 initialized for receiving data into the receive buffers in chained block mode
- DMAC channel 1 initialized for transmitting data from the transmit buffers using chained block transfer mode
- DMAC channels, MSCI receiver and MSCI transmitter are enabled in that order

Preparation for transmitting data

Both DMAC channels prepare to transfer data between the MSCI and memory. For DMAC channel 1, the following occurs:

- DMAC compares CDA1 and EDA1, and continues if not equal
- DMAC reads information from the transmit descriptor pointed to by CDA1. The starting address information is stored in the DMAC's Source Address Register (SAR). The link address and status information are transferred to DMAC internal registers for future reference.
- The DMAC loads the data length value from the descriptor into the BCR
- DMAC responds to the MSCI internal request to accept transmit data

Transmitting data

- DMAC transfers one byte of data from buffer to the MSCI transmitter
- DMAC decrements the BCR1 and increments the DAR1
- Transfers continue until BCR1 = 0
- When BCR1 = 0, buffer switching is initiated

Buffer switching during transmission

- The link field information of the descriptor for the buffer just transferred (descriptor #1, buffer #1) is read from the internal working register and written to the CDA1 register.
- The status field information of descriptor #1, which was placed in an internal working register prior to transmitting buffer #1's data, is checked to determine if the EOM bit is set. If so, transfer ends. If this bit is not set, tasks continue. (If Multi-frame mode was selected, the status field would be checked for EOT bit set, and transmission would complete only if this bit was set).
- This new CDA1 value is compared with the value in EDA1. If they are not equal, the buffer referenced by the descriptor addressed by the CDA1 contains data that should be transmitted. (If CDA = EDA, DMA transfer stops)
- The new descriptor's link field is stored in the internal working register. The buffer pointer stored in the descriptor

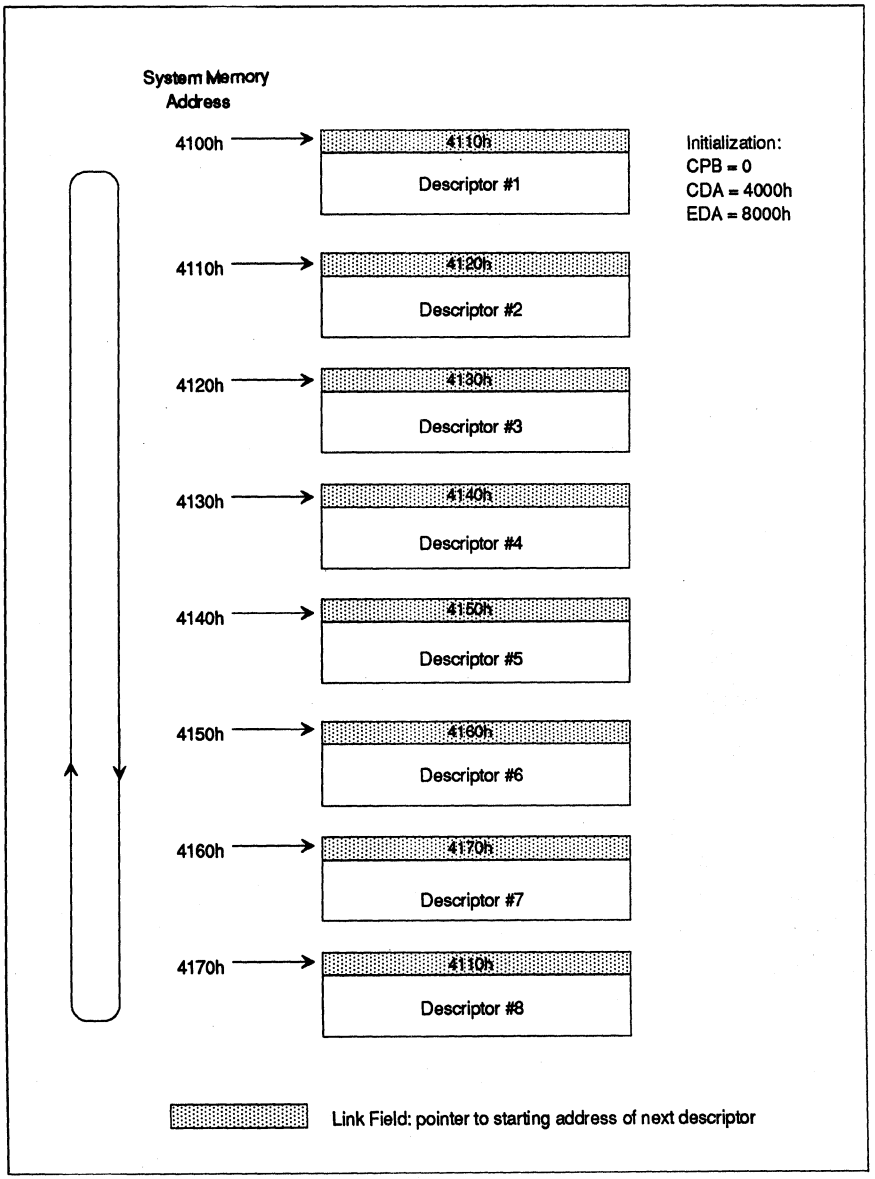


Figure 5 - Buffer Descriptors set up for continuous receipt of MSCI data

is loaded into the DMAC's BAR

- Data length field of the descriptor pointed to by the link field of descriptor #1 is loaded into the BCR.
- Status field of this next descriptor is loaded into an internal working register.
- Data transmission from buffer #2 begins.

33 bus states are required to complete the buffer switching tasks when the switch occurs during the middle of a frame. 37 states are required when the buffer switch occurs at the end of a frame.

Preparation for receiving data

For DMAC Channel 0 (DMA receive data), the following actions occur:

- DMAC compares CDA0 and EDA0, and continues if not equal
- DMAC reads information from the receive descriptor pointed to by CDA0. The starting address information is stored in the DMAC's Destination Address Register (DAR0). The link address is transferred to a DMAC internal register for future reference.
- The DMAC loads the data length value from BFL0 into BCR0.
- DMAC responds to the MSCI internal request to receive data.

Receiving data

- DMAC transfers one byte of data from the MSCI receiver to the receive buffer
- DMAC decrements BCR0 and increments DAR0
- Transfers continue until end of frame is detected or BCR1 = 0, when buffer switching is initiated

Receive Buffer Switching

- The number of bytes received into the buffer is written to the descriptor's data length field.
- If the end of the frame/packet was not received into this buffer, the DMAC writes 00h to the status field of the descriptor. If the end of the frame was received, the DMAC requests the MSCI to put its MFST value onto the internal data bus. This information is transferred by the DMAC to the descriptor's status field.
- The descriptor's link field information, which was placed into an internal working register when the descriptor was examined prior to receiving the data, is transferred to the CDA register.
- This new CDA value is compared with the value in the EDA. If they are not equal, the buffer referenced by the descriptor addressed by the CDA is available to store receive data. (If CDA = EDA, DMA transfer stops)
- The new descriptor's link field is stored in the internal working register. The buffer pointer stored in the descriptor is loaded into the DMAC's BAR.
- The DMAC's BFL value is copied to the BCR
- Data reception into the new buffer begins

34 states are required to perform these buffer switching tasks.

Closing the loop

After one or more receive buffers have been filled by DMA transfers, the user program can begin processing the buffer data. When processing of first buffer's data is complete, the user program can update the EDA0 to point to the address of the next buffer's descriptor. The receive buffers will be filled sequentially, and when the eighth buffer is filled, its link address, which will be placed in CDA0, will point to the descriptor of the first buffer. Since EDA0 does not point to this first buffer's descriptor, upon comparison CDA0 will not equal EDA0 so reception from the MSCI will continue and receive data will be transferred to this first buffer again, closing the loop.

In this example program, the processing performed on the received data is to move it from the receive buffer to another location in memory, where the entire frame will be assembled in contiguous bytes. Following enabling of the DMAC and the MSCI, the user program polls the CDA0 register to determine when the first buffer has been filled and the DMAC has switched to filling the second buffer. The user program detects that this has occurred when the CDA0 register points to the second descriptor. The program continues this polling then moving for each receive buffer in the descriptor table.

Although eight receive buffers are available to the DMAC, a total of ten buffers worth of data will be received to process the transmitted frame. To accomplish this, the first two receive buffers must be reused. Upon initialization, EDA0 points to the descriptor for the eighth buffer. The DMAC will therefore stop receiving data when the eighth descriptor is reached, unless EDA0 is reprogrammed.

Once the first buffer has been filled and the data transferred, this buffer is available for reuse. At this point in the program, EDA0 can be reprogrammed to point to the address of the descriptor of the second buffer. When the second buffer has been filled and transferred, EDA0 can safely be reprogrammed to point to the address of the descriptor of the third buffer. When this second buffer has been filled the second time and buffer switching is initiated, the DMAC will detect that the

CDA = EDA, and transfer will stop.

Questions and Answers

The following paragraphs contain questions and answers pertaining to the use of the Chained Block Transfer Mode of DMA operation.

How does the DMAC know that the next descriptor references a buffer that is available for transfer?

During buffer switching, the current buffer's chain pointer becomes the new CDA. The DMAC compares this new CDA with the current EDA value. If they are equal, the end of the chain has been reached. If they are not equal, the new buffer is available for use by the DMAC.

How does the user program know that a particular buffer has been used by the DMAC?

The program can read the DMAC channel's CDA. If, for instance, the CDA contains the starting address of the third descriptor in the linked list, the user can determine from this that:

1. The DMAC is ready to access buffer #3 OR
2. The DMAC is in the process of accessing buffer #3.

In either case, the user can be certain that the DMAC is finished with buffers corresponding to the first and second descriptors in the list, and these buffers can be processed by the user program and then reused as DMAC data buffers.

How can the DMAC reclaim buffers written with data received in a bad frame?

The MSCI can be programmed to cause an interrupt upon receipt of a bad frame, for instance by signalling a CRC error. In the service routine for this interrupt, the user program can determine which buffer contains the start of the bad frame detected. The program can then disable the DMA channel handling the receive, and reprogram the CDA with the start address of the descriptor referencing this buffer. Upon reenabling the DMA channel, new data will be written over the data of the bad frame.

How can a frame be retransmitted?

The user program can determine the start address of frames being transmitted from the transmit buffer descriptor table. If the need to retransmit a frame occurs, the DMAC can be disabled and the CDA register can be reprogrammed with the address of the descriptor corresponding to the buffer containing the start of the frame to be retransmitted. When DMA is reenabled, transmission will start with the beginning of the frame.

```
0000          CPU     "64180.TBL"
0000          HOF  "INT8"

              ;*************************************************************
              ;  TITLE:  NPU DMA BUFFER MANAGEMENT DEMOSTRATION PROGRAM
              ;*************************************************************
              ;        DEFSEG ROM, ABSOLUTE
              ;        SEG    ROM
              ;
              ; DATA TO BE TRANSMITTED
              ;
3000                  ORG    3000H
3000 030B465241       DFB     03,0BH,"FRAME1 B1-BUFFER MANAGEMENT DEMOSTRATION PROGRAM";50 BYTES
3100                  ORG    3100H
3100 204652414D       DFB     " FRAME1 B2-ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklm"    ;50 BYTES
              ;*****************************************************
              ;  TRANSMIT DESCRIPTOR SET UP
              ;*****************************************************
              ; TX BUFFER 1
4000                  ORG    4000H
4000 1040             DWL     4010H      ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 4010H
4002 0030             DWL     3000H      ; LOWER 16 BIT OF TX BUFFER POINTER = 3000H
4004 00               DFB     00         ; HIGHER 4 BIT OF 20 BIT TX BUFFER POINTER = 00
4005 00               DFB     0          ; RESERVED
4006 3200             DWL     50         ; DATA LENGTH OF BLOCK 1 = 50
4008 00               DFB     0          ; STATUS ===> NO EOM , NO EOT
4009 00               DFB     0          ; RESERVED

              ; TX BUFFER 2
4010                  ORG    4010H
4010 2040             DWL     4020H      ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 4020H
4012 0031             DWL     3100H      ; LOWER 16 BIT OF TX BUFFER POINTER = 3100H
4014 00               DFB     00         ; HIGHER 4 BIT OF 20 BIT TX BUFFER POINTER = 00
4015 00               DFB     0          ; RESERVED
4016 3200             DWL     50         ; DATA LENGTH OF BLOCK 2 = 50
4018 81               DFB     81H        ; STATUS ===>  EOM ,  EOT
4019 00               DFB     0          ; RESERVED

              ; TX BUFFER 3
4020                  ORG    4020H
4020 3040             DWL     4030H      ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 4030H
4022 0032             DWL     3200H      ; LOWER 16 BIT OF TX BUFFER POINTER = 3200H
4024 00               DFB     00         ; HIGHER 4 BIT OF 20 BIT TX BUFFER POINTER = 00
4025 00               DFB     0          ; RESERVED
4026 3200             DWL     50         ; DATA LENGTH OF BLOCK 3 = 50
4028 80               DFB     80H        ; STATUS ===> EOM , NO EOT (END OF FRAME 1)
4029 00               DFB     0          ; RESERVED
              ;*****************************************************
              ;  RECEIEVE DESCRIPTOR SET UP
              ;*****************************************************
              ; RX BUFFER 1
4100                  ORG    4100H
```

```
4100 1041          DWL    4110H     ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 4110H
4102 0030          DWL    3000H     ; LOWER 16 BIT OF RX BUFFER POINTER = 3000H
4104 03            DFB    03        ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 03
4105 00            DFB    0         ; RESERVED
4106 0000          DWL    0         ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
4108 00            DFB    0         ; STATUS ===> TO BE WRITTEN BY DMA
4109 00            DFB    0         ; RESERVED
           ; RX BUFFER 2
4110               ORG    4110H
4110 2041          DWL    4120H     ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 4120H
4112 2030          DWL    3020H     ; LOWER 16 BIT OF RX BUFFER POINTER = 3020H
4114 03            DFB    03        ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 03
4115 00            DFB    0         ; RESERVED
4116 0000          DWL    0         ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
4118 00            DFB    0         ; STATUS ===> TO BE WRITTEN BY DMA
4119 00            DFB    0         ; RESERVED
           ; RX BUFFER 3
4120               ORG    4120H
4120 3041          DWL    4130H     ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 4130H
4122 4030          DWL    3040H     ; LOWER 16 BIT OF RX BUFFER POINTER = 3040H
4124 03            DFB    03        ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 03
4125 00            DFB    0         ; RESERVED
4126 0000          DWL    0         ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
4128 00            DFB    0         ; STATUS ===> TO BE WRITTEN BY DMA
4129 00            DFB    0         ; RESERVED
           ; RX BUFFER 4
4130               ORG    4130H
4130 4041          DWL    4140H     ; STARTING ADDRESS OF NEXT DESCRIPTOR = 4140H
4132 6030          DWL    3060H     ; LOWER 16 BIT OF RX BUFFER POINTER = 3060H
4134 03            DFB    03        ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 03
4135 00            DFB    0         ; RESERVED
4136 0000          DWL    0         ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
4138 00            DFB    0         ; STATUS ===> TO BE WRITTEN BY DMA
4139 00            DFB    0         ; RESERVED
           ; RX BUFFER 5
4140               ORG    4140H
4140 5041          DWL    4150H     ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 4150H
4142 8030          DWL    3080H     ; LOWER 16 BIT OF RX BUFFER POINTER = 3080H
4144 03            DFB    03        ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 03
4145 00            DFB    0         ; RESERVED
4146 0000          DWL    0         ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
4148 00            DFB    0         ; STATUS ===> TO BE WRITTEN BY DMA
4149 00            DFB    0         ; RESERVED
           ; RX BUFFER 6
4150               ORG    4150H
4150 6041          DWL    4160H     ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 4160H
4152 A030          DWL    30A0H     ; LOWER 16 BIT OF RX BUFFER POINTER = 30A0H
4154 03            DFB    03        ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 03
4155 00            DFB    0         ; RESERVED
4156 0000          DWL    0         ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
4158 00            DFB    0         ; STATUS ===> TO BE WRITTEN BY DMA
```

```
4159 00          DFB     0          ; RESERVED
            ; RX BUFFER 7
4160             ORG    4160H
4160 7041        DWL     4170H      ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 4170H
4162 C030        DWL     30C0H      ; LOWER 16 BIT OF RX BUFFER POINTER = 30C0H
4164 03          DFB     03         ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 03
4165 00          DFB     0          ; RESERVED
4166 0000        DWL     0          ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
4168 00          DFB     0          ; STATUS ===> TO BE WRITTEN BY DMA
4169 00          DFB     0          ; RESERVED
            ; RX BUFFER 8
4170             ORG    4170H
4170 0041        DWL     4100H      ; STARTING ADDRRESS OF NEXT DESCRIPTOR = 4100H
4172 E030        DWL     30E0H      ; LOWER 16 BIT OF RX BUFFER POINTER = 30E0H
4174 03          DFB     03         ; HIGHER 4 BIT OF 20 BIT RX BUFFER POINTER = 03
4175 00          DFB     0          ; RESERVED
4176 0000        DWL     0          ; DATA LENGTH OF BLOCK 1 TO BE WRITTEN BY DMA
4178 00          DFB     0          ; STATUS ===> TO BE WRITTEN BY DMA
4179 00          DFB     0          ; RESERVED
            ;*****************************************************
            ;  CPU ROUTINE
            ;*****************************************************
F000             ORG    0F000H
F000        RESET:

            ;*******************************************************
            ;       MSCI TRANSMIT IN DMA SET UP
            ;*******************************************************
            ;
            ;  TRANSMITTER AND RECEIEVER SET UP
            ;
F000 3E21          LD   A,21H
F002 ED392A        OUT0 (MCMD),A    ; CHANNEL RESET
F005 3E87          LD   A,87H
F007 ED392B        OUT0 (MMD0),A    ; BIT-SYNC HDLC,AUTO ENABLE=0,CRC-CCITT=1 INITIALLY
F00A 3E00          LD   A,00H
F00C ED392C        OUT0 (MMD1),A    ; ADDRESS NOT CHECKED
F00F 3E03          LD   A,03H
F011 ED392D        OUT0 (MMD2),A    ; FULL DUPLEX, NRZ CODE, LOCAL LPBK
F014 3EB1          LD   A,0B1H
F016 ED392E        OUT0 (MCTL),A    ; DMA,FLAG AND IDLE, -RTSM=HIGH
                                    ; SPECIFIES FCS NO-LOAD
F019 3E43          LD   A,43H
F01B ED3933        OUT0 (MRXS),A    ; RXCM FROM BRG
F01E 3E43          LD   A,43H
F020 ED3934        OUT0 (MTXS),A    ; TXCM FROM BRG
F023 3E80          LD   A,80H
F025 ED3932        OUT0 (MTMC),A    ; SET TMC=128 -> BAUD RATE SELECTED=9.6K

F028 3E00          LD   A,000H
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

```
F02A ED3926          OUT0 (MIE0),A    ; TXINT AND RXINT DISABLED
F02D 3E83            LD   A,83H
F02F ED3927          OUT0 (MIE1),A    ; UNDERRUN,ABORT,IDLE DETECTION INTERRUPT ENABLED
F032 3E32            LD   A,32H
F034 ED392F          OUT0 (MSA0),A    ; SECONDARY STATION ADDRESS = 02H
F037 3E7E            LD   A,7EH
F039 ED3931          OUT0 (MIDL),A    ; SET FLAG PATTERN = 01111110 AS IDLE PATTERN
            ;*****************************************************
            ;  DMA CHANNEL 0 SET UP   (RECEIEVER)
            ;*****************************************************
F03C 3E94          LD    A,94H            ; DMA MODE REGISTER A
F03E ED3969        OUT0  (DMRA0),A        ; MSCI,CBSA,MSCI->MEMORY,MULTI FRAME
F041 3E00          LD    A,0
F043 ED395D        OUT0  (CPB0),A         ; 4 HIGHER BIT OF THE 20-BIT DESCRIPTOR ADDR

F046 3E70          LD    A,70H            ;
F048 ED3960        OUT0  (EDA0L),A        ; 4170H
F04B 3E41          LD    A,41H            ; STARTING ADDR (LOW-ORDER 16 BITS) OF THE
F04D ED3961        OUT0  (EDA0H),A        ; RX DESCRIPTOR 8

F050 3E00          LD    A,00H            ;
F052 ED395E        OUT0  (CDA0L),A        ; 4100H
F055 3E41          LD    A,41H            ; STARTING ADDR (LOW-ORDER 16 BITS) OF THE
F057 ED395F        OUT0  (CDA0H),A        ; FIRST RX DESCRIPTOR

F05A 3E0A          LD    A,10
F05C ED3962        OUT0  (BUFL0L),A
F05F 3E00          LD    A,0
F061 ED3963        OUT0  (BUFL0H),A       ; ALLOWING ONLY 10 BYTES IN EACH RX BUFFER

F064 3E80          LD    A,80H
F066 ED396C        OUT0  (DIR0),A         ; EOT INTERRUPT ENABLED

            ;*****************************************************
            ;  DMA CHANNEL 1 SET UP   (TRANSMITTER)
            ;*****************************************************
F069 3E9C          LD    A,09CH           ; DMA MODE REGISTER A
F06B ED3981        OUT0  (DMRA1),A        ; MSCI,CHAINED,MEMORY->MSCI,MULTI FRAME
F06E 3E00          LD    A,0
F070 ED3975        OUT0  (CPB1),A         ; 4 HIGHER BIT OF THE 20-BIT DESCRIPTOR ADDR

F073 3E20          LD    A,20H            ;
F075 ED3978        OUT0  (EDA1L),A        ; 4020H
F078 3E40          LD    A,40H            ; STARTING ADDR (LOW-ORDER 16 BITS) OF THE
F07A ED3979        OUT0  (EDA1H),A        ; DESCRIPTOR NEXT TO THE LAST TX BUFFER

F07D 3E00          LD    A,00H            ;
F07F ED3976        OUT0  (CDA1L),A        ; 4000H
F082 3E40          LD    A,40H            ; STARTING ADDR (LOW-ORDER 16 BITS) OF THE
F084 ED3977        OUT0  (CDA1H),A        ; FIRST DESCRIPTOR OF THE FIRST TX BUFFER
```

SECTION 2

```
F087 3E80          LD    A,80H
F089 ED3984        OUT0  (DIR1),A          ; EOT INTERRUPT ENABLED
            ;
            ; TO ENABLE DMA CHANNEL 0, 1 AND MSCI TX,RX
            ;
F08C 3E02          LD    A,02H
F08E ED3968        OUT0  (DSR0),A          ; DMA CHANNEL 0 ENABLE
F091 3E02          LD    A,02H
F093 ED3980        OUT0  (DSR1),A          ; DMA CHANNEL 1 ENABLE
F096 3E12          LD    A,12H
F098 ED392A        OUT0  (MCMD),A          ; RX ENABLE
F09B 3E02          LD    A,02H
F09D ED392A        OUT0  (MCMD),A          ; TX ENABLE
            ;
            ; TEST CDA OF RECEIEVER
            ; CHANGE EDA AFTER EACH BLOCK IS DONE
            ; RECONSTRRUCT DATA INTO ONE PIECE
            ;
F0A0 3E30          LD      A,30H
F0A2 ED3902        OUT0    (BBR),A         ; BBR OF MMU SET AT 30
F0A5        BLK1:
F0A5 ED385E        IN0     A,(CDA0L)       ; CHECK LOWER BYTE OF CDA0
F0A8 FE00          CP      0
F0AA 28F9          JR      Z,BLK1
            ; MOVE BLOCK1
F0AC 210030        LD      HL,3000H
F0AF 11003A        LD      DE,3A00H
F0B2 010A00        LD      BC,10
F0B5 EDB0          LDIR

F0B7        BLK2:
F0B7 ED385E        IN0     A,(CDA0L)       ; CHECK LOWER BYTE OF CDA0
F0BA FE10          CP      10H
F0BC 28F9          JR      Z,BLK2
            ; MOVE BLOCK2
F0BE 212030        LD      HL,3020H
F0C1 010A00        LD      BC,10
F0C4 EDB0          LDIR
            ;       CALL    PUTSCR
            ;       DFB       'BLOCK2 MOVED',0

            ; SET NEW EDA
F0C6 3E10          LD    A,10H             ;
F0C8 ED3960        OUT0  (EDA0L),A         ; NEW EDA0=4110H
F0CB 3E41          LD    A,41H             ;  buffer of descr at 4100h
F0CD ED3961        OUT0  (EDA0H),A         ;  ready for reuse
F0D0        BLK3:
F0D0 ED385E        IN0     A,(CDA0L)       ; CHECK LOWER BYTE OF CDA0
F0D3 FE20          CP      20H
F0D5 28F9          JR      Z,BLK3
            ; MOVE BLOCK3
```

```
F0D7 214030          LD      HL,3040H
F0DA 010A00          LD      BC,10
F0DD EDB0            LDIR

F0DF          BLK4:

              ; SET NEW EDA
F0DF 3E20            LD    A,20H              ;
F0E1 ED3960          OUT0  (EDA0L),A          ; NEW EDA0=4120H
F0E4 3E41            LD    A,41H              ;  buffer of descr 4110h
F0E6 ED3961          OUT0  (EDA0H),A          ;  can be reused

F0E9 ED385E          IN0     A,(CDA0L)        ; CHECK LOWER BYTE OF CDA0
F0EC FE30            CP      30H
F0EE 28EF            JR      Z,BLK4
              ; MOVE BLOCK4
F0F0 216030          LD      HL,3060H
F0F3 010A00          LD      BC,10
F0F6 EDB0            LDIR

F0F8          BLK5:
              ; SET NEW EDA
F0F8 3E30            LD    A,30H              ;
F0FA ED3960          OUT0  (EDA0L),A          ; NEW EDA0=4130H
F0FD 3E41            LD    A,41H              ;  buffer of descr 4120h
F0FF ED3961      OUT0  (EDA0H),A              ;  can now be reused

F102 ED385E          IN0     A,(CDA0L)        ; CHECK LOWER BYTE OF CDA0
F105 FE40            CP      40H
F107 28EF            JR      Z,BLK5
              ; MOVE BLOCK5
F109 218030          LD      HL,3080H
F10C 010A00          LD      BC,10
F10F EDB0            LDIR

F111          BLK6:
F111 ED385E          IN0     A,(CDA0L)        ; CHECK LOWER BYTE OF CDA0
F114 FE50            CP      50H
F116 28F9            JR      Z,BLK6
              ; MOVE BLOCK6
F118 21A030          LD      HL,30A0H
F11B 010A00          LD      BC,10
F11E EDB0            LDIR

F120          BLK7:
F120 ED385E          IN0     A,(CDA0L)        ; CHECK LOWER BYTE OF CDA0
F123 FE60            CP      60H
F125 28F9            JR      Z,BLK7
              ; MOVE BLOCK7
F127 21C030          LD      HL,30C0H
F12A 010A00          LD      BC,10
```

```
F12D EDB0               LDIR

F12F            BLK8:
F12F ED385E             IN0     A,(CDA0L)       ; CHECK LOWER BYTE OF CDA0
F132 FE70               CP      70H
F134 28F9               JR      Z,BLK8
                ; MOVE BLOCK8
F136 21E030             LD      HL,30E0H
F139 010A00             LD      BC,10
F13C EDB0               LDIR

F13E            BLK9:
F13E ED385E             IN0     A,(CDA0L)       ; CHECK LOWER BYTE OF CDA0
F141 FE00               CP      00H
F143 28F9               JR      Z,BLK9
                ; MOVE BLOCK9
F145 210030             LD      HL,3000H
F148 010A00             LD      BC,10
F14B EDB0               LDIR

F14D            BLK10:
F14D ED385E             IN0     A,(CDA0L)       ; CHECK LOWER BYTE OF CDA0
F150 FE10               CP      10H
F152 28F9               JR      Z,BLK10
                ; MOVE BLOCK10
F154 212030             LD      HL,3020H
F157 010A00             LD      BC,10
F15A EDB0               LDIR

F15C            BLK11:
F15C 3E00               LD      A,0H
F15E ED3902             OUT0    (BBR),A         ; BBR OF MMU SET AT 0

F161            DONE:
F161 C361F1       JP    $                       ; loop here when done

                ;-----------------------------
                ;  NPU EV BOARD IO ASSIGNMENT
                ;-----------------------------
                ;       npu io address assignments 10-11-88 06:23:00
                ;
0000 =          ICR:    EQU     000H    ; INTERRUPT CONTROL REG
0001 =          CBR:    EQU     001H    ; MMU COMMON BASE REG
0002 =          BBR:    EQU     002H    ; MMU BANK BASE REG
0003 =          CBAR:   EQU     003H    ; MMU COMMON/BANK AREA REG
0004 =          OMCR:   EQU     004H    ; OPERATION MODE CTRL REG
0005 =          IOCR:   EQU     005H    ; I/O CTRL REG
                ;
0020 =          MTRB:   EQU     020H    ; MSCI TX/RX BUFFER REG
0021 =          MST0:   EQU     021H    ; MSCI STATUS REG 0
0022 =          MST1:   EQU     022H    ; MSCI STATUS REG 1
```

```
0023 =          MST2:    EQU     023H    ; MSCI STATUS REG 2
0024 =          MST3:    EQU     024H    ; MSCI STATUS REG 3
0025 =          MFST:    EQU     025H    ; MSCI FRAME STATUS REG
0026 =          MIE0:    EQU     026H    ; MSCI INTERRUPT ENABLE REG 0
0027 =          MIE1:    EQU     027H    ; MSCI INTERRUPT ENABLE REG 1
                ;
0028 =          MIE2:    EQU     028H    ; MSCI INTERRUPT ENABLE REG 2
0029 =          MFEI:    EQU     029H    ; MSCI FRAME INTERRUPT ENABLE REG
002A =          MCMD:    EQU     02AH    ; MSCI COMMAND REG
002B =          MMD0:    EQU     02BH    ; MSCI MODE REG 0
002C =          MMD1:    EQU     02CH    ; MSCI MODE REG 1
002D =          MMD2:    EQU     02DH    ; MSCI MODE REG 2
002E =          MCTL:    EQU     02EH    ; MSCI CONTROL REG
002F =          MSA0:    EQU     02FH    ; MSCI SYNCHRONOUS ADDRESS REG 0
                ;
0030 =          MSA1:    EQU     030H    ; MSCI SYNCHRONOUS ADDRESS REG 1
0031 =          MIDL:    EQU     031H    ; MSCI IDLE PATTERN REG
0032 =          MTMC:    EQU     032H    ; MSCI TIME CONSTANT REG
0033 =          MRXS:    EQU     033H    ; MSCI RX CLOCK SOURCE REG
0034 =          MTXS:    EQU     034H    ; MSCI TX CLOCK SOURCE REG
                ;
0058 =          DAR0L:   EQU     058H    ; DESTINATION ADDRESS REG CH 0 LOW
0058 =          BAR0L:   EQU     058H    ; BUFFER ADDRESS REG CH 0 LOW
0059 =          DAR0H:   EQU     059H    ; DESTINATION ADDRESS REG CH 0 HI
0059 =          BAR0H:   EQU     059H    ; BUFFER ADDRESS REG CH 0 HI
005A =          DAR0B:   EQU     05AH    ; DESTINATION ADDRESS REG CH 0 BANK
005A =          BAR0B:   EQU     05AH    ; BUFFER ADDRESS REG CH 0 BANK
005B =          SAR0L:   EQU     05BH    ; SOURCE ADDRESS REG CH 0 LOW
005B =          DRWR0L:  EQU     05BH    ; DESCRIPTOR READ/WRITE REG CH 0 LOW
005C =          SAR0H:   EQU     05CH    ; SOURCE ADDRESS REG CH 0 HI
005C =          DRWR0H:  EQU     05CH    ; DESCRIPTOR READ/WRITE REG CH 0 HI
005D =          SAR0B:   EQU     05DH    ; SOURCE ADDRESS REG CH 0 BANK
005D =          CPB0:    EQU     05DH    ; CHAIN POINTER BASE CH 0
005E =          CDA0L:   EQU     05EH    ; ACCESS DESCRIPTOR ADDRESS REG CH 0 LOW
005F =          CDA0H:   EQU     05FH    ; ACCESS DESCRIPTOR ADDRESS REG CH 0 HI
                ;
0060 =          EDA0L:   EQU     060H    ; ERROR DESCRIPTOR ADDRESS REG CH 0 LOW
0061 =          EDA0H:   EQU     061H    ; ERROR DESCRIPTOR ADDRESS REG CH 0 HI
0062 =          BUFL0L:  EQU     062H    ; RX BUFFER LENGTH CH 0 LOW
0063 =          BUFL0H:  EQU     063H    ; RX BUFFER LENGTH CH 0 HI
0064 =          BCR0L:   EQU     064H    ; BYTE COUNT REG CH 0 LOW
0065 =          BCR0H:   EQU     065H    ; BYTE COUNT REG CH 0 HI
                ;
0068 =          DSR0:    EQU     068H    ; DMA STATUS REG CH 0
0069 =          DMRA0:   EQU     069H    ; DMA MODE REG A CH 0
006A =          DMRB0:   EQU     06AH    ; DMA MODE REG B CH 0
006B =          ICNT0:   EQU     06BH    ; FRAME END INTERRUPT COUNTER CH 0
006C =          DIR0:    EQU     06CH    ; DMA INTERRUPT ENABLE REG CH 0
006D =          DCR0:    EQU     06DH    ; DMA COMMAND REG CH 0
                ;
0070 =          DAR1L:   EQU     070H    ; DESTINATION ADDRESS REG CH 1 LOW
```

```
0070 =          BAR1L:   EQU     070H    ; BUFFER ADDRESS REG CH 1 LOW
0071 =          DAR1H:   EQU     071H    ; DESTINATION ADDRESS REG CH 1 HI
0071 =          BAR1H:   EQU     071H    ; BUFFER ADDRESS REG CH 1 HI
0072 =          DAR1B:   EQU     072H    ; DESTINATION ADDRESS REG CH 1 BANK
0072 =          BAR1B:   EQU     072H    ; BUFFER ADDRESS REG CH 1 BANK
0073 =          SAR1L:   EQU     073H    ; SOURCE ADDRESS REG CH 1 LOW
0073 =          DRWR1L:  EQU     073H    ; DESCRIPTOR READ/WRITE REG CH 1 LOW
0074 =          SAR1H:   EQU     074H    ; SOURCE ADDRESS REG CH 1 HI
0074 =          DRWR1H:  EQU     074H    ; DESCRIPTOR READ/WRITE REG CH 1 HI
0075 =          SAR1B:   EQU     075H    ; SOURCE ADDRESS REG CH 1 BANK
0075 =          CPB1:    EQU     075H    ; CHAIN POINTER BASE CH 1
0076 =          CDA1L:   EQU     076H    ; ACCESS DESCRIPTOR ADDRESS REG CH 1 LOW
0077 =          CDA1H:   EQU     077H    ; ACCESS DESCRIPTOR ADDRESS REG CH 1 HI
                ;
0078 =          EDA1L:   EQU     078H    ; ERROR DESCRIPTOR ADDRESS REG CH 1 LOW
0079 =          EDA1H:   EQU     079H    ; ERROR DESCRIPTOR ADDRESS REG CH 1 HI
007A =          BUFL1L:  EQU     07AH    ; RX BUFFER LENGTH CH 1 LOW
007B =          BUFL1H:  EQU     07BH    ; RX BUFFER LENGTH CH 1 HI
007C =          BCR1L:   EQU     07CH    ; BYTE COUNT REG CH 1 LOW
007D =          BCR1H:   EQU     07DH    ; BYTE COUNT REG CH 1 HI
                ;
0080 =          DSR1:    EQU     080H    ; DMA STATUS REG CH 1
0081 =          DMRA1:   EQU     081H    ; DMA MODE REG A CH 1
0082 =          DMRB1:   EQU     082H    ; DMA MODE REG B CH 1
0083 =          ICNT1:   EQU     083H    ; FRAME END INTERRUPT COUNTER CH 1
0084 =          DIR1:    EQU     084H    ; DMA INTERRUPT ENABLE REG CH 1
0085 =          DCR1:    EQU     085H    ; DMA COMMAND REG CH 1
                ;
                ;-----------------------------
                ; DEFINITIONS
                ;-----------------------------
FFFF =          USRSP:   EQU     0FFFFH
0005 =          PUTSCR:  EQU     5
00FE =          INTV:    EQU     0FEH
000D =          CR:      EQU     0DH
000A =          LF:      EQU     0AH
0011 =          XON:     EQU     11H
0013 =          XOFF:    EQU     13H
0003 =          EOT:     EQU     03H
0017 =          EOB:     EQU     17H
0000                     END
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD64180 Family

Application Note

Memory Read and Write Timing

Marnie Mar

Revision B

This note provides designers of HD64180-family device based systems with CPU timing information for Memory Read/Write cycles.

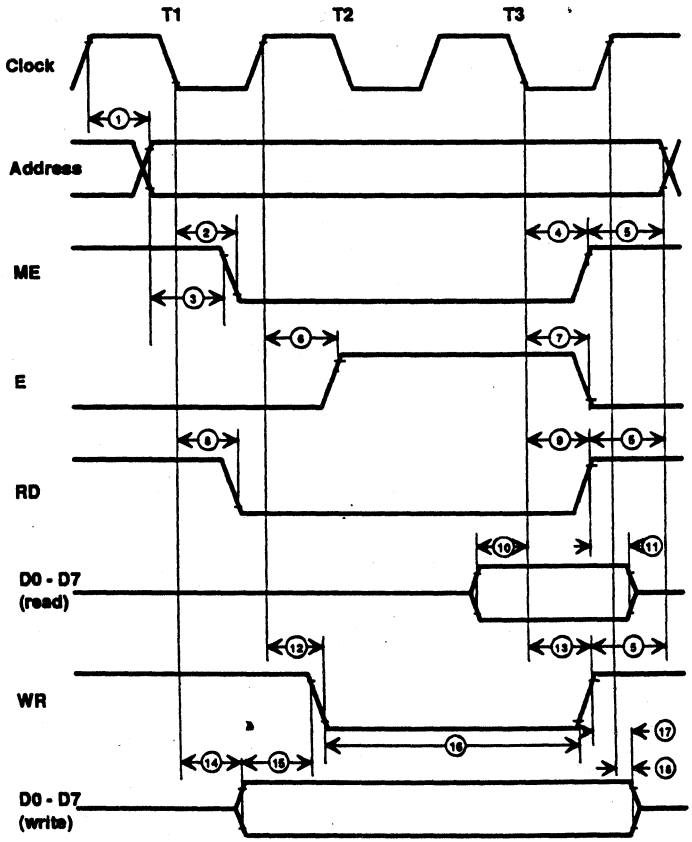
This timing differs from Opcode Fetch cycle in that for Memory Read cycles, read data is latched on the falling edge of T3. For Opcode Fetch, data is latched on the rising edge of T3. Opcode Fetch and I/O cycle timing diagrams are shown in the 64180 family data books.

Although the timing parameters shown on the following pages are contained in the data books, a timing diagram showing their relation to the system clock is not included.

Please refer to the following data books for more information on these timing parameters.

* HD64180 8-Bit Microprocessor Hardware Manual (#U77) (HD64180R0, HD64180R1, and HD64180Z)
*HD647180X 8-Bit Microcontroller Hardware Manual (#U94)

SECTION 2



CPU Memory Read/Write Timing

HD64180R0 AC Characteristics

Vcc = 5V +/- 10%, V_{SS} = 0 V, Ta= -20 to +75°C

#	Symbol	Item	HD64A180R0 (4MHz) Min	HD64A180R0 (4MHz) Max	HD64B180R0 (6MHz) Min	HD64B180R0 (6MHz) Max
1	t_{AD}	Address Delay Time*	-	110	-	105 nS
2	t_{MED1}	ME Delay Time 1	-	85	-	75 nS
3	t_{AS}	Address Set-up Time*	45	-	10	- nS
4	t_{MED2}	ME Delay Time 2	-	85	-	75 nS
5	t_{AH}	Address Hold Time	80	-	35	- nS
6	t_{ED1}	Enable Delay Time 1	-	100	-	95 nS
7	t_{ED2}	Enable Delay Time 2	-	100	-	95 nS
8	t_{RDD1}	RD Delay Time 1	-	85	-	75 nS
9	t_{RDD2}	RD Delay Time 2	-	85	-	75 nS
10	t_{DRS}	Data Read Set-up Time	50	-	45	- nS
11	t_{DRH}	Data Read Hold Time	0	-	0	- nS
12	t_{WRD1}	WR Delay Time 1	-	90	-	80 nS
13	t_{WRD2}	WR Delay Time 2	-	90	-	80 nS
14	t_{WDD}	Write Data Delay Time	-	110	-	90 nS
15	t_{WDS}	Write Data Set-up Time	60	-	40	- nS
16	t_{WRP}	WR Pulse Width	220	-	135	- nS
17	t_{WDH}	Write Data Hold Time	60	-	40	- nS
18	t_{WDZ}	Write Data Floating Delay Time	-	100	-	95 nS

* See data book for additional information on these items

HD 64180R1 AC Characteristics

Vcc = 5V +/- 10%, V_{SS} = 0 V, Ta= -20 to +75°C

#	Symbol	Item	HD64180R1-4 Min	HD64180R1-4 Max	HD64180R1-6 Min	HD64180R1-6 Max	HD64180R1-8 Min	HD64180R1-8 Max	HD64180R1-10* Min	HD64180R1-10* Max
1	t_{AD}	Address Delay Time	-	110	-	90	-	80	-	70 nS
2	t_{MED1}	ME Delay Time 1	-	85	-	60	-	50	-	50 nS
3	t_{AS}	Address Set-up Time	50	-	30	-	20	-	10	- nS
4	t_{MED2}	ME Delay Time 2	-	85	-	60	-	50	-	50 nS
5	t_{AH}	Address Hold Time	80	-	35	-	20	-	10	- nS
6	t_{ED1}	Enable Delay Time 1	-	100	-	95	-	70	-	60 nS
7	t_{ED2}	Enable Delay Time 2	-	100	-	95	-	70	-	60 nS
8	t_{RDD1}	RD Delay Time 1	-	85	-	60	-	50	-	50 nS
9	t_{RDD2}	RD Delay Time 2	-	85	-	60	-	50	-	50 nS
10	t_{DRS}	Data Read Set-up Time	50	-	40	-	30	-	25	- nS
11	t_{DRH}	Data Read Hold Time	0	-	0	-	0	-	0	- nS
12	t_{WRD1}	WR Delay Time 1	-	90	-	65	-	60	-	50 nS
13	t_{WRD2}	WR Delay Time 2	-	90	-	80	-	60	-	50 nS
14	t_{WDD}	Write Data Delay Time	-	110	-	90	-	80	-	60 nS
15	t_{WDS}	Write Data Set-up Time	60	-	40	-	20	-	15	- nS
16	t_{WRP}	WR Pulse Width	280	-	170	-	130	-	110	- nS
17	t_{WDH}	Write Data Hold Time	60	-	40	-	15	-	10	- nS
18	t_{WDZ}	Write Data Floating Delay Time	-	100	-	95	-	70	-	60 nS

* - Preliminary Specification

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD64180Z AC Characteristics

Vcc = 5V +/- 10%, V_{ss} = 0 V, Ta= -20 to +75°C

#	Symbol	Item	HD64180Z-4 Min	HD64180Z-4 Max	HD64180Z-6 Min	HD64180Z-6 Max	HD64180Z-8 Min	HD64180Z-8 Max	HD64180Z-10* Min	HD64180Z-10* Max	
1	t_{AD}	Address Delay Time	-	110	-	90	-	80	-	70	nS
2	t_{MED1}	ME Delay Time 1	-	85	-	60	-	50	-	50	nS
3	t_{AS}	Address Set-up Time	50	-	30	-	20	-	10	-	nS
4	t_{MED2}	ME Delay Time 2	-	85	-	60	-	50	-	50	nS
5	t_{AH}	Address Hold Time	80	-	35	-	20	-	10	-	nS
6	t_{ED1}	Enable Delay Time 1	-	100	-	95	-	70	-	60	nS
7	t_{ED2}	Enable Delay Time 2	-	100	-	95	-	70	-	60	nS
8	t_{RDD1}	RD Delay Time 1 IOC=1	-	85	-	60	-	50	-	50	nS
		IOC=0	-	85	-	65	-	60	-	55	nS
9	t_{RDD2}	RD Delay Time 2	-	85	-	60	-	50	-	50	nS
10	t_{DRS}	Data Read Set-up Time	50	-	40	-	30	-	25	-	nS
11	t_{DRH}	Data Read Hold Time	0	-	0	-	0	-	0	-	nS
12	t_{WRD1}	WR Delay Time 1	-	90	-	65	-	60	-	50	nS
13	t_{WRD2}	WR Delay Time 2	-	90	-	80	-	60	-	50	nS
14	t_{WDD}	Write Data Delay Time	-	110	-	90	-	80	-	60	nS
15	t_{WDS}	Write Data Set-up Time	60	-	40	-	20	-	15	-	nS
16	t_{WRP}	WR Pulse Width	280	-	170	-	130	-	110	-	nS
17	t_{WDH}	Write Data Hold Time	60	-	40	-	15	-	10	-	nS
18	t_{WDZ}	Write Data Floating Delay Time	-	100	-	95	-	70	-	60	nS

* Preliminary Specification

SECTION 2

HD647180X AC Characteristics

Vcc = 5V +/- 10%, V_{ss} = 0 V, Ta= -20 to +75°C

#	Symbol	Item	HD647180X-4 Min	HD647180X-4 Max	HD647180X-6 Min	HD647180X-6 Max	HD647180X-8 Min	HD647180X-8 Max	
1	t_{AD}	Address Delay Time	-	110	-	90	-	80	nS
2	t_{MED1}	ME Delay Time 1	-	85	-	60	-	50	nS
3	t_{AS}	Address Set-up Time	50	-	30	-	20	-	nS
4	t_{MED2}	ME Delay Time 2	-	85	-	60	-	50	nS
5	t_{AH}	Address Hold Time	80	-	35	-	20	-	nS
6	t_{ED1}	Enable Delay Time 1	-	100	-	95	-	70	nS
7	t_{ED2}	Enable Delay Time 2	-	100	-	95	-	70	nS
8	t_{RDD1}	RD Delay Time 1 IOC=1	-	85	-	60	-	50	nS
		IOC=0	-	85	-	65	-	60	nS
9	t_{RDD2}	RD Delay Time 2	-	85	-	60	-	50	nS
10	t_{DRS}	Data Read Set-up Time	50	-	40	-	30	-	nS
11	t_{DRH}	Data Read Hold Time	0	-	0	-	0	-	nS
12	t_{WRD1}	WR Delay Time 1	-	90	-	65	-	60	nS
13	t_{WRD2}	WR Delay Time 2	-	90	-	80	-	60	nS
14	t_{WDD}	Write Data Delay Time	-	110	-	90	-	80	nS
15	t_{WDS}	Write Data Set-up Time	60	-	40	-	20	-	nS
16	t_{WRP}	WR Pulse Width	280	-	170	-	130	-	nS
17	t_{WDH}	Write Data Hold Time	60	-	40	-	15	-	nS
18	t_{WDZ}	Write Data Floating Delay Time	-	100	-	95	-	70	nS

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD64180 Family

Application Note

Task Switching Using the 64180 MMU

Marnie Mar

Introduction

Hitachi's HD64180 family of devices combine a wealth of on-chip I/O features with a Memory Management Unit (MMU) which allows access to 512K or 1Mbyte of memory 64Kbytes at a time. The MMU features can be used to demonstrate a method of switching between multiple tasks residing in different areas of physical memory.

This task switching method is examined here in a program which bases the requests for task switches on events caused by on-chip peripherals. The HD64180 family features used by this program are also described, and flow charts for each task and the final coded program are also presented.

Some familiarity with the HD64180 devices is assumed. For further information, please refer the HD64180 Microprocessor Hardware Manual, the HD647180X Microcontroller Hardware Manual, and the HD64180 Software Manual.

HD64180 Family Devices

The HD64180 features highlighted in this note are available on the HD64180R1, HD64180Z and HD647180X members of the family, and the code example shown will run on any of these devices. References to the HD64180 throughout this article refer to any of these three devices.

Thanks to the high integration of these devices, execution of the code written for this note depends mainly on on-chip features. Because of this, the code presented here will run on many systems based on these devices. A HD64180R1-based single board computer was used to test this example code. This board computer consists of the HD64180R1 interfaced to 64K of EPROM in physical locations 0h - FFFFh, 64K of SRAM in locations 10000h - 1FFFFh, and RS232 drivers and receivers connected for ASCI port 1 communications. A switch was connected to the HD64180's /IRQ1 input to allow simulation of external events.

Sidenote: Managing the 64180's Memory Space

The 64180 upgrades users of Z80-type processors by allowing access to a full 1 MByte of memory. However, to maintain code compatibility with the Z80, addresses in the 64180 instruction set must be specified in 16 bits, corresponding to the Z80's 64K address space.

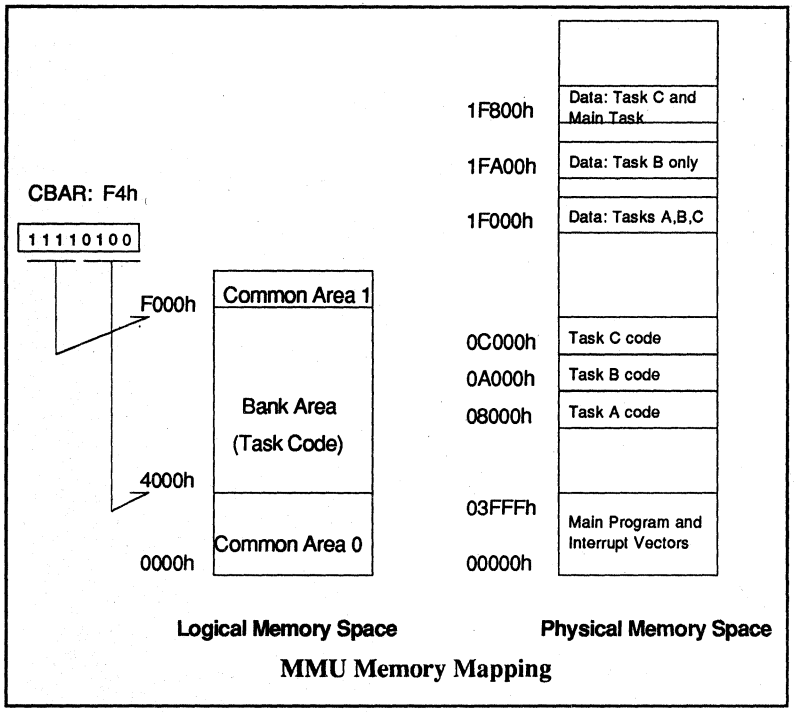
The 64180 bridges the gap between the Z80's 64K address space and a 1 Mbyte address space by providing an on chip Memory Management Unit. This Memory Manager automatically maps 16-bit logical addresses specified in instructions to locations in the 1 Mbyte physical address space.

The user has direct access to 64K of memory at any time, by using 16-bit logical addresses. This 64K can be broken into up to three sections, which will map to three different sections of physical memory. These three sections are named Common Area 0, Bank Area, and Common Area 1.

Division of this 64K of logical memory is determined by the user, who must program the Common/Bank Area Register (CBAR) with two 4-bit values which indicate the base of the Bank area and the base of Common Area 1 (See the Figure on page 2). These four bits specify the most significant bits of a 16-bit address which indicates the starting address of the area. Common Area 0 is always located at logical and physical address 0000h. Note that if the four bits specifying the Bank Base are set to 0000, only a Bank area and Common Area 1 are defined. If the CBAR is programmed with 00000000, then only a Common Area 1 of 64Kbytes is specified. The base of the Bank Area must always be programmed to be less than or equal to the base of Common Area 1.

Translation of logical to physical address is automatically performed using values programmed into the Common Base Register (CBR) and the Bank Base Register (BBR). These registers specify the upper eight bits of a 20 bit address (the remaining 12 least significant bits are 0) that is added to the logical address to obtain the physical address. See the Figure on page 3 for a translation example.

SECTION 2



MMU Memory Mapping

Application: Time Control Using the HD64180

The features of the HD64180 are used in this application to perform the following tasks:

- control an elapsed time clock in memory
- display the elapsed time on a display device (terminal) connected to ASCI port 1, using the DMAC to transfer the time information to the ASCI
- acknowledge the occurrence of an external event by incrementing a count of the number of external events and recording the time of the last occurrence
- display external event occurrence information to the display device
- accept and service interrupts based on these events

These tasks are performed in response to requests made to the HD64180 CPU by external and internal events. Three of these events (Programmable Reload Timer (PRT) channel 0 and PRT channel 1 count match, and external interrupt level 1) cause interrupts to occur, which are handled by the on-chip interrupt controller. The HD64180 polls for the fourth event, which is the receipt of a character by the SCI (RXRDY flag set).

In order to demonstrate the HD64180's ability to handle task switching using the MMU, each of the interrupt driven tasks is assigned to a different area of physical memory. Task switching is handled by reprogramming the MMU to access the area of memory where the task handler exists.

Memory Management

The HD64180 Memory Management Unit (MMU) allows access to 1Mbyte of address space or 512Kbyte of address space (for HD64180R1 DIP package, due to pin limitation). To maintain compatibility with Z80-type devices, at any point in time the device has access to 64Kbytes of memory. These 64Kbytes are addressed directly as a contiguous block of logical memory.

This logical memory actually maps into up to three different portions of physical memory. This mapping of logical addresses to physical addresses is accomplished using the three registers related to the MMU: the Common/Bank Area Register (CBAR), the Common Base Register (CBR), and the Bank Base Register (BBR).

Task A Code Execution Start Address:

BBR: 04h	0 0 0 0 0 1 0 0	
Logical Addresses:	+ 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	(4000h, logical base of code)
Physical Addresses:	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	= 8000h

Task A Data Storage Start Address:

CBR: 10h	0 0 0 1 0 0 0 0	
Logical Addresses:	+ 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0	(F000h, logical base of data)
Physical Addresses:	0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0	= 1F000h

64180 Family MMU Logical to Physical Address Calculation

These registers divide the logical address space into up to three areas, referred to as Common Area 0, Common Area 1, and Bank Area. The starting logical and physical address of Common Area 0 is always location 0000h. The starting logical addresses of Bank Area and Common Area 1 are specified by the CBAR. The physical starting address of Common Area 1 and the Bank Area are specified using the CBR and BBR, respectively.

Keep in mind that the actual physical starting address of the Bank Area and Common Area 1 is not determined solely from the CBR and BBR. These registers specify an eight-bit value that is added to most significant four bits of the logical address to provide the most significant 8 bits of the physical address. See the MMU Memory Mapping figure shown above.

180's features can be used to trigger task servicing

The HD64180's on-chip peripheral features provide the capabilities required to carry out this application. The two channels of programmable reload timer are each capable of counting down from a programmed value, and generating an interrupt when a count of zero is reached.

PRT0 is programmed to count down every .01 second. The interrupt service routine requested when PRT0's countdown interrupt occurs increments the elapsed time clock stored in memory.

PRT1 is used by this application to trigger the display of the elapsed time stored in memory. PRT1 is programmed to cause an interrupt every tenth of a second. Service of this interrupt involves initializing and initiating DMAC channel 1, which is used to display the elapsed time.

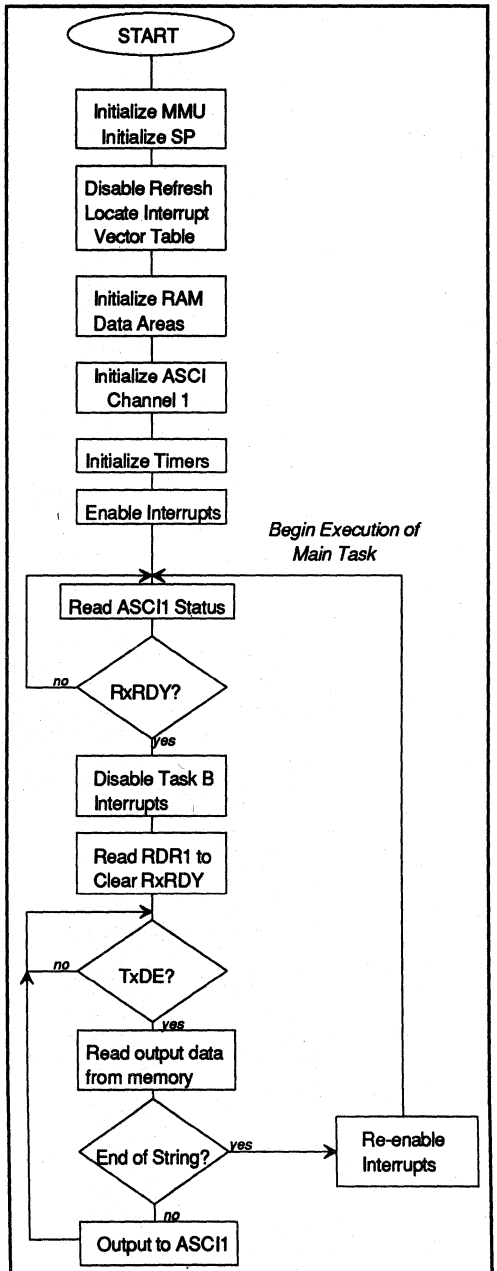
The full duplex ASCI channel serves two purposes for this application. The transmitter portion is used to transfer elapsed time (using the DMAC, as mentioned above) and external event information to the terminal screen. The RDRF bit of the receiv er, set when data is received, can be used to initiate a task (display of event data) whenever a key press occurs on a terminal connecter to the ASCI port.

Interrupt Handling

The HD64180's interrupt controller handles interrupts from four external sources and eight internal souces. These interrupts are maskable (except for the Non Maskable Interrupt, NMI, and the Undefined Op-code Trap, TRAP), and are prioritized with fixed priority. Upon receiving and accepting an interrupt, the controller in most cases determines the location of an interrupt service routine using a vector table that is programmably located in memory.

Two other modes of vector generation are available with external interrupt 0 (INT0), one fetches vector information from the bus, the other vectors directly to a fixed location.

The HD64180 prioritizes interrupts using fixed priority levels. The priorities of the interrupts used for this program are



Main Task Flow Diagram

(from highest to lowest): \IRQ1, PRT0 countdown, followed by PRT1 countdown. Note that for correct operation, PRT0 interrupt must always be accepted in order to maintain the elapsed time.

Program Design Notes

CPU Activity During Program Operation

Once the CPU and on-chip peripherals are initialized for this application, no further activity need be undertaken until an event (PRT time-out, external interrupt, receiver data ready) occurs. Since all these events can generate an interrupt to the CPU, the HD64180 could be placed in a sleep mode, to be awakened when an interrupt occurs.
Another option is to have the CPU poll for one of the events following initialization. This method was selected, and the least important task, in this case, reporting external event information, was chosen to be the task request polled. The remainder of the tasks interrupt this task whenever the event triggering them occurs.

Memory Configuration Considerations

The physical hardware to be used must be considered in designing the program. Since the program is to be placed in EPROM, data areas that are modified by the program must be singled out and placed in RAM. These data areas must also be initialized by the program to bring them to a known state.

Additionally, the program's use of logical and physical memory must be determined. Program initialization and the polling loop were selected for execution out of common area 1. Note that although this area starts at logical and physical address 0000h, the starting address of program code in this area must be selected carefully. The HD64180 expects a reset vector to be located at physical address 0000h, and the interrupt vector table may also be located starting in this area, depending on the programmed values for the IL (Interrupt vector Low) register.

The bank area was selected for execution of the tasks. Logically, this was set to 4000h. The MMU is reprogrammed by the task switcher to access each of the different tasks at this location, so the physical address of each task differs.

Finally, Common Area 1 was selected for use as the data areas accessed by the different tasks. Since only a small amount of data area is required, this logical area is programmed to start at F000h. This logical address space maps to physical addresses in RAM for this program.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

The program to carry out the tasks outlined above was written in sections. The main program, to execute out of Common Area 0, consists of initialization of the HD64180 core and the on-chip peripherals functions to be used, and initialization of the RAM areas to be used for display data and the elapsed time clock. Following initialization, this main program enters a routine that continually checks for a keystroke on the terminal keyboard, which is signalled by the RXD bit of the serial channel 1 being set to 1.

The second section of code performs task switching, and is entered each time one of the enabled interrupts occurred. This section reprograms the MMU and preserves register contents prior to initiating the execution of interrupt service routines. This section also handles return from exception, by restoring register contents.

The other code sections perform the different tasks signalled for by interrupts. Task A increments the clock every .01 second. Task B counts the PRT1 interrupts requested and initializes the DMAC and enables the DMAC on the appropriate counts.

Execution of Task C results from an external event indicated by interrupt \IRQ1 going active. Task C increments a count of external events, and record the current value of the elapsed time clock.

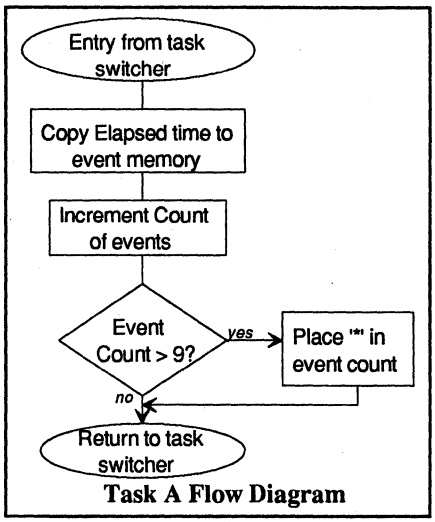
The resulting program is shown in the Appendix.

Initialization

At power-on, the HD64180 control registers take on default values that allow the device to operate according to default conditions. If deviation from these defaults is desired, the individual registers must be programmed with new initialization values. The stack pointer must be programmed to a valid address for stack operations to be performed correctly.

For this program, the MMU control registers were programmed to divide the logical space into three areas, accessing three different sections of physical memory. Since SRAMs were used, the refresh capability of HD64180 was turned off by writing to the Refresh Control Register (RCR).

The Interrupt Vector Table and I/O Registers are relocatable by programming their associated control registers. The Interrupt Vector Table was located at 20H by programming the IL register, so that the reset vector located at 00H would not overlay the vector entry for /INT1. Note that the interrupt vector table was initialized in EPROM using the capabilities of the assembler.



Task A Flow Diagram

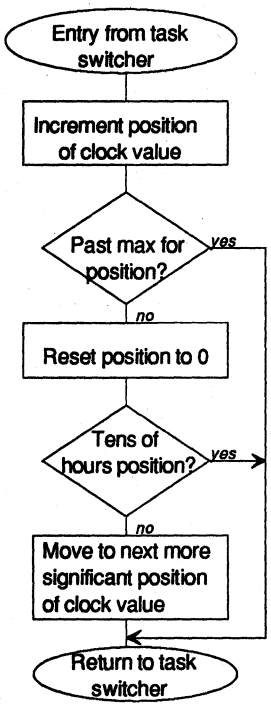
For this program, the default reset value for the I/O Registers base address was taken. Similarly, the Operation Mode Control Register (OMCR) available in the HD64180Z and HD647180X was not reprogrammed from its' default value.

On-chip Peripheral Initialization

ASCI channel 1 and the PRT channels are initialized by writing the appropriate values into the associated control registers. PRT0 is programmed for a count based on the CPU clock and a start count and a reload count of h'1200, which results in an interrupt once every .01 seconds with a CPU clock of 9.216 MHz. During service of this interrupt, the elapsed time clock is updated.

Note that DMAC channel 0 must be used in order to cause DMA transfer to or from the on-chip serial channel. This channel is connected to the serial channels internally, allowing these transfers to occur. For this transfer to operate correctly, DMA channel 0 must be used, bits 0-2 of the DMA Destination Address Register DAR0B must be programmed for the correct destination, and the channel must be programmed to respond to edge-sense requests using the DMS (DMA request Sense) bits of the DMA/WAIT Control Register (DCNTL).

Values for ASCI baud rate (in CNTRLB1) and PRT data and reload registers were determined based on the CPU clock



Task B Flow Diagram

input crystal frequency of 9.216MHz.

Interrupt Enable

Once the HD64180 has been initialized for operation, interrupts can be enabled, and the main task can be initiated. Enabling interrupts is a two level process.

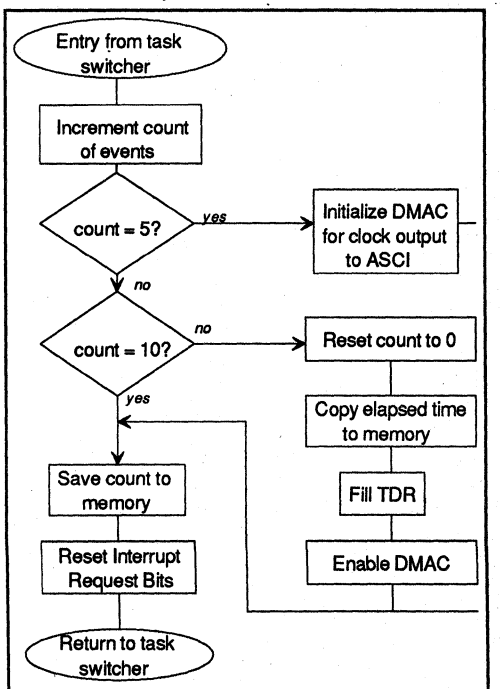
First, individual bits in control registers must be set to enable interrupts. For the PRT channels, the Timer Interrupt Enable (TIE) bits of the Timer Con. Register (TCR) must be set. At the same time, the Timer countdown can be started by setting the Timer Down Count Enable (TDE) bits. To enable INT1 and disable INT0, which is enabled at reset, the appropriate bits of the ITC must be programmed.

Second, the Interrupt Enable Flags (IEF) of the interrupt controller must be set. This is accomplished by executing the EI instruction.

Task switching using the MMU

As mentioned previously, the HD64180's logical address space is divided into the three areas allowed by the MMU. Common Area 0 is used for RAM initialization data, interrupt vectors, initialization code, and the main task execution. Common Area 1 is used for shared RAM space, which stores the elapsed time clock data and other data that is manipulated by the tasks. The Bank area is used for executing the subtask code segments.

Since the subtask execution code is always mapped to the logical addresses of the Bank area, task switching can be accomplished by storing the existing values of the MMU registers to the stack, reprogramming them to access the physical locations of the new task and its' associated data RAM area, and jumping to the first location of the bank area. With this scheme, each task can be located in a different area



Task C Flow Diagram

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

of physical memory, yet still be accessed using the same addresses in logical memory.

In more complex applications, this method of switching tasks allows easy handling of the non-contiguous memory space. Although the entire code for this application could have fit in the directly addressable 64Kbyte memory space, more complex code would have to be broken into sections that would fit within the logical address space.

In this example, all tasks accessed the same data area, and the CBR was not reprogrammed during task switching. Note that by reprogramming the CBR register, physical data area locations can be switched at the same time as physical code areas.

Coding the Tasks

The main task operates continuously, and is interrupted by the other tasks based on events. This tasks consists simply of polling the RDRF bit of the ASCI1 status register to detect when a key has been pressed on the terminal. When a key input is detected (RDRF set), a routine is entered which displays the current external event information stored in RAM by transferring the data character by character through ASCI channel 1.

Task A is executed in response to the PRT0 countdown interrupt. This task increments the elapsed time clock that is stored in RAM. Note that Task B interrupt requests are disabled during the main task execution, to prevent time data from being updated to the screen during the event display. However, note also that the elapsed time and external events can be accepted.

Task B is executed in response to the PRT1 countdown interrupt. The DMA controller is initiated as part of this task for data transfer to the ASCI1 transmit data register. When the Task B interrupt count reaches 10, the elapsed time clock value is copied to the memory location indicated as the DMA source address, and DMA transfer for channel 0 is enabled. This causes the current elapsed time to be display to the terminal through ASCI1. Copying the elapsed time clock value during Task B execution eliminates the possibility of a PRT0 interrupt changing the time during output to the terminal. Since DMA transfer continues beyond the time that Task B ends and interrupts are reinabled, it is possible that Task A could increment the elapsed time clock in memory before the entire DMA transfer completed.

Task C interrupts are serviced in response to an INT1 interrupt, which signals an external event. This routine simply copies the current elapsed time to a buffer in memory to record the time of the most recent event, and increments the event counter. To simplify programming, the event counter can increment to a maximum of nine. This event information is displayed by the main task in response to a key entry. If more than nine external events have occurred, an "*" is displayed in place of the number of external events.

Assembling Code for MMU Task Switching

Since the logical and physical addresses differ for the sections of code and data that are located outside of Common Area 0, care must be taken when specifying code addresses during assembly. Since the processor uses the logical addresses for execution, code should be located at the logical address space during assembly. For this example, all task code should be ORGed or located at h'4000, the logical address for task execution.

When downloading code to a development system or EPROMs, an offset should be specified so that the code is physically loaded to the correct address. For instance, Task C would be ORGed to location 4000h. When downloading to development system memory, an offset of 6000h should be specified so that the code would be loaded to the correct physical address of A000h, where the MMU will expect it to be.

Summary

This application identifies how the HD64180R, HD64180Z and HD647180X devices' on-chip features can be used in a multi-tasking operation. Reprogramming the MMU to initiate task switching allows for a simple method of allocating physical address space to executable code. The on-chip Programmable Reload Timers can generate interrupts which, during interrupt service accumulate elapsed time. The on-chip Serial Communications Interface controls transfer of data to a terminal, either using DMA or program transfer of data to the SCI transfer registers.

APPENDIX : Main Task Code Listing

```
;*********************************************************
;
;     64180 Application Program
;             Elapsed time clock and external event logger
;
;     Main Program:  (in common area 0)
;             - Initializes the 64180 as follows:
;                     Interrupt vectors are loaded to h'20
;                     MMU is set up so that:
;                             Common Area 0 is fixed at h'0000
;                             Bank Area is at logical 4000h
;                             Common Area 1 is at logical F000h
;                     PRT0 - generates an interrupt every .01 sec
;                     PRT1 - generates an interrupt every .1 sec
;                     ASCI0 - displays elapsed time clock
;                     DMAC0 - handles transfer of data to terminal
;             - Enters a loop polling for RxD condition
;
;     RxD Handler: (follows main program)
;             - Initializes pointers for display of event data
;             - Polls for TDRE, displays event data
;             - Returns to loop polling for RxD condition
;             - PRT1 interrupt disabled during this routine
;
;     Task Switcher:  (in common area 0)
;             - stores current MMU context
;             - loads MMU values for next task
;             - executes task code
;             - executes return from interrupt
;
;     Task A: (in bank area, physical 8000h)
;             - Results from PRT0 timer interrupt (every .01s)
;             - Increments elapsed time clock in common area 1
;             - Returns to Task Switcher
;
;     Task B: (in bank area, physical A000h)
;             - Results from PRT1 timer interrupt (every .1s)
;             - Increments count of task B interrupts
;             - If count = 5, then
;                     initialize DMAC0 for displaying clock
;             - If count = 10, then
;                     reads and stores elapsed time clock
;                     starts DMAC0 to display time
;                     resets count
;             - Returns to Task Switcher
;
;     Task C: (in bank area, physical C000h)
;             - Results from external event (INT1)
;             - Records Elapsed Time Clock value (time of event)
;             - Increments event count
;             - Returns to Task Switcher
```

APPENDIX: Main Task Code Listing (continued)

```
                        ;
000000                  PAGE
000000                  CPU   "64180.TBL"
000000                  INCL  "INTIO.LIB"
                        ;
                        ;  64180 Internal I/O register equates
                        ;
                        ;  Copyright 1988, SOFTAID, INC., Columbia, MD.
                        ;
                        ;  Note: BASE must be defined as the internal I/O
                        ;        relocation offset of 00h, 40h, 80h or 0C0h
                        ;        and register ICR must be set appropriately.
                        ;
00000000     =          BASE:        EQU    00h

00000000     =          CNTLA0:      EQU    00H + BASE     ; ASCI CONTROL REGISTER A CH 0
00000001     =          CNTLA1:      EQU    01H + BASE     ; ASCI CONTROL REGISTER A CH 1
00000002     =          CNTLB0:      EQU    02H + BASE     ; ASCI CONTROL REGISTER B CH 0
00000003     =          CNTLB1:      EQU    03H + BASE     ; ASCI CONTROL REGISTER B CH 1
00000004     =          STAT0:       EQU    04H + BASE     ; ASCI STATUS REGISTER CH 0
00000005     =          STAT1:       EQU    05H + BASE     ; ASCI STATUS REGISTER CH 1
00000006     =          TDR0:        EQU    06H + BASE     ; ASCI TRANSMIT DATA REGISTER CH 0
00000007     =          TDR1:        EQU    07H + BASE     ; ASCI TRANSMIT DATA REGISTER CH 1
00000008     =          RDR0:        EQU    08H + BASE     ; ASCI RECEIVE DATA REGISTER CH 0
00000009     =          RDR1:        EQU    09H + BASE     ; ASCI RECEIVE DATA REGISTER CH 1
0000000A     =          CNTR:        EQU    0AH + BASE     ; CSI/O CONTROL REGISTER
0000000B     =          TRDR:        EQU    0BH + BASE     ; CSI/O TRANSMIT/RECEIVE DATA REGISTER
0000000C     =          TMDR0L:      EQU    0CH + BASE     ; TIMER DATA REGISTER CH 0L
0000000D     =          TMDR0H:      EQU    0DH + BASE     ; TIMER DATA REGISTER CH 0H
0000000E     =          RLDR0L:      EQU    0EH + BASE     ; RELOAD REGISTER CH 0L
0000000F     =          RLDR0H:      EQU    0FH + BASE     ; RELOAD REGISTER CH 0H
00000010     =          TCR:         EQU    10H + BASE     ; TIMER CONTROL REGISTER
00000014     =          TMDR1L:      EQU    14H + BASE     ; TIMER DATA REGISTER CH 1L
00000015     =          TMDR1H:      EQU    15H + BASE     ; TIMER DATA REGISTER CH 1H
00000016     =          RLDR1L:      EQU    16H + BASE     ; RELOAD REGISTER CH 1L
00000017     =          RLDR1H:      EQU    18H + BASE     ; FREE RUNNING COUNTER
00000020     =          SAR0L:       EQU    20H + BASE     ; DMA SOURCE ADDRESS REGISTER CH 0L
00000021     =          SAR0H:       EQU    21H + BASE     ; DMA SOURCE ADDRESS REGISTER CH 0H
00000022     =          SAR0B:       EQU    22H + BASE     ; DMA SOURCE ADDRESS REGISTER CH 0B
00000023     =          DAR0L:       EQU    23H + BASE     ; DMA DESTINATION ADDRESS REGISTER CH 0L
00000024     =          DAR0H:       EQU    24H + BASE     ; DMA DESTINATION ADDRESS REGISTER CH 0H
00000025     =          DAR0B:       EQU    25H + BASE     ; DMA DESTINATION ADDRESS REGISTER CH 0B
00000026     =          BCR0L:       EQU    26H + BASE     ; DMA BYTE COUNT REGISTER CH 0L
00000027     =          BCR0H:       EQU    27H + BASE     ; DMA BYTE COUNT REGISTER CH 0H
00000028     =          MAR1L:       EQU    28H + BASE     ; DMA MEMORY ADDRESS REGISTER CH 1L
00000029     =          MAR1H:       EQU    29H + BASE     ; DMA MEMORY ADDRESS REGISTER CH 1H
0000002A     =          MAR1B:       EQU    2AH + BASE     ; DMA MEMORY ADDRESS REGISTER CH 1B
0000002B     =          IAR1L:       EQU    2BH + BASE     ; DMA I/O ADDRESS REGISTER CH 1L
0000002C     =          IAR1H:       EQU    2CH + BASE     ; DMA I/O ADDRESS REGISTER CH 1H
0000002E     =          BCR1L:       EQU    2EH + BASE     ; DMA BYTE COUNT REGISTER CH 1L
```

APPENDIX: Main Task Code Listing (continued)

```
0000002F   =   BCR1H:       EQU   2FH + BASE    ; DMA BYTE COUNT REGISTER CH 1H
00000030   =   DSTAT:       EQU   30H + BASE    ; DMA STATUS REGISTER
00000031   =   DMODE:       EQU   31H + BASE    ; DMA MODE REGISTER
00000032   =   DCNTL:       EQU   32H + BASE    ; DMA/WAIT CONTROL REGISTER
00000033   =   IL:          EQU   33H + BASE    ; IL REGISTER(INTERRUPT VECTOR LOW REG)
00000034   =   ITC:         EQU   34H + BASE    ; INT/TRAP CONTROL REGISTER
00000036   =   RCR:         EQU   36H + BASE    ; REFRESH CONTROL REGISTER
00000038   =   CBR:         EQU   38H + BASE    ; MMU COMMON BASE REGISTER
00000039   =   BBR:         EQU   39H + BASE    ; MMU BANK BASE REGISTER
0000003A   =   CBAR:        EQU   3AH + BASE    ; MMU COMMON/BANK AREA REGISTER
0000003F   =   ICR:         EQU   3FH + BASE    ; I/O CONTROL REGISTER
               ;
               ;  Programmable reload timer enable equates
               ;
00000001   =   PRT_COUNT:   EQU   01H            ; TIMER COUNT ENABLE
00000010   =   PRT_INT:     EQU   10H            ; TIMER INTERRUPT ENABLE
               ;
               ;  ASCI initialization and status equates
               ;
00000000   =   ASCI_38400:  EQU   00H            ; BAUD RATE
00000001   =   ASCI_19200:  EQU   01H            ; BAUD RATE
00000002   =   ASCI_9600:   EQU   02H            ; BAUD RATE
00000003   =   ASCI_4800:   EQU   03H            ; BAUD RATE
00000004   =   ASCI_2400:   EQU   04H            ; BAUD RATE
00000005   =   ASCI_1200:   EQU   05H            ; BAUD RATE
00000006   =   ASCI_600:    EQU   06H            ; BAUD RATE
               ;
00000004   =   ASCI_8BITS:  EQU   04H            ; DATA BITS
00000002   =   ASCI_PARITY: EQU   02H            ; PARITY ENABLE
00000001   =   ASCI_2STOP:  EQU   01H            ; 2 STOP BITS
               ;
00000002   =   ASCI_TDRE:   EQU   02H            ; TDRE BIT MASK
00000080   =   ASCI_RDRF:   EQU   80H            ; RDRF BIT MASK
  000000                    HOF   "INT16"

00000000   =   NULL:        EQU   00h
0000000D   =   CR:          EQU   0Dh
0000000A   =   LF:          EQU   0Ah
00000000   =   T1LOW:       EQU   000h
000000B4   =   T1HI:        EQU   0B4h
00000000   =   T0LOW:       EQU   00h
00000012   =   T0HI:        EQU   012h
00004000   =   BANK:        EQU   04000h ;bank logical location for task
                            ; handling routines
```

APPENDIX: Main Task Code Listing (continued)

```
                    ;
                    ;*********************************************************
                    ;
                    ;     Data Storage
                    ;             - System Data
                    ;
000000                              ORG  00h              ;initialize reset information
000000 C30001                       JP      CLSTART

000020                    ORG       020h           ;interrupt vector table
000020 B901         INT1: DWL       TASKC   ;external event interrupt
000022 DE01         INT2: DWL       INTEND
000024 AB01         PRT0: DWL       TASKA   ;timer 0 countdown interrupt
000026 B201         PRT1: DWL       TASKB   ;timer 1 countdown interrupt
000028 DE01         DMA0: DWL       INTEND
00002A DE01         DMA1: DWL       INTEND
00002C DE01         CSIO: DWL       INTEND
00002E DE01         ASCI0:          DWL     INTEND
000030 DE01         ASCI1:          DWL     INTEND

                    ;
                    ;     Preliminary Storage area for task data - data in this
                    ;             area (ROM) is moved to RAM during initialization
                    ;
000040                              ORG     040h            ;storage area (Common 1)
                                                            ;used by all tasks
000040 30303A30303A30ELCLK:         DFB     "00:00:00.0"
00004A 300D                         DFB     "0",CR
                    ;
                                    ;storage area X (Common 1)
                                    ;used by Task C only
00004C 0D0A0A       EVDATA:         DFB  CR,LF,LF
00004F 30                           DFB     30h             ;storage for count of events (<= 9)
000050 206576656E7473                DFB     " events occurred",CR,LF
000062 204C6173742065                DFB     " Last event occurred at "
00007A 30303A30303A30                DFB     "00:00:00.00",CR,LF,LF
000088 00                           DFB     00h             ;end of data
                    ;
                    ;                                       ;storage area Y (Common 1)
                    ;                                       ;used by Task B
000090                              ORG     090h
000090 00           CTRDATA:        DFB     0h
000091 30303A30303A30                DFB     "00:00:00",CR
```

SECTION 2

APPENDIX: Main Task Code Listing (continued)

```
                        ;
                        ;     RAM DATA TABLE:  the following sets up logical
                        ;             addresses to correspond with those used in
                        ;             tasks.  These addresses map to physical RAM,
                        ;             which must be intialized by the main program
                        ;
00F000                              ORG     0F000h          ;storage area (Common 1)
                                                            ;used by all tasks
00F000 30303A30303A30ET_CLK:        DFB     "00:00:00.0"
00F00A 300D             ET_CLKend:  DFB     "0",CR
                        ;
00F800                              ORG     0F800h
                                    ;storage area X (Common 1)
                                    ;used by Task C only
00F800 0D0A0A           EV_START:   DFB  CR,LF,LF
00F803 30               EVNT_CT:    DFB     30h             ;storage for count of events (<= 9)
00F804 206576656E7473               DFB     " events occurred",CR,LF
00F816 204C6173742065               DFB     " Last event occurred at "
00F82E 30303A30303A30LAST_EV:       DFB     "00:00:00.00",CR,LF,LF
00F83C 00                           DFB     00h             ;end of data
                        ;
00FA00                              ORG     0FA00h
                        ;                                   ;storage area Y (Common 1)
                        ;                                   ;used by Task B
00FA00 00               CTR:        DFB     0h
00FA01 30303A30303A30D_TIME:        DFB     "00:00:00",CR
                        ;
00FA0A      PAGE
                       ;*************************************************************
                        ;
                        ;     Initialization routines for 64180
                        ;
000100                      ORG     100h
                        ;
                        ;     Initialize MMU: common 0 at location 0000h (logical)
                        ;                           common 1 at location F000h (logical)
                        ;                             bank at location 4000h (logical)
                        ;
000100 3EF4             CLSTART:    LD      A,0F4h  ;Common 1 at F000h, Bank at 4000h
000102 ED393A                       OUT0    (CBAR),A
000105 3E10                         LD      A,10h   ;Common1 physical at 1F000h
000107 ED3938                       OUT0    (CBR),A
00010A DD21FFFF                     LD      IX,0FFFFh       ;stack at high memory
00010E DDF9                         LD      SP,IX
000110 3E00                         LD      A,00h   ;turn off refresh insertion
000112 ED3936                       OUT0    (RCR),A
                        ;
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDIX: Main Task Code Listing (continued)

```
000115      PAGE
                    ;*********************************************************
                    ;
                    ;     Initialize interrupt vector registers, table

000115 3E20                 LD      A,20h          ;use low memory page
000117 ED3933               OUT0 (IL),A            ;I initialized to 00h on reset
                    ;
                    ;*********************************************************
                    ;
                    ;     Initialize common RAM areas
                    ;
00011A 214000               LD      HL,ELCLK               ;data to be moved to RAM
00011D 1100F0               LD      DE,ET_CLK              ;data area to be loaded
000120 010C00               LD      BC,12                  ;count of characters
000123 EDB0                 LDIR
                    ;
000125 214C00               LD      HL,EVDATA              ;data to be moved to RAM
000128 1100F8               LD      DE,EV_START    ;data area to be loaded
00012B 013D00               LD      BC,61          ;count of characters
00012E EDB0                 LDIR
                    ;
000130 219000               LD      HL,CTRDATA     ;data to be moved to RAM
000133 1100FA               LD      DE,CTR         ;data area to be loaded
000136 010A00               LD      BC,10          ;count of characters
000139 EDB0                 LDIR

00013B      PAGE
                    ;*********************************************************
                    ;
                    ;     Initialize ASCI1 for serial communications
                    ;           9600 baud, 7 data bits, 1 stop bits, no parity
                    ;
0001    70          SCINIT:     LD      A,70h          ;RE,TE,7,N,1
0001    3901                    OUT0    (CNTLA1),A             ;
0001    21                      LD      A,21h          ;9600 at 9.216MHz
00014Z  ED3903                  OUT0    (CNTLB1),A
000145  3E00                    LD      A,00h
000147  ED3905                  OUT0    (STAT1),A              ;disable receiver interrupts
                    ;
                    ;*********************************************************
                    ;
                    ;     Initialize PRT0 for interrupt every .01 second
                    ;     Interrupts not yet enabled

00014A 011000       ^0INIT:     LD      BC,TCR         ;load address of register
00014D 3E00                     LD      A,T0LOW
00014F ED390C                   OUT0    (TMDR0L),A     ;timer data register 0L
000152 ED390E                   OUT0    (RLDR0L),A     ;reload data register 0L
000155 3E12                     LD      A,T0HI
```

APPENDIX: Main Task Code Listing (continued)

```
000157 ED390D                    OUT0    (TMDR0H),A      ;timer data register 0H
00015A ED390F                    OUT0    (RLDR0H),A      ;reload data register 0H
                        ;
00015D      PAGE
                          ;*******************************************************
                        ;
                        ;       Initialize PRT1 for interrupt every .1 second
                        ;       Interrupts not yet enabled

00015D 3E00                      LD      A,T1LOW
00015F ED3914                    OUT0    (TMDR1L),A      ;timer data register 1L
000162 ED3916                    OUT0    (RLDR1L),A      ;reload data register 1L
000165 3EB4                      LD      A,T1HI
000167 ED3915                    OUT0    (TMDR1H),A      ;timer data register 1H
00016A ED3917                    OUT0    (RLDR1H),A      ;reload data register 1H
                        ;
00016D      PAGE
                        ;*******************************************************
                        ;
                        ;       Enable all Interrupts, start clock, and wait for RxRDY
                        ;
00016D FB               EI
00016E 3E02             LD      A,02h
000170 ED3934                    OUT0    (ITC),A         ;enable INT1 interrupts
000173 3E33             LD      A,33h
000175 ED3910                    OUT0    (TCR),A         ;start PRT timers
                        ;

                          ;*******************************************************
                        ;
                        ;       MAIN TASK:
                        ;               Loop here until RxRDY received from ASCI0

                        ;
000178 010500           KEYLOOP:         LD      BC,STAT1                ;ASCI status register
00017B ED7480                            TSTIO   80h                     ;tests if RDRF
00017E 28F8                              JR      Z,KEYLOOP
                        ;
                        ;       Reach here when RxRDY received
                        ;
000180 ED3809                            IN0     A,(RDR1)                ;read RDR to clear RDRF
000183 3E13                              LD      A,13h
000185 ED3910                            OUT0    (TCR),A                 ;disable Task B irq
000188 3E20                              LD      A,20h
00018A ED3930                            OUT0    (DSTAT),A               ;disable DMA in progress

00018D DD2100F8         KEYIN:           LD      IX,0F800h               ;
000191 ED7402           OUTLP:           TSTIO   02h                     ;tests if TDRE
000194 28FB                              JR      Z,OUTLP
000196 DD7E00                            LD      A,(IX+0)                ;get memory to output
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDIX: Main Task Code Listing (continued)

```
000199 FE00                    CP      NULL                    ;test if end of string
00019B 2807                    JR      Z,KEYEND
00019D ED3907                  OUT0    (TDR1),A                ;output to asci port 1
0001A0 DD23                    INC     IX                      ;increment memory pointer
0001A2 18ED                    JR      OUTLP

0001A4 3E33          KEYEND:   LD      A,33h
0001A6 ED3910                  OUT0    (TCR),A                 ;enable timer irqs
0001A9 18CD                    JR      KEYLOOP
                     ;
                       ;*************************************************************
                     ;     Task Switcher - interrupt service
                     ;             entry point to this section depends on IRQ recv'd
                     ;
                     ;     enter here for PRT0 irq
0001AB C5            TASKA:        PUSH BC
0001AC 0E04                LD      C,04h           ;task A at physical 8000h
0001AE 0610                LD      B,10h           ;common 1 at physical 10000h
0001B0 180C                JR      SWITCH
                     ;
                     ;     enter here for PRT1 irq
0001B2 C5            TASKB:        PUSH BC
0001B3 0E06                LD      C,06h           ;task B at physical A000h
0001B5 0610                LD      B,10h           ;common 1 at physical 10000h
0001B7 1805                JR      SWITCH
                     ;
                     ;     enter here for IRQ1 request
0001B9 C5            TASKC:        PUSH BC
0001BA 0E08                LD      C,08h           ;task C at physical C000h
0001BC 0610                LD      B,10h           ;common 1 at physical 10000h
                     ;
                     ;     Task switches handled here
0001BE ED1838        SWITCH: IN0   E,(CBR) ;read common1 base address
0001C1 ED1039              IN0     D,(BBR)         ;read bank base address
0001C4 D5                  PUSH    DE              ;save to stack

0001C5 ED0939              OUT0    (BBR),C         ;initialize BBR
0001C8 ED0138              OUT0    (CBR),B         ;initialize CBR

0001CB E5                  PUSH    HL
0001CC F5                  PUSH    AF              ;save context prior to int svc
0001CD DDE5                PUSH    IX
                     ;
                     ;     Memory switched, call task handler routine
                     ;
0001CF CD0040              CALL    BANK            ;switch to task in bank area
```

SECTION 2

APPENDIX: Task Switcher Code Listing

```
                    ;
                    ;     Task complete, perform clean-up for return from task
                    ;
0001D2 DDE1         RETURN:     POP     IX
0001D4 F1                 POP     AF
0001D5 E1                 POP     HL
0001D6 D1                 POP     DE
0001D7 ED1139             OUT0    (BBR),D        ;restore MMU
0001DA ED1938             OUT0    (CBR),E        ;restore MMU
0001DD C1                 POP     BC
                    ;
0001DE FB           INTEND: EI
0001DF ED4D               RETI
000000                    END
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDIX: Task A Code Listing

```
                    ;*********************************************************
                    ;
                    ;     TASK A ROUTINES:  Interrupt service for timer 0
                    ;             - Causes elapsed time clock to increment by
                    ;               one hundredth of a second
                    ;
                    ;*********************************************************
                    ;
                    ;     Task A Code: executed upon receipt of PRT0 interrupt
                    ;
000000              CPU     "64180.TBL"
000000              INCL    "INTIO.LIB"
                    ;
                    ;     Use same table of equates as for Main Task
                    ;*********************************************************
                    ;
                    ;     RAM area loaded with data by main program
                    ;
00F000                      ORG     0F000h          ;storage area (Common 1)

00F000 30303A30303A30ET_CLK:        DFB     "00:00:00.0"
00F00A 300D         ET_CLKend:DFB "0",CR

00F00C      PAGE
                    ;*********************************************************
                    ;
                    ;     Task A: increment elapsed time clock
                    ;

004000                      ORG     4000h                   ;org at logical address

004000 DD210AF0     CLK_CHG:        LD      IX,ET_CLKend    ;IX contains address of clock
004004 3E01                 LD      A,01h
004006 DD8600               ADD     A,(IX+0)                ;increment
004009 FE39                 CP      39h                     ;if 3A, then update
00400B 2078                 JR      NZ,UPDATE
00400D 3E30                 LD      A,30h                   ;reset to 0
00400F DD7700               LD      (IX+0),A                ;prepare to incr preceeding
004012 DD2B                 DEC     IX
004014 3E01                 LD      A,01h
004016 DD8600               ADD     A,(IX+0)                ;increment
004019 FE39                 CP      39h                     ;if 3A, then update
00401B 2068                 JR      NZ,UPDATE
00401D 3E30                 LD      A,30h                   ;reset to 0
00401F DD7700               LD      (IX+0),A                ;prepare to incr preceeding
004022 DD2B                 DEC     IX
004024 DD2B                 DEC     IX                      ;skip period
004026 3E01                 LD      A,01h
```

SECTION 2

APPENDIX: Task A Code Listing (continued)

```
                    ;       update seconds

004028 DD8600               ADD     A,(IX+0)            ;increment
00402B FE39                 CP      39h                 ;if 3A, then update
00402D 2056                 JR      NZ,UPDATE
00402F 3E30                 LD      A,30h               ;reset to 0
004031 DD7700               LD      (IX+0),A            ;prepare to incr preceeding
004034 DD2B                 DEC     IX
004036 3E01                 LD      A,01h
004038 DD8600               ADD     A,(IX+0)            ;increment
00403B FE36                 CP      36h                 ;if 37 (>60), then update
00403D 2046                 JR      NZ,UPDATE
00403F 3E30                 LD      A,30h               ;reset to 0
004041 DD7700               LD      (IX+0),A            ;prepare to incr preceeding
004044 DD2B                 DEC     IX
004046 DD2B                 DEC     IX                  ;skip period
004048 3E01                 LD      A,01h

                    ;       update minutes

00404A DD8600               ADD     A,(IX+0)            ;increment
00404D FE39                 CP      39h                 ;if 3A, then update
00404F 2034                 JR      NZ,UPDATE
004051 3E30                 LD      A,30h               ;reset to 0
004053 DD7700               LD      (IX+0),A            ;prepare to incr preceeding
004056 DD2B                 DEC     IX
004058 3E01                 LD      A,01h
00405A DD8600               ADD     A,(IX+0)            ;increment
00405D FE36                 CP      36h                 ;if 37 (>60), then update
00405F 2024                 JR      NZ,UPDATE
004061 3E30                 LD      A,30h               ;reset to 0
004063 DD7700               LD      (IX+0),A            ;prepare to incr preceeding
004066 DD2B                 DEC     IX
004068 DD2B                 DEC     IX                  ;skip period
00406A 3E01                 LD      A,01h

                    ;       update hours

00406C DD8600               ADD     A,(IX+0)        ;increment
00406F FE39                 CP      39h             ;if 3A, then update
004071 2012                 JR      NZ,UPDATE
004073 3E30                 LD      A,30h           ;reset to 0
004075 DD7700               LD      (IX+0),A        ;prepare to incr preceeding
004078 DD2B                 DEC     IX
00407A 3E01                 LD      A,01h
00407C DD8600               ADD     A,(IX+0)        ;increment
00407F FE39                 CP      39h             ;if >100, then reset to 0
004081 2002                 JR      NZ,UPDATE
004083 3E30                 LD      A,30h           ;reset to 0
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDIX: Task A Code Listing (continued)

```
004085 DD7700        UPDATE:      LD      (IX+0),A        ;prepare to incr preceeding
004088 ED3810              IN0     A,(TCR)          ;read timer registers to reset
00408B ED380C              IN0  A,(TMDR0L)          ;   timer interrupt

00408E C9                  RET                      ;return from task routine
                                              ; to task switcher
                     ;*********************************************************
000000               END
```

APPENDIX: Task B Code Listing

```
                    ;*******************************************************
                ;
                ;       Task B - Executed upon receipt of PRT1 interrupt
                ;
000000                  CPU     "64180.TBL"
000000                  INCL    "INTIO.LIB"
                ;       Include same table of equates used for Main Task
                ;
                    ;*******************************************************
                ;
                ;       Data Storage
                ;               - System Data
                ;
00F000                          ORG     0F000h  ;storage area (Common 1)
                                ;used by all tasks
00F000 30303A30303A30ET_CLK:    DFB     "00:00:00.0"
00F00A 300D     ET_CLKend:      DFB     "0",CR

00FA00                          ORG  0FA00h     ;storage area Y (Common 1)
00FA00 00       CTR:            DFB  0h         ;used by Task B
00FA01 30303A30303A30D_TIME:    DFB     "00:00:00",CR
                ;*******************************************************
                ;
                ;       Task B - Prepare for and perform DMA to ASCI1

004000                  ORG     4000h                   ;org at logical address

004000 DD2100FA         CLK_DSP:        LD      IX,CTR          ;IX contains address of count
004004 3E01             LD      A,01h
004006 DD8600           ADD     A,(IX+0)                ;increment
004009 FE05             CP      05h                     ;compare to 5
00400B 2005             JR      NZ,SEC_CHK
                ;
                ;       reach here if count equals 5
                ;
00400D CD3740           CALL    DMINIT                  ;if = 5, initialize DMAC
004010 181B             JR      DSP_END
004012 FE0A             SEC_CHK:        CP      0Ah             ;compare to 10
004014 2017             JR      NZ,DSP_END
                ;
                ;       reach here if count equals 10
                ;
004016 3E00             LD      A,00h                   ;reset count value in A
                ;
004018 2100F0           LD      HL,ET_CLK
00401B 1101FA           LD      DE,D_TIME
00401E 010800           LD      BC,8
004021 EDB0             LDIR                            ;Copy current clock
                ;
```

APPENDIX: Task B Code Listing (continued)

```
004023 0620              LD      B,20h                   ;output a space
004025 ED0107            OUT0    (TDR1),B                ;prior to starting ASCI ch 0.
004028 0640              LD      B,40h
00402A ED0130            OUT0    (DSTAT),B       ;start DMAC ch 0
                    ;
                    ;    exit interrupt service here
                    ;
00402D DD7700       DSP_END:     LD      (IX+0),A        ;restore counter to memory
004030 ED3810            IN0     A,(TCR)         ;read registers to clear
004033 ED3814            IN0     A,(TMDR1L)      ;    interrupt request
004036 C9                RET                     ;return from task B
                    ;
                     ;****************************
                    ;
                    ;            DMAC Channel 0 Initialization
                    ;                   for displaying elapsed time to terminal
                    ;
004037 0101FA       DMINIT:      LD      BC,D_TIME       ;prepare to load source addr
00403A ED0920            OUT0    (SAR0L),C               ; source address at ET_CLK
00403D ED0121            OUT0    (SAR0H),B
004040 1601              LD      D,01h
004042 ED1122            OUT0    (SAR0B),D
004045 010700            LD      BC,TDR1                 ;prepare to load dest addr
004048 ED0923            OUT0    (DAR0L),C
00404B ED0124            OUT0    (DAR0H),B
00404E 1602              LD      D,02h                   ;TDRE1 generates request
004050 ED1125            OUT0    (DAR0B),D
004053 0600              LD      B,0h
004055 ED0127            OUT0    (BCR0H),B       ;byte count high = 0
004058 0609              LD      B,9
00405A ED0126            OUT0    (BCR0L),B       ;byte count low = 9 chars
00405D 0630              LD      B,30h
00405F ED0131            OUT0    (DMODE),B       ;source (mem) incs, i/o stays
004062 06F4              LD      B,0F4h
004064 ED0132            OUT0    (DCNTL),B               ;edge sense on channel 0

004067 C9                RET

                    ;*************************************************************

000000              END
```

APPENDIX: Task C Code Listing

```
                    ;*********************************************************
                    ;
                    ;     Task C - executed upon receiving IRQ1 (external event)
                    ;            Causer count of external events to be incremented
                    ;
                    ;*********************************************************
000000                    CPU     "64180.TBL"
000000                    INCL    "INTIO.LIB"
                    ;     Include equates file used for Main Task
                    ;*********************************************************
                    ;
                    ;     Data Storage
                    ;            - System Data loaded from EPROM at initialization
                    ;
00F000                            ORG     0F000h ;storage area (Common 1)
                                             ;used by all tasks
00F000 30303A30303A30ET_CLK:      DFB     "00:00:00.0"
00F00A 300D         ET_CLKend:    DFB     "0",CR

00F800                            ORG  0F800h    ;storage area X (Common 1)
                                             ;used by Task C only
00F800 0D0A0A       EV_START:  DFB  CR,LF,LF
00F803 30           EVNT_CT:      DFB     30h            ;storage for count of events (<= 9)
00F804 206576656E7473             DFB     " events occurred",CR,LF
00F816 204C6173742065             DFB     " Last event occurred at "
00F82E 30303A30303A30LAST_EV:     DFB     "00:00:00.00",CR,LF,LF
                    ;
                    ;*********************************************************
                    ;
                    ;     Task C
                    ;            copy current value of real time clock to memory
                    ;            increment event counter
                    ;
004000                    ORG     04000h         ;org at logical address
                    ;
004000 210BF0       EVNT_SAV:     LD  HL,ET_CLK+11       ;end of ET_CLK string
004003 1139F8             LD      DE,LAST_EV+11  ;end of LAST_EV string
004006 010C00             LD      BC,12          ;number of chars to move
004009 EDB8         LDDR                         ;move information
00400B 2103F8             LD      HL,EVNT_CT     ;
00400E 34                 INC     (HL)           ;increment event count
00400F 3E39               LD      A,39h          ;load '9' + 1
004011 BE                 CP      (HL)           ;compare to value in Acc
004012 3003               JR      NC,SAV_END     ;if not equal, then return
004014 3E30               LD      A,"0"          ;if equal, then reset
004016 77                 LD      (HL),A
004017              SAV_END:
004017 C9                 RET
                    ;*********************************************************
000000              END
```

Reading of a Short Data Byte from the 64180 CSIO

Tech Notes

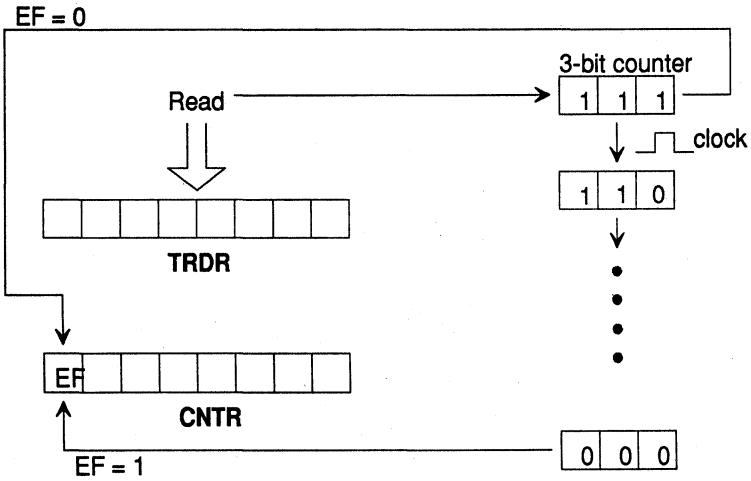
Application Engineering

Amelia Lam

In 64180, the clock synchronous serial I/O (CSIO) port is used for fixed 8-bit data transfer in half duplex mode. Both transmit and receive use the same Data Register (TRDR) and the same Control/ Status Register (CNTR). In the CNTR, an End flag (EF) is provided to indicate the status of an 8-bit data transfer; it is set to one if the transfer is completed, or reset to zero when data is read from or written to the TRDR.

In some cases, the user might have a link established with data less than 8-bit. When that happens, the data will still be available in the Data Register, and the user can read the shorter byte, e.g. 6-bit, as soon as it is done. Question may arise as to what will happen to the EF bit, since this bit get set at the end of 8-bit data transfer.

In 64180, whenever a read occurs, it causes an CSIO internal 3-bit counter to be reset. This 3-bit counter will set the EF flag after it has counted to zero and reset the flag after data is read. And it will perform the down-counting at the next available clock. But if the read occurs right after the 6-bit transfer, the counter does not have a chance to reach zero. Because of this, the EF flag will never get set. Hence, the user **cannot** rely on the CNTR for status information and has to ensure the correct data is read with external circuitry.



Start Bit Detection in the 64180's ASCI

Tech Notes

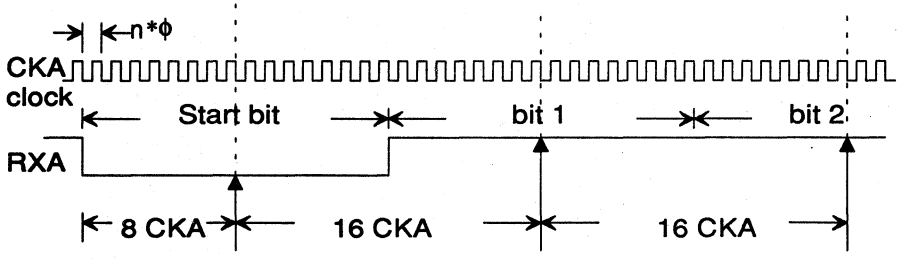
Application Engineering

Amelia Lam

(Note : This technote is to replace #TN-0041, March 1992)
The ASCI detects a start bit by monitoring the low level of the receive data after the following sequences have taken place to prevent the wrong interpretation of the noise signal as the start bit:

1. The receiver is enabled
2. On the falling edge of the serial clock (CKA)
3. a delay of 10 system clocks (ϕ)

If a low level is detected, the ASCI will sample the RXD again at 1/2 of a bit time later; that is 8 CKA in ÷16 mode or 32 CKA in ÷64 mode. The start bit will become valid if the sampling level is still low. From this point on, the ASCI then samples the data bit at one-bit time interval, which happens to take place in the middle of each bit. If a valid start bit is not detected, the ASCI will repeat the search at each falling edge of the clock.

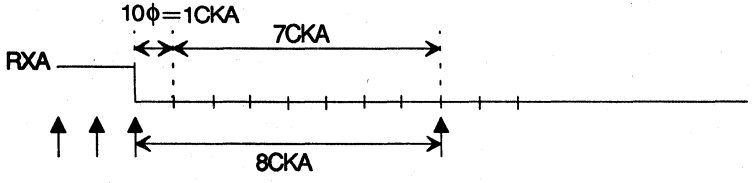


* n is determined from SS0,1,2 bits & PS bit of ASCI Control Register B

Data sampling in ÷ 16 mode

The delay of 10 ϕ clock is determined so that the next sampling point will be aligned at the center of the bit even in the worst case scenario with small prescale factor:

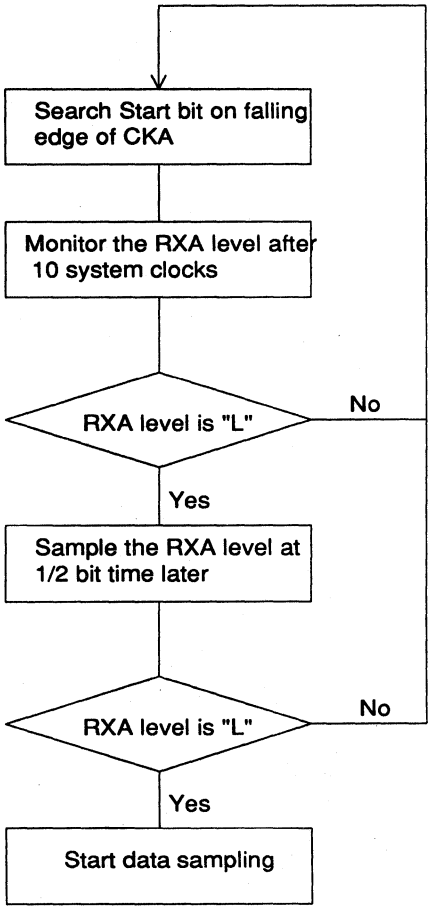
divide ratio is ÷ 1 (SS0, SS1, SS2 = 000) ; prescale by 10 (PS=0); sampling clock is CKA ÷ 16 (DR=0)



HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Here, 10 ϕ equals 1 CKA, and the 64180 actually waits after 7 CKA elapsed then performs the second sampling of the start bit. With this, the total durations still equal 8 CKA. This is not so critical if the prescale factor is higher, since 10 ϕ takes a less significant weight in the entire sampling cycle.

The following flow chart highlights the detection of a start bit in an Asynchronous data transfer.



Differences Between HD64180S and HD64180S2

Tech Notes

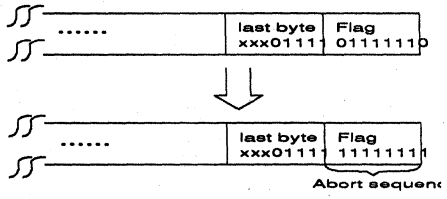
Application Engineering

Amelia Lam

The new product marking for the NPU is HD64180S2. It is the same as HD64180S except with the following differences :

	HD64180S	HD64180S2
1. Condition - Bit synchronous Loop Mode	Cannot be used	Can be used
2. Condition - Bit or Byte synchronous FM encoding (FM0, FM1, Manchester) - When the continous frames received by the MSCI are out of phase, the synchronization can be done by the ADPLL if an "enter search mode" command is generated in between the idle state of each frame	The user is required to generate an "enter search mode" command	An "enter search mode" command is automatically issued by the MSCI. No software intervention
3. Condition - Bit synchronous HDLC or Loop mode - If an FCS is not included in the transmit frame, and the last data immediately before the closing flag is in the range of F0 - F7H (i.e. the bit sequence right before the flag is seen as xxx01111)	The closing flag sent by the transmitter becomes seven 1's. The receiver thus interprets this as an Abort sequence and results in the wrong frame.	The MSCI will correctly transmit the closing flag even if the last data byte is between F0 - F7H.

in 64180S



HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Start Bit Detection in the 64180's ASCI

Tech Notes

Application Engineering

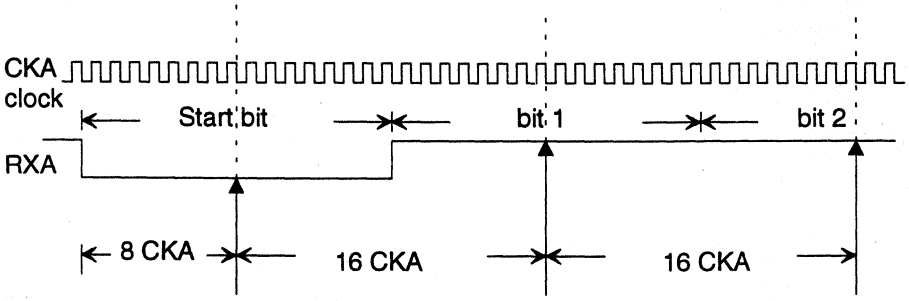
Amelia Lam

The ASCI detects a start bit by monitoring the low level of the receive data after the following sequences have taken place:

1. The receiver is enabled
2. On the falling edge of the serial clock (CKA)
3. a delay of 10 system clocks (Φ)

These prevent the wrong interpretation of the noise signal as the start bit.

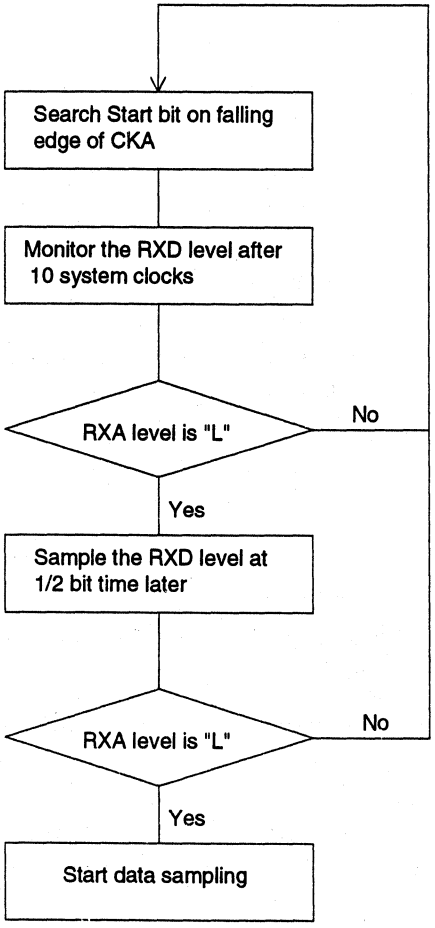
If a low level is detected, the ASCI will sample the RXD again at 1/2 of a bit time later; that is 8 CKA in ÷16 mode or 32 CKA in ÷64 mode. The start bit will become valid if the sampling level is still low. From this point on, the ASCI then samples the data bit at one-bit time interval, which happens to take place in the middle of each bit. If a valid start bit is not detected, the ASCI will repeat the search at each falling edge of the clock.



Data sampling in ÷ 16 mode

Start Bit Detection in the 64180's ASCI

The following flow chart highlights the detection of a start bit in an Asynchronous data transfer.



HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD64180 Family—DCD0 Line Operation

Tech Notes

Application Engineering

Marnie Mar

Hardware Manual Supplement

When 64180 designs using ASCI channel 0 require DCD0 to be active, users should be aware of the following if receive interrupts are enabled.

A receive interrupt will be generated whenever receive interrupts are enabled and the ASCI detects a low to high transmition of the DCD0 bit of Status Register 0 (STAT0). This bit transition will occur if either the external DCD0 input line transitions from low to high, or if the DCD0 bit of STAT0 is cleared by reading STAT0, but the external DCD0 line remains high. The DCD0 bit will be cleared by the STAT0 read, but the bit will be reset as soon as the external DCD0 line is detected high.

If the external DCD0 line is to be held high in an application, recieve interrupts should be disabled by clearing the RIE bit of STAT0. Otherwise, receive interrupts will be requested continuously until the external DCD0 line is set low.

ASCI Status Register 0 (STAT0 : I/O Address = 04H

bit	7	6	5	4	3	2	1	0
	RDRF	OVRN	PE	FE	RIE	DCD0	TDRE	TIE
	R	R	R	R	R/W	R	R	R/W

R: read only
R/W: read and write

64180 DMAC: Memory-Mapped I/O Transfers

Tech Notes

Application Engineering

A fixed memory location can be specified as the source or destination for a DMAC channel 0 transfer on the HD64180R, Z and HD647180X. When this occurs, the DMAC assumes that the fixed address is a register in an external memory-mapped I/O device which is capable of generating an external hardware request signal to be input on the $DREQ_0$ pin. Once the channel has been initialized and enabled, transfers will occur as requested by the $DREQ_0$ input which can be programmed to be edge or level sensitive.

If the user's system does not have the capability of providing a $DREQ_0$ input to trigger transfers to or from this location, transfers can be triggered by program control. In order for this to happen, program $DREQ_0$ to be level sensitive (set the DMS0 bit of the DCNTL register to 0) and tie the $DREQ_0$ pin low. This will cause DMA transfers to occur in burst mode whenever channel 0 is enabled (by writing 1 and 0 to the DSTAT register's DME0 and DWE0 bits, respectively).

Note on tying the $DREQ_0$ pin low: Since the $DREQ_0$ line is multiplexed with the CKA_0 input/output line, the user must not enable the ASCI0 baud rate generator prior to initializing DMAC channel 0 for memory-mapped I/O transfers. Initializing the system in this order would cause the CKA_0 line to output the baud clock, and this pin in the output state should not be tied low.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Notes on HD64180S (NPU) Bit-Sync Loop Mode

Tech Notes

Application Engineering

Marnie Mar

Abnormal transmission of data in secondary stations

The following problem may be found when using the HD64180S Network Processing Unit (NPU). The Multiprotocol Serial Communications Interface (MSCI) of this device may not operate correctly when it is used in the Bit Synchronous Loop Mode in support of secondary stations for bit synchronous loop transmission of data. For details on the correct operation of the Bit-Synchronous Loop Mode, please refer to the HD64180S NPU Hardware Manual, #U16.

The MSCI operating in this mode may transmit data abnormally when the Go active On Poll (GOP) bit of the MSCI Control Register (MCTL) is changed from 0 to 1. If operating correctly, data transmission would not begin until both the GOP bit is changed to 1 and a Go Ahead (GA) pattern has been detected. Figure 1 shows this incorrect operation.

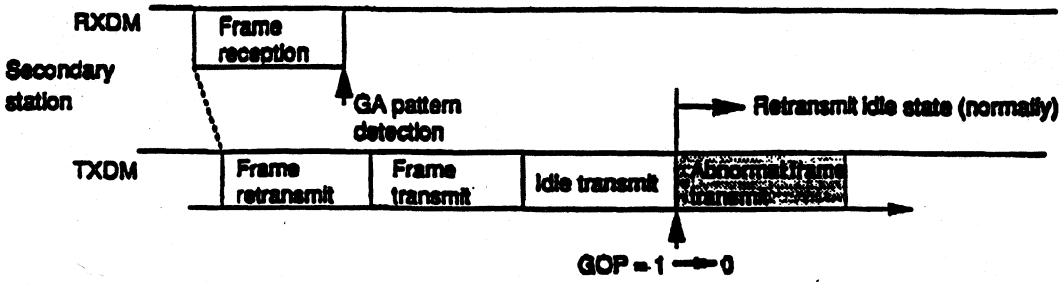


Figure 1. Failure in data transmission in a secondary station

Countermeasure

This abnormal transmission can be prevented by modifying the application software that controls the NPU operating in bit-synchronous loop mode.

<u>Warning on using the GOP bit</u>

The GOP bit should be changed from 0 to 1 only during the TX disable state immediately after hardware reset. After that, writing to the GOP bit should be avoided in the application software.

Figure 2 shows a portion of the State Transition Diagram for Transmission in Bit Synchronous Loop Mode. To avoid abnormal transmission of data, the GOP bit should never be written by application software after it has been initialization to "1". Therefore, GOP remains in the "1" state during operation. To cause transition from the Idle state to the Retransmit Idle state without writing "0" to the GOP bit requires the following steps:

1) Issuing the "TX Reset" command in the Idle state causes transition to the TX Disable state
2) Issuing the "TX Enable" command in the TX disable state causes transition to the Retransmit Idle state

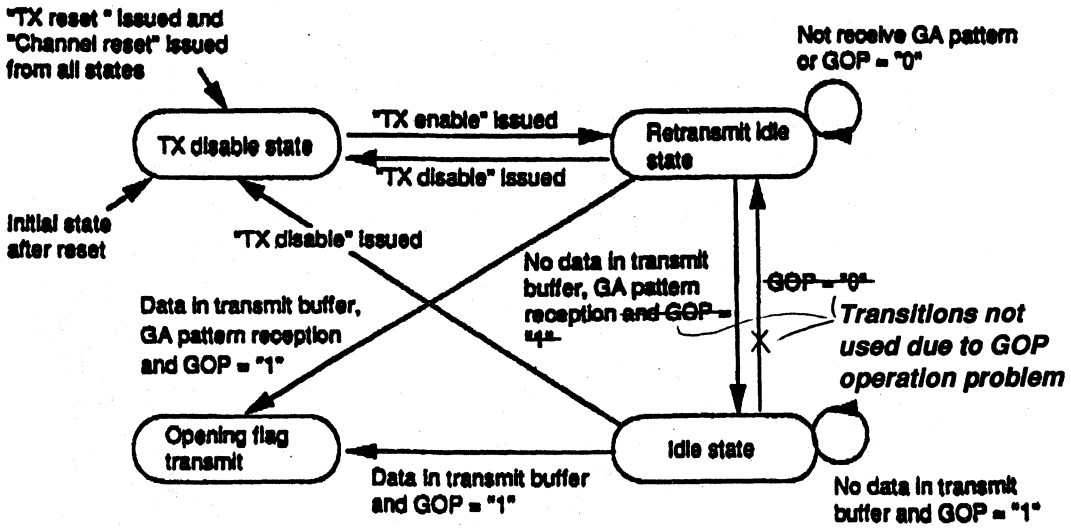


Figure 2. State transition

Countermeasure by software

Figure 3 shows the results of the countermeasure for the loop mode operation problem in software:

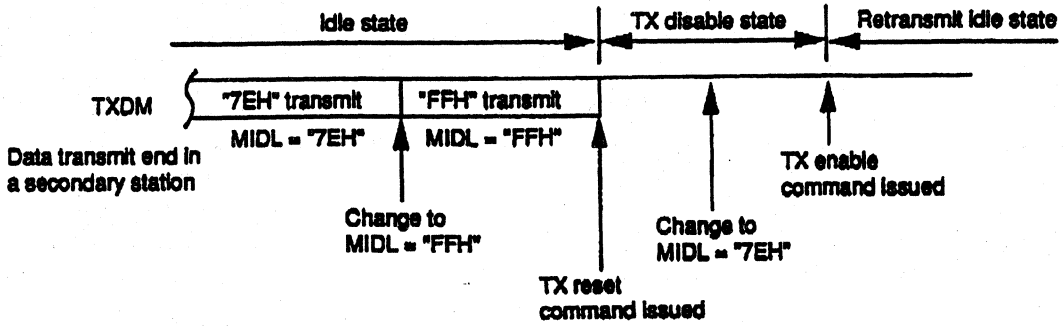


Figure 3. Example

Notes:

1) The MSCI Idle pattern register (MIDL) is programmed to FFh from 7Eh during the Idle state to cause the change from flag transmit to mark transmit.

2) The TXDM output pin is in mark state during the TX Disable state, which is the same as it would be if direct transition to Retransmit Idle state occurred.

3) When the GA pattern is detected, a secondary station transitions to the Idle state even though it does not have transmit data (refer to Figure 2). When this occurs, the above operation needs to be repeated to return to the retransmit idle state, since writing "0" to the GOP bit could cause problems.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

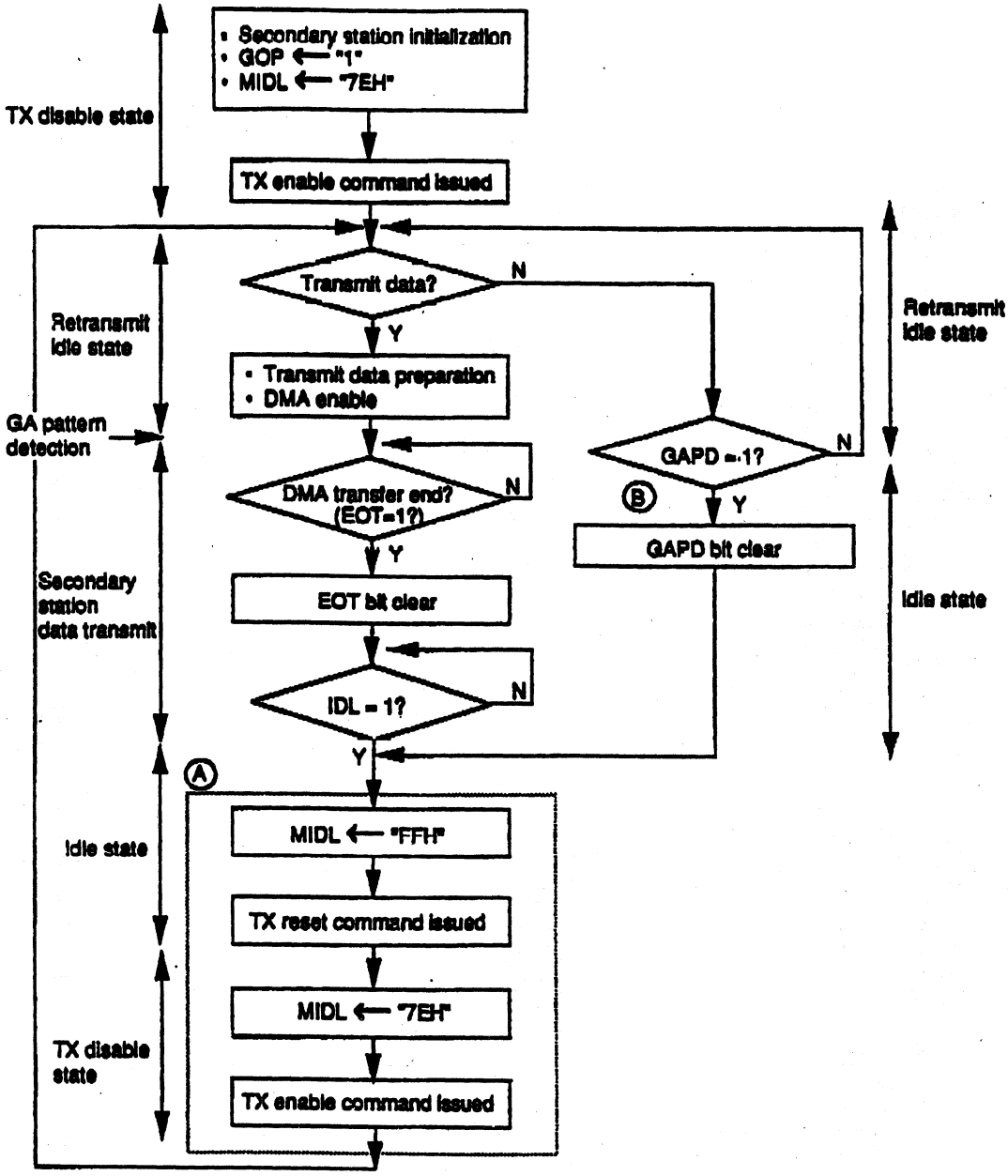


Figure 4. State transition control flow at bit synchronous loop mode

Flow Chart

Figure 4 shows the flow chart of the transmit operation in Bit Synchronous Loop Mode using the recommended software countermeasure.

Notes:
1) Condition branches in the flow chart occur as a result of interrupts, or by CPU polling.
2) Operation shown in the box marked A replaces writing "0" to the GOP bit.
3) The GOP bit is always "1", so a secondary station transitions to an Idle State when the GA pattern is detected in the Retransmit Idle State whether or not transmit data is available. When this occurs, operation should return to the Retransmit Idle State by clearing the Go Ahead Pattern Detect (GAPD) bit of MSCI Status Register 1 (MST1).

Port A Register Programming

Tech Notes

Application Engineering

Marnie Mar

Writing to the DERA (Port A Disable Register) can affect the values programmed into the DDRA (Data Direction Register A). For instance, if the DDRA is programmed to set up directions for the I/O port pins, and the DERA register is later programmed with a value, it is possible for the DDRA value to change causing the data directions for the I/O pins to change.

To ensure that the directions of I/O pins are preserved, the following steps should be taken:

1. DERA should be programmed prior to programming the DDRA.

2. If the DERA is reprogrammed, the DDRA should also be reprogrammed.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Section 3

4-Bit Family

HITACHI®

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Effect of TMA2 on Timer A Operation

Tech Notes

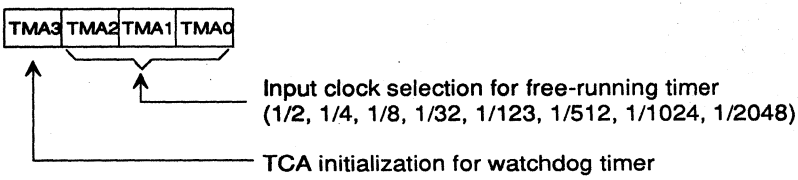
Application Engineering

Amelia Lam

In the Compact 400 series, Timer A can be configured by mask option in two operation modes, namely Free-running timer or Watchdog timer. According to the databook, bit 3 of the Timer Mode Register A (TMA) is used for "TCA initialization for watchdog timer". This technote is to explain the usage of this bit 3 in watchdog mode, and the additional feature it provides in free-running mode.

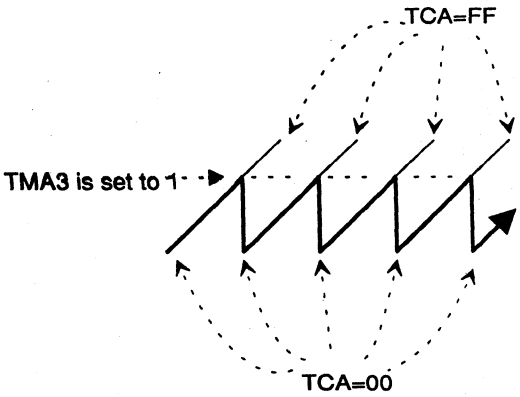
Timer Mode Register (TMA)

This is a 4-bit write-only register. The prescaled input clock to Timer A is determined from bit0 to bit2, and bit 3 is for resetting the counter TCA.



Watchdog Timer

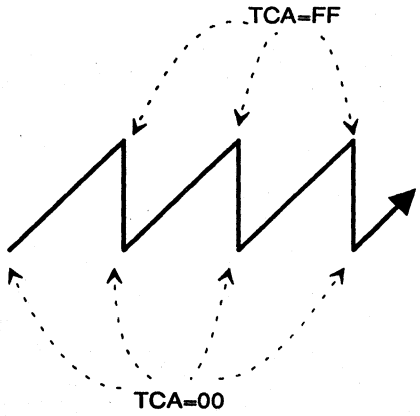
In the watchdog timer mode, Timer A counts up on every 1/2048 of the system clock signal and generates an overflow when the counter reaches FF, causing the MCU to reset at the same time. Therefore, TMA3 is used to reset the counter before the overflow situation occurs.



Effect of TMA2 on Timer A Operation

Free-running Timer

In the free-running timer mode, Timer A counts up on the clock signal selected by the TMA register. As soon as the counter reaches FF, it generates an overflow and sets the interrupt request flag. Timer A is then reloaded with 00 and starts counting again. But if TMA3 is set, the counter will get reset before it has the chance to reach FF, thus defeats the purpose of Timer A serving as a free-running timer. The databook refers this phenomenon as " the MCU malfunction". In fact, it is the timer, not the MCU, that is being referred. The real issue is the counter does not function as expected in the free-running timer mode. On the contrary, setting this bit 3 provides an additional feature, an event counter for Timer A in the free-running mode.



HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

4-Bit ZTAT Microcomputer PROM Programming

Tech Notes

Application Engineering

Amelia Lam

The on-chip PROM of the HMCS400 ZTAT microcontrollers is programmed in the same way as a standard 27256 EPROM does. Since each instruction of the HMCS400 is 10 bits wide, a special programming sequence is employed in order to properly convert a 10-bit program code onto an 8-bit memory locations.

However, there may be times when the PROM programmer reports a device or programming error even there isn't one. This may be caused by some of the records in the object file not having the entire memory space occupied with data. After the object file is downloaded into the PROM programmer, those unoccupied areas are then filled with data of '00' byte. This violates the HMCS400 programming specification which requires the upper three bits of each byte be '111'.

The following text illustrates a method to circumvent this problem.

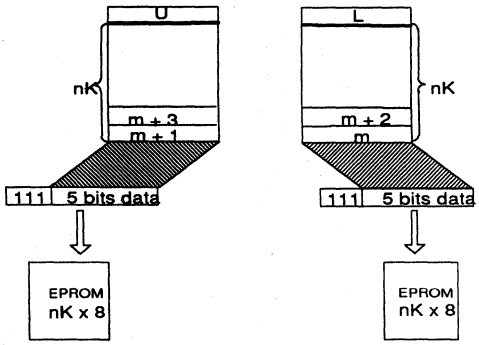
ZTAT Vs 27256 EPROM Code Assemblying

By using the Cross-Assembler, the 10-bit instruction source code will be transformed into an 8-bit object file ready for downloading. This is done by splitting 10-bit word into two halves and each half is padded with '111' in the most significant three bits to form a byte.

If an 8-bit EPROM is used for programming, the command is:

"ASSEMBLE *filename* "

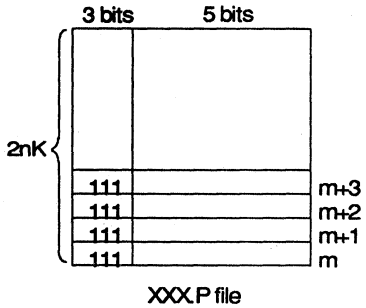
this will create two 8-bit wide files to be burned into two EPROMs; one contains all the even address code, and the other one contains all the odd address code.



If the ZTAT is used for programming, the command is:

"ASSEMBLE *filename* /P"

this will generate an object file 'xxx.P' which contains both odd and even addresses interleaving with each other.



Downloading

When using a menu-driven software such as "PROMLINK" for downloading, first **fill the programmer RAM space with FF** prior to loading with the object code. This ensures all ones in the most significant three bits on each byte even after 00 is loaded to the RAM.

PROM Programming

Adjust the operation boundaries by **setting the device block size** to the code size. Instead of programming the entire ROM which the ZTAT part comprises , it only program the selected ROM space. By doing this, it adds efficiency in the burning process.

Alternative

Besides filling the programmer RAM with FF, an alternative to eliminate the gap between each record is to pad the source code with FF so that the starting address to the ending address of a whole record will have data in it.

HD404272 User Cable Conversion Board SW1 Pin Layout

Tech Notes

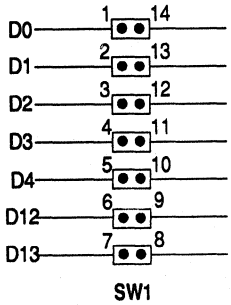
Application Engineering

Amelia Lam

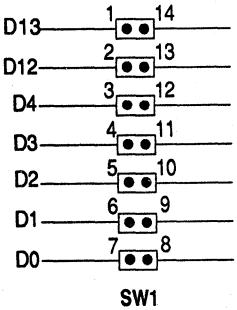
HS4274ECS28H is a User System Interface Cable connecting to a general-purpose target probe (HS400ETA01H). It is used specifically for the high-voltage I/O Compact device HD404272 family. The unit includes a conversion board with header plugged into the target probe, one set of flexible ribbon cables with 28-pin shrink DIP connector plus protection socket, a power supply cable and a spare protection socket (No.3).

Rather than software programmable, the high-voltage pins on this user cable must be fixed to either input or output. The way to do this is by means of the on-board switches (SW1, SW3 & SW4). However, there is a mismatch in the SW1 switch layouts between the schematic and the board itself.

On the schematic of the HS4274ECS28H user's manual, the SW1 pinouts are connected to the databus as follows:



The correct settings should be the one showing on the board:



HITACHI

Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Section 4

Display Devices

- Graphics
- LCD

HITACHI®

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Section 4

Display Devices Graphics

SECTION 4

HITACHI®

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD63484 ACRTC AND HD63487 MIVAC Data

Tech Notes

Application Engineering

Kash Yajnik

Q / A , Test , and Reliability Information

The most frequently requested Q/A, test, and reliability data by the Hitachi customers using the parts listed above, is summarized below;

1.0 ACRTC (HD63484) - Transistor Count = 117,000

2.0 MTBF

- o HD63484 (ALL Packages) = 3.8×10^6 Hours at $T_A = 55\ ^{oC}$, and Confidence Level = 60%
- o HD63487 (ALL Packages) = 3.4×10^6 Hours at $T_A = 55\ ^{oC}$, and Confidence Level = 60%

3.0 ACRTC (HD63484) - Junction to Case Thermal Resistance

PRODUCT	PACKAGE	COMMENT	θ_{J-C}
HD63484P8	DP-64	Plastic DIP	75 ^{oC}/ W
HD63484-8	DC-64	Ceramic DIP	35 ^{oC}/ W
HD63484CP8	CP-68	PLCC	45 ^{oC}/ W

4.0 ACRTC (HD63484) - Power Consumption Vs Speed

SPEED	I_{CC}(Max) @ V_{CC} = +5V ± 5 %
9.8MHz	120 mA
8 MHz	100 mA
6 MHz	80 mA
4 MHz	60 mA

HD63484 ACRTC

Tech Notes

Application Engineering

Kash Yajnik

PRODUCT : **HD63484 ACRTC**

Mask History

The details of limitations on the usage of the "R", "S", and "U" mask are shown below along with the product mask diagram for identifying different masks.

■ HD63484 ACRTC Limitations on Usage

The status of the item numbers described in this paragraph is summarized in the following table.

Limitation on the ACRTC function.

No.	Limitation for the ACRTC function	R Mask	S Mask	U Mask
1	Light Pen Interface	Unusable	Usable	Usable
2	RS Signal during DMA Transfer	Unusable	Usable	Usable
3	AREA Mode for AFRCT, RFRCT and PAINT Commands	Unusable	Usable	Usable
4	DRD Command	Unusable	Usable	Usable
5	DMOD Command	Unusable	Usable	Usable
6	PAUSE Bit	Unusable	Usable	Usable
7	AS Output Timing during Zooming	Unusable	Usable	Usable
8	BLINK Feature	Unusable	Usable	Usable
9	CLR, WT and DWT Command	Unusable	Usable	Usable
10	Writing to Registers during DRD Command Execution	Unusable	Unusable	Usable
11	Command DMA Transfer Mode	Unusable	Unusable	Usable
12	Tiling Using the PAINT Command	Unusable	Usable	Usable
13	Displaying WINDOW	Unusable	Unusable	Usable
14	PARAMETERs for ELLIPSE Command	Unusable	Unusable	Unusable
15	PARAMETERs for ELLIPSE ARC Command	Unusable	Unusable	Unusable
16	Light Pen Strobe Detect (LPD) Status Bit	Unusable	Unusable	Usable

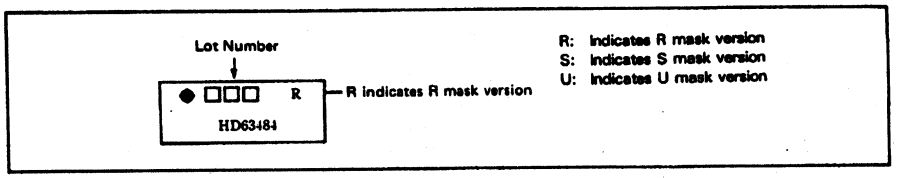


Figure N5 Product Mark

EV63487

Technical Brief

MIVAC Evaluation Board

Kash Yajnik

The EV63487 MIVAC Evaluation Board was designed by Hitachi Europe Ltd., Munich, Germany, and can be ordered through Hitachi America Ltd. in U.S.A. The board is shipped in a foil cover with a User Manual, and the associated software diskette capable of demonstrating graphics patterns on a 6.3" or 10.4" TFT color Liquid Crystal Module from Hitachi.

Up to eight colors may be displayed depending upon the selected LCD panel from Hitachi's ELT Division. The EV63487 MIVAC Evaluation Board can reside inside IBM PC-AT or a compatible system running later than DOS version 2.0. It is also possible to run the EV63487 Board with an external power supply. This board takes one slot in the chasis and its size is half that of the standard card. The back light power is also supplied by this card. Identical color images can be displayed on the CRT monitor as well as the color LCD panels.

The EV66387 Evaluation board uses Hitachi's Advanced CRT Controller (ACRTC) HD63484, and Memory Interface Video Attribute Controller (MIVAC) HD63487. This board has no LCD controller part as the CRT data is serially sent to the color panel for display.

This technical brief is written to complement the EV63487 User's Guide for one specific application using the 6.3" color TFT module (TM16D01HC) from Hitachi's Electron Tube Division (ELT). A copy of the schematic is also included to provide the design implementation detail. A system diagram is also included to high light the laboratory environment. Similarly, each user may tailor display subsystem requirements for the desired application.

The scope of this document is to help make the design task easier and quicker. The circuit minimization tasks are left to each user and are not attempted. This is intended as an illustrative example for the Hitachi field and technical staff, and their customers.

The following pages cover system configuration and componenents, EV63487 Board set up, System debug, and Demonstration software. The Appendix "A" covers LCD cable and the Appendix "B" shows the Back light power connections. The Appendix "C" lists the schematic.

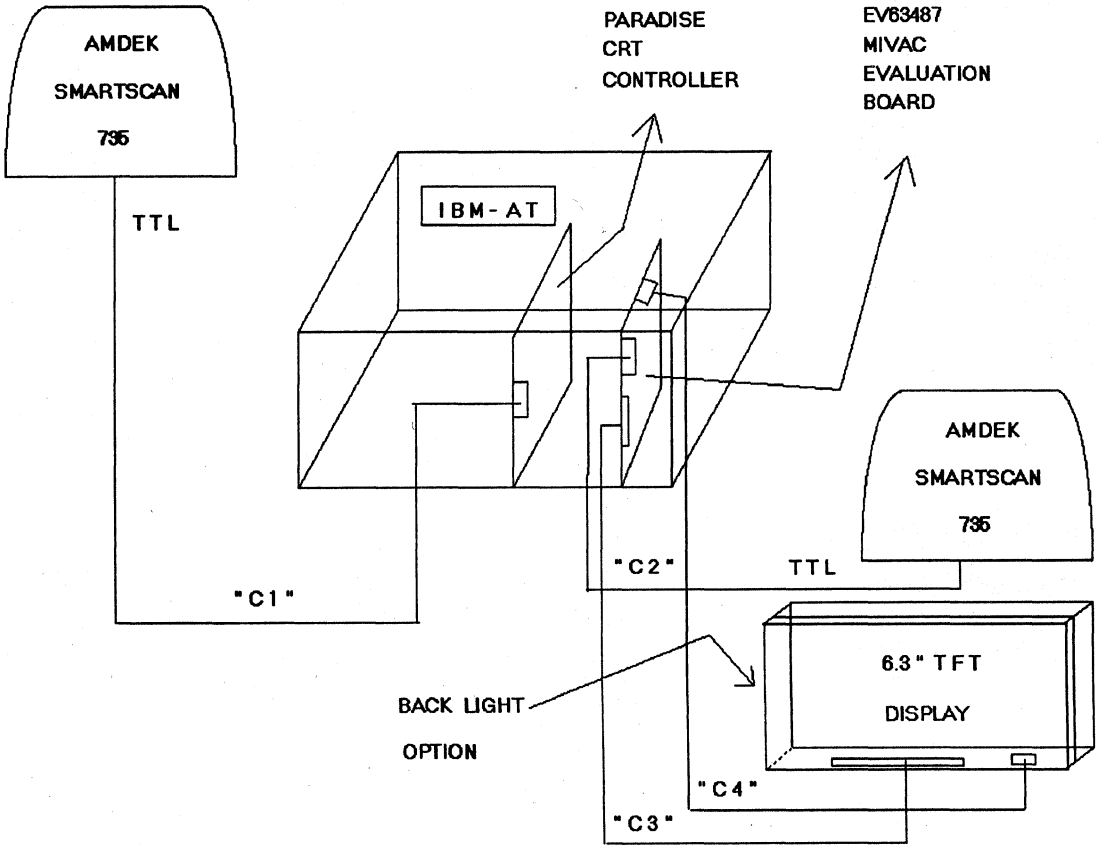
Refer to the subsequent pages for more detail.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

SYSTEM CONFIGURATION

The development system was configured with IBM PC-AT or compatible machine, Paradise Autoswitch EGA 480 card, EV63487 MIVAC Evaluation Board, Smartscan Amdek 735 digital color monitors, and Hitachi TFT active matrix, 8 color, 6.3" LCD display (TM16D01HC) from the ELT division. The 9 pin TTL video cables required to provide CRT video signals from the EV63487 Evaluation board or the Paradise video board are **not** provided. The MIVAC Evaluation board output connector cable to the 6.3" TFT display **is** shielded to increase noise immunity and to make the LCD display connection task easier. A separate +12V DC cable is also required for the back light option (#BLS-006M). The back light is easily mounted with the four corner screws of the 6.3" TFT display.

The system diagram is shown below :



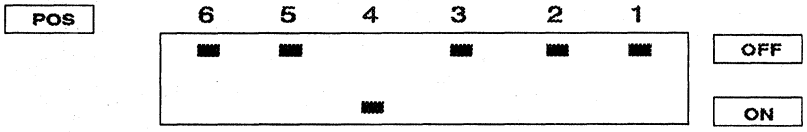
NOTE : 1.0 " C1 " = " C2 " = "C3" = "C4" cables are **not** provided.

SYSTEM COMPONENTS

The hardware components are described in this section while the " C3 "," C4", cable wiring diagrams and schematic are shown in the Appendices.

PC-AT : AST Premium 286 model 70 was operating at 10 MHZ, with 512KB memory, 20MB hard disk drive, and 1.2 MB, 5.25 " floppy drive. It was also running DOS version 3.2.

VIDEO CONTROLLER : It provides the video signals to the Amdek monitor # 1 over the cable " C1 ". Paradise Autoswitch EGA is card used in the CGA mode at (640H x 200V) resolution to generate the TTL level signals to the monitor. The switch settings for 80 column, RGB monitor in CGA mode are shown below in its diagram:



NOTE : 1.0 For more details refer to the Paradise CRT controller manual.
2.0 Make sure this switch is correctly set.

EV63487 MIVAC EVALUATION BOARD

This board has no switches and its settings are built in. So, please refer to the EV66841 User's Guide for details. Only the 6.3" TFT LCD display was used, eventhough the manual describes the 10.3" display.

The EV63487 board provides TTL level output signals carried by the 9 pin cable " C2 " to the the video monitor Amdek # 2. The R,G,B, HSYNC, and VSYNC signals are included in that cable. The output signals are also sent over the 34 pin cable " C3" to the 6.3" TFT, 8 colors, Hitachi display. This board also supplies +12 Volts required by the back light through the cable " C4 ".

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HITACHI COLOR LCD TFT MODULE (TM16D01HC) :

Refer to the display data sheet for detail. The page 14 of it shows how the sub-pixels are designated for LVIC HD66841 interface with 160 dots (H) and 200 dots (V) resolution. The cable " C3 " provides the signals to the display while cable " C4 " provides the back light power. The display tilt and swivel angles provide different contrast ratios in the ambient light, so it should be adjusted for the most desirable viewing angle.

AMDEK MONITORS

The two CRT color monitors #1 and #2 are used in the CGA mode. They show the DOS commands dialogue on monitor #1 while the monitor #2 displays the demonstration program output. Observe that the monitor #2 and the 6.3" TFT display show identical color images in the 640H x 200V mode.

SYSTEM DEBUG

First power up the system in CGA mode using the AMDEK color monitor #1 and the EGA board. Only the cable "C1" is plugged while the cables "C2", "C3", and "C4" are not connected effectively disconnecting the 6.3" TFT LCD display. After the system is up in the CGA mode, with a DOS prompt, verify that it works correctly. Then reconnect the EV63487 board cables "C2", "C3", and "C4". Also, verify that the cable "C2" is properly connected to the display, since there is **no** key in the connector. If the cable "C4" is properly connected, back florescent light should come on and it is clearly visible.

If every thing is working correctly, one can execute all the DOS commands when appropriate prompts are displayed on the CRT monitor #1 screen.

DEMONSTRATION SOFTWARE

After DOS 3.2 or later is installed, load the programs from the software diskette after creating MIVAC and SOURCE directories. Modify the AUTOEXEC.BAT file to run file INI20-32.EXE to initialize the ACRTC (HD63484) and the MIVAC (HD63487) from Hitachi. After the initialization program is run, execute the DEMO200.EXE program to show identical graphic images on the color 6.3" LCD panel as well as the video monitor #2.

The AUTOEXEC.BAT file sample is shown below:

```
PATH = C:\;C:\DOS;
SET PROMPT = $P$G
CLS
C:\MIVAC\EV63487\INI-32.EXE
ECHO HIT RETURN FOR TFT DEMO
ECHO OTHERWISE TYPE ^C
```

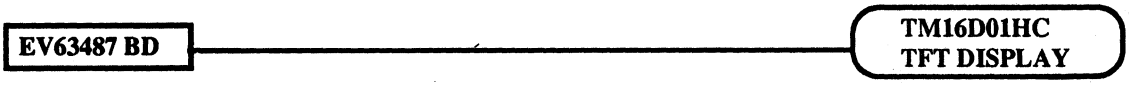
DEMONSTRATION SOFTWARE (CNTD.)

PAUSE
C:\MIVAC\EV63487\DEMO200.EXE

The DEMO200.EXE program screen output on the DOS monitor #1 is described below:
1.0 1000 Random filled circles
2.0 1000 Random filled rectangles
3.0 1000 Random rectangles
4.0 Lines
5.0 Color bars
6.0 16 filled ellipses
7.0 Logo
The corresponding images should be displayed on the 6.3" TFT color LCD display as the demonstration program is executed in the IBM PC-AT or compatible machine.

APPENDIX " A "

The 34 pin LCD shielded cable " C3 " is shown below :

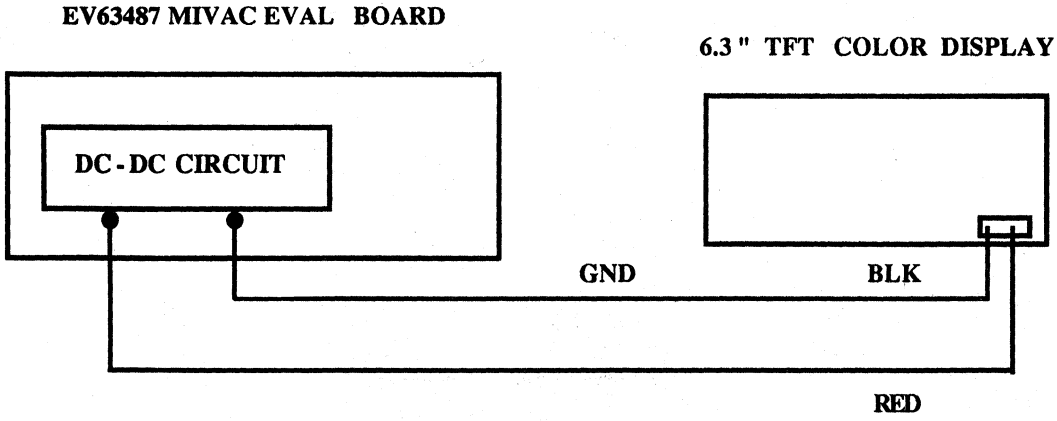


" C3 "

V_{EE} (-20V)	Pin 1	Pin 2	V_{EE} (-20V)
V_{DDA} (+5V)	Pin 3	Pin 4	V_{DDA} (+5V)
NC	Pin 5	Pin 6	V_{DD} (+5V)
IM0 (+5V)	Pin 7	Pin 8	V_{DD} (+5V)
IM1(G)	Pin 9	Pin 10	V_{DD} (+5V)
DOTE	Pin 11	Pin 12	G
VSYNC	Pin 13	Pin 14	G
HSYNC	Pin 15	Pin 16	G
DTMG	Pin 17	Pin 18	G
G	Pin 19	Pin 20	G
G	Pin 22	Pin 22	GRN
G	Pin 23	Pin 24	G
G	Pin 25	Pin 26	RED
G	Pin 27	Pin 28	G
G	Pin 29	Pin 30	BLU
G	Pin 31	Pin 32	DCLK
G	Pin 33	Pin 34	LCLK

APPENDIX " B "

Back light power cable " C4 " , " C1 ", and " C2 " are shown in this Appendix :



" C4 "

NOTE : Back light power is to be externally supplied through the " C4 " cable.

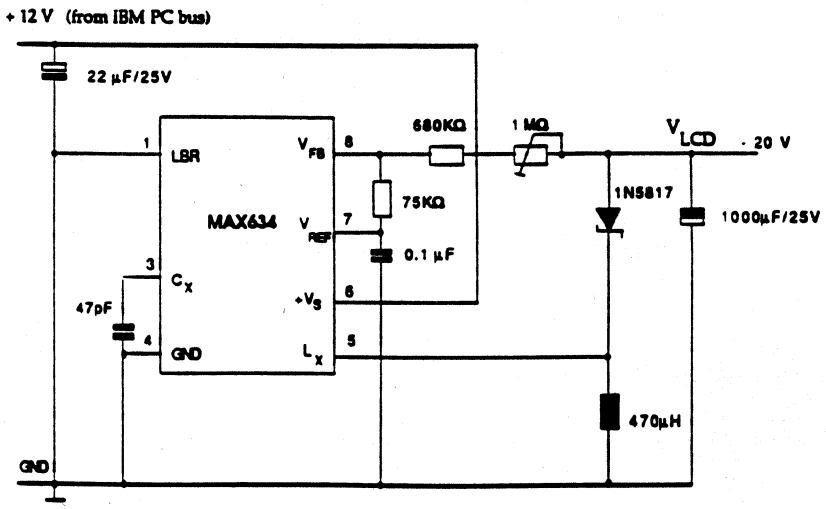
" C1 " AND " C2 " CABLES :

They are atached to the two monitors and are provided by Amdek, the manufacturer.

APPENDIX " C "

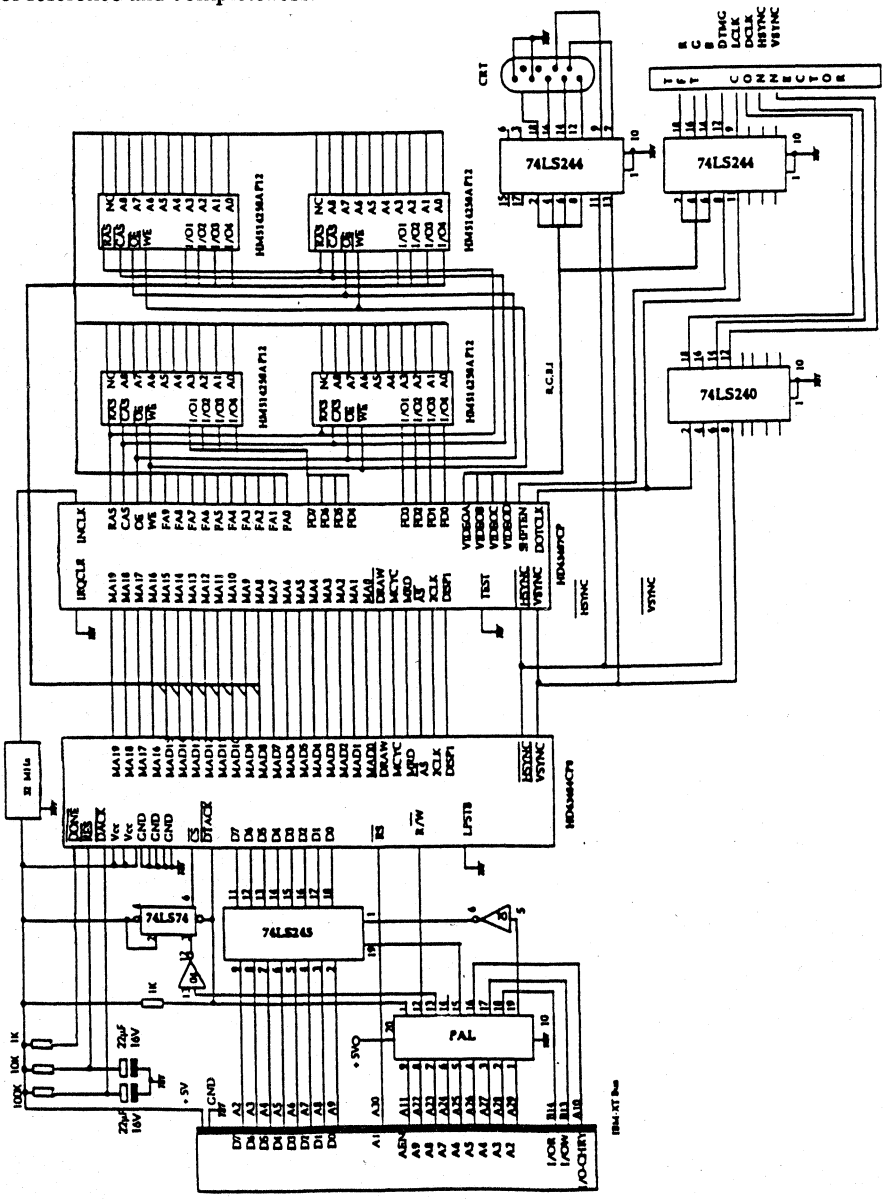
This section shows a copy of the schematic supplied by Hitachi Europe Ltd.. Munich. Germany. It is inluded for reference and completeness.

DC-DC converter :



APPENDIX " C "

This section shows a copy of the schematic supplied by Hitachi Europe Ltd., Munich, Germany. It is merely included for reference and completeness.



HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

EV63487

MIVAC Evaluation Board

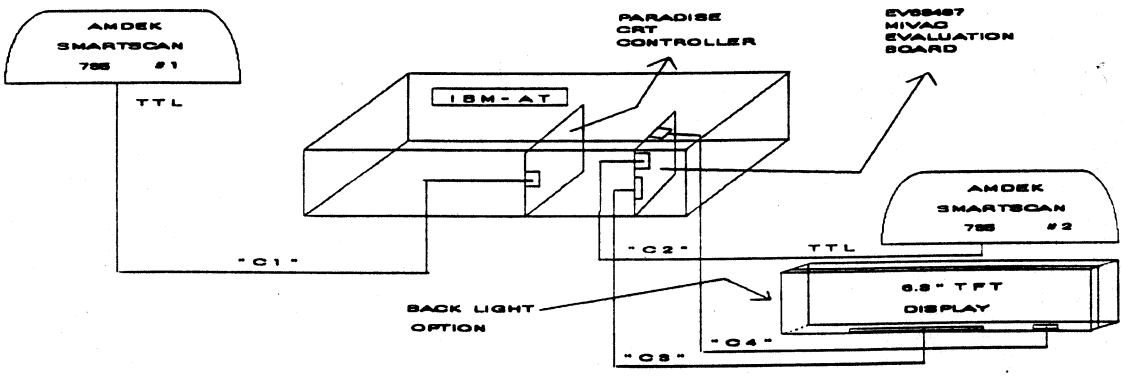
This document presents information for a 6.3" or 10.4" color active matrix LCD subsystem implementation using Hitachi semi conductor products ACRTC HD63484 and MIVAC HD63487. Its major components include IBM PC-AT, EV63487 MIVAC Evaluation Board, Paradise Video Controller Board, and the color LCD display (TM16D01HC) from Hitachi's ELT Division. It can be further enhanced by adding demonstration software that runs on the IBM PC-AT or a compatible machine.

■ FEATURES

--Hardware--
(1) IBM PC-AT or compatible machine
(2) EV63487 MIVAC Evaluation Board from Hitachi Europe in Germany.
(3) Color LCD 6.3" Active Matrix display from Hitachi with Back Light option
(4) Paradise Video Controller Board

--Software--
(1) DOS 3.2 Version or later
(2) ACRTC and MIVAC Inilialization programs
(3) Source code for the programs in " C " or BASIC
(4) Demonstration programs for 6.3" or 10.4" TFT active matrix color LCD display

■



■ OBJECTIVES

(1) To display EV63487 MIVAC Evaluation Board
(2) To demonstrate 6.3" color active matrix display
(3) To show demonstration software running on the EV63487 MIVAC Evaluation Board
(4) To high light PC-AT bus interface

■ ADDITIONAL INFORMATION

The details of the system configuration and its design along with the associated software, are available in the Hitachi America Ltd. Technical Brief #TB0101.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Section 4

Display Devices
LCD

HITACHI®

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD44780 / LCD PANEL

Design Tutorial

H8/325 Application Note

Kash Yajnik

OVERVIEW

This tutorial is written to complement the H8/325 micro processor literature and also **illustrate** the HD44780 development for one specific application i.e. interfacing to a selected number of LCD panels from Hitachi's ELT Division.

A copy of the schematic and software listing is included to provide the design implementation detail. A system diagram is also included to high light the laboratory environment. Similarly, each user may tailor requirements for the desired application.

The scope of this document is to help make the customizing task easier and quicker. The circuit minimization tasks are left to each user and are **not** attempted. This is intended as an illustrative example for the Hitachi field and technical staff, and their customers.The H8/325 Series Model-I ASE (Adaptive System Evaluator) was designed by Hitachi Ltd., Tokyo, Japan, and can be ordered through Hitachi America Ltd., in U.S.A. The associated Emulator Box (HS328ABX01H) for H8/325 microprocessor based product development was used to send digital information. A HD44780 LCD Controller Driver located on the Hitachi panel from Electorn Tube (ELT) Division, Chicago, Illinois, processed the displayed message.

Black and white character information can be shown on selected LCD panels from Hitachi's ELT Division. Among the many products offered by the Hitachi's ELT Division, for **this** application, LCD panels LM016XML, LM016L, LM041L, LM044L, and LM054 were selected and tested.

An Emulator Inter connect Board is required to enable the H8/325 ASE to talk to the LCD display panels. The character data is sent to the LCD panel for processing as well as display. The HD44780 LCD Controller Driver from Hitachi, SICD, located on the panel, processes the data sent by the H8/325 development system for display.

The H8/325 Emulator Interconnect Board resides on a bench connected to a LCD panel. The other end of the board is connected to the H8/325 Emulator User probe. It also requires external power supply. After power on, a demonstration program is loaded in the ASE system. It is then run to display a character message.

The following pages cover system configuration and components, H8/325 ASE Development System, Hardware Design, Software, and Demonstration Program. The Appendix " A " covers H8/325 ASE system details, and the Appendix " B " shows the Emulator Interconnect Board schematic. Also, Appendix " C " lists the LCD Panel data sheets, and Appendix " D " shows the H8/325 Initialization software listing. The appendices " E ", and " F " show HD44780 code listing and the reference literature respectively.

Refer to the subsequent pages for more detail.

TABLE OF CONTENTS

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD44780 / LCD PANEL

INTRODUCTION :

This section describes the design goals and provides a general overview of this presentation, along with a software development listing.

The design goals established for this project are briefly listed below:

- o To use H8/325 ASE system with software.
- o To display with LM016XML, LM016L, LM041L, LM044L, and LM054 panels from Hitachi.
- o To display four data bytes in the character mode using HD44780.
- o To design Emulator Interconnect Board.
- o To write programs for debug and test.
- o To use Hitachi H8/325 Emulator and User probe.
- o To use readily available software at Hitachi Field Offices for development.
- o To generate HD44780 / LCD Panel Tutorial.

A brief description of the LCD display system components listed above is provided in the next section as an overview. To complete the overview, a system block diagram is also presented. The rest of the sections described in the Table Of Contents are expanded in greater details along with their programming data. The Appendices give additional information, the program listing, and also list the referenced literature. A copy of the Emulator Interconnect Board schematic is also provided to illustrate the implementation details of this application.

SYSTEM CONFIGURATION

The display system was configured with H8/325 ASE Unit, Emulator Box, User cable, and a variety of LCD panels from the ELT Division, along with an Emulator Interconnect Board. The required cable lengths are shown in the schematic for CMOS signal levels. The LCD power pins are a part of the 14 pin panel cable, so a separate power cable is not required. The system block diagram for the Emulator Interconnect Board is shown in the Appendix " B ". The system block diagram is shown below in Figure 1:

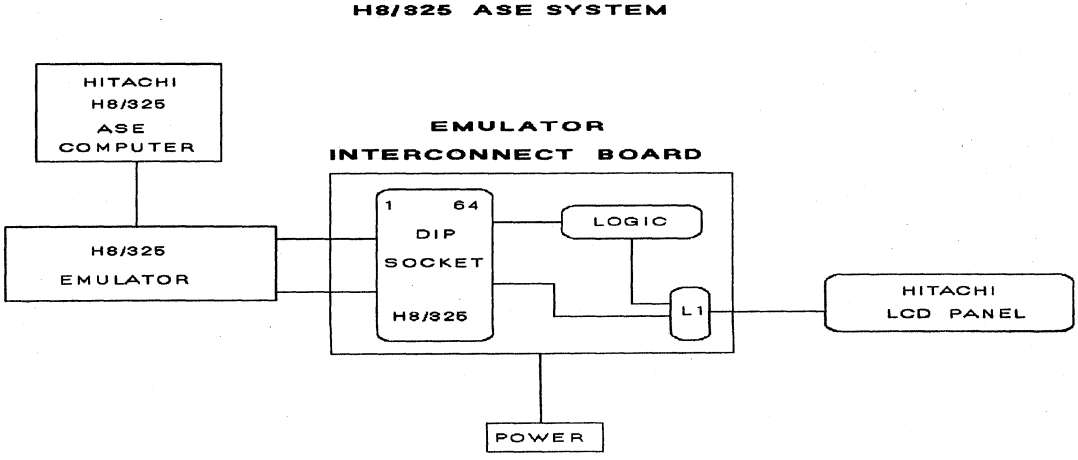


FIGURE 1

NOTE:

1.0 The required ASE and Emulator cables are provided by Hitachi Ltd.

2.0 The Emulator Intercconnect Board power and panel cables are built from the available documentation.

SYSTEM COMPONENTS

The LCD display system components such as H8 / 325 ASE Unit, Emulator, User cable, Emulator Interconnect Board, a variety of display panels, External Power Supply and the related software are described in this section.

H8 / 325 ASE Unit : This product was designed by Hitachi Ltd., Tokyo, Japan. It is used as a demonstration and development tool. Refer to the Appendix " A " for its features and other details including a picture. The system software allows line assembly, disassembly, editing, trace, break setting, and other debug facility.
It is used in transparent mode and the host port is connected to a VAX computer. The CRT port is connected to DEC VT320 terminal. The Motorola S record compatible programs are loaded by 1.2 MB, 5 1/4" diskette. They are run to develop the code and associated demonstration software. The ASE is run at the emulation clock speed of 7.3 MHz and H8/325 mode 2 operation. The development system comes with ASE system program, control program, configuration file, edit command program, and diagnostic program. For more details, refer to the ASE manual (HS328ASE01HE).

LCD Display Panels : These character display panels are provided by the Hitachi's ELT Division. Although, LM016XML, LM016L, LM041L, LM044L, and LM054 panels were used and tested in the laboratory, most of the code development was done using LM016XML. The appendix " C " lists the panels and their features. Their cable pin outs are identical and so, switching between them is easier. Note that the display orientation for panels LM041L and LM044L is upside down from the other panels. The same demonstration program was run on all the LCD panels to show " S ", " I ", " C ", and " D ".
All the panels mentioned above are capable of displaying 1 or 2 or 4 lines of eight or sixteen or twenty 5x7 alpha numeric characters. Their resolution varies from 40 dots to 100 dots in width and 8 dots to 32 dots in height. The duty cycle may be 1/8 or 1/16.
The parallel data may be clocked in at a maximum " E " clock rate of 1 MHz . They run from +5V power supply. The customer has to solder 14 pins on each of the panels for the appropriate connector used on the Emulator Interconnect Board. The LCD panel mounting and the proper viewing angles are critical to a strain free LCD display. Please, handle the panels according to the care recommended by the LCD display manufacturer. The logic signals sent to the LCD panel are at CMOS levels.

Emulator Interconnect Board : A wire wrap board was built to send parallel data, control signals, and power to the LCD panel over the "L1" cable. The 64 pin male DIP User Cable was connected to the DIP socket on the Emulator Interconnect Board. The LCD panel contrast adjust potentiometer was also put on this board. The data bus and gating logic were also located. The power on reset pulse was provided by the H8 / 325 Emulator unit. Refer to the Appendix " B " for its schematic.

Power Supply : Open frame switcher power supply from Kępco, Model # ECM-021K-CB was used to power up the Emulator Interconnect Board as well as the display panel. Its rating was +5V @ 2A, +12V @ 0.3A, and -12V @ 0.2A. Note that the Emulator also sources power as shown in the schematic in Appendix " B ".

Software : The H8/325 ASE system and PC resident software development tools, packages, and utilities are described very briefly:

H8 / 325 Cross Assembler : It is designed for DOS environment inside the IBM PC-AT compatible Personal Computer. When the user program is submitted as the source file, it assembles the code. Consequently, it produces Object and List files of the source program.

H8 / 325 Linker : To link various object code segments (" *.OBJ " extention) developed in parallel for a larger program. The linked file has " *.ABS " extention. Motorola " S " record conversion utility is also included with the linker, and is used as output file with " S " record format.

Load : To Load " S " Record file " INIT780B.ABS ", after the ASE system is powered up, the floppy load command shown below is issued:
:FL INIT780B.ABS;S [CR]

NOTE: H8/325 ASE COMMANDS ARE **NOT** DOS EQUIVALENT.

Demonstration File : After the program file " INIT780B.ABS " is loaded from the floppy diskette, the following commands are given to run the program:
:.pc 300 [CR]
: go [CR]

Screen Editor : Any word processing package is acceptable. In this application, Microsoft "WORD" package was used. The source programs are created and edited with this package. The source program files have " *.SRC " extention.

HARDWARE DESIGN

This section covers H8/325 microprocessor design high lights, H8/325 initialization, operation mode selection, I/O port assignments, and HD44780 design guide lines.

H8 / 325 MPU Design : This HD6473258 product was designed by SICD, Hitachi Ltd., Tokyo, Japan. Refer to the Appendix " F" for all the required product design manuals for the associated circuit design. Only high lights are addressed in this illustrative application, since LCD controller peripheral design is the main goal.

H8 / 325 Initialization : Refer to the Appendix " D " for code sample. This program was developed to scope the H8/325 waveforms in operating mode 2. The " E " and " ø " clocks were measured. RESET Emulation command can be issued used during the debug process as required when ASE is used.

" E " Clock Determination : The maximum " E " clock rate of 1 MHz is specified in the panel specs. as well as the HD44780 data sheet. Based upon it, the maximum " ø " rate of 8 MHz is established. Therefore, the H8/325 crystal should be set at 16 MHz. Then ASE system is used, the " ø " clock is set at 7.37MHz using the CLOCK Emulation command.

Operating Mode Selection : Operating Mode 2 i.e. Expanded mode with on chip ROM (32 K Bytes) address space was chosen. The associated external address space and address map is defined in the H8/320 Series Hardware Manual. The peripheral addressing is memory mapped, so please refer to it for details.
The H8/325 " E " clock timing generation was done by this MPU and so external logic was **not** required. This is one of the **strengths** of the Hitachi H8/320 Series micro processors. It was decided to exploit this feature.

I/O Port Assignments : The operating mode 2 selection also pre determined the I/O port selection. They are briefly summarized below :

- o Ports 1 and 2 ; Address Bus
- o Port 3 ; Data Bus
- o Port 7 (Partly) ; Bus Control signals
- o Port 4 (Bits 6 and 7) ; " ø " and " E " Clocks
- o Port 5 ; Serial Communication
- o Port 6 ; Interrupt Request and Free Running Timer

All the ports were listed to make sure that they were initialized **correctly** (specially port 7) to match the input output requirements of the HD44780 on the LCD panel.

HD44780 LCD Controller Driver : The reset conditions and busy flag check areas are discussed for more clarity:

Reset : The internal reset conditions or the hardware chip reset signal, timing sequence is specified in its data sheet. They are based upon the VCC on or off power sequence. If these can be assured at **all** times, no other reset e.g. software reset, is necessary for the panel. However, in case of doubt or for reliability purposes, a software power up sequence specified in the HD44780 data sheet may be executed. When contrast pot is correctly set, the panel will power up with visible character grid but no character display. Note that the software reset sequence depends upon 8 or 4 bit MPU interface. However, for this application 8 bit software reset flow chart was used.

Busy Flag Check : The HD44780 instruction execution times are shown in a table in the data sheet. When the software is designed to ensure that these execution times are **guaranteed** , to meet or exceed the specifications, **no** busy flag check is required. This will reduce the software code size but will **not** optimize the panel **data transfer** rate. Since, **minimizing** the LCD data transfer delay was **not** one of the objectives of this application, busy flag was not checked. The associated software had the built in delay to **exceed** the table of required instruction execution times.

LCD Display Panels : Although, LM016XML, LM016L, LM041L, LM044L, and LM054 panels tested in the Applications Engineering Laboratory, refer to the LM016XML specs for the remainder of this tutorial. The software coding was developed with it in mind. Minor panel dependent code changes are not shown and are left to each user for customizing the desired panel.

SOFTWARE

This section covers HD44780 software initialization code as well as the command sequence flow chart. For more coding details refer to the associated listing for the demonstration program " INIT780B.ABS " in the Appendix " E ".

HD44780 Initialization : The data sheet defines the desirable flow chart for 8 bit initialization sequence. However the actual implementation code is shown below :

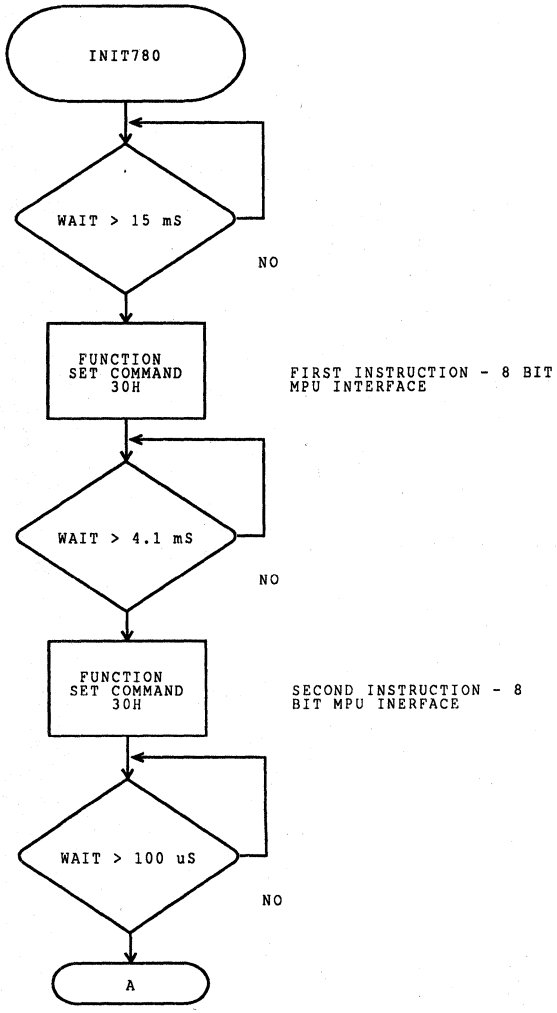


FIGURE 2

HD44780 Initialization :

This is continued in Figure 3 from the previous page :

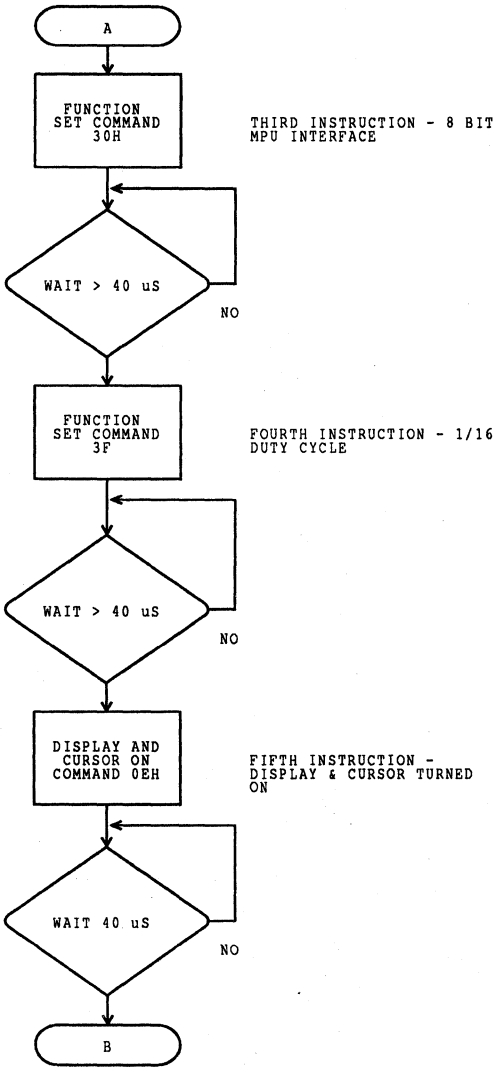


FIGURE 3

HD44780 Initialization :

This is continued in Figure 4 from the previous page :

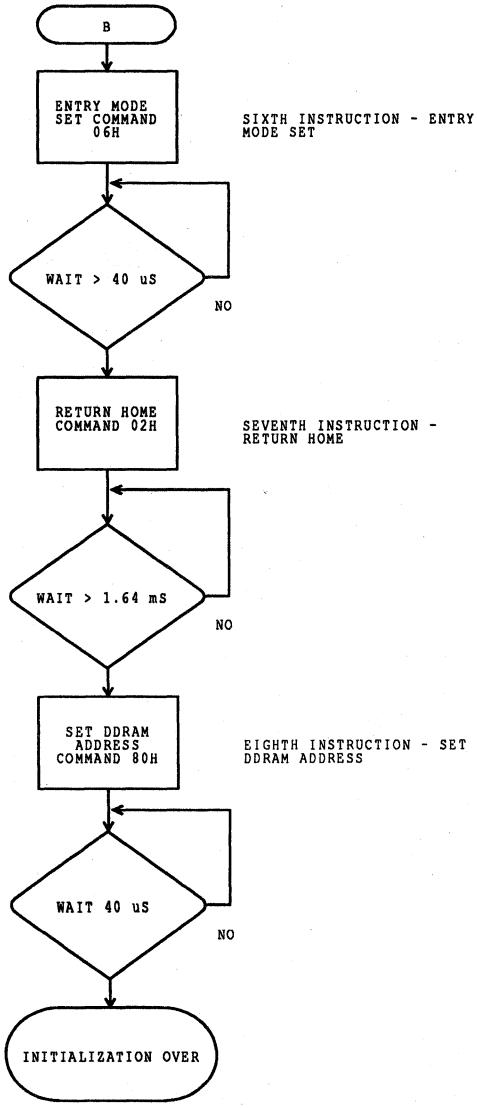


FIGURE 4

HD44780 Data Transfer :

This is continued in Figure 5 from the previous page :

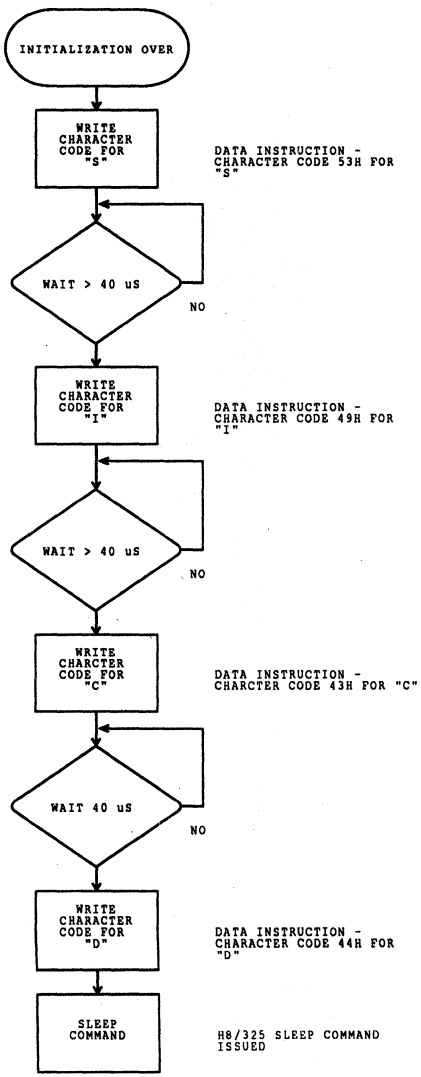
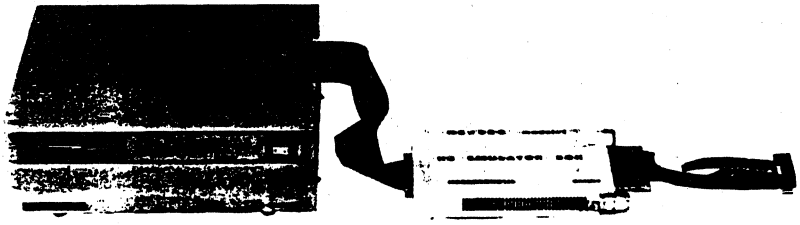


FIGURE 5

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDICES

APPENDIX " A "



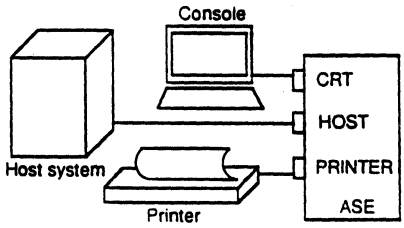
Features

- Realtime emulation
- A wide selection of emulation commands, promoting efficient development for many functions
- Operability as a stand-alone system, connected to an RS-232C interface console
- A 5.25-inch floppy disk drive, which facilitates:
  - — Loading, saving, and verifying user system memory contents
  - — Saving emulation results
  - — Input, edit, and execution of commands using a floppy disk for external storage
- An RS-232C interface to a host system which enables:
  - — Using a host system console as an ASE console
  - — Loading, saving, and verifying the user program using host system facilities
- A Centronics printer interface for printouts of emulation results.
- Usability of the ASE station compatibility with all H-Series microprocessors
- HELP functions to assist command usage without a manual
- Command execution during emulation (called parallel mode), for example:
  - — Trace data display
  - — User memory display and modification
- Memory and clock options
  - — Emulation memory (substitute user system memory) : 64 kbytes
  - — Clock (emulation clock): 3.6864 MHz, 4.9152 MHz, 7.3728 MHz, and 9.8304 MHz

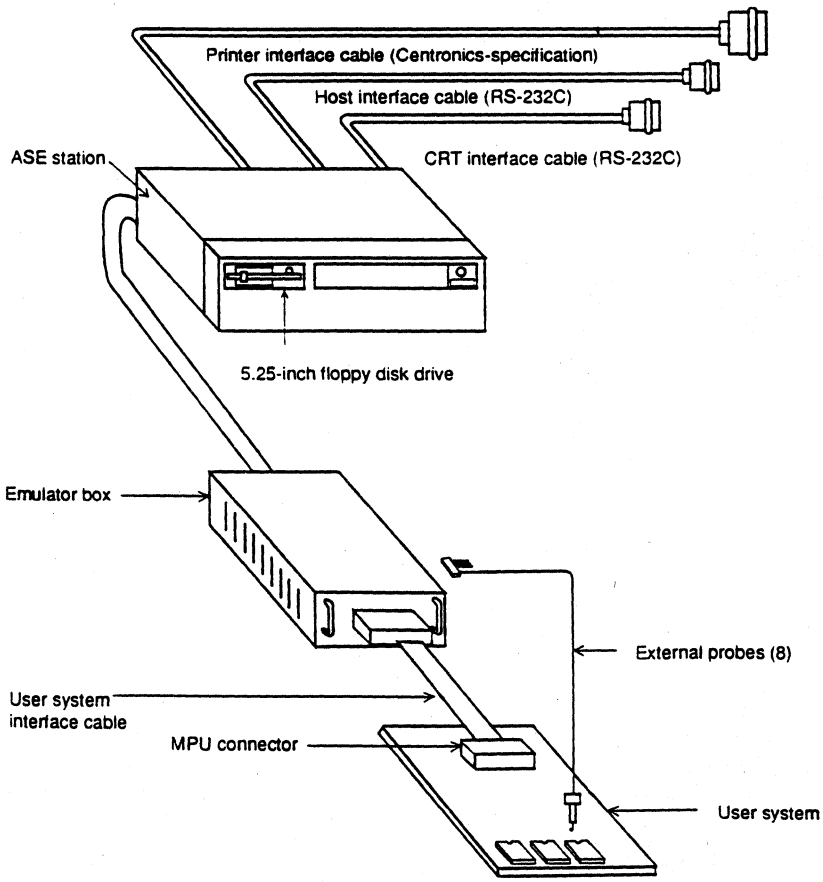
HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDIX " A "

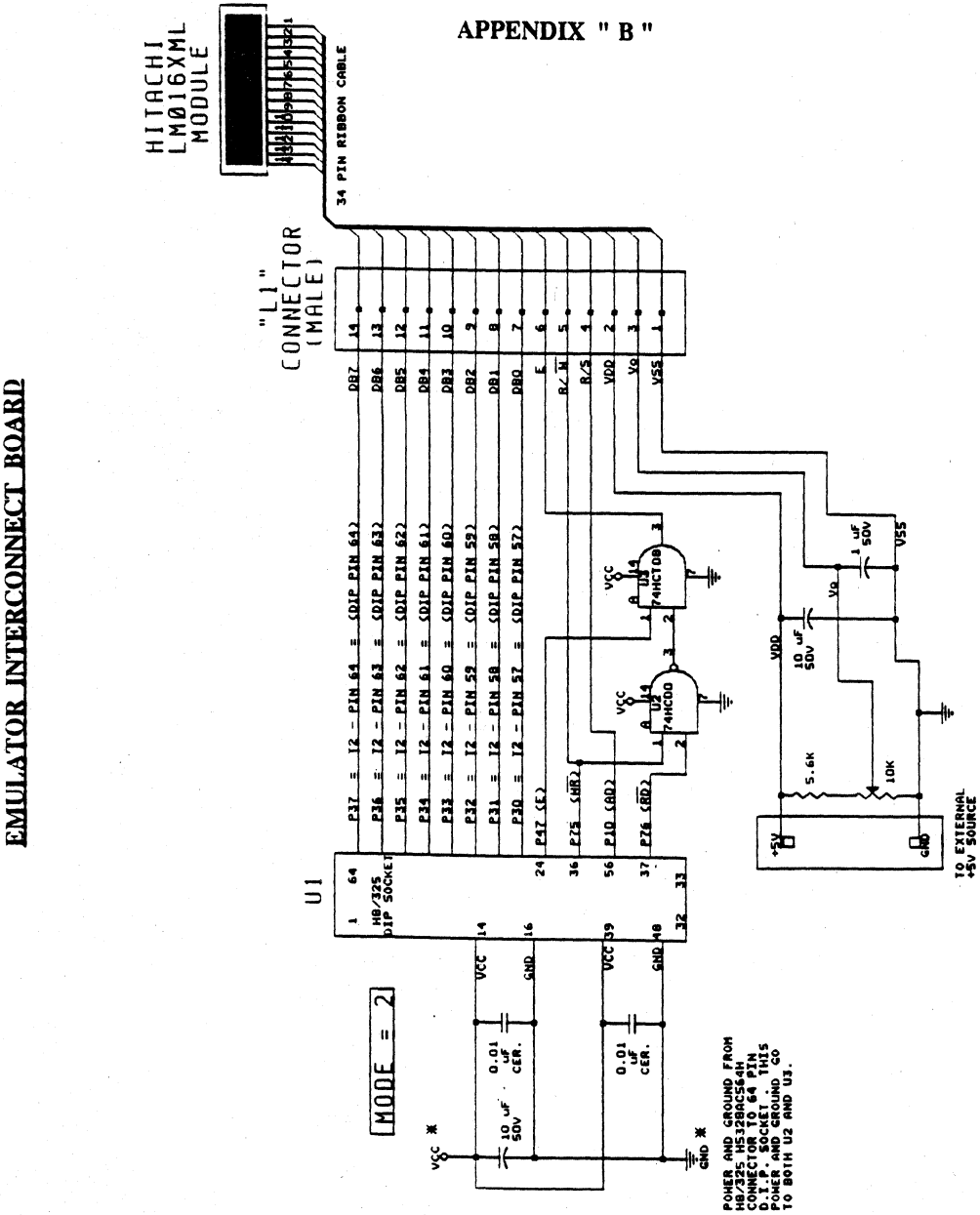
Transparent mode



ASE Components



APPENDIX " B "



HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDIX " C "

LM016XML

- 16 character x 2 lines
- Controller LSI HD44780 is built-in (See page 97).
- +5V single power supply
- Color tone ..... New gray

MECHANICAL DATA (Nominal dimensions)

Module size ..... 84W x 44H x 12T (max.) mm
Effective display area ..... 61W x 15.8H mm
Character size (5 x 7 dots) ..... 2.96W x 4.86H mm
Character pitch ..... 3.55 mm
Dot size ..... 0.56W x 0.66H mm
Weight ..... about 35 g

ABSOLUTE MAXIMUM RATINGS

	min.	max.
Power supply for logic ($V_{DD}-V_{SS}$)	0	6.5 V
Power supply for LCD drive ($V_{DD}-V_O$)	0	6.5 V
Input voltage (Vi)	V_{SS}	V_{DD} V
Operating temeprature (Ta)	0	50°C
Storage temperature (Tstg)	−20	70°C

ELECTRICAL CHARACTERISTICS

Ta = 25°C, V_{DD} = 5.0 V ± 0.25 V

Input "high" voltage (Vi_H) ..... 2.2 V min.
Input "low" voltage (Vi_L) ..... 0.6 V max.
Output "high" voltage (V_{OH}) ($-I_{OH}$ = 0.2 mA) .. 2.4 V min.
Output "low" voltage (V_{OL}) (I_{OL} = 1.2 mA) .... 0.4 V max.
Power supply current (I_{DD}) (V_{DD} = 5.0 V) .. 1.0 mA typ.
3.0 mA max.
Power supply for LCD drive (Recommended) ($V_{DD}-V_O$)
Duty = 1/16
Range of $V_{DD} - V_O$ ..... 1.5 ~ 5.25 V
Ta = 0°C ..... 4.6 V typ.
Ta = 25°C ..... 4.4 V typ.
Ta = 50°C ..... 4.2 V typ.

OPTICAL DATA ..... See page 7

INTERNAL PIN CONNECTION

Pin No.	Symbol	Level	Function	
1	V_{SS}	–	0V	Power supply
2	V_{DD}	–	+5V	
3	V_O	–	–	
4	RS	H/L	L: Instruction code input H: Data input	
5	R/W	H/L	H: Data read (LCD module→MPU) L: Data write (LCD module←MPU)	
6	E	H, H→L	Enable signal	
7	DB0	H/L	Data bus line Note (1), (2)	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		

Notes:

In the HD44780, the data can be sent in either 4-bit 2-operation or 8-bit 1-operation so that it can interface to both 4 and 8 bit MPU's.

(1) When interface data is 4 bits long, data is transferred using only 4 buses of DB_4~DB_7 and DB_0~DB_3 are not used. Data transfer between the HD44780 and the MPU completes when 4-bit data is transferred twice. Data of the higher order 4 bits (contents of DB_4~DB_7 when interface data is 8 bits long) is transferred first and then lower order 4 bits (contents of DB_0~DB_3 when interface data is 8 bits long).

(2) When interface data is 8 bits long, data is transferred using 8 data buses of DB_0~DB_7.

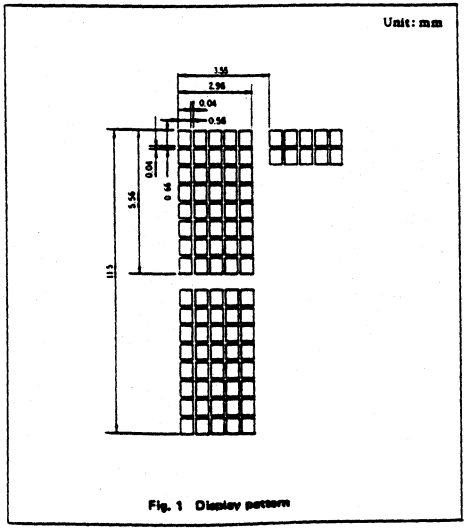


Fig. 1 Display pattern

SECTION 4

LM016L

APPENDIX " C "

- 16 character x 2 lines
- Controller LSI HD44780 is built-in (See page 97).
- +5V single power supply

MECHANICAL DATA (Nominal dimensions)

Module size ..... 84W x 44H x 12T (max.) mm
Effective display area ..... 61W x 15.8H mm
Character size (5 x 7 dots) ..... 2.96W x 4.86H mm
Character pitch ..... 3.55 mm
Dot size ..... 0.56W x 0.66H mm
Weight ..... about 35 g

ABSOLUTE MAXIMUM RATINGS min. max.

Power supply for logic ($V_{DD}-V_{SS}$) ..... 0 6.5 V
Power supply for LCD drive
($V_{DD}-V_O$) ..... 0 6.5 V
Input voltage (Vi) ..... V_{SS} V_{DD} V
Operating temeprature (Ta) ..... 0 50°C
Storage temperature (Tstg) ..... −20 70°C

ELECTRICAL CHARACTERISTICS

Ta = 25°C, V_{DD} = 5.0 V ± 0.25 V
Input "high" voltage (Vi_H) ..... 2.2 V min.
Input "low" voltage (Vi_L) ..... 0.6 Vmax.
Output high voltage (V_{OH}) ($-I_{OH}$ = 0.2 mA) .. 2.4 V min.
Output low voltage (V_{OL}) (I_{OL} = 1.2 mA) .... 0.4 Vmax.
Power supply current (I_{DD}) (V_{DD} = 5.0 V) .. 1.0 mA typ.
3.0 mA max.
Power supply for LCD drive (Recommended) ($V_{DD}-V_O$)
Duty = 1/16
Range of $V_{DD}-V_O$ ..... 1.5~5.25 V
Ta = 0°C ..... 4.6 V typ.
Ta = 25°C ..... 4.4 V typ.
Ta = 50°C ..... 4.2 V typ.

OPTICAL DATA ..... See page 7

INTERNAL PIN CONNECTION

Pin No.	Symbol	Level	Function	
1	V_{SS}	–	0V	Power supply
2	V_{DD}	–	+5V	
3	V_O	–	–	
4	RS	H/L	L: Instruction code input H: Data input	
5	R/W	H/L	H: Data read (LCD module→MPU) L: Data write (LCD module←MPU)	
6	E	H, H→L	Enable signal	
7	DB0	H/L	Data bus line Note (1), (2)	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		

Notes:

In the HD44780, the data can be sent in either 4-bit 2-operation or 8-bit 1-operation so that it can interface to both 4 and 8 bit MPU's.

(1) When interface data is 4 bits long, data is transferred using only 4 buses of DB_4~DB_7 and DB_0~DB_3 are not used. Data transfer between the HD44780 and the MPU completes when 4-bit data is transferred twice. Data of the higher order 4 bits (contents of DB_4~DB_7 when interface data is 8 bits long) is transferred first and then lower order 4 bits (contents of DB_0~DB_3 when interface data is 8 bits long).

(2) When interface data is 8 bits long, data is transferred using 8 data buses of DB_0~DB_7.

Unit: mm

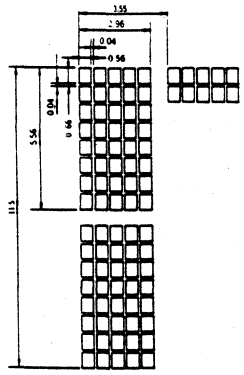


Fig. 1 Display pattern

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDIX " C "

LM041L

- 16 character x 4 lines
- Controller LSI HD44780 is built-in (See page 97).
- +5V single power supply

MECHANICAL DATA (Nominal dimensions)

Module size ..... 87W x 60H x 12T (max.) mm
Effective display area ..... 61.8W x 25.2H mm
Character size (5 x 7 dots) ..... 2.95W x 4.15H mm
Character pitch ..... 3.55 mm
Dot size ..... 0.55W x 0.55H mm
Weight ..... about 60g

ABSOLUTE MAXIMUM RATINGS

	min.	max.
Power supply for logic ($V_{DD}-V_{SS}$)	0	6.5 V
Power supply for LCD drive ($V_{DD}-V_O$)	0	6.5 V
Input voltage (V_I)	V_{SS}	V_{DD} V
Operating temperature (Ta)	0	50°C
Storage temperature (Tstg)	−20	70°C

ELECTRICAL CHARACTRISTICS

Ta=25°C, V_{DD}=5.0V±0.25V
Input "high" voltage (V_{IH}) ..... 2.2V min.
Input "low" voltage (V_{IL}) ..... 0.6V max.
Output "high" voltage (V_{OH}) ($-I_{OH}$=0.2mA) . 2.4V min.
Output "low" voltage (V_{OL}) (I_{OL}=1.2mA) ... 0.4V max.
Power supply current (I_{DD}) (V_{DD}=5.0V) ... 2.0 mA typ.
3.0 mA max.
Power supply for LCD drive (Recommended) ($V_{DD}-V_O$)
Duty = 1/16
Range of $V_{DD}-V_O$ ..... 1.5~5.25 V
Ta=0°C ..... 4.6 V typ.
Ta=25°C ..... 4.4 V typ.
Ta=50°C ..... 4.2 V typ.

OPTICAL DATA ..... See page 7

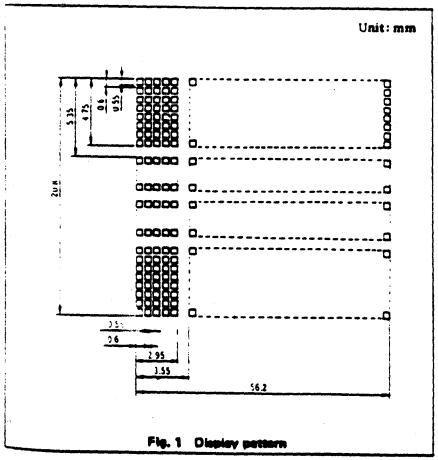


Fig. 1 Display pattern

INTERNAL PIN CONNECTION

Pin No.	Symbol	Level	Function	
1	V_{SS}	–	0V	Power supply
2	V_{DD}	–	+5V	
3	V_O	–	–	
4	RS	H/L	L: Instruction code input H: Data input	
5	R/W	H/L	H: Data read (LCD module→MPU) L: Data write (LCD module←MPU)	
6	E	H, H→L	Enable signal	
7	DB0	H/L	Data bus line Note (1), (2)	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		

Notes:

In the HD44780, the data can be sent in either 4-bit 2-operation or 8-bit 1-operation so that it can interface to both 4 and 8 bit MPU's.

(1) When interface data is 4 bits long, data is transferred using only 4 buses of DB_4~DB_7 and DB_0~DB_3 are not used. Data transfer between the HD44780 and the MPU completes when 4-bit data is transferred twice. Data of the higher order 4 bits (contents of DB_4~DB_7 when interface data is 8 bits long) is transferred first and then lower order 4 bits (contents of DB_0~DB_3 when interface data is 8 bits long).

(2) When interface data is 8 bits long, data is transferred using 8 data buses of DB_0~DB_7.

DISPLAY POSITION AND DD RAM ADDRESS

Character No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1st line	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
2nd line	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
3rd line	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
4th line	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF

Notes:

(1) 80 ~ DF are described in hexidecimal for DD RAM address.
(2) The set to HD44780 are "N = "1", F = "0" (2 lines 5 x 7 + cursol)."
(3) DD RAM address is no series in line. Address set is necessary to change the lines.
(4) Circuit is equal to 32 characters by 2 lines type.
(5) In case of executing shift, first line and third line are shifted continuously, also second line and fourth line. Therefore it happens that display of third line is transferred to first line.

SECTION 4

LM044L

APPENDIX " C "

- 20 character x 4 lines
- Controller LSI HD44780 is built-in (See page 97).
- +5V single power supply

MECHANICAL DATA (Nominal dimensions)

Module size ..... 98W x 60H x 12T (max.) mm
Effective display area ..... 76.0W x 25.2H mm
Character size (5 x 7 dots) ..... 2.95W x 4.15H mm
Character pitch ..... 3.55 mm
Dot size ..... 0.55W x 0.55H mm
Weight ..... about 65 g

ABSOLUTE MAXIMUM RATINGS

	min.	max.
Power supply for logic ($V_{DD}-V_{SS}$)	0	6.5 V
Power supply for LCD drive ($V_{DD}-V_O$)	0	6.5 V
Input voltage (Vi)	V_{SS}	V_{DD} V
Operating temperature (Ta)	0	50°C
Storage temperature (Tstg)	−20	70°C

ELECTRICAL CHARACTERISTICS

Ta = 25°C, V_{DD} = 5.0 V ± 0.25 V
Input "high" voltage (Vi_H) ..... 2.2 V min.
Input "low" voltage (Vi_L) ..... 0.6 V max.
Output "high" voltage (V_{OH}) ($-I_{OH}$ = 0.2 mA) 2.4 V min.
Output "low" voltage (V_{OL}) (I_{OL} = 1.2 mA) .. 0.4 V max
Power supply current (I_{DD}) (V_{DD} = 5.0 V) .. 1.0 mA typ.
3.5 mA max.
Power supply for LCD drive (Recommended) ($V_{DD}-V_O$)
Duty = 1/16
Range of $V_{DD}-V_O$ ..... 1.5~5.25 V
Ta = 0°C ..... 4.6 V typ.
Ta = 25°C ..... 4.4 V typ.
Ta = 50°C ..... 4.2 V typ.

OPTICAL DATA ..... See page 7

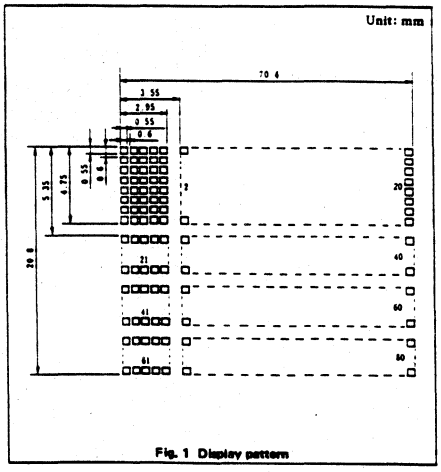


Fig. 1 Display pattern

INTERNAL PIN CONNECTION

Pin No.	Symbol	Level	Function	
1	V_{SS}	–	0V	Power supply
2	V_{DD}	–	+5V	
3	V_O	–	–	
4	RS	H/L	L: Instruction code input H: Data input	
5	R/W	H/L	H: Data read (LCD module→MPU) L: Data write (LCD module←MPU)	
6	E	H, H→L	Enable signal	
7	DB0	H/L	Data bus line Note (1), (2)	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		

Notes:

In the HD44780, the data can be sent in either 4-bit 2-operation or 8-bit 1-operation so that it can interface to both 4 and 8 bit MPU's.

(1) When interface data is 4 bits long, data is transferred using only 4 buses of DB_4~DB_7 and DB_0~DB_3 are not used. Data transfer between the HD44780 and the MPU completes when 4-bit data is transferred twice. Data of the higher order 4 bits (contents of DB_4~DB_7 when interface data is 8 bits long) is transferred first and then lower order 4 bits (contents of DB_0~DB_3 when interface data is 8 bits long).

(2) When interface data is 8 bits long, data is transferred using 8 data buses of DB_0~DB_7.

DISPLAY POSITION AND DD RAM ADDRESS

Character No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1st line	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
2nd line	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3
3rd line	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
4th line	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7

Notes:

(1) 80 ~ E7 are described in hexidecimal for DD RAM address.

(2) Function setting of HD44780 should be 'N = "1"', F = "0" (2 lines of 5 x 7 + cursor).

(3) DD RAM address is no series in line. Address setting is necessary to change the lines.

(4) Circuit is equal to 40 characters by 2 lines type.

(5) In case of executing shift, first line and third line are shifted continuously, also second line and fourth line. Therefore it happens that display of third line is transferred to first line.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDIX " C "

LM054

- 8 character x 1 line
- Controller LSI HD44780 is built-in (See page 97).
- +5 V single power supply

MECHANICAL DATA (Nominal dimensions)

Module size ..... 84W x 44H x 12D (max.) mm
Effective display area ..... 61W x 15.8H mm
Character size (5 x 7 dots) ..... 6.45W x 9.4H mm
Character pitch ..... 7.15 mm
Dot size ..... 1.25W x 1.3H mm
Weight ..... about 35 g

ABSOLUTE MAXIMUM RATINGS

	min.	max.
Power supply for logic ($V_{DD}-V_{SS}$)	0	7.0 V
Power supply for LCD drive ($V_{DD}-V_O$)	0	13.5 V
Input voltage (Vi)	V_{SS}	V_{DD} V
Operating temperature (Ta)	0	50°C
Storage temperature (Tstg)	−20	70°C

ELECTRICAL CHARACTERISTICS

Ta = 25°C, V_{DD} = 5.0 V ± 0.25 V

Input "high" voltage (Vi_H) ..... 2.2 V min.
Input "low" voltage (Vi_L) ..... 0.6 V max.
Output high voltage (V_{OH}) ($-I_{OH}$ = 0.2 mA) . 2.4 V min.
Output low voltage (V_{OL}) (I_{OL} = 1.2 mA) ... 0.4 V max.
Power supply current (I_{DD}) (V_{DD} = 5.0 V) .. 1.0 mA typ.
2.0 mA max.
Power supply for LCD drive (Recommended) ($V_{DD}-V_O$)
Duty = 1/8
Range of $V_{DD}-V_O$ ..... 1.5~5.25V
Ta = 0°C ..... 4.2 V typ.
Ta = 25°C ..... 3.7 V typ.
Ta = 50°C ..... 3.2 V typ.

OPTICAL DATA ..... See page 7

INTERNAL PIN CONNECTION

Pin No.	Symbol	Level	Function	
1	V_{SS}	–	0V	Power supply
2	V_{DD}	–	+5V	
3	V_O	–	–	
4	RS	H/L	L: Instruction code input H: Data input	
5	R/W	H/L	H: Data read (LCD module→MPU) L: Data write (LCD module←MPU)	
6	E	H, H→L	Enable signal	
7	DB0	H/L	Data bus line Note (1), (2)	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		

Notes:

In the HD44780, the data can be sent in either 4-bit 2-operation or 8-bit 1-operation so that it can interface to both 4 and 8 bit MPU's.

(1) When interface data is 4 bits long, data is transferred using only 4 buses of DB_4~DB_7 and DB_0~DB_3 are not used. Data transfer between the HD44780 and the MPU completes when 4-bit data is transferred twice. Data of the higher order 4 bits (contents of DB_4~DB_7 when interface data is 8 bits long) is transferred first and then lower order 4 bits (contents of DB_0~DB_3 when interface data is 8 bits long).

(2) When interface data is 8 bits long, data is transferred using 8 data buses of DB_0~DB_7.

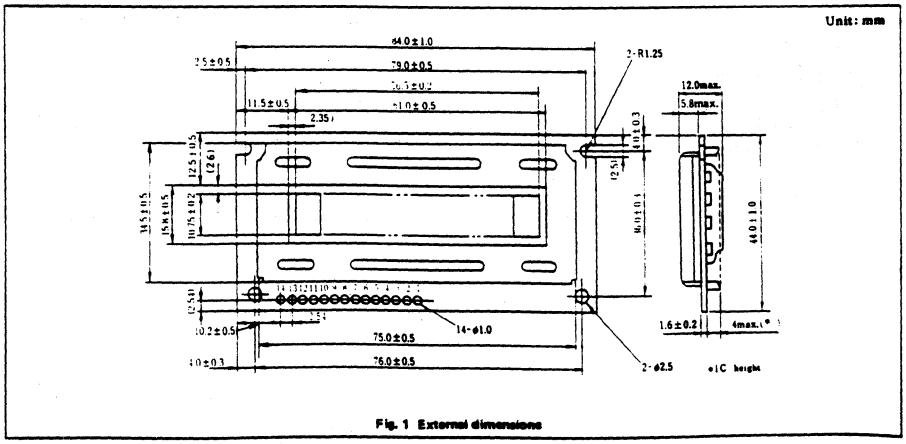


Fig. 1 External dimensions

APPENDIX " D "

```
Microtec Research ASMH83     Version 1.0A     Sep 26 11:38:06 1991     Page 1

Command line:  C:\ASMH83\ASMH83.EXE -l init.src
Line        Addr
1
2                                       ; ASE SYSTEM TO PROVIDE RESET PULSE
3
4                                       ; H8/325  INITIALIZATION  -  MODE  2
5
6                                       ;
7
8                                       .PROGRAM INIT
9                                       .SECTION CODE
10                                      .ALIGN 2
11
12                                              .ORG            H'100
13
14          0100 F8FF                           MOV.B   #H'FF, R0L      ; PORT 1 DDR - (A7-A0) ADR BUS OUT
15          0102 38B0                           MOV.B   R0L, @H'FFB0    ; ON CHIP ADDRESS
16          0104 0000                           NOP
17          0106 F8FF                           MOV.B   #H'FF, R0L      ; PORT 2 DDR - (A15-A0) ADR BUS OUT
18          0108 38B1                           MOV.B   R0L, @H'FFB1    ; ON CHIP ADDRESS
19          010A 0000                           NOP
20          010C 0000                           NOP
21                                      ; PORT 3 - DATA BUS - SET AUTOMATICALLY
22                                      ; PORT 4 - INITIALIZED TO OUTPUT CLOCKS "PHI" & "E"
23                                      ; PORT 5 - COMMUNICATION PORT - NOT USED
24                                      ; PORT 6 - FREE RUNNING TIMER, NOT (INTRQ 0 TO 3) - NOT USED
25                                      ;
26          010E 0000                           NOP
27                                      ;
28          0110 F87A                           MOV.B   #H'7A, R0L      ; PORT 7 DDR - CONTROL BUS
29          0112 38BC                           MOV.B   R0L, @H'FFBC    ; ON CHIP ADDRESS
30                                      ;
31          0114 6AC8 9001              LABEL   MOVTPE  R0L, @H'9001:16
32                                      ;
33          0118 6B00 8002                      MOV.W   @H'8002, R0
34          011C 6B00 8000                      MOV.W   @H'8000, R0
35          0120 40F2                           BRA             LABEL
36                                      ;
37          0122 0000                           NOP                     ; SCOPE THE H8/325 WAVEFORMS ON ALL
'INS
 8                                              .END
```

APPENDIX " E "

```
Microtec Research ASMH83    Version 1.0A    Oct 09 10:25:36 1991    Page 1

Command line:  C:\ASMH83\ASMH83.EXE -l INIT780B.SRC
Line        Addr
1                                   ; HITACHI LCD CHARACTER DISPLAY CONTROLLER
2
3                                       ; HD44780 INITIALIZATION FOR 8 BIT MCU INTERFACE
4
5                                       ; MCU - HITACHI H8/325 PROCESSOR
6
7                                       .PROGRAM INIT780B
8                                       .SECTION.CODE
9                                       .ALIGN 2
10
11                                                  .ORG        H'300       ; SET PC AT ADDRESS 300H
12                                  ;
13          0300 F8FF                       MOV.B   #H'FF, R0L      ; PORT 1 DDR - (A7-A0) ADR BUS OUT
14          0302 38B0                       MOV.B   R0L, @H'FFB0    ; ON CHIP ADDRESS
15          0304 0000                       NOP
16          0306 F8FF                       MOV.B   #H'FF, R0L      ; PORT 2 DDR - (A15-A0) ADR BUS OUT
17          0308 38B1                       MOV.B   R0L, @H'FFB1    ; ON CHIP ADDRESS
18          030A 0000                       NOP
19          030C 0000                       NOP
20                                      ; PORT 3 - DATA BUS - SET AUTOMATICALLY
21                                      ; PORT 4 - INITIALIZED TO OUTPUT CLOCKS "PHI" & "E"
22                                      ; PORT 5 - COMMUNICATION PORT - NOT USED
23                                      ; PORT 6 - FREE RUNNING TIMER, NOT (INTRQ 0 TO 3) - NOT USED
24                                      ;
25          030E 0000                       NOP                     ; SCOPE THE H8/325 WAVEFORMS ON ALL PINS
26
27                                  ; PRESET DELAY COUNTS IN GENERAL REGISTERS
28                                  ;
29          0310 F164                           MOV.B   #H'64, R1H      ; COUNT EQ 100H FOR 15 mS
30          0312 F21C                           MOV.B   #H'1C, R2H      ; COUNT EQ 28 FOR 4.1 mS
31          0314 FA0B                           MOV.B   #H'0B, R2L      ; COUNT EQ 11 FOR 1.64mS
32          0316 F9FF                           MOV.B   #H'FF, R1L      ; BASE TICKER PRESET - 150 uS
33                                  ;
34                                  ; START DELAY 1 - EXEEDS 15 mS
35
36          0318 1A09               A           DEC         R1L                 ; DECR BASE TICKER
37          031A 46FC                           BNE         A                   ; R1L COUNT DOWN CONTINUES
                                                                                ; BASE TICKER COUNT OVER
39          031C 1A01                           DEC         R1H                 ; DECR SIGNIFICANT TIMER
40          031E 46F8                           BNE         A                   ; R1H COUNT DOWN CONTINUES
41                                                                              ; 15 mS DELAY OVER
42                                  ;
43                                  ;    FIRST INSTRUCTION - 8 BIT INTERFACE = 30H
44                                  ;
45          0320 FB30                       MOV.B   #H'30, R3L      ; INSTRUCTION CODE = 30H
46          0322 6ACB 9000                  MOVTPE  R3L,@H'9000:16  ; PERIPHERAL WRITE WITH E CLOCK
47          0326 0000                       NOP
48                                  ; START DELAY 2 - EXEEDS 4.1 mS
49                                  ;
50          0328 F9FF                       MOV.B   #H'FF, R1L      ; BASE TICKER PRESET COUNT= FFH
51                                  ;
52          032A 1A09               B           DEC         R1L                 ; DECR BASE TICKER
53          032C 46FC                           BNE         B                   ; R1L COUNT DOWN CONTINUES
54                                                                              ; BASE TICKER COUNT OVER
55          032E 1A02                           DEC         R2H                 ; DECR SIGNIFICANT TIMER
56          0330 46F8                           BNE         B                   ; R2H COUNT DOWN CONTINUES
57                                                                              ; 4.1 mS DELAY OVER
```

APPENDIX " E "

```
Microtec Research ASMH83    Version 1.0A    Oct 09 10:25:36 1991    Page 2

Line     Addr
58                                ;    SECOND INSTRUCTION - 8 BIT INTERFACE = 30H
60                                ;
61       0332 6ACB 9000                   MOVTPE  R3L,@H'9000:16 ; PERIPHERAL WRITE WITH E CLOCK
62       0336 0000                        NOP
63                                ; START DELAY 3 - EXEEDS 100 uS
64                                ;
65       0338 F9FF                        MOV.B   #H'FF, R1L     ; BASE TICKER PRESET COUNT= FFH
66                                ;
67       033A 1A09                C       DEC             R1L                    ; DECR BASE TICKER
68       033C 46FC                        BNE             C                      ; R1L COUNT DOWN CONTINUES
69                                                                               ; BASE TICKER COUNT OVER
70                                ;
71                                ;    THIRD INSTRUCTION - 8 BIT INTERFACE = 30H
72                                ;
73       033E 6ACB 9000                   MOVTPE  R3L,@H'9000:16 ; PERIPHERAL WRITE WITH E CLOCK
74       0342 0000                        NOP
75                                ; START DELAY 4 - EXEEDS 40 uS
76                                ;
77       0344 F9FF                        MOV.B   #H'FF, R1L     ; BASE TICKER PRESET COUNT= FFH
78                                ;
79       0346 1A09                D       DEC             R1L                    ; DECR BASE TICKER
80       0348 46FC                        BNE             D                      ; R1L COUNT DOWN CONTINUES
81                                                                               ; BASE TICKER COUNT OVER
82                                ;
83                                ;    FOURTH INSTRUCTION -FUNCTION MODE SET- DUTY CYCLE 1/16 = 3FH
84                                ;
85       034A FB3F                        MOV.B   #H'3F, R3L     ; INSTRUCTION CODE = 3FH
86       034C 6ACB 9000                   MOVTPE  R3L,@H'9000:16 ; PERIPHERAL WRITE WITH E CLOCK
87       0350 0000                        NOP
88
89                                ; START DELAY 5 - EXEEDS 40 uS
90                                ;
91       0352 F9FF                        MOV.B   #H'FF, R1L     ; BASE TICKER PRESET COUNT= FFH
92                                ;
93       0354 1A09                E       DEC             R1L                    ; DECR BASE TICKER
94       0356 46FC                        BNE             E                      ; R1L COUNT DOWN CONTINUES
95                                                                               ; BASE TICKER COUNT OVER
96
97                                ;    FIFTH INSTRUCTION - DISPLAY AND CURSOR ON = 0EH
98                                ;
99       0358 FB0E                        MOV.B   #H'0E, R3L     ; INSTRUCTION CODE = 0EH
100      035A 6ACB 9000                   MOVTPE  R3L,@H'9000:16 ; PERIPHERAL WRITE WITH E CLOCK
101      035E 0000                        NOP
102
103                               ; START DELAY 6 - EXEEDS 40 uS
104                               ;
105      0360 F9FF                        MOV.B   #H'FF, R1L     ; BASE TICKER PRESET COUNT= FFH
106                               ;
107      0362 1A09                G       DEC             R1L                    ; DECR BASE TICKER
108      0364 46FC                        BNE             G                      ; R1L COUNT DOWN CONTINUES
109                                                                               ; BASE TICKER COUNT OVER
110                               ;
111      0366 0000                        NOP
112                               ;    SIXTH INSTRUCTION - ENTRY MODE SET  = 06H
113                               ;
114      0368 FB06                        MOV.B   #H'06, R3L     ; INSTRUCTION CODE = 06H
115      036A 6ACB 9000                   MOVTPE  R3L,@H'9000:16 ; PERIPHERAL WRITE WITH E CLOCK
```

APPENDIX " E "

```
Microtec Research ASMH83    Version 1.0A    Oct 09 10:25:36 1991    Page 3

Line        Addr
116         036E 0000                       NOP
117         0370 F9FF                       MOV.B   #H'FF,R1L           ; PRESET BASE TICKER COUNT
118         0372 1A09               J       DEC         R1L                 ; DECR R1L
119         0374 46FC                       BNE         J                   ; COUNT DOWN CONTINUES
120
121                                 ;   DELAY EXEEDS 40uS
122
123
124
125                                 ;   SEVENTH INSTRUCTION - CURSOR HOME & DDRAM ADR SET TO 0 = 02H
126                                 ;
127         0376 FB02                       MOV.B   #H'02, R3L      ; INSTRUCTION CODE = 02H
128         0378 6ACB 9000                  MOVTPE  R3L,@H'9000:16 ; PERIPHERAL WRITE WITH E CLOCK
129         037C 0000                       NOP
130
131                                 ; START DELAY 7 - EXEEDS 1.64 mS
132                                 ;
133         037E F9FF                       MOV.B   #H'FF, R1L      ; BASE TICKER PRESET COUNT= FFH
134                                 ;
135         0380 1A09               H       DEC         R1L                 ; DECR BASE TICKER
136         0382 46FC                       BNE         H                   ; R1L COUNT DOWN CONTINUES
137                                                                         ; BASE TICKER COUNT OVER
138         0384 1A0A                       DEC         R2L                 ; DECR SIGNIFICANT TIMER
139         0386 46F8                       BNE         H                   ; R2L COUNT DOWN CONTINUES
140                                 ;
141                                                                         ; 1.64mS DELAY OVER
142                                 ;
143         0388 0000                       NOP
144
145                                 ;   HD44780 INITIALIZATION COMPLETE
146                                 ;   FIRST COMMAND - SET DDRAM ADDRESS - 80H
147
148         038A 0000                       NOP
149         038C FB80                       MOV.B   #H'80, R3L      ; WRITE DDRAM ADR CODE = 80H
150         038E 6ACB 9000                  MOVTPE  R3L, @H'9000:16 ; PERIPHERAL WRITE WITH E CLOCK
151         0392 0000                       NOP
152
153                                 ;   DELAY EXEEDS 40uS
154
155         0394 F9FF                       MOV.B   #H'FF,R1L           ; PRESET BASE TICKER COUNT
156         0396 1A09               K       DEC         R1L                 ; DECR R1L
157         0398 46FC                       BNE         K                   ; COUNT DOWN CONTINUES
158
159                                 ;   SECOND COMMAND - WRITE CHARACTER CODE 53H TO DDRAM
160         039A 0000                       NOP
161         039C FB53                       MOV.B   #H'53, R3L      ; CHARACTER CODE FOR "S" = 53H
162         039E 6ACB 9001                  MOVTPE  R3L, @H'9001:16 ; PERIPHERAL WRITE WITH E CLOCK
163         03A2 0000                       NOP
164                                 ;   DELAY EXEEDS 40uS
165                                 ;       NOP
166         03A4 F9FF                       MOV.B   #H'FF,R1L           ; PRESET BASE TICKER COUNT
167         03A6 1A09               L       DEC         R1L                 ; DECR R1L
168         03A8 46FC                       BNE         L                   ; COUNT DOWN CONTINUES
169
170
171                                 ;   THIRD COMMAND - WRITE CHARACTER CODE - 49H
172
173         03AA 0000                       NOP
```

SECTION 4

APPENDIX " E "

```
Microtec Research ASMH83    Version 1.0A    Oct 09 10:25:36 1991    Page 4

Line      Addr
174       03AC FB49                         MOV.B   #H'49, R3L        ; CHARACTER CODE FOR I = 49H
175       03AE 6ACB 9001                    MOVTPE  R3L, @H'9001:16   ; PERIPHERAL WRITE WITH E CLOCK
176       03B2 0000                         NOP
177
178                                 ;    DELAY EXEEDS 40uS
179
180       03B4 F9FF                         MOV.B   #H'FF,R1L              ; PRESET BASE TICKER COUNT
181       03B6 1A09                 M       DEC             R1L                    ; DECR R1L
182       03B8 46FC                         BNE             M                      ; COUNT DOWN CONTINUES
183
184                                 ;    FOURTH COMMAND - WRITE CHARACTER CODE - 43H
185       03BA 0000                         NOP
186       03BC FB43                         MOV.B   #H'43, R3L      ; CHARACTER CODE FOR C
187       03BE 6ACB 9001                    MOVTPE  R3L,@H'9001:16 ; PERIPHERAL WRITE WITH E CLOCK
188       03C2 0000                         NOP
189                                 ;
190                                 ;    DELAY EXEEDS 40uS
191
192       03C4 F9FF                         MOV.B   #H'FF,R1L              ; PRESET BASE TICKER COUNT
193       03C6 1A09                 N       DEC             R1L                    ; DECR R1L
194       03C8 46FC                         BNE             N                      ; COUNT DOWN CONTINUES
195
196                                 ;    FIFTH COMMAND - WRITE CHARACTER CODE - 44H
197       03CA 0000                         NOP
198       03CC FB44                         MOV.B   #H'44, R3L      ; CHARACTER CODE FOR D
199       03CE 6ACB 9001                    MOVTPE  R3L,@H'9001:16 ; PERIPHERAL WRITE WITH E CLOCK
200       03D2 0000                         NOP
201
202                                 ; H8/325 IN SLEEP MODE
203
204       03D4 0180                         SLEEP
205
206                                         .END
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDIX " F "

The reference literature and other documents used in this design are summarized below :

- o H8/325 Series ASE Model-I User Manual # HS328ASE01HE
- o H8/320 Series Hardware Manual # M21T102
- o H8/300 Series Programming Manual # M21T103
- o Hitachi LCD Controller/Driver LSI Data Book # M24T013
- o Hitachi Liquid Crystal Character Display Module Catalog # XX-E138
- o Hitachi ASMH83 H8/300 Assembler Manual
- o Hitachi LSI Support Tools XRAYH83 H8/300 debugger Manual
- o Hitachi MCCH83 H8/300 C Compiler Manual

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD61830B / LM200

Split Panel Scanning

Application Note

Kash Yajnik

The first tutorial described in the Hitachi document #AE150 presents in depth design process for a LCD subsystem. The HD61830B / LM200 Design Tutorial Part II document # AE151 describes custom character generation. This Application Note illustrates how LM200 panel can be considered as two panels with the displayed information scanned across them.

Its major components include H8/532 Evaluation board as the local processor, LCD Controller HD61830B, and the display panel LM200 from Hitachi ELT Division.

The HD61830B controller is designed to run in the graphics mode. The H8/532 Evaluation Board is designed by Hitachi Microsystems. The LM200 LCD panel can display 240 Dots(W) by 64 Dots(H) character or graphics data as a single panel. However, as a split panel, its upper and lower halves can each display 240 Dots (W), and 32 Dots (H) information in the graphics mode. Hitachi Monitor firmware resident on the H8/532 Evaluation Board provides the program debugging and host computer communication facilities.

By adding a laptop computer to down load the programs to the Evaluation Board, a program development station can be readily built. The H8/532 Cross Assembler, Linker, any word processor package e.g. "WORD" as screen editor, and Motorola "S" record conversion utility inside the Hitachi laptop PC complete the software development environment. The "PROCOMM" communication package is used to facilitate down load or up load of programs to the H8/532 Evaluation board.

The split scan program is listed in the Appendix "A". **No effort is made for either code or logic minimization.**

This application note is intended for the technical staff at customer sites and other Hitachi employees who are fairly familiar with LCD design guide lines. Therefore, basic LCD design principles are **not** covered.

The previously published tutorials include HD61830B LCD Controller Design, Introduction, Design Overview, Custom Character definition and display, LCD Interface Board Schematics, along with their associated Software. This application note is the last of the series and is a continuation of the Tutorial Part II.

Only the details not available in the reference section are explained at greater length in this publication, while the page 2 lists the Table Of Contents.

Refer to the subsequent pages for more information on the LM200 LCD Panel split scanning technique.

HD61830B / LM200 DESIGN - SPLIT PANEL SCANNING

TABLE OF CONTENTS

HD61830B / LM200 DESIGN - SPLIT PANEL SCANNING

1.0 INTRODUCTION :

This section describes the design goals and provides a general overview of this presentation, along with a software development listing.

The **design goals** established for this project are briefly listed below:

1.1 To use H8/532 Evaluation Board with Monitor Software.
1.2 To provide split panel LCD display with LM200 from Hitachi.
1.3 To display four data bytes in the **graphics mode** using HD61830B.
1.4 To design Interface Board for the LM200 LCD panel.
1.5 To write programs for debug and test.
1.6 To use Hitachi Laptop Personal Computer "HL320".
1.7 To use readily available software at Hitachi Field Offices for development.
1.8 To generate HD61830B / LM200 split panel scanning application note.

A brief description of the LCD display subsystem components listed above is provided in the next section as an overview. To complete the overview, a subsystem block diagram is also presented. The rest of the sections described in the Table Of Contents are expanded in greater details along with their programming data. The Appendices give the program listing, and also list the referenced literature. A copy of the LCD Interface Board schematic is also provided to illustrate the implementation details of this application.

HD61830B / LM200 DESIGN - SPLIT PANEL SCANNING

2.0 DESIGN OVERVIEW :

The LCD display subsystem components such as H8 / 532 Evaluation Board, LM200 display, LCD Interface Board, Hitachi Laptop Computer, and the related software are described in this section. At the end, a subsystem block diagram is also presented. For the HD61830B LCD Controller, and the LM200 LCD panel data sheets, as well as other related documentation refer, to the Appendix "B". This description from the HD61830B / LM200 Design Tutorial Part I and II is included only for completeness of this document, and can therefore be skipped by those familiar with them.

2.1 H8/532 Evaluation Board : This board was designed by Hitachi Micro Systems. It is provided as a training and development tool. On-board EPROM contains the Hitachi Monitor firmware used for single line assembly, disassembly, line editing, and debug purposes. Of the two serial ports, only the Terminal port is used to down load, up load, and run the programs. The I/O extention connectors "J1" and "J2" are used to connect to the LCD Interface Board. The partially decoded extented I/O space is further decoded on the LCD Interface Board. This board is designed to run at 10MHz and uses a 20 MHz crystal for that purpose. **However, in this application a 16 MHz crystal is used to provide 1MHz "E" clock to the LCD Controller HD61830B.** All the jumpers on this board are set at the factory according to their default states.

2.2 LM200 LCD Panel display : This display is provided by the Hitachi ELT Division. It is capable of displaying alpha-numeric characters as well as the graphics data. However, only graphics mode is used in this application. It is 240 dots wide and 64 dots high for single panel display. It has 1 /32 duty cycle. The serial data is clocked in at 500KHz. It runs from +5V, and -5V power supply. The customer has to solder the pins on LM200 for the appropriate connector used on the LCD Interface Board. The LM200 LCD panel mounting and the proper viewing angles are critical to a strain free LCD display. Please, handle the panels according to the care recommended by the LCD display manufacturer. The logic signals sent to the LCD panel are at CMOS levels.

2.3 LCD Interface Board : A wire wrap board was built to control the LCD panel LM200. It also exchanged data with the H8/532 Evaluation Board over the I/O extention cables "J1" and "J2". The Hitachi LCD controller HD61830B was used on the LCD Interface Board. A 4,096 byte display buffer memory was also designed to store the character data. The 500KHz dot clock required by the display was also provided on this board. The LM200 LCD panel contrast adjust potentiometer was also put on this board. Set the jumper "J10" on this board to the "C-2" position. Test connectors were also provided to help debug this board.

2.4 Hitachi Laptop Personal Computer "HL320" : It is connected to the serial terminal port of the H8/532 Evaluation Board. The connector RJ-12 is attached to the Terminal port while a male to female 25 pin adapter cable is required at the Laptop PC end. The Hitachi "HL320" PC provides the software development tools for the user programs. The program up load and down load capability is also provided by the laptop PC. The communication link is full duplex, 9600 baud, 8 bits, 1 stop bit, and no parity check.

2.0 DESIGN OVERVIEW : (CNTD.)

2.5 Software Tools : The laptop PC resident software development tools, packages, and utilities are described very breifly.

H8 / 532 Cross Assembler : It is designed for DOS environment inside the laptop Personal Computer. When the user program is submitted as the source file, it assembles the code. Consequently, it produces Object and List files of the source program. The list files with " *.LIS " extentions are reproduced in the appendices for the programs developed on the software work station.

H8/ 532 Linker : To link various object code segments (" *.OBJ " extention) developed in parallel for a larger program. The linked file has " *.ABS " extention.

Motorola " S " record Conversion Utility : It is used to convert the machine code into Motorola " S " record format for uploading it to the H8 / 532 Evaluation Board. The converted file has " *.MOT " extention.

Up Loading Of Laptop PC " S " Record file : Push " EDIT SHIFT " Key down. Depress the " PG UP " key when using " PROCOMM " package for communications. Also, select ASCII format.

Screen Editor : Any word processing package is acceptable. In this application, Microsoft "WORD" package is used. The source programs are created and edited with this package. The source program files have " *.SRC " extentions.

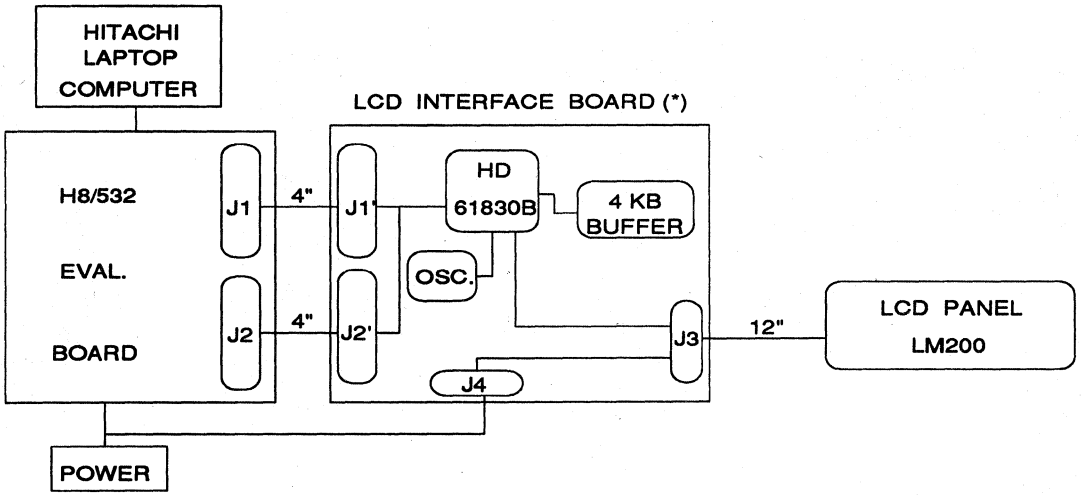
File Management Utilities : To help aid the program development, packages such as " XTREE ", or " TREE86 " may also be used.

Back -Up Utility : It is a good practice to back up program files. Such packages as " FASTBACK ", OR " FASTBACK PLUS " can also be used.

The display subsystem block diagram is shown on the next page.

HD61830B / LM200 DESIGN - SPLIT PANEL SCANNING

HD61830B / LM200 SOFTWARE STATION



* NOTE : 1.0 8 MHZ OSC. DIVIDED DOWN.
2.0 SET "J1" JUMPER TO " C-2 " POSITION.

BLOCK DIAGRAM

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

3.0 SPLIT PANEL DISPLAY :

Hitachi LCD panel LM200 with 240 dots(W) and 64 dots(H) resolution and 1 / 32 duty cycle is shown in the Figure 1. The upper and lower halves of the LM200 panel are **each** configured as separate panels with 240 dots (W) and 32 dots (H) resolution. Both, upper and lower panels display data in the **graphics** mode.

The upper half of the panel is scanned at "D1" time while the lower half of the panel is scanned at "D2" time. Both, "D1", and "D2" signals are generated by the HD61830B, operating at 1 / 32 duty cycle with horizontal pitch (HP) set at 8.

Each row can display 30 bytes (240 / 8) of graphics data and so, each half of the panel can display 32 x 30 = 960 bytes of information. If the display memory start address is set to zero, the first row (R1) will display the contents of the first 30 addresses contiguously. The Appendix " A " shows the listing of the split scan program for the LM200 panel. Similarly, the contents of the 1200 address (4B0H) are displayed in the 40 th row or the 8th row of the lower panel. Four bytes of graphics data (0H,FFH,66H, and 77H) are contiguoulsly displayed in the 40th row at positions #1, #2, #3, and #4 respectively. Refer to the Figure 1 for details of the displayed patterns :

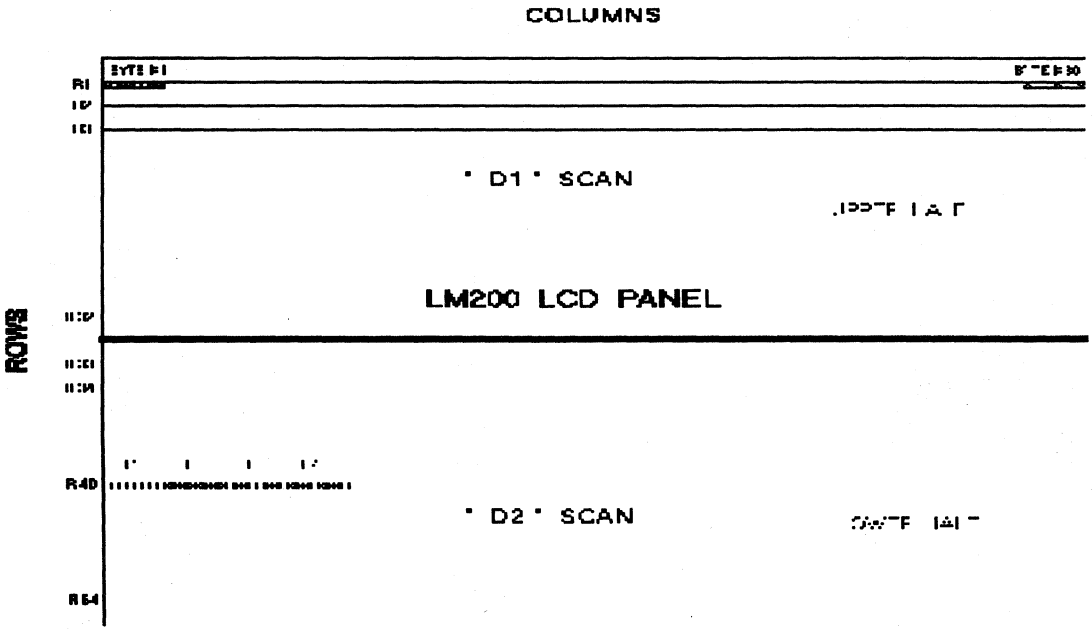


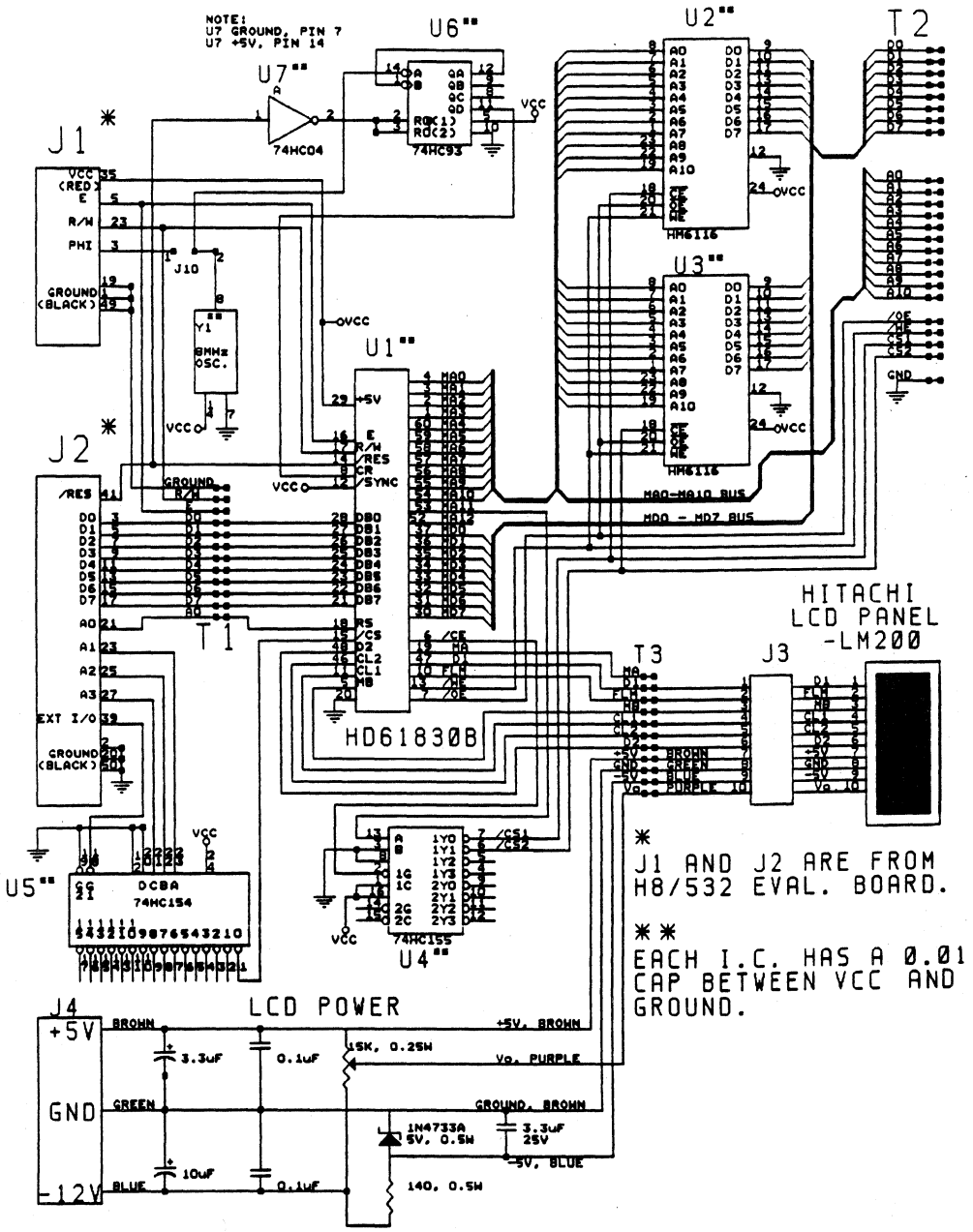
FIGURE 1

4.0 LCD INTERFACE BOARD SCHEMATIC :

The next page shows the schematic of the LCD interface board used in split panel scanning circuit. Also, note that the LCD PANEL LM200 DC power supply (+5V,G, -5V) can also be tied to the H8/532 Evaluation Board power source at **one** point. In such a case, the display contrast resistor may have to be re-adjusted. A zener regulator is added to derive the -5V DC required by the LCD panel. It generates -5V DC from the -12 V power supply provided by the external power source.

HD61830B / LM200 DESIGN - SPLIT PANEL SCANNING

LCD INTERFACE BOARD - SCHEMATIC



SECTION 4

5.0 SOFTWARE :

The software section covering the HD61830B / LM200 panel design tutorial part I shows I/O address, Busy Flag Varification, Initialization Flow Chart, Code Assembly Procedure while its Appendices show the program listings. Similarly, the software section of the Tutorial part II shows the custom character code patterns and their associated display.

For this tutorial, the "GRA-BCS.MOT" program " is listed in the Appendix "A". The "GRA-BCS.MOT" program is located at the address E000H in the H8/532 processor evaluation board memory space. When it is run, it initailizes the LCD controller HD61830B. Then clears the screen by writing 0H i.e. code for a blank graphic byte in the LCD display memory. Following a screen clear routine, the four graphics data bytes "0H","FFH","66H", and "77H" are written in the LCD display frame memory starting at address 1200 i.e.(4B0H). Since the display is memory mapped, the four graphics data bytes get displayed in the lower half of the LCD panel LM200 at the row R40 at positions #1, #2, #3, and #4 respectively. For more information on the "GRA-BCS.MOT " program refer to the Appendix "A". Although, a total of 1920 = (64 x 30) bytes of graphical bytes can be displayed on this panel in the split scan mode, for demonstartion purposes only four bytes were chosen for this application.

For register programming details , refer to the HD61830B data sheet.

The code developed in the Appendix "A" for "GRA-BCS.MOT" program can also be transported to the H8/532 Evaluation Board. It can reside within a HN27512AG -25 EPROM which **replaces** the HMS V1.2 debugger EPROM located at "U6" on the H8/532 Evaluation Board. In this manner, a stand alone display unit can also be built.

APPENDIX " A "

1.0 PROGRAM NAME - " GRA-BCS.MOT "

2.0 ADDRESS RANGE - " E000H - E199H "

3.0 PROGRAM DESCRIPTION - CLEARS SCREEN, CHECKS BUSY FLAG, AND DISPLAYS 4 GRAPHICS BYTES ON THE LCD LM200 PANEL STARTING AT THE 40TH ROW IN POSITIONS #1, #2, #3, AND #4.

HD61830B / LM200 DESIGN - SPLIT PANEL SCANNING

```
1                          .HEADING  "GRA-BCS"
2 GRA   C 0000             .SECTION  GRA,CODE,ALIGN=2
3                          .EXPORT          X
4 GRA   C E000             .ORG             H'E000          ;LOC CNTR =E000H
5
6                                           ; BUSY FLAG CHECKED
7
8 GRA   C    0000E000      X:       .EQU          $                ;X = E000H
9 GRA   C E000 A013        CLR.B            R0              ;CLEAR R0
10 GRA  C E002 A113                 CLR.B         R1               ;CLEAR R1
11 GRA  C E004 A213                 CLR.B         R2               ;CLEAR R2
12 GRA  C E006 A313                 CLR.B         R3               ;CLEAR R3
13 GRA  C E008 A413                 CLR.B         R4               ;CLEAR R4
14 GRA  C E00A 00          NOP
15 GRA  C E00B 00          NOP              ; INITIALIZATION  START
16 GRA  C E00C 00          NOP
17 GRA  C E00D 157FF10090           MOVTPE        R0,@H'7FF1       ;0H TO 7FF1
18 GRA  C E012 5112                 MOV:E         #H'12,R1   ;LOAD R1 =12H
19 GRA  C E014 157FF00091           MOVTPE        R1,@H'7FF0       ;12H TO 7FF0
20 GRA  C E019 00          NOP
21 GRA  C E01A 157FF10084     X1:   MOVFPE        @H'7FF1,R4       ;READ 7FF1 DATA TO R4
22 GRA  C E01F ACF7                 BTST          #7,R4            ;BIT TEST #7 OF R4
23 GRA  C E021 26F7                 BNE           X1               ;IF B/FLAG =Z=1 GO TO X1
24 GRA  C E023 00          NOP                                  ;B/FLAG NOT SET
25 GRA  C E024 5001                 MOV:E         #H'1,R0          ;LOAD R0=1H
26 GRA  C E026 157FF10090           MOVTPE        R0,@H'7FF1       ;1H TO 7FF1
27 GRA  C E02B 5107                 MOV:E         #H'7,R1          ;LOAD R1=7H
28 GRA  C E02D 157FF00091           MOVTPE        R1,@H'7FF0       ;7H TO 7FF0
29 GRA  C E032 00          NOP
30 GRA  C E033 157FF10084     X2:   MOVFPE        @H'7FF1,R4       ;READ 7FF1 DATA TO R4
31 GRA  C E038 ACF7                 BTST          #7,R4            ;BIT TEST #7 OF R4
32 GRA  C E03A 26F7                 BNE           X2               ;IF B/FLAG =Z=1 GO TO X2
33 GRA  C E03C 00          NOP                                  ;B/FLAG NOT SET
34 GRA  C E03D 5002                 MOV:E         #H'2,R0          ;LOAD R0=2H
35 GRA  C E03F 157FF10090           MOVTPE        R0,@H'7FF1       ;2H TO 7FF1
36 GRA  C E044 511D                 MOV:E         #H'1D,R1   ;LOAD R1=1DH
37 GRA  C E046 157FF00091           MOVTPE        R1,@H'7FF0       ;1DH TO 7FF0
38 GRA  C E04B 00          NOP
39 GRA  C E04C 157FF10084     X3:   MOVFPE        @H'7FF1,R4       ;READ 7FF1 DATA TO R4
40 GRA  C E051 ACF7                 BTST          #7,R4            ;BIT TEST #7 OF R4
41 GRA  C E053 26F7                 BNE           X3               ;IF B/FLAG =Z=1 GO TO X3
42 GRA  C E055 00          NOP                                  ;B/FLAG NOT SET
43 GRA  C E056 5003                 MOV:E         #H'3,R0          ;LOAD R0=3H
44 GRA  C E058 157FF10090           MOVTPE        R0,@H'7FF1       ;3H TO 7FF1
45 GRA  C E05D 511F                 MOV:E         #H'1F,R1   ;LOAD R1=1FH
46 GRA  C E05F 157FF00091           MOVTPE        R1,@H'7FF0       ;1FH TO 7FF0
47 GRA  C E064 00          NOP
48 GRA  C E065 157FF10084     X4:   MOVFPE        @H'7FF1,R4       ;READ 7FF1 DATA TO R4
49 GRA  C E06A ACF7                 BTST          #7,R4            ;BIT TEST #7 OF R4
50 GRA  C E06C 26F7                 BNE           X4               ;IF B/FLAG =Z=1 GO TO X4
51 GRA  C E06E 00          NOP                                  ;B/FLAG NOT SET
52 GRA  C E06F 5008                 MOV:E         #H'8,R0          ;R0=8H
53 GRA  C E071 157FF10090           MOVTPE        R0,@H'7FF1       ;8H TO 7FF1
54 GRA  C E076 157FF00092           MOVTPE        R2,@H'7FF0       ;0H TO 7FF0
55 GRA  C E07B 00          NOP
56 GRA  C E07C 157FF10084     X5:   MOVFPE        @H'7FF1,R4       ;READ 7FF1 DATA TO R4
57 GRA  C E081 ACF7                 BTST          #7,R4            ;BIT TEST #7 OF R4
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

```
58 GRA   C E083 26F7              BNE       X8          ;IF B/FLAG=Z=1 GO TO X8
59 GRA   C E085 00        NOP                     ;B/FLAG NOT SET
60 GRA   C E086 5009              MOV:E     #H9,R0      ;LOAD R0=9H
61 GRA   C E088 157FF10090        MOVTPE    R0,@H7FF1   ;9H TO 7FF1
62 GRA   C E08D 157FF00092        MOVTPE    R2,@H7FF0   ;0H TO 7FF0
63 GRA   C E092 00        NOP
64                                        ; SCREEN CLEAR ROUTINE START
65
66 GRA   C E093 AD13              CLR.W     R5          ; CLEAR R5
67 GRA   C E095 00        NOP
68 GRA   C E096 157FF10084   C1:  MOVFPE    @H7FF1,R4   ;READ 7FF1 DATA TO R4
69 GRA   C E09B ACF7              BTST      #7,R4       ;BIT TEST #7 OF R4
70 GRA   C E09D 26F7              BNE       C1          ;IF B/FLAG=Z=1 GO TO C1
71 GRA   C E09F 00        NOP                     ;B/FLAG NOT SET
72 GRA   C E0A0 500A              MOV:E     #HA,R0      ;R0=AH
73 GRA   C E0A2 157FF10090        MOVTPE    R0,@H7FF1   ;AH TO 7FF1
74 GRA   C E0A7 5100              MOV:E     #H0,R1      ;R1=0H
75 GRA   C E0A9 157FF00091        MOVTPE    R1,@H7FF0   ;0H TO 7FF0-CUR L/S=0H
76 GRA   C E0AE 00        NOP
77 GRA   C E0AF 157FF10084   C2:  MOVFPE    @H7FF1,R4   ;READ 7FF1 DATA TO R4
78 GRA   C E0B4 ACF7              BTST      #7,R4       ;BIT TEST #7 OF R4
79 GRA   C E0B6 26F7              BNE       C2          ;IF B/FLAG=Z=1 GO TO C2
80 GRA   C E0B8 00        NOP                     ;B/FLAG NOT SET
81 GRA   C E0B9 500B              MOV:E     #HB,R0      ;R0=BH
82 GRA   C E0BB 157FF10090        MOVTPE    R0,@H7FF1   ;BH TO 7FF1
83 GRA   C E0C0 5100              MOV:E     #H0,R1      ;R1=0H
84 GRA   C E0C2 157FF00091        MOVTPE    R1,@H7FF0   ;0H TO 7FF0-CUR H/S=0H
85 GRA   C E0C7 00        NOP
86 GRA   C E0C8 5DFFFF            MOV:I     #HFFFF,R5;COUNT=R5=FFFFH
87 GRA   C E0CB 00        C3:     NOP
88 GRA   C E0CC 157FF10084   C4:  MOVFPE    @H7FF1,R4   ;READ 7FF1 DATA TO R4
89 GRA   C E0D1 ACF7              BTST      #7,R4       ;BIT TEST #7 OF R4
90 GRA   C E0D3 26F7              BNE       C4          ;IF B/FLAG=Z=1 GO TO C4
91 GRA   C E0D5 00        NOP                     ;B/FLAG NOT SET
92 GRA   C E0D6 500C              MOV:E     #HC,R0      ;R0=CH
93 GRA   C E0D8 157FF10090        MOVTPE    R0,@H7FF1   ;CH TO 7FF1
94 GRA   C E0DD 5100              MOV:E     #H0,R1      ;R1=0H=CODE FOR "DOT OFF"
95 GRA   C E0DF 157FF00091        MOVTPE    R1,@H7FF0   ;0H TO 7FF0
96 GRA   C E0E4 00        NOP
97 GRA   C E0E5 01BDE3            SCB/F     R5,C3
98 GRA   C E0E8 00        NOP
99                                        ; SCREEN CLEAR ROUTINE COMPLETED
100 GRA  C E0E9 00        NOP
101 GRA  C E0EA 00                  NOP
102 GRA  C E0EB 157FF10084   X7:  MOVFPE    @H7FF1,R4   ;READ 7FF1 DATA TO R4
103 GRA  C E0F0 ACF7              BTST      #7,R4       ;BIT TEST #7 OF R4
104 GRA  C E0F2 26F7              BNE       X7          ;IF B/FLAG=Z=1 GO TO X7
105 GRA  C E0F4 00        NOP                     ;B/FLAG NOT SET
106 GRA  C E0F5 00        NOP
107 GRA  C E0F6 00        NOP                     ;INITIALIZATION DONE
108 GRA  C E0F7 00        NOP
109 GRA  C E0F8 500A              MOV:E     #HA,R0      ;R0=AH
110 GRA  C E0FA 157FF10090        MOVTPE    R0,@H7FF1   ;AH TO 7FF1
111 GRA  C E0FF 51B0              MOV:E     #HB0,R1     ;R1=B0H
112 GRA  C E101 157FF00091        MOVTPE    R1,@H7FF0   ;0H TO 7FF0
113 GRA  C E106 00        NOP
114 GRA  C E107 157FF10084   X8:  MOVFPE    @H7FF1,R4   ;READ 7FF1 DATA TO R4
```

```
115 GRA   C E10C ACF7                BTST     #7,R4          ;BIT TEST #7 OF R4
116 GRA   C E10E 26F7                BNE      X8             ;IF B/FLAG =Z=1 GO TO X8
117 GRA   C E110 00        NOP                        ;B/FLAG NOT SET
118 GRA   C E111 500B                MOV:E    #H'B,R0        ;R0=BH
119 GRA   C E113 157FF10090          MOVTPE   R0,@H'7FF1     ;BH TO 7FF1
120 GRA   C E118 5204                MOV:E    #H'4,R2        ;R2=04H
121 GRA   C E11A 157FF00092          MOVTPE   R2,@H'7FF0     ;04H TO 7FF0
122 GRA   C E11F 00        NOP
123 GRA   C E120 157FF10084    X9:   MOVFPE   @H'7FF1,R4     ;READ 7FF1 DATA TO R4
124 GRA   C E125 ACF7                BTST     #7,R4          ;BIT TEST #7 OF R4
125 GRA   C E127 26F7                BNE      X9             ;IF B/FLAG =Z=1 GO TO X9
126 GRA   C E129 00        NOP                        ;B/FLAG NOT SET
127 GRA   C E12A 500C                MOV:E    #H'C,R0        ;R0=CH
128 GRA   C E12C 157FF10090          MOVTPE   R0,@H'7FF1     ;CH TO 7FF1
129 GRA   C E131 5100                MOV:E    #H'0,R1        ;R1=00H=GRAPHIC BYTE #1
130 GRA   C E133 157FF00091          MOVTPE   R1,@H'7FF0     ;0H TO 7FF0
131 GRA   C E138 00        NOP
132 GRA   C E139 157FF10084    X10:  MOVFPE   @H'7FF1,R4     ;READ 7FF1 DATA TO R4
133 GRA   C E13E ACF7                BTST     #7,R4          ;BIT TEST #7 OF R4
134 GRA   C E140 26F7                BNE      X10            ;IF B/FLAG =Z=1 GO TO X10
135 GRA   C E142 00        NOP                        ;B/FLAG NOT SET
136 GRA   C E143 157FF10090          MOVTPE   R0,@H'7FF1     ;CH TO 7FF1
137 GRA   C E148 51FF                MOV:E    #H'FF,R1  ;R1=FF=GRAPHIC BYTE #2
138 GRA   C E14A 157FF00091          MOVTPE   R1,@H'7FF0     ;FF TO 7FF0
139 GRA   C E14F 00        NOP
140 GRA   C E150 157FF10084    X11:  MOVFPE   @H'7FF1,R4     ;READ 7FF1 DATA TO R4
141 GRA   C E155 ACF7                BTST     #7,R4          ;BIT TEST #7 OF R4
142 GRA   C E157 26F7                BNE      X11            ;IF B/FLAG =Z=1 GO TO X11
143 GRA   C E159 00        NOP                        ;B/FLAG NOT SET
144 GRA   C E15A 157FF10090          MOVTPE   R0,@H'7FF1     ;CH TO 7FF1
145 GRA   C E15F 5166                MOV:E    #H'66,R1  ;R1=66=GRAPHIC BYTE #3
146 GRA   C E161 157FF00091          MOVTPE   R1,@H'7FF0     ;66 TO 7FF0
147 GRA   C E166 00        NOP
148 GRA   C E167 157FF10084    X12:  MOVFPE   @H'7FF1,R4     ;READ 7FF1 DATA TO R4
149 GRA   C E16C ACF7                BTST     #7,R4          ;BIT TEST #7 OF R4
150 GRA   C E16E 26F7                BNE      X12            ;IF B/FLAG =Z=1 GO TO X12
151 GRA   C E170 00        NOP                        ;B/FLAG NOT SET
152 GRA   C E171 157FF10090          MOVTPE   R0,@H'7FF1     ;CH TO 7FF1
153 GRA   C E176 5177                MOV:E    #H'77,R1  ;R1=77=GRAPHIC BYTE #4
154 GRA   C E178 157FF00091          MOVTPE   R1,@H'7FF0     ;77 TO 7FF0
155 GRA   C E17D 00        NOP
156 GRA   C E17E 157FF10084    X13:  MOVFPE   @H'7FF1,R4     ;READ 7FF1 DATA TO R4
157 GRA   C E183 ACF7                BTST     #7,R4          ;BIT TEST #7 OF R4
158 GRA   C E185 26F7                BNE      X13            ;IF B/FLAG =Z=1 GO TO X13
159 GRA   C E187 00        NOP                        ;B/FLAG NOT SET
160 GRA   C E188 157FF10090          MOVTPE   R0,@H'7FF1     ;CH TO 7FF1
161 GRA   C E18D 5132                MOV:E    #H'32,R1  ;LOAD R1=32H=DISP=CH
162 GRA   C E18F 157FF00091          MOVTPE   R1,@H'7FF0     ;32H TO 7FF0
163
164 GRA   C E194 00        NOP                  ; DISPLAY DOT LIGHT = LOGIC "0"
165 GRA   C E195 00        NOP                  ; DISPLAY DOT DARK  = LOGIC "1"
166
167 GRA   C E196 00        NOP
168 GRA   C E197 00        NOP
169 GRA   C E198 1A        SLEEP                          ;HD6302 ASLEEP
170 GRA   C E199 00        NOP
```

APPENDIX " B "

REFERENCE LITERATURE

HD61830B / LM200 DESIGN - SPLIT PANEL SCANNING

The literature and other documents used in this design are summarized below :

- o H8/532 Cross Assembler Manual #S085CPC and " C " compiler for IBM PC
- o H8/532 Evaluation Board User's Manual # US538EVB21H
- o H8/532 Software User's Manual # HS538EMSS1E
- o MS " WORD " User Manual and other reference manuals
- o " PROCOMM " User Manual and other reference manuals
- o LCD Data Book #M24T013 from Hitachi America Ltd.
- o Memory Data Books from Hitachi America Ltd.
- o Hitachi Graphic Module Catalog # XX-E139 from ELT Division
- o H8/532 Hardware User's Manual #M21T002 from Hitachi America, Ltd.
- o H8/500 Programming Manual #M21T001 from Hitachi America, Ltd.
- o H8/500 Software Application Note #M21T003 from Hitachi America, Ltd.
- o H8/532 Overview #M21T173 from Hitachi America, Ltd.
- o Hitachi Laptop Personal Computer HL320 - Operator Manual
- o Hitachi Laptop Personal Computer HL320 - MSDOS V3.2 User's Manual
- o Hitachi HD61830B / LM200 Panel Design Tutorial Part I (in this manual)
- o Hitachi HD61830B / LM200 Panel Design Custom Character Generation Tutorial Part II (in this manual)

HD4074308 / PIXIE SWITCH

Automobile Dash Board Indicator

APPLICATION NOTE

Kash Yajnik

This document shows the design and implementation of the Hitachi 4 bit single chip Micro Processor HD4074308 used in automobile dash board indicator display application. Its major components include Micro Processor Module, Pixie Graphic Display Switch, and AC Power Adapter.

Fuel, Oil, Battery Voltage, and Engine Temperature gauges along with levels are displayed. The gauges are simulated by different input voltages. When they drop below or exeed their preset levels, an alarm sounds warning. By depressing the Pixie switch, the display cycles through all the four gauges.

The Hitachi Micro Processor Module contains 4 bit Micro processor HD4074308 with its instruction code and data storage. Variable voltages represent the gauge transducer inputs. The AC Power Adapter provides the required DC voltage. The gauges are graphically displayed on a Pixie switch. A buzzer is provided to sound the alarm. When the corrective action simulated by normal input volage levels, is taken, it stops the alarm.

The indicator display demonsration program is stored in the Hitachi HD4074308 Micro Processor. After power on, the Hitachi logo is displayed. When the Pixie switch is subsequently pushed; Fuel, Battery, Engine Temperature, and Oil gauge graphic is displayed on the Pixie switch one at a time. Likewise, a horizontal bar displayed above the gauge shows the associated gauge reading. The bar length changes as transducer voltages change. These levels are simulated by the appropriate potentiometers.

No effort is spent in either code or circuit minimization.

This application note is intended for the technical staff at customer sites and other Hitachi employees who are fairly familiar with design guide lines. Therefore, basic design principles are **not** covered.

This document includes Introduction, Design, Schematic, Silk Screen, Software, and the Appendices. The Appendix "A" lists the reference literature. The HD4074308 Micro Processor Features are described in the Appendix "B", while the Appendix "C" shows the IEE Pixie Switch features. The page 2 lists Table Of Contents.

Refer to the subsequent pages for more information on the details of this design.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

TABLE OF CONTENTS

1.0 INTRODUCTION :

This section describes the design goals and provides a general overview of this presentation.

The **design goals** established for this project are briefly listed below:

1.1 To develop HD4974308 Processor Module
1.2 To provide custom four gauge digital graphic display on a **single** IEE Pixie Switch
1.3 To simulate gauge transducer viltages
1.4 To provide Indicator Display Demonstration Program
1.5 To build a stand alone Pixie Switch Display Demonstration Unit
1.6 To generate Automobile Dash Board Indicator Apllication Note

A brief design description is provided in the section 2.0. Its schematic and silk screen copy are shown in the section 3.0. The software section describes the available tools and demonstration program details. The Appendix "A" lists the reference literature used in this project. The HD4074308 4 bit single chip microprocessor features are described in the Appendix "B". The IEE Pixie LCD Graphic Switch features are briefly stated in the Appendix "C".

2.0 DESIGN :

The LCD display subsystem components such as Hitachi Micro Processor Module, IEE Pixie Switch Display, and the AC Power Adapter details, along with a subsystem block diagram are presented in this section. For the Hitachi 4 bit single chip micro processor HD4074308 data sheet, and the related hardware or software information, call your nearest Hitachi America Ltd. sales office. Refer to the IEE Pixie Switch data sheet features in the appendix "C" as required. In addition, the AC Power adapter rating information is also provided for completeness.

2.1 Hitachi Micro Processor Module :

It is a 2"(W) x5"(L) PCB module with different circuit blocks. The Hitachi 4 bit micro processor HD4074308P (Z-TAT Version) is located on it and is in Dual In Line Package. 4 channel Analog to Digital inputs, and upto 25 high voltage (40V) I/O pins are built inside the HD4074308P device. It runs from a 4 MHz crystal. On board audio buzzer, and automobile dash board variable voltage gauge simulating circuit is built around the four adjustible potentiometers. Each sensor gauge potentiometer (R16 through R19) is clearly marked in the silk screen drawing. A +12V DC voltage input jack for the AC Power Adapter is also provided on this assembly. A +5V DC regulator, and a -5V DC to DC convertor are also designed to provide the required voltage for the LCD display on the Pixie switch. A 7 " ribbon cable is also supplied to connect the IEE Pixie switch assembly. For more details refer to the schematic, and silk screen drawing in the section 3.0.

2.2 Pixie Switch / LCD Display :

Each Pixie switch provides two switch terminals, back light LEDs, LCD drivers, and 864 pixels (24 dots high x 36 dots wide) LCD display. It operates in the graphics mode. A display contrast adjustment potentiometer "R1" is also located on the Hitachi Micro Processor Module. By changing "R1" the LCD display contrast can be changed for different ambient lighting conditions. The Pixie switch is mounted on a break away PCB with 4 stand offs and is connected to the Micro Processor Module by a ribbon cable. For more information on the IEE Pixie Switch, see the Appendix "C".

2.3 AC Power Adapter :

It is available from Radio Shack, A Division Of Tandy Corporation. This unit is listed as "ARCHER AC Adapter" catalog # 273-1652A. Its input is rated at 120V AC, 60 Hz, and 14 Watts. The output rating is 12V DC at 500 mA.

2.0 DESIGN : (CNTD.)

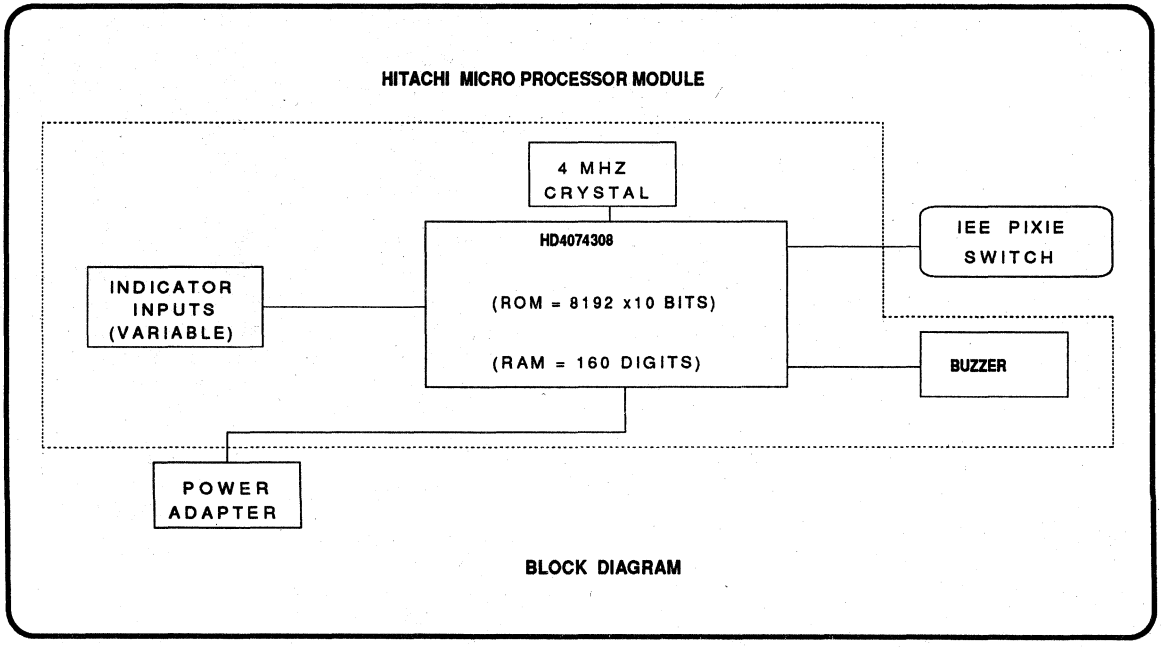
2.4 Software Tools :

For the HD4074308 program development following support tools are available from Hitachi America Ltd. ;

- o Cross Assembler and Simulator Software for use with IBM PCs and compatilbles.
- o In-Circuit emulator for IBM PC.
- o Programming Socket Adapter for programming the on-chip EPROM during firmware development.
- o Emulator can be used for both HMCS400 Series and AS micro computer.

For more information, call your Hitachi FAE at nearest sales office.

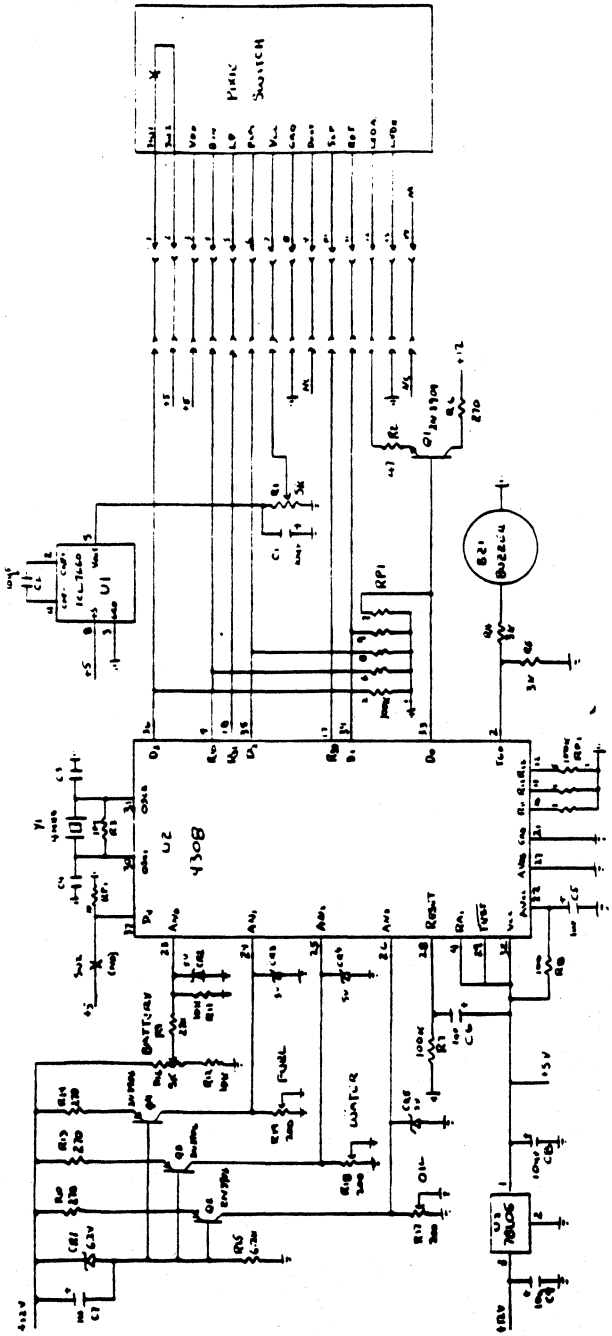
The stand alone Pixie Switch Display Unit block diagram is shown below:



Automobile Dash Board Indicator Application Note

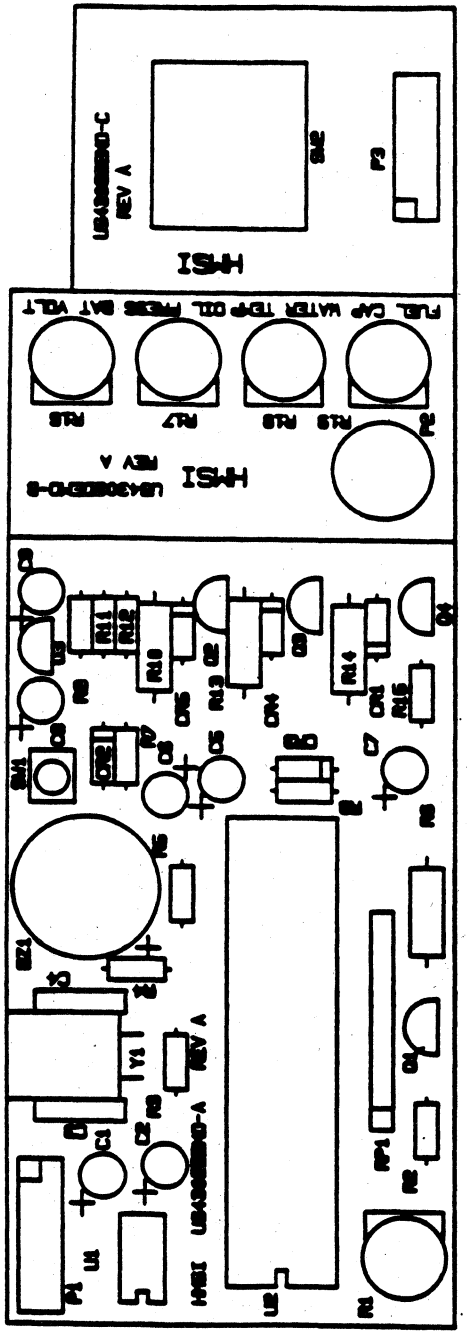
3.0 SCHEMATIC / SILKSCREEN

The Pixie switch Display unit schematic and its associated silk screen drawing are shown in this section.



3.0 SCHEMATIC / SILKSCREEN

The Pixie switch Display unit schematic and its associated silk screen drawing are shown in this section.



4.0 SOFTWARE :

The Indicator Display Demonstration Program was developed by Hitachi Micro Systems Inc. (HMSI). The customers who wish to obtain a copy of this program may do so by calling your Hitachi FAE at the nearest sales office. The source program is resident on the Hitachi Bulletin Board. Due to its length and size, the program is not reproduced in this application note.

APPENDIX " A "

REFERENCE LITERATURE

The literature and other documents used in this design are summarized below :

- Hitachi 4 Bit Single Chip. Micro Computer Data Book dated August 1989, #U76.
- IEE Pixie Graphic LCD Switch data sheet dated April 1988. For more information, call IEE, Van Nuys, CA 91409. Tel # (818) 787 0311.
- For other components used in this assembly, call their respective companies.
- For Archer AC Power Adapter information, call your nearest Radio Shack Store (Catalog # 273-1652A).

APPENDIX " B "

HD4074308 FEATURES

- 4-bit architecture
- 2048 words of 10-bit ROM (mask ROM version)
  8192 words of 10-bit ROM (ZTAT version)
- 160 digits of 4-bit RAM
- 33 I/O pins, including 25 High voltage I/O pins (40 V max)
- Two Timers/Counters
  —11-bit prescaler
  —8-bit timer (free-run timer/watchdog timer)
  —8-bit timer (reload timer/event counter)
- Five interrupt sources
  —External 2
  —Timer 2
  —A/D 1
- A/D converter: 8 bits × 4 channels
- Two Tone generator outputs: 2
- Subroutine Stack
  —Up to 16 levels including interrupts
- Two low power dissipation modes
  —Standby mode
  —Stop mode
- On-chip oscillator
  —Crystal or ceramic filter (externally drivable)
- Package
  —42-pin plastic DIP (DP-42)
  —42-pin ceramic DIP with window (DC-42)
- Instruction set compatible with HMCS412C: 100 instructions
- High programming efficiency with 10-bit/word
  ROM: 78 single-word instructions
- Direct branch to all ROM areas
- Direct or indirect addressing of all RAM areas

Program Development Support Tools

- H68/H680SD series macro assembler
- IBM-PC cross assembler
- ZTAT microcomputer (HD4074308)
- H400CMIX3 emulator
- Emulator unit, target probe, and user cable can be purchased individually.
- Emulator unit can be used for both HMCS400 series and AS microcomputer

Ordering Information

	Part No.	ROM(Words)	Package
Mask ROM type	HD404302P	2,048	DP-42
ZTAT type	HD4074308P	8,192	DP-42
	HD4074308C		DC-42 (Note)

HD4074308/PIXIE SWITCH

Automobile Dash Board Indicator Application Note

APPENDIX " C "

PIXIE SWITCH FEATURES

MECHANICAL SPECIFICATIONS

Pixels:	864 (24 x 36) LCD display with super-twist
Characters:	18 Max. (using 5 x 7 matrix, 3 lines by 6 digits)
Pixel Size:	.014 in. sq.
Viewing Area:	.490 x .590 in.

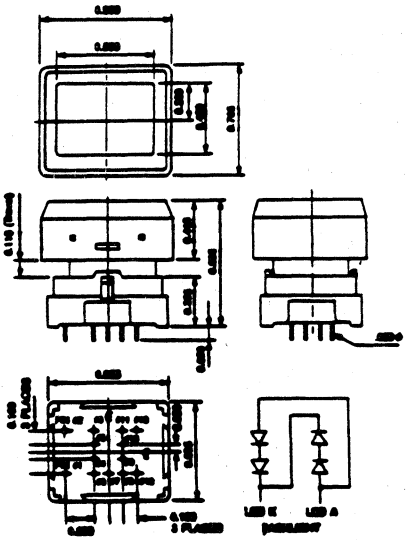
SWITCH DATA

Nominal Rating:	DC12V 50mA
Contact Resistance:	< 1Ω
Isolation Resistance:	> 50MΩ (DC200V)
Withstanding Voltage:	AC250V (1 minute)
Switch Bounce:	< 5msec
Switch Travel:	0.118 in.
Actuation Force:	130 g ±20 g
Life Expectancy:	> 1,000,000 actuations

CONNECTOR PIN ASSIGNMENTS

No.	Pin	Function	Connection
1	SW1	switch	user defined
2	SW2	switch	user defined
3	VDD	supply voltage for logic +5V	power supply
4	DIN	data input	DOUT or controller
5	LP	latch pulse	controller
6	FLM	first line marker	controller
7	VLC	supply voltage for LCD	power supply
8	GND	ground (±0V)	power supply
9	DOUT	data output	DIN
10	SCP	serial clock pulse	controller
11	RST	reset signal	controller
12	LED A	LED anode	power supply
13	LED K	LED cathode	power supply

DIMENSIONAL DRAWINGS



HITACHI

Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD61830B / LM200

Custom Character Generation

TUTORIAL PART II

Kash Yajnik

The first tutorial described in the Hitachi document #AE150 presents in depth design process for a LCD subsystem. This document is part II of that tutorial and describes custom character generation. Its major components include H8/532 Evaluation board as the local processor, LCD Controller HD61830B, and the display panel LM200 from Hitachi ELT Division, and EPROM resident custom character set.

The HD61830B controller is designed to run in the character mode. The H8/532 Evaluation Board is designed by Hitachi Microsystems. The LM200 LCD panel can display 240 Dots(W) by 64 Dots(H) character or graphics data. Hitachi Monitor firmware resident on the H8/532 Evaluation Board provides the program debugging and host computer communication facilities.

By adding a laptop computer to down load the programs to the Evaluation Board, a program development station can be readily built. The H8/532 Cross Assembler, Linker, any word processor package e.g. "WORD" as screen editor, and Motorola "S" record conversion utility inside the Hitachi laptop PC complete the software development environment. The "PROCOMM" communication package is used to facilitate down load or up load of programs to the H8/532 Evaluation board.

The custom character generation program is listed in the Appendix "A". **No effort is spent in either code or logic minimization.**

This tutorial is intended for the technical staff at customer sites and other Hitachi employees who are fairly familiar with LCD design guide lines. Therefore, basic LCD design principles are **not** covered.

The HD61830B LCD Controller design tutorial includes Introduction, Design Overview, Custom Character definition and display, LCD Interface Board Schematic, along with the associated Software.

While a lot of programs were developed, **only** one is listed as an example in the Appendix "A". The Appendix "B" covers EPROM font data while the Appendix "C" lists the reference literature.

Only the details not available in the reference section are explained at greater length in this article. The page 2 shows the Table Of Contents.

Refer to the subsequent pages for more information on the part II of this design.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

TABLE OF CONTENTS

1.0 INTRODUCTION :

This section describes the design goals and provides a general overview of this presentation, along with a software development listing.

The **design goals** established for this project are briefly listed below:

1.1 To use H8/532 Evaluation Board with Monitor Software.
1.2 To provide custom character LCD display with LM200 panel from Hitachi.
1.3 To display largest characters in the **character mode** of the HD61830B.
1.4 To design Interface Board for the LM200 LCD panel.
1.5 To write programs for debug and test.
1.6 To use Hitachi Laptop Personal Computer "HL320".
1.7 To use readily available software at Hitachi Field Offices for development.
1.8 To build a stand alone display unit.
1.9 To generate HD61830B / LM200 panel design tutorial part II.

A brief description of the LCD display subsystem components listed above is provided in the next section as an overview. To complete the overview, a subsystem block diagram is also presented. The rest of the sections described in the Table Of Contents are expanded in greater details along with their programming data. The Appendices give the program listing, EPROM font data, and also list the referenced literature. A copy of the LCD Interface Board schematic is also provided to illustrate the implementation details of this tutorial.

2.0 DESIGN OVERVIEW :

The LCD display subsystem components such as H8 / 532 Evaluation Board, LM200 display, LCD Interface Board, Hitachi Laptop Computer, and the related software are described in this section. At the end, a subsystem block diagram is also presented. For the HD61830B LCD Controller, and the LM200 LCD panel data sheets, as well as other related documentation refer, to the Appendix "C". This description from the HD61830B / LM200 Design Tutorial Part I is included only for completeness of this document, and can therefore be skipped by those familiar with the Part I.

2.1 H8/532 Evaluation Board : This board was designed by Hitachi Micro Systems. It is provided as a training and development tool. On-board EPROM contains the Hitachi Monitor firmware used for single line assembly, disassembly, line editing, and debug purposes. Of the two serial ports, only the Terminal port is used to down load, up load, and run the programs. The I/O extention connectors "J1" and "J2" are used to connect to the LCD Interface Board. The partially decoded extented I/O space is further decoded on the LCD Interface Board. This board is designed to run at 10MHz and uses a 20 MHz crystal for that purpose. **However, in this application a 16 MHz crystal is used to provide 1MHz "E" clock to the LCD Controller HD61830B.** All the jumpers on this board are set at the factory according to their default states.

2.2 LM200 LCD Panel display : This display is provided by the Hitachi ELT Division. It is capable of displaying alpha-numeric characters as well as the graphics data. However, only character mode is used in this application. It is 240 dots wide and 64 dots high. It has 1 /32 duty cycle. The serial data is clocked in at 500KHz. It runs from +5V, and -12V power supply. The customer has to solder the pins on LM200 for the appropriate connector used on the LCD Interface Board. The LM200 LCD panel mounting and the proper viewing angles are critical to a strain free LCD display. Please, handle the panels according to the care recommended by the LCD display manufacturer. The logic signals sent to the LCD panel are at CMOS levels.

2.3 LCD Interface Board : A wire wrap board was built to control the LCD panel LM200. It also exchanged data with the H8/532 Evaluation Board over the I/O extention cables "J1" and "J2". The Hitachi LCD controller HD61830B was used on the LCD Interface Board. A 4,096 byte display buffer memory was also designed to store the character data. The 500KHz dot clock required by the display was also provided on this board. The LM200 LCD panel contrast adjust potentiometer was also put on this board. Set the jumper "J10" on this board to the "C-2" position. Test connectors were also provided to help debug this board.

2.4 Hitachi Laptop Personal Computer "HL320" : It is connected to the serial terminal port of the H8/532 Evaluation Board. The connector RJ-12 is attached to the Terminal port while a male to female 25 pin adapter cable is required at the Laptop PC end. The Hitachi "HL320" PC provides the software development tools for the user programs. The program up load and down load capability is also provided by the laptop PC. The communication link is full duplex, 9600 baud, 8 bits, 1 stop bit, and no parity check.

2.0 DESIGN OVERVIEW : (CNTD.)

2.5 Software Tools : The laptop PC resident software development tools, packages, and utilities are described very breifly.

H8 / 532 Cross Assembler : It is designed for DOS environment inside the laptop Personal Computer. When the user program is submitted as the source file, it assembles the code. Consequently, it produces Object and List files of the source program. The list files with " *.LIS " extentions are reproduced in the appendices for the programs developed on the software work station.

H8/ 532 Linker : To link various object code segments (" *.OBJ " extention) developed in parallel for a larger program. The linked file has " *.ABS " extention.

Motorola " S " record Conversion Utility : It is used to convert the machine code into Motorola " S " record format for uploading it to the H8 / 532 Evaluation Board. The converted file has " *.MOT " extention.

Up Loading Of Laptop PC " S " Record file : Push " EDIT SHIFT " Key down. Depress the " PG UP " key when using " PROCOMM " package for communications. Also, select ASCII format.

Screen Editor : Any word processing package is acceptable. In this application, Microsoft "WORD" package is used. The source programs are created and edited with this package. The source program files have " *.SRC " extentions.

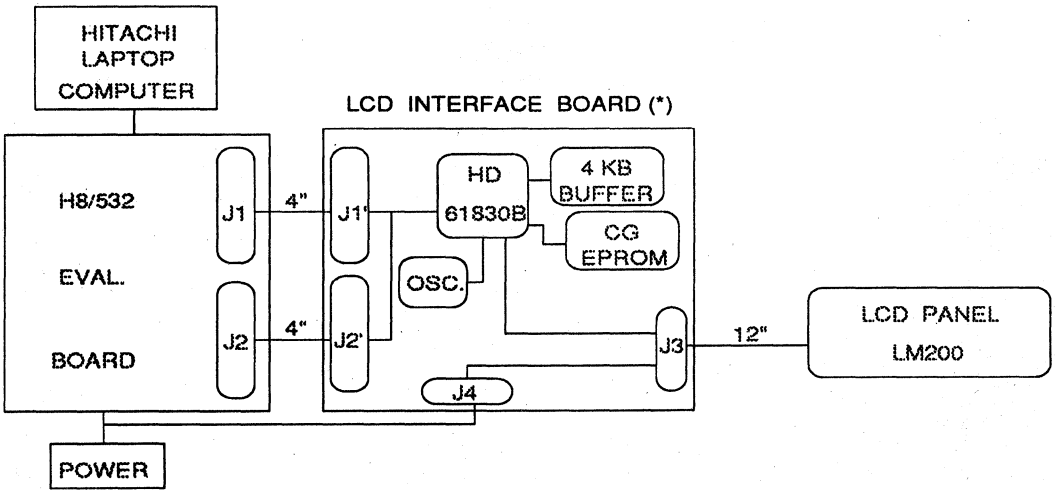
File Management Utilities : To help aid the program development, packages such as " XTREE ", or " TREE86 " may also be used.

Back -Up Utility : It is a good practice to back up program files. Such packages as " FASTBACK ", OR " FASTBACK PLUS " can also be used.

EPROM Programming : Data I/O Model 212 Multi-programmer was used to program the 32KByte or 16KByte UV erasable EPROMS.

The stand alone display unit block diagram is shown on the next page.

HD61830B / LM200 STAND ALONE UNIT



* NOTE : 1.0 8 MHZ OSC. DIVIDED DOWN.
2.0 SET "J1" JUMPER TO "C-2" POSITION.

BLOCK DIAGRAM

CUSTOM CHARACTER DEFINITION:

In character mode, for visual comparison, the maximum programmable font size of 8 (Columns) x 16 (Rows) using the Hitachi LCD controller HD61830B was chosen for display. Also, note that in the **graphics** mode of the HD61830B, **larger** font sizes beyond the character mode limits are possible. However, for this tutorial, only the character mode is considered to illustrate custom character generation using an EPROM.

The custom characters were displayed on the LCD panel LM200 from Hitachi America Ltd. Arbitrarily, four custom characters "K", "A", "S", and "H" were chosen for font generation using a 32K bytes EPROM. The Figures 1,2,3, and 4 show each character, its character code, line position, and the EPROM data output. The character shapes were **purposely** chosen to be slightly different from the standard character shapes defined in the HD61830B data sheet. This makes character verification easier and distinguishes custom characters from the standard character set.

The custom character font line position address bits (A3-A0) were connected to the (MA15-MA12) signals from the HD61830B. These four bits form the lowest address nibble of the 32KB character generator EPROM. The character code address bits (A11-A4) are also provided by the (MD7-MD0) signals from the HD61830B. These character code bits determine the **square** in the character map where the custom character is located. For more information on the character map refer to the HD61830B data sheet. The standard character in the character map is **replaced** by the custom character at the location addressed by the character code bits (A11-A4) of the EPROM.

Note that **only one** 4,096 bytes long page can be addressed by the HD61830B from the available 8 pages in the 32K Bytes EPROM space in the character mode. Therefore, the upper address bits (A12-A14) of the EPROM are grounded to select page 0. The character font EPROM output data bits (D7'-D0') are sent to the HD61830B ROM data input bus (RD7-RD0).

It is left up to the reader to come up with a scheme to locate 8 different character sets on a 4KB page boundary e.g. Eight different languages for eight countries in Europe within the 32KB EPROM character space. The appropriate font page for a country should be activated when a LCD display panel is used for a product in the specified country. Such a font design would make the product saleable in the international markets.

For the implementation details on the four custom character EPROM refer to the schematic in the section 5.

CHR	EPROM ADDRESS												EPROM OUTPUT							
	CHARACTER CODES								LINE POSITION											
	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	07	06	05	04	03	02	01	00
"K"	0	1	0	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	1	0
									0	0	0	1	0	1	0	0	0	0	1	0
									0	0	1	0	0	0	1	0	0	0	1	0
									0	0	1	1	0	0	0	1	0	0	1	0
									0	1	0	0	0	0	0	0	1	0	1	0
									0	1	0	1	0	0	0	0	0	1	1	0
									0	1	1	0	0	0	0	0	0	1	1	0
									0	1	1	1	0	0	0	0	0	1	1	0
									1	0	0	0	0	0	0	0	0	1	1	0
									1	0	0	1	0	0	0	0	0	1	1	0
									1	0	1	0	0	0	0	0	1	0	1	0
									1	0	1	1	0	0	0	1	0	0	1	0
									1	1	0	0	0	0	1	0	0	0	1	0
									1	1	0	1	0	1	0	0	0	0	1	0
									1	1	1	0	1	0	0	0	0	0	1	0
									1	1	1	1	0	0	0	0	0	0	0	0

FIGURE 1

CHR	EPROM ADDRESS												EPROM OUTPUT							
	CHARACTER CODES								LINE POSITION											
	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	07	06	05	04	03	02	01	00
	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
									0	0	0	1	0	0	1	0	1	0	0	0
									0	0	1	0	0	1	0	0	0	1	0	0
									0	0	1	1	1	0	0	0	0	0	1	0
									0	1	0	0	1	0	0	0	0	0	1	0
									0	1	0	1	1	0	0	0	0	0	1	0
"A"									0	1	1	0	1	0	0	0	0	0	1	0
									0	1	1	1	1	0	0	0	0	0	1	0
									1	0	0	0	1	1	1	1	1	1	1	0
									1	0	0	1	1	0	0	0	0	0	1	0
									1	0	1	0	1	0	0	0	0	0	1	0
									1	0	1	1	1	0	0	0	0	0	1	0
									1	1	0	0	1	0	0	0	0	0	1	0
									1	1	0	1	1	0	0	0	0	0	1	0
									1	1	1	0	1	0	0	0	0	0	1	0
									1	1	1	1	0	0	0	0	0	0	0	0

FIGURE 2

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

CHR	EPROM ADDRESS												EPROM OUTPUT							
	CHARACTER CODES								LINE POSITION											
	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	O7	O6	O5	O4	O3	O2	O1	O0
	0	1	0	1	0	0	1	1	0	0	0	0	1	1	1	1	1	1	1	0
									0	0	0	1	0	0	0	0	0	0	1	0
									0	0	1	0	0	0	0	0	0	0	1	0
									0	0	1	1	0	0	0	0	0	0	1	0
									0	1	0	0	0	0	0	0	0	0	1	0
									0	1	0	1	0	0	0	0	0	0	1	0
"S"									0	1	1	0	0	0	0	0	0	0	1	0
									0	1	1	1	1	1	1	1	1	1	1	0
									1	0	0	0	1	0	0	0	0	0	0	0
									1	0	0	1	1	0	0	0	0	0	0	0
									1	0	1	0	1	0	0	0	0	0	0	0
									1	0	1	1	1	0	0	0	0	0	0	0
									1	1	0	0	1	0	0	0	0	0	0	0
									1	1	0	1	1	0	0	0	0	0	0	0
									1	1	1	0	1	1	1	1	1	1	1	0
									1	1	1	1	0	0	0	0	0	0	0	0

FIGURE 3

SECTION 4

CHR	EPROM ADDRESS												EPROM OUTPUT							
	CHARACTER CODES								LINE POSITION											
	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	07	06	05	04	03	02	01	00
	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
									0	0	0	1	1	0	0	0	0	0	1	0
									0	0	1	0	1	0	0	0	0	0	1	0
									0	0	1	1	1	0	0	0	0	0	1	0
									0	1	0	0	1	0	0	0	0	0	1	0
									0	1	0	1	1	0	0	0	0	0	1	0
"H"									0	1	1	0	1	0	0	0	0	0	1	0
									0	1	1	1	1	1	1	1	1	1	1	0
									1	0	0	0	1	0	0	0	0	0	1	0
									1	0	0	1	1	0	0	0	0	0	1	0
									1	0	1	0	1	0	0	0	0	0	1	0
									1	0	1	1	1	0	0	0	0	0	1	0
									1	1	0	0	1	0	0	0	0	0	1	0
									1	1	0	1	1	0	0	0	0	0	1	0
									1	1	1	0	1	0	0	0	0	0	1	0
									1	1	1	1	0	0	0	0	0	0	0	0

FIGURE 4

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

CUSTOM CHARACTER DISPLAY :

The horizontal character pitch (HP) is 8. Also, the inter character space is 1 row horizontally and 1 column vertically. Programming a logic 1 inside the EPROM corresponds to a lighted dot on the LCD panel LM200. It appears as a dark dot on the plain background. A variable resistor is also provided to adjust the contrast ratio. The character display for each of the four characters "K", "A", "S", and "H" is shown in the Figures 5,6,7, and 8 respectively. Observe the display pattern inversion from the corresponding programmed patterns of the EPROM illustrated in the Figures 1,2,3, and 4 of the section 3. The Appendix "B" shows the EPROM font data and its addresses along with a check sum. The Data I/O model 212 Multi Programmer was used for programming the 32K Bytes Hitachi EPROM HNC256AG-15 or 16K Bytes Hitachi EPROM HN27128AG-17. Both the EPROMS were used for generating the same four custom characters defined in the section 3 and displayed in the Figures of the section 4.

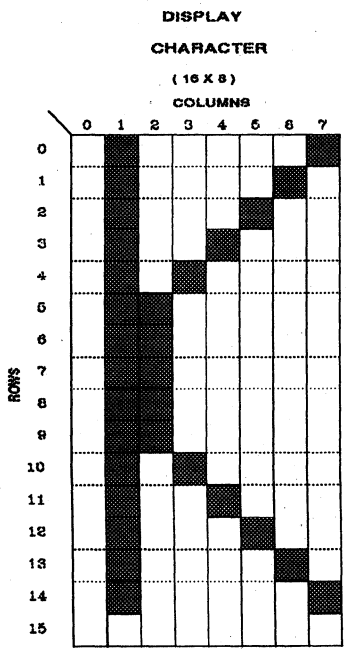


FIGURE 5

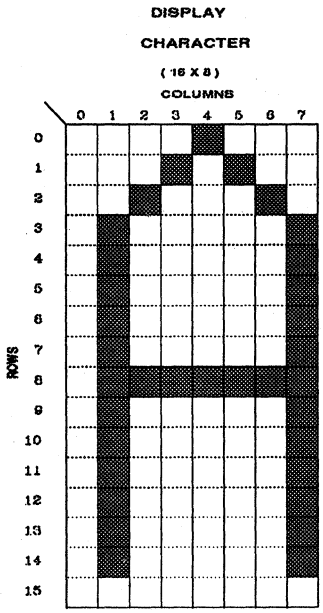


FIGURE 6

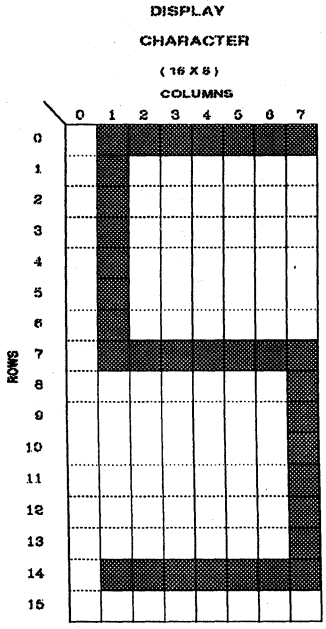


FIGURE 7

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

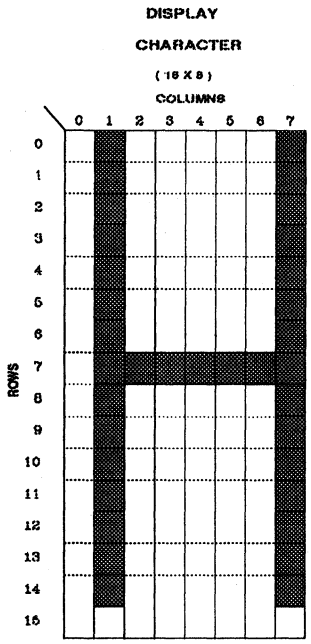


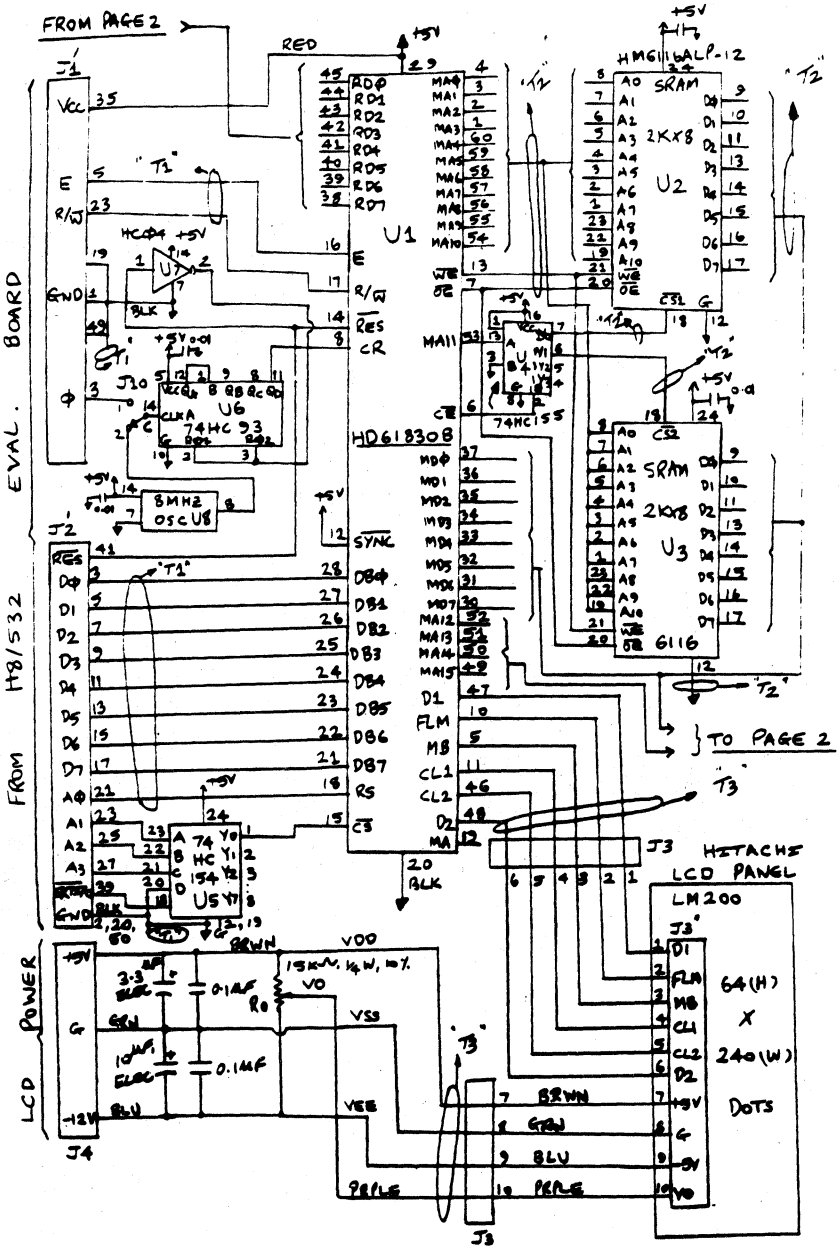
FIGURE 8

5.0 LCD INTERFACE BOARD SCHEMATIC :

The next two pages show the schematic of the LCD interface board used in custom character generation. The Hitachi UV EPROM HN27C256AG -15 (32KBx8) resident in a ZIF socket was used as a character generator. As an alternate part, EPROM HN27128AG-17 (16KBx8) was also tested for the same application. Also, note that the LCD PANEL LM200 DC power supply (+5V,G, -12V) can also be tied to the H8/532 Evaluation Board power source at **one** point. In such a case, the display contrast resistor may have to be re-adjusted.

HD61830B / LM200 DESIGN - CUSTOM CHARACTER GENERATION

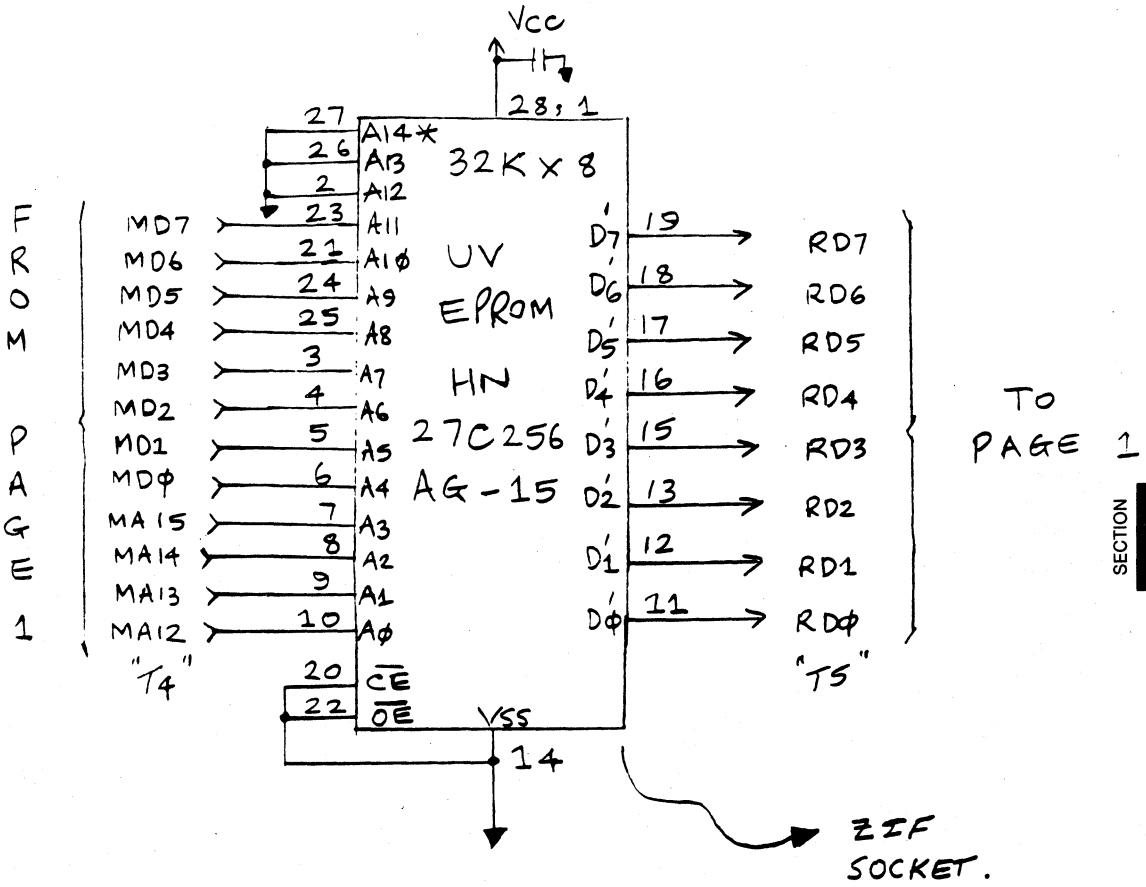
LCD INTERFACE BOARD - SCHEMATIC



HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD61830B / LM200 DESIGN - CUSTOM CHARACTER GENERATION

LCD INTERFACE BOARD - SCHEMATIC

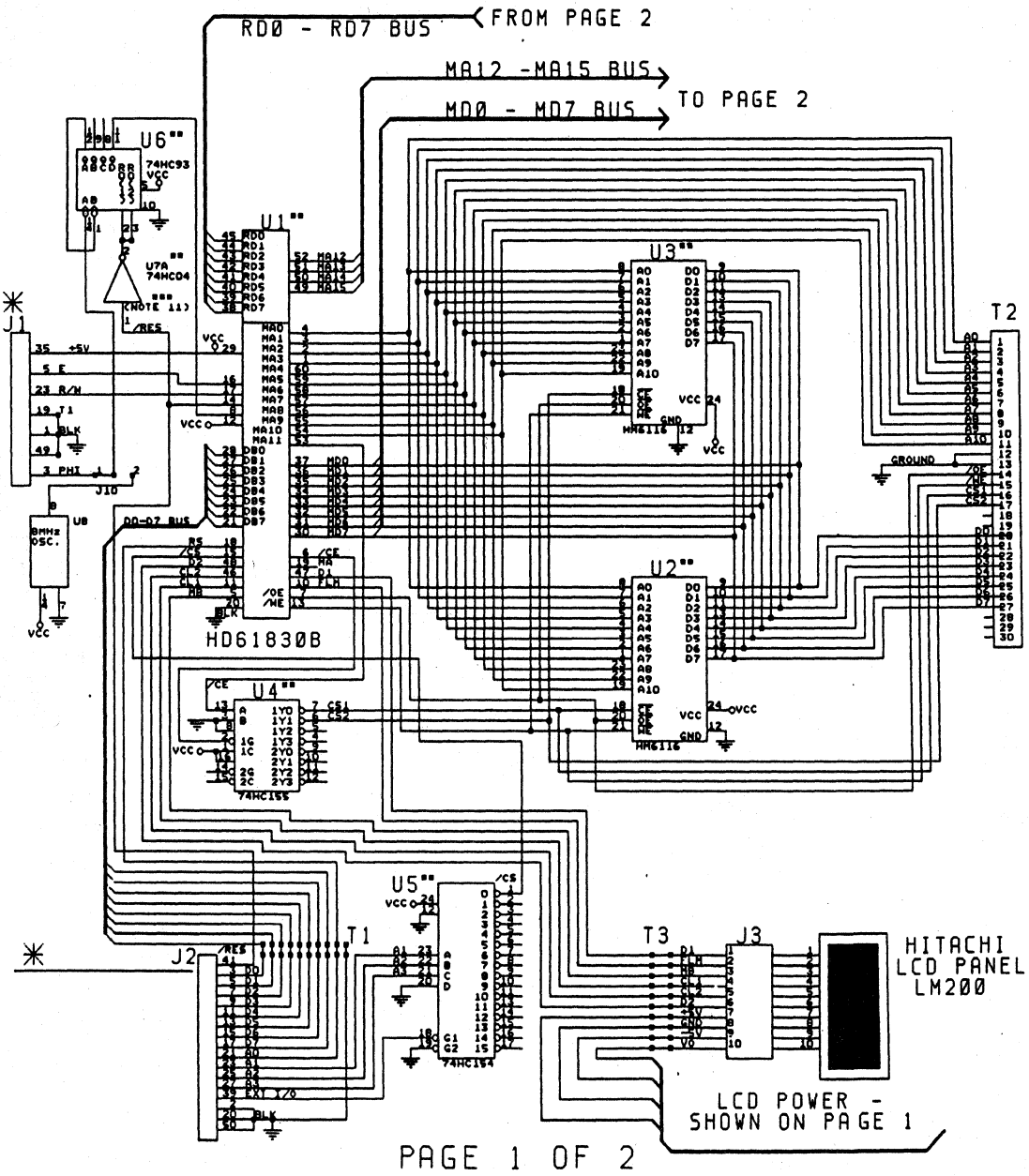


NOTE:

1.0 Test Connectors T1,T2, and T3 are for test and debug.

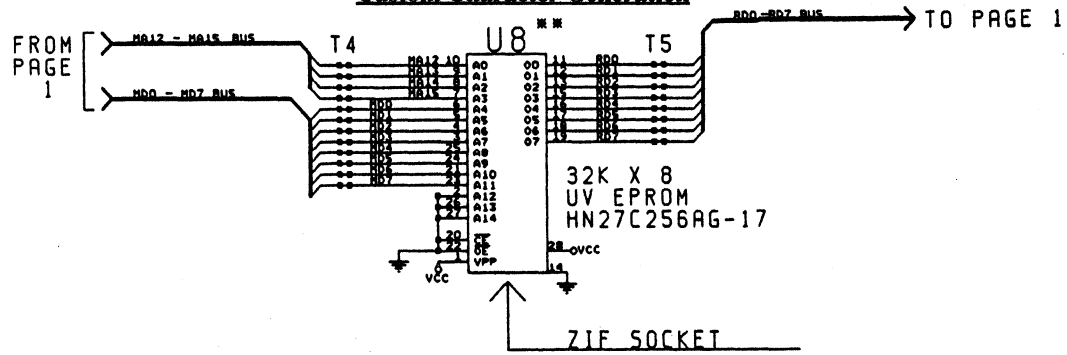
2.0 After power on reset, Display is "OFF", Slave Mode "ON", and Hp = 6.

Custom Character Generation—Schematic

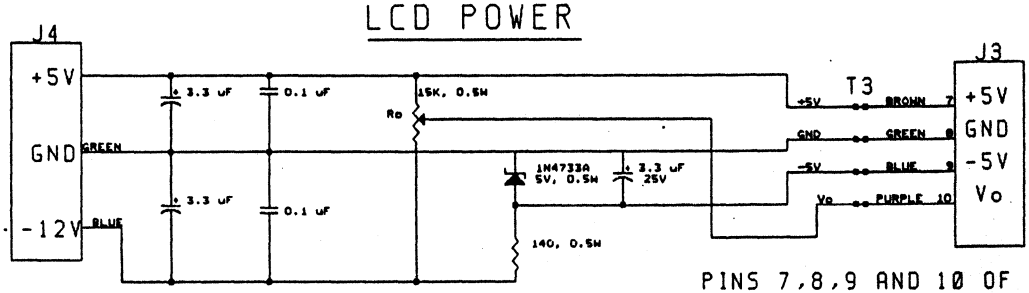


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Custom Character Generation



LCD POWER



PINS 7,8,9 AND 10 OF J3 PROVIDE POWER FOR THE HITACHI LM200 LCD PANEL SHOWN ON PAGE 1.

NOTES:

1.0 Test connectors T1, T2, T3, T4, and T5 are for test and debug.

2.0 On H8/325 EVAL. BOARD change "Y1" crystal from 20 MHz to 16 MHz.

3.0 Power on reset: Display off, Slave Mode On, and Hp = 6.

4.0 Install 28 pin ZIF wire-wrap socket.

5.0 Color code wires.

6.0 Decouple VCC pin 28.

7.0 Keep wire lengths as short as possible.

8.0 For HN27128AG-17 EPROM, A14 (pin 27) is tied to VCC.

9.0 * J1 and J2 are from H8/532 Evaluation board.

10.0 ** U1 - U8 each have a 0.01 uF capacitor between Vcc and ground.

11.0 *** U7 has ground at pin 7 and Vcc at pin 14.

PAGE 2 OF 2

7.0 SOFTWARE :

The software section covering the HD61830B / LM200 panel design tutorial part I shows I/O address, Busy Flag Varification, Initialization Flow Chart, Code Assembly Procedure while its Appendices show the program listings.

In the tutorial part II, the custom character generation program " XCG.MOT " is listed in the Appendix "A". The Appendix "B" shows the character font patterns loaded in a 32KB EPROM. The " XCG.MOT " program is located at the address 8000H in the H8/532 processor Evaluation Board memory space. After initailizing the LCD controller HD61830B, it enables the external character generator. Then clears the screen by writing character code 20H i.e. code for a blank in the LCD display memory. Following a screen clear routine, the character for the four custom character "K","A","S", and "H" are written in the LCD display frame memory. Since the display is memory mapped, the four custom characters get displayed. For more information on the " XCG.MOT " program refer to the Appendix "A". The number of custom characters can be expanded from 4 to **256** with a maximum font size of 16 (Rows) x 8 (columns) only in the character mode. However, for demonstartion purposes only four custom characters were chosen for this tutorial.

For register programming details , refer to the HD61830B data sheet.

The code developed in the Appendix "A" for external character generator program can also be transported to the H8/532 Evaluation Board. It can reside within a HN27512AG -25 EPROM which **replaces** the HMS V1.2 debugger EPROM located at "U6" on the H8/532 Evaluation Board. In this manner, a stand alone custom character display unit can also be built.

Listing 2: START.LIS (continued)

```
  398                                   ;enable input delay timer
  399 program  C 00AE F808                      mov.b   #8,r0l
  400 program  C 00B0 3890                      mov.b   r0l,@frt_tier   ;enable delay timer interrupts
  401
*** H8/300 ASSEMBLER              VER 1.1 ***   03/20/91 08:11:20                                        PAGE     3
PROGRAM NAME =                                Buffer Initialization Routine

  402                                   ;main loop
  403 program  C 00B2                   main:
  404 program  C 00B2 0180                      sleep                   ;wait for interrupts
  405
  406 program  C 00B4 40FC                      bra     main
  407
  408                                           .end
  *****TOTAL ERRORS        0
  *****TOTAL WARNINGS      0
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Listing 2: START.LIS (continued)

```
 319                                    ;re-initialize buffer pointers
 320 program  C 0048 0D56                       mov.w   r5,r6           ;clear IDP and ODP
*** H8/300 ASSEMBLER                 VER 1.1 ***   03/20/91 08:11:20                                    PAGE     2
PROGRAM NAME =                                  Buffer Initialization Routine

 321
 322                                    ;initialize program constants and flags
 323
 324                                    ;initialize margin area
 325 program  C 004A F410                       mov.b   #h'10,r4h       ;margin = 16 bytes
 326
 327                                    ;initialize flags
 328 program  C 004C FC15                       mov.b   #h'15,r4l       ;set buf_mt, buf_init, & on_line active
 329
 330                                    ;initialize free-running timer for input watchdog timing
 331 program  C 004E 79000000                   mov.w   #0,r0
 332 program  C 0052 6B80FF92                   mov.w   r0,@frt_frc     ;reset counter
 333
 334 program  C 0056 F801                       mov.b   #1,r0l
 335 program  C 0058 3891                       mov.b   r0l,@frt_tcsr   ;clear counter on match a
 336
 337 program  C 005A F800                       mov.b   #0,r0l
 338 program  C 005C 3896                       mov.b   r0l,@frt_tcr    ;use phi/2
 339
 340 program  C 005E 790001F4                   mov.w   #500,r0
 341 program  C 0062 6B80FF94                   mov.w   r0,@frt_ocra    ;set count for 100 usec
 342
 343                                    ;initialize multi-function timer 0 for output strobe generation
 344 program  C 0066 F801                       mov.b   #1,r0l
 345 program  C 0068 38C8                       mov.b   r0l,@tmr0_tcr   ;use phi/8, no interrupts
 346
 347 program  C 006A F800                       mov.b   #0,r0l
 348 program  C 006C 38C9                       mov.b   r0l,@tmr0_tcsr  ;negative strobe
 349
 350 program  C 006E F800                       mov.b   #0,r0l
 351 program  C 0070 38CC                       mov.b   r0l,@tmr0_tcnt  ;clear counter
 352
 353 program  C 0072 F802                       mov.b   #2,r0l
 354 program  C 0074 38CB                       mov.b   r0l,@tmr0_tcorb
 355 program  C 0076 F805                       mov.b   #5,r0l
 356 program  C 0078 38CA                       mov.b   r0l,@tmr0_tcora ;generate strobe 2.4 usec wide
 357
 358                                    ;initialize multi-function timer 1 for initialization timing
 359 program  C 007A F801                       mov.b   #1,r0l
 360 program  C 007C 38D0                       mov.b   r0l,@tmr1_tcr   ;use phi/8, no interrupt
 361
 362 program  C 007E F806                       mov.b   #6,r0l
 363 program  C 0080 38D1                       mov.b   r0l,@tmr1_tcsr  ;negative strobe
 364
 365 program  C 0082 F800                       mov.b   #0,r0l
 366 program  C 0084 38D4                       mov.b   r0l,@tmr1_tcnt  ;set counter to 0
 367
 368 program  C 0086 F802                       mov.b   #2,r0l
 369 program  C 0088 38D3                       mov.b   r0l,@tmr1_tcorb
 370 program  C 008A F834                       mov.b   #h'34,r0l
 371 program  C 008C 38D2                       mov.b   r0l,@tmr1_tcora ;generate strobe 40 usec wide
 372
 373                                    ;initialize interrupt structure
 374 program  C 008E F807                       mov.b   #7,r0l          ;set maskable interrupts
 375 program  C 0090 38C6                       mov.b   r0l,@iscr       ; for falling edge
 376 program  C 0092 38C7                       mov.b   r0l,@ier        ;enable maskable interrupts
 377 program  C 0094 0700                       ldc     #h'00,ccr       ;global interrupt enable
 378
 379                                    ;enable init pulse timer
 380 program  C 0096 F800                       mov.b   #0,r0l
 381 program  C 0098 38D4                       mov.b   r0l,@tmr1_tcnt  ;reset init pulse count
 382
 383 program  C 009A F841                       mov.b   #h'41,r0l
 384 program  C 009C 38D0                       mov.b   r0l,@tmr1_tcr   ;use phi/8, interrupt on match a
 385
 386                                    ;test for initialization complete
 387 program  C 009E                    init_loop:
 388 program  C 009E 732C                       btst.b  #buf_init_flag,r4l      ;init done ?
 389 program  C 00A0 46FC                       bne     init_loop               ;no
 390
 391                                    ;enable input handshaking
 392 program  C 00A2 7FB27240                   bclr.b  #ibusy_bit,@in_hs       ;release input busy signal
 393
 394                                    ;set status indicators
 395 program  C 00A6 7FB27200                   bclr.b  #ready_bit,@stat_port   ;Light Ready LED
 396 program  C 00AA 7FB27210                   bclr.b  #online_bit,@stat_port  ;Light On Line LED
 397
```

SECTION 5

Listing 2: START.LIS

```
*** H8/300 ASSEMBLER                    VER 1.1 ***   03/20/91 08:11:20                                    PAGE     1
PROGRAM NAME =                                  Buffer Initialization Routine

  1                                                 .heading        "Buffer Initialization Routine"
  2
  3                                         ;H8/330 Print Buffer Routine
  4                                         ;revision 2.0
  5
  6                                         ;written by:
  7                                         ;       Tom Hampton
  8                                         ;       Hitachi America, Ltd.
  9                                         ;       Application Engineering
 10
248                                                .output         dbg,obj
249                                                .print          nocref,nosct
250
251                                                .global         start
252
253 program  C 0000                                .section        program,code
254
255                                         ;initialization routines
256 program  C 0000                         start:
257
258                                         ;initialize stack pointer
259 program  C 0000 7907FF80                       mov.w   #top_ram,r7     ;set SP to top of ram
260
261                                         ;initialize input handshake and status indicators
262 program  C 0004 F8FF                           mov.b   #h'ff,r0l       ;set IBUSY active to keep
263 program  C 0006 38B2                           mov.b   r0l,@p1_dr      ; istb interrupts inactive
264                                                                        ;clear LEDs
265 program  C 0008 38B0                           mov.b   r0l,@p1_ddr     ;set port as outputs
266
267                                         ;initialize Memory Control Port
268 program  C 000A 38BA                           mov.b   r0l,@p5_dr      ;set WE\, CS1\, & CS0\ inactive
269 program  C 000C 38B8                           mov.b   r0l,@p5_ddr     ;set pins as outputs
270
271                                         ;initialize Memory Address Bus
272 program  C 000E 38B6                           mov.b   r0l,@p3_dr
273 program  C 0010 38BF                           mov.b   r0l,@p8_dr      ;set buffer address as FFFF
274 program  C 0012 38B4                           mov.b   r0l,@p3_ddr
275 program  C 0014 38BD                           mov.b   r0l,@p8_ddr     ;set ports as outputs
276
277                                         ;initialize Output Data Port
278 program  C 0016 38B9                           mov.b   r0l,@p6_ddr     ;set port as output
279
280                                         ;initialize Input Port Controls and Pause
281 program  C 0018 38C1                           mov.b   r0l,@p9_dr      ;turn-on MOS Pull-Ups
282
283                                         ;initialize Output Port Controls
284 program  C 001A F813                           mov.b   #h'13,r0l
285 program  C 001C 38B7                           mov.b   r0l,@p4_dr      ;set OINIT\ active and OSTB\ inactive
286 program  C 001E F812                           mov.b   #h'12,r0l       ;set port as follows:
287 program  C 0020 38B5                           mov.b   r0l,@p4_ddr     ; Bit 4 as output (OINIT)
288                                                                        ; Bit 1 as output (OSTB\)
289                                                                        ; Bit 0 as input (OBUSY), MOS Pull-Up active
290
291                                         ;initialize memory buffer
292 program  C 0022 F8FF                           mov.b   #write,r0l
293 program  C 0024 38B1                           mov.b   r0l,@mem_dir    ;set memory data port as output
294 program  C 0026 F800                           mov.b   #0,r0l          ;clearing value
295 program  C 0028 79050000                       mov.w   #0,r5           ;buffer pointer
296 program  C 002C 38B3                           mov.b   r0l,@mem_data   ;set data
297
298 program  C 002E                         clear_buffer:
299                                         ;clear buffer location
300 program  C 002E 3DB6                           mov.b   r5l,@addr_lo
301 program  C 0030 35BF                           mov.b   r5h,@addr_hi    ;set address
302 program  C 0032 4B04                           bmi     wr_cs1
303 program  C 0034                         wr_cs0:
304 program  C 0034 F802                           mov.b   #wrcs0,r0l      ;write to chip 0
305 program  C 0036 4002                           bra     wr_cont
306 program  C 0038                         wr_cs1:
307 program  C 0038 F801                           mov.b   #wrcs1,r0l      ;write to chip 1
308 program  C 003A                         wr_cont:
309 program  C 003A 38BA                           mov.b   r0l,@mem_ctrl   ;activate write pulse
310 program  C 003C F807                           mov.b   #7,r0l
311 program  C 003E 38BA                           mov.b   r0l,@mem_ctrl   ;de-activate write pulse
312
313                                         ;increment buffer pointer
314 program  C 0040 8D01                           add.b   #1,r5l
315 program  C 0042 44EA                           bcc     clear_buffer    ;loop until buffer cleared
316 program  C 0044 8501                           add.b   #1,r5h
317 program  C 0046 44E6                           bcc     clear_buffer    ;loop until buffer cleared
318
```

Listing 1: BUFFER.LIS

```
*** H8/300 ASSEMBLER                    VER 1.1 ***   03/20/91 08:11:13                                    PAGE     1
PROGRAM NAME =                                    Vector Table Definitions

                                                  .heading        "Vector Table Definitions"

                                        ;H8/330 Print Buffer Routine
                                        ;revision 2.0

                                        ;written by:
                                        ;       Tom Hampton
                                        ;       Hitachi America, Ltd.
                                        ;       Application Engineering

                                                  .output         dbg,obj
                                                  .print          nocref,nosct

                                                  .global         start
                                                  .global         online_int,pause_int,input_int,iinit_int
                                                  .global         output_int,ostb_int,oinit_int

 vector   D 0000                                  .section        vector,data,locate=0

                                        ;vector table initialization

                                                  .org    0
 vector   D 0000 0000                             .data.w start          ;reset vector

 vector   D 0006                                  .org    nmi_vec
 vector   D 0006 0000                             .data.w iinit_int      ;input init strobe detect

 vector   D 0008                                  .org    irq0_vec
 vector   D 0008 0000                             .data.w online_int     ;online pushbutton detect

 vector   D 000A                                  .org    irq1_vec
 vector   D 000A 0000                             .data.w pause_int      ;pause pushbutton detect

 vector   D 000C                                  .org    irq2_vec
 vector   D 000C 0000                             .data.w input_int      ;input strobe detect

 vector   D 0020                                  .org    ocia_vec
 vector   D 0020 0000                             .data.w output_int     ;output service request

 vector   D 0026                                  .org    cmi0a_vec
 vector   D 0026 0000                             .data.w ostb_int       ;output data strobe generator

 vector   D 002C                                  .org    cmi1a_vec
 vector   D 002C 0000                             .data.w oinit_int      ;output init strobe generator

                                                  .end
*****TOTAL ERRORS       0
*****TOTAL WARNINGS     0
```

SECTION 5

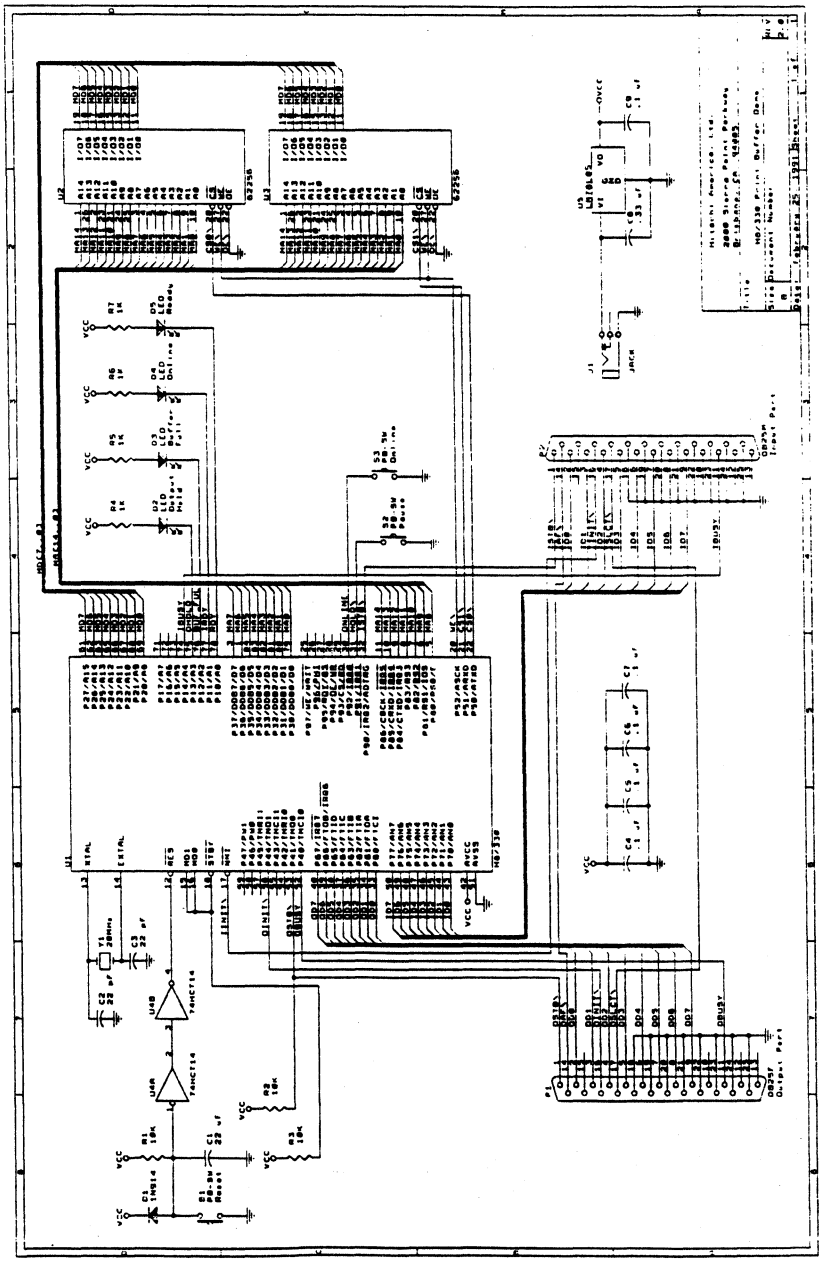


Figure A-1: Schematic Diagram

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDIX A

TABLE OF CONTENTS

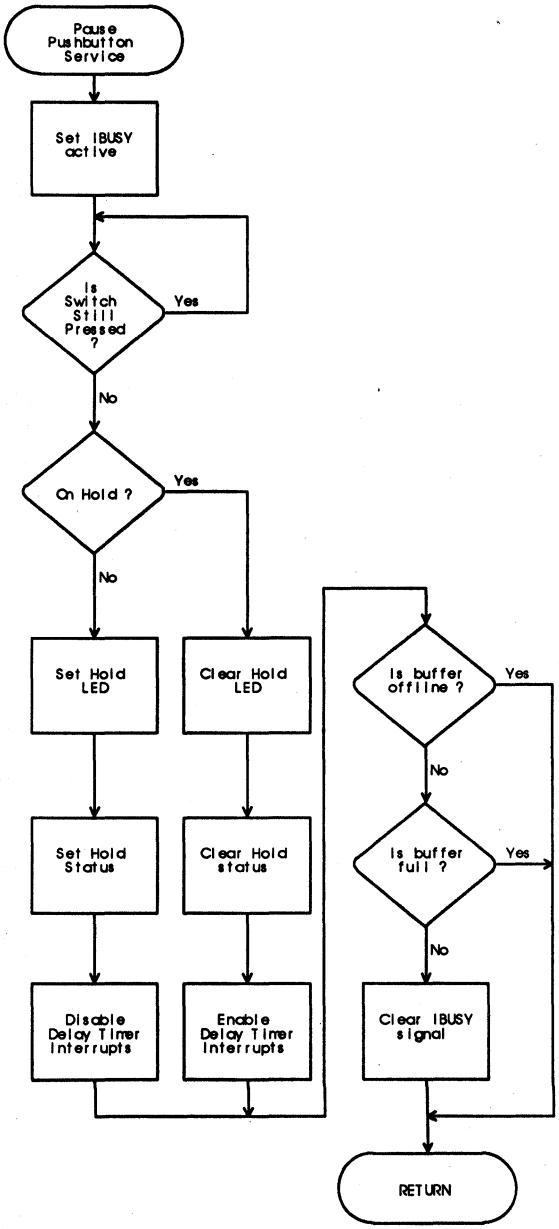


Figure 9: Pause Pushbutton Service Routine

Pause Pushbutton Service Routine

This routine executes whenever the "Pause" pushbutton is pressed. Whenever this button is pressed, an interrupt request is generated that allows the software to control the ability of the print buffer to output any input data to the printer. This would be very similar to having pressed the "Online" pushbutton at the printer itself.

If the print buffer output is currently active when this pushbutton is pressed, then this routine will make it inactive. This is done by setting the "output hold" status flag and disabling input watchdog interrupts.

If the print buffer output is currently inactive when this pushbutton is pressed, then this routine will make it active. This is done by resetting the "output hold" status flag and enabling input watchdog interrupts. For a flow chart of this service routine, refer to Figure 9.

CONCLUSION

While this example does not use all of the peripheral features of the H8/330, it does provide examples of programming for both timers and I/O ports, as well as features of the individual I/O ports. Also included are methods for initializing the interrupts structure of the H8/330. Enhancements can most certainly be made to this example by doing some rearranging of the I/O port choices. A serial input or output option can be made by using the on-chip SCI and moving the memory buffer control functions to another I/O port. More memory could be added by using more I/O bits from another port to expand the address field. This would also required a little extra address manipulation in determining buffer conditions, but it is achievable.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

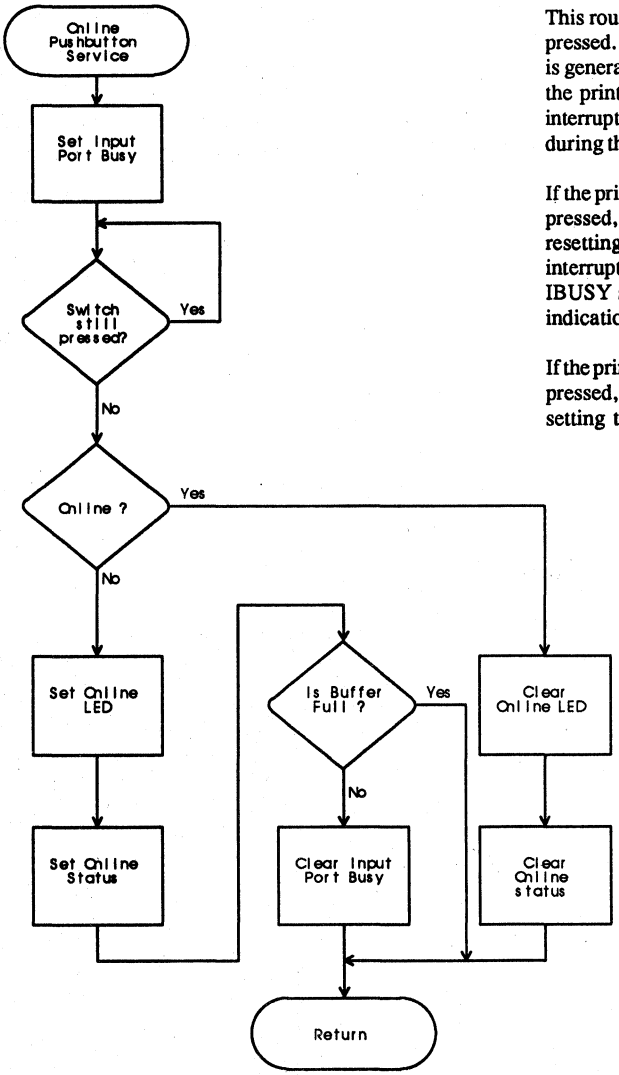


Figure 8: Online Pushbutton Service Routine

Online Pushbutton Service Routine

This routine executes whenever the "Online" pushbutton is pressed. Whenever this button is pressed, an interrupt request is generated that allows the software to control the ability of the print buffer to accept any input data. To inhibit input interrupts from being requested, the IBUSY signal is set active during the processing of this routine.

If the print buffer is currently online when this pushbutton is pressed, then this routine will take it off-line. This is done by resetting the "online" status flag and disabling input strobe interrupts. The $\overline{\text{ISTB}}$ interrupt itself is disabled as well as the IBUSY signal left active so that the sending device has an indication that the buffer is now "off-line."

If the print buffer is currently off-line when this pushbutton is pressed, then this routine will take it online. This is done by setting the "online" status flag and enabling input strobe interrupts. The $\overline{\text{ISTB}}$ interrupt itself is enabled as well as the IBUSY signal made inactive so that the sending device has an indication that the buffer is now "online." For a flow chart of this service routine, refer to Figure 8.

SECTION 5

accept such data. If no data is in the buffer or the printer is not ready to accept the buffered data, this routine merely resets the input watchdog timer and checks to determine if the IBUSY signal should be activated before returning to the main program.

If there is data in the buffer and the printer is ready to accept the data, then this routine goes through the process of getting the data from the buffer and sending it to the output parallel port. In getting the data from the memory buffer, this routine must change the direction of the memory buffer's data bus to be input as well as set the address bus with the output data pointer. The proper $\overline{CS}$ signal is then generated in order to "read" the data to be output. That data is moved to the output parallel port and the multi-function timer that generates the output strobe ($\overline{OSTB}$) is enabled. The H8/330 then goes to "sleep" until the output strobe interrupt occurs.

After returning from the output strobe interrupt routine, the data output service routine has to determine whether or not the memory buffer is in a either the "empty" or "full" condition. If the buffer is in the empty condition, then this routine sets the "buffer empty" flag, deactivates the IBUSY signal, and resets the input watchdog timer in completing its operations. If the buffer is not "empty," then this routine must determine whether or not the buffer is in the "full" condition. If the buffer is in a "full" condition, then this routine just resets the input watchdog timer and completes its service with the IBUSY signal still being set. If the buffer is not in the "full" condition, then the routine clears the "buffer full" flag, deactivates the IBUSY signal, and resets the input watchdog timer in completing its operations.

Output Strobe Service Routine

During the execution of the data output service routine, one of the 8-bit multi-function timers is programmed to generate the $\overline{OSTB}$ signal, and also an interrupt on the trailing edge of that strobe. During this service routine, this timer is programmed not to generate any more interrupts and also to keep its output at a high level (thus generating no more strobes). Execution then returns to the data output service routine. For a flow chart of this service routine, refer to Figure 5.

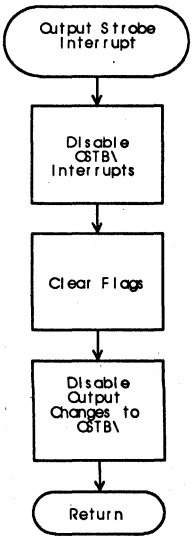


Figure 5: Output Strobe Service Routine

Input INIT Pulse Service Routine

The input $\overline{INIT}$ pulse ($\overline{IINIT}$) signal is connected directly to the $\overline{NMI}$ input of the H8/330. Whenever the sending device sets this signal active, the print buffer will disable all timers from generating any of their interrupts. It then restarts the software just as if the reset pushbutton had been pressed. This allows the sending device to control the reset of the buffer and printer through its hardware signal. It does not interfere with any software reset commands sent directly to the printer. For a flow chart of this service routine, refer to Figure 6.

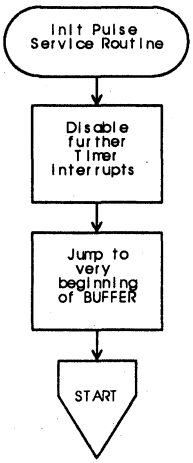


Figure 6: $\overline{IINIT}$ Pulse Service Routine

Output INIT Pulse Service Routine

During the execution of the main routine, one of the 8-bit multi-function timers is programmed to generate the $\overline{OINIT}$ signal, and also an interrupt on the trailing edge of that strobe. During this service routine, this timer is programmed not to generate any more interrupts and also to keep its output at a high level (thus generating no more strobes). The routine also clears the "oinit" status bit so that the main routine can complete it operation. For a flow chart of this service routine, refer to Figure 7.

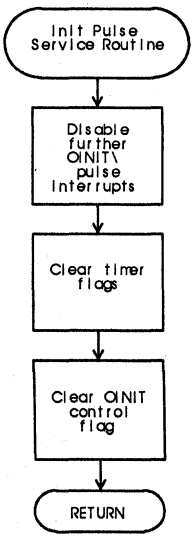


Figure 7: $\overline{OINIT}$ Pulse Service Routine

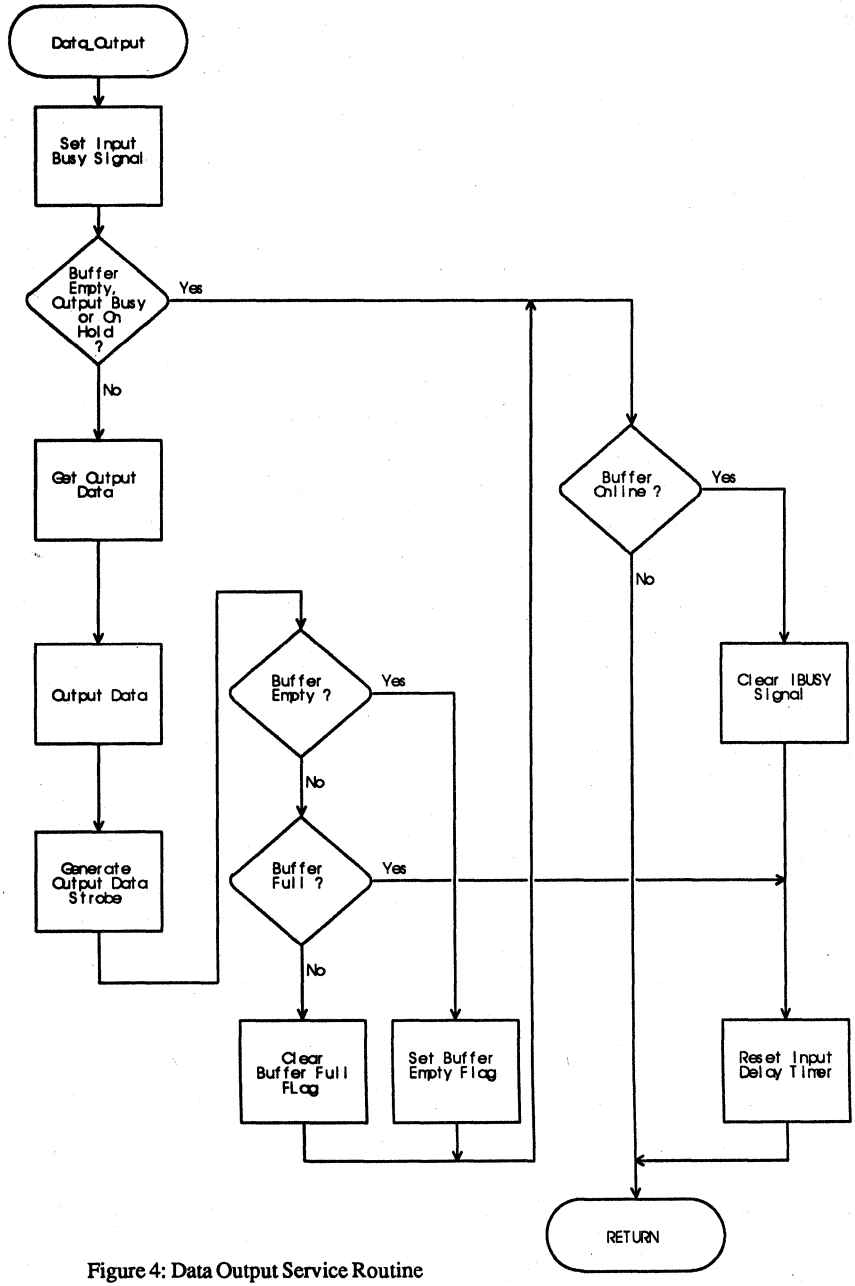


Figure 4: Data Output Service Routine

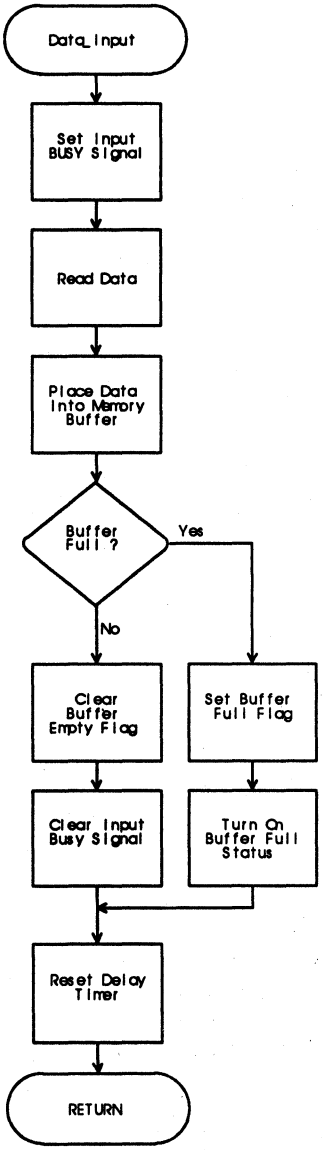


Figure 3: Data Input Service Routine

Now we can enable the second 8-bit multi-function timer to generate the $\overline{\text{OINIT}}$ signal and also the interrupts. Here we wait in a loop until the "oinit" status bit has been cleared to indicate that the pulse has occurred. Now we clear out the IBUSY signal so that input data strobes can occur, and set the status indicators to show that the print buffer is "online" and "ready."

The last thing we do in the main routine is to enable the input watchdog timer so that its interrupts can be generated.

Data Input Service Routine

The input data strobe ($\overline{\text{ISTB}}$) is input to the H8/330 as the lowest level maskable interrupt. Whenever the falling edge of the $\overline{\text{ISTB}}$ signal is detected, the H8/330 goes through the process of inputting data from the parallel port and placing into the print buffer. For a flow chart of this service routine, refer to Figure 3. In order to keep further $\overline{\text{ISTB}}$ interrupts from occurring before the print buffer is ready to accept them, this routine first sets the IBUSY signal active before it can do anything else. It then goes through the process of getting the data and writing it to the memory buffer.

A separate pointer is maintained for the input position of the buffer. This position is checked against the output data pointer to determine when the buffer is to full to accept any more data. As long as the buffer is not full, the "buffer empty" flag is cleared, the input watchdog timer is reset, and the IBUSY signal is deactivated. This would complete this service routine. If the memory buffer is determined to be full, then the "buffer full" flag is set, the "Buffer Full" status indicator is "turned on," and the service routine completes with the IBUSY signal remaining active after resetting the input watchdog timer. This inhibits further input strobe interrupts from occurring until the buffer has been emptied of some of its data.

Data Output Service Routine

One of the timers is initialized such that it will generate an interrupt to the H8/330 if no input or output activity takes place within 100 µsec. Both the data input and data output service routines reset this counter in order to keep both activities going. For a flow chart of this service routine, refer to Figure 4. To ensure that no input data requests are generated while the output service is taking place, this routine also sets the IBUSY signal active immediately.

Since a timer generated this request rather the printer itself, this routine must determine whether or not there is data in the buffer to be output and whether or not the printer is ready to

could possibly be requested before the H8/330 is initialized [except for $\overline{NMI}$ ($\overline{IINIT}$), which performs the same function as a reset], we can now go about the business of initializing the H8/330 without worrying about missing a request for buffering.

At this point, we go through the process of initializing all of the I/O ports for proper usage. Since all of the I/O ports of the H8/330 are initialized as input ports at reset, we must go through each port and setup both direction and functions.

I/O Port 1 is used for two functions; status indication and input busy (IBUSY) signal control. All of these signals are outputs. The initial status for outputs on this port should be all high. This "turns" the LEDs off and sets the IBUSY signal active. To ensure that this happens as soon as the port is programmed for output function, we write to the data register prior to setting the direction.

The second set of ports we initialize are for the memory buffer. In order to make sure that the memory is inactive when we program the ports, we write to the I/O port (Port 5) used for the control signals to make the $\overline{WE}$ and $\overline{CS}$ signals inactive when the direction is changed to outputs. We are then able to program the I/O ports to be used as the address bus. Both of these ports (Port 3 and Port 8) are to be used as outputs. The only other port used in connection with the memory buffer is for data access. Since this port will be used bidirectionally, the direction will be programmed as we need to use it.

We can now program the ports that we intend to use for the parallel input and output ports. Since we have chosen Port 7 for the input port, no direction programming is necessary because Port 7 is input only anyway. Port 6 is used for the output parallel port so its direction must be changed. Port 9 is used for control over both the input and output parallel ports. The signals involved here are the input strobe ($\overline{ISTB}$) and the two control panel switches for online and output hold. Since all three of these signals are inputs, no direction change is required. We can, however, use the internal MOS pull-up feature of the port so that external resistors are not needed. To control this feature, we can write to the port data register with "1s" to enable the pull-ups (this feature is available only with port bits that are inputs).

Port 4 is used for control signals directly affecting the output parallel port. These signals include the output strobe ($\overline{OSTB}$), output busy (OBUSY), and output initialization ($\overline{OINIT}$). Since the OBUSY signal is an input while the other two signals are outputs, we must program this port for a mixture of input and output lines. Initially, we would like to ensure that the $\overline{OSTB}$ signal is inactive while the $\overline{OINIT}$ is active (to make certain that the printer itself gets reset) and to enable the MOS pull-up on the OBUSY signal. To do this we program the data register of the port with the value H'13 before programming the direction of the port. For setting up bits 4 and 1 as outputs while bit 0 is an input, we must program the direction register with the value H'12.

At this point in the main routine we finally come to where we get to use the I/O ports for controlling the memory buffer. We use this opportunity to clear the memory buffer contents. In performing this operation, we must first set the I/O port used for the data bus to the output direction. We can then setup the ports used for the memory address with a valid address as well as the data port with the clearing value to be written. Next we write to the I/O port used for the control signals to activate the $\overline{WE}$ and correct $\overline{CS}$ signal. Since were are using different instructions to set and clear these bits, we can immediately deactivate the signals after having activated them. This sequence provides plenty of time for proper SRAM operation. This function is positioned inside a loop that executes until the entire buffer has been cleared.

We are now at a position where we can set the program constants and operation flags, as well as initializing the timers for their uses. The 16-bit free-running timer is used for watching input activity. This timer is programmed to generate an interrupt every 100 µsec, but this occurs only if it is not reset in a service routine. One of the 8-bit multi-function timers is used to generate the output data strobe. This timer is programmed to generate a negative-going strobe that is 2.4 µsec in width, and an interrupt on the trailing edge of that pulse. Since we don't want to generate the strobe at this time, the timer output is programmed to remain at a high level with no interrupts generated. We control the output level during the output service routine. The second 8-bit multi-function timer is used to generate the output initialization ($\overline{OINIT}$) pulse. This timer is programmed to generate a negative-going strobe that is 40 µsec in width, and an interrupt on the trailing edge of that pulse. Again, we do not wish to generate the strobe right now so the timer output is programmed to remain at a high level with no interrupts generated. We will generate the strobe right after we initialize the interrupt structure.

The interrupt structure must be setup to handle three (3) external interrupts and also for those external interrupts generated by the falling edge of a signal. For the external interrupts, this is accomplished by programming the Interrupt Sense Control Register (ISCR) for edge selection prior to programming the Interrupt Enable Register (IER). The interrupt mask is then released in the Condition Codes Register (CCR) of the H8/330 in order to enable all interrupts.

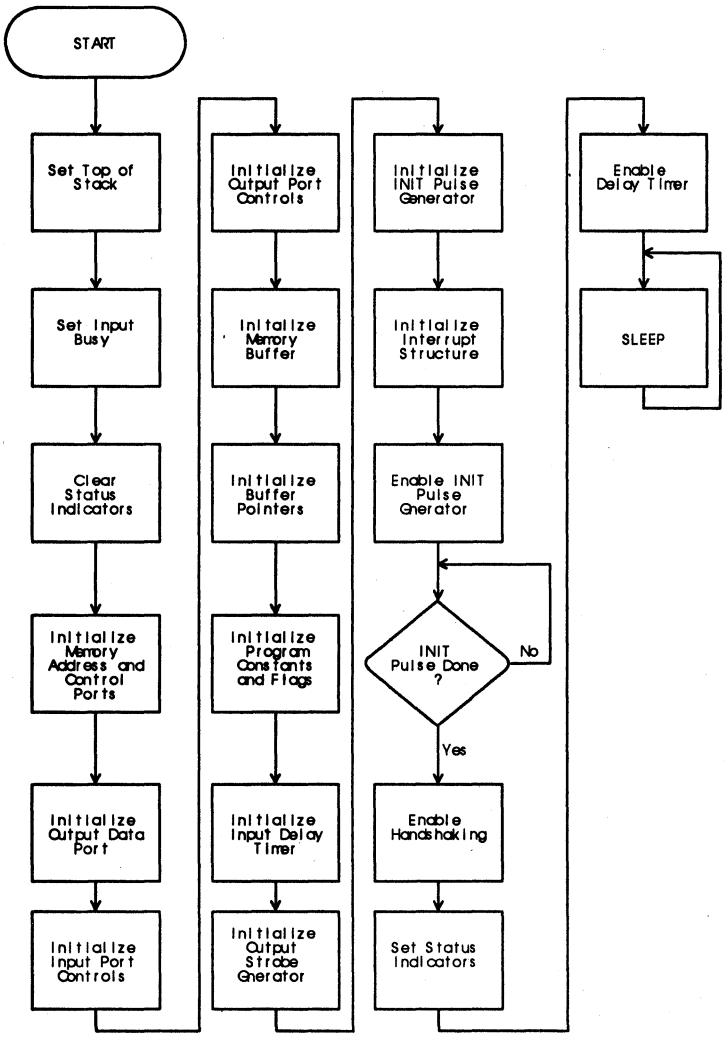


Figure 2: Main Routine

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Three of the on-chip timers are used for strobe generation and event monitoring. One of the 8-bit multi-function timers is used to generate an output data strobe while the second multi-function timer is used to generate an output printer initialization strobe. Since the multi-function timers can control both of these outputs directly, Port 4 was used for this function. The 16-bit free-running timer is used for event monitoring; its function is basically that of a watchdog timer. This timer is started and allowed to run continuously until an input service is requested. During the service of the input interrupt request, this timer is reset. If the timer is allowed to overflow, then the software assumes that there was no input activity and will check to determine if any output service can be performed. This timer is also reset during the output service routine. The interrupt generated by this timer is used internally only and not brought out to the rest of the system.

Memory Buffer

The interface to the memory buffer requires fifteen (15) lines for address (32K), eight (8) lines for data access, two (2) lines for chip selection, and one (1) line for write control. Since the memory devices draw the most power when they are chip selected, it was unnecessary to use an I/O line to control the output enables. These lines were left tied to ground so that they would always be active. In this manner, the chip select (CS_n) signal activate the appropriate memory device and keep power consumption to a minimum. I/O Port 5 happens to be a 3-bit port and is ideal for use for the three control signals.

Since Port 8 is already a 7-bit port, it was convenient to use it for the most significant portion (MA_{14}-MA_8) of the address bus. Two more 8-bit ports are required to complete the address bus as well as the data access bus. Since only two full 8-bit ports remained, we used Port 2 for the data bus (MD_7-MD_0) and Port 3 for the least significant byte (MA_7-MA_0) of the address bus.

SOFTWARE

The print buffer software basically consists of eight separate routines.

1. The main routine performs initialization of the I/O ports, timers, and the memory buffer as well as the generation of an initialization pulse for the printer attached to the buffer.

2. The input strobe (ISTB) service routine is responsible for placing data from the input data port into the memory buffer. This routine has the lowest priority of all the external interrupt routines, but does take precedence over all internal interrupts.

3. The output service routine is responsible for placing data from the memory buffer out to the output port. This routine is allowed to occur only when no other higher priority activity ($\overline{IINIT}$, $\overline{ISTB}$, or control panel switch press) has requested service within 100 µsec. This routine is also responsible for generating the output data strobe ($\overline{OSTB}$).

4. The output strobe service routine takes care of disabling the multi-function timer from generating further output strobes ($\overline{OSTB}$) as requested by the output service routine.

5. The input initialization ($\overline{IINIT}$) service routine provides a means by which the sending device can reset the print buffer as well as the printer connected to it. This includes generation of an output initialization strobe ($\overline{OINIT}$) as well as initializing the print buffer

6. The initialization strobe routine takes care of disabling the multi-function timer from generating further output initialization strobes ($\overline{OINIT}$) as requested by the input initialization service routine.

7. The online pushbutton service routine monitors an external switch which allows the user control over whether or not to allow data to be input to the print buffer. This performs a function similar to a printer's "online" switch.

8. The pause pushbutton service routine also monitor an external switch. This routine allows the user control over allowing the data in the memory buffer to be output to the printer. This would be useful in instances where the printer's "online" switch might not be readily accessible.

Complete source listings for each routine, as well as supporting files, can be foundin Appendix A.

Main Routine

This routine performs the function of initializing the print buffer for operation. For a flow chart of its sequence, please refer to Figure 2. In order to prevent any interrupts from being requested by the input port, the main routine sets the IBUSY signal active. By setting this signal active, the sending device is inhibited from generating any strobe ($\overline{ISTB}$) signals to the print buffer. Since the ISTB interrupt is the only interrupt that

DESIGN CONSIDERATIONS

One of the main considerations in the demo design was to keep the parts count to a minimum. This meant that we would have to use the features of the H8/330 wherever possible rather than an external device. A simple block diagram of the print buffer is shown in Figure 1.

The required parts had to consist of the H8/330 and some memory chips. We chose to use two (2) HM62256 SRAMs (32Kx8) to provide us with a 64K byte buffer. In order to clean up the power-on reset circuitry, we chose to add a 74HC14 although it probably wasn't necessary. This kept our parts count to only four ICs.

We also wanted to have some control over the operation of this print buffer. For this reason, three (3) switches were added to provide for an external reset, a means of taking the buffer "off-line" (just as if it were a printer), and a means of halting the buffer's output. We also wanted to have an indication of this control, so four (4) LEDs were added to indicate the status of the buffer.

You will find a complete schematic of the H8/330 Print Buffer in Appendix A (Figure A-1). You may want to refer to this schematic as we discuss our decisions for devices and I/O port usage.

I/O Port Usage

In the expanded modes of operation, the H8/330 has the capability of directly addressing external memory through the use of twenty-seven (27) of its I/O lines. We could have used one of these modes of operation, but that would limit the size of our storage buffer to considerably less than 64K bytes. Also, in these modes of operation, the only two ports on the H8/330 that have the capability to drive LEDs directly also serve as the external address bus. In order not to require the use of an external device to drive LEDs, and also to allow a large storage buffer (we chose 64K bytes for simplicity), the single-chip mode of operation was used. This forces us to use individual I/O ports to control buffers addressing, memory control, and data access. We are, however, not losing the use of any I/O lines because of this.

Status Display

In this design, we have four (4) LEDs that are used to display the status of the print buffer. These status indicators include Ready, Online, Buffer Full, and Output Hold. Since only ports 1 and 2 have the capability of driving LEDs directly, neither could be used to address the external memory buffer. Port 1 was chosen to indicate the status.

Control Panel

Also in this design, we have two (2) switches which are used to provide user control over the actions of the print buffer. These two switches allow the user to halt (or restart) the buffer's output, and take the print buffer on-line or off-line. Since continual polling of these switches would take too much time out of the spooling action, it was decided to use external interrupts as the switch inputs. This meant that Port 9 would be used for this function. I was also convenient since Port 9 has internal MOS pull-up resistors, thus keeping with our constraint of minimizing the parts counts.

The third switch of the control panel controls a hardware reset to the print buffer in the event that the user wishes to reset the buffer during its normal operation.

Parallel Input and Output

Since this print buffer is parallel-in and parallel-out, three (3) 8-bit ports are required to allow for this interface (data plus handshake). The A/D converter of the H8/330 is not being utilized for this application so I/O Port 7 is ideal for the parallel input port (since it happens to be an input only port anyway). Also, since no other external interrupts are required and the free-running timer interrupt is internal only, I/O Port 6 is an ideal selection for the parallel output port.

Additionally, three control signals ($\overline{\text{INIT}}$, $\overline{\text{STB}}$, and BUSY) from both the input and output ports are necessary for proper operation.

The INIT strobe ($\overline{\text{IINIT}}$) from the input port is fed directly to the NMI input of the H8/330. When the personal computer generates this strobe, it is an indication that the system hardware wishes to reset the printer. For this reason, this event takes precedence over all others. When this occurs, the print buffer will generate an INIT pulse ($\overline{\text{OINIT}}$) for the output port to reset the printer. This pulse is also generated during the initialization sequence of the print buffer.

The input STB signal ($\overline{\text{ISTB}}$) is accepted as a maskable interrupt to Port 9. This strobe has the lowest priority of all maskable external interrupts in order to ensure that the initialization pulse and the switch press interrupts take precedence. This event causes the generation of the input BUSY signal (IBUSY) so that no other input $\overline{\text{STB}}$s can occur until the data is properly buffered.

H8/330

Application Note

Parallel-to-Parallel Print Buffer Controller

Tom Hampton

INTRODUCTION

The HD6473308 (H8/330) is a highly integrated 8-bit microcomputing unit. Along with a central processing unit utilizing a reduced instruction set designed for speed, the H8/330 incorporates several system peripheral devices and memory onto a single chip. These on-chip functions include 16K bytes of ROM or EPROM, 512 bytes of RAM, 15 bytes of dual-port RAM, 5 timers, a UART channel, 8 channels of A/D conversion, and 9 I/O Ports.

These features allow the H8/330 to be used in many applications; a print buffer is merely one of the vast possibilities. In this application, we are able to examine the usage of several of the on-chip peripherals as well as I/O ports and interrupt control. While this application does not use all of the peripheral features of the H8/330, it does provide programming examples for many of the peripherals as well as the CPU itself.

Three of the on-chip timers are used to control such events as strobe generation for both output data and printer initialization, and also for event monitoring. This is accomplished through the exception processing features of the H8/330. Four external interrupts are utilized to monitor input data strobes, input initialization strobes, and pushbutton (control panel) events.

In order to maximize the external memory addressing capabilities for this application, the H8/330 device is used in the single-chip mode of operation. In this mode, many of the I/O ports are used to control the memory buffer itself as well as for status displays. One of the I/O ports is even used for a bidirectional data bus even though the H8/330 does not have that feature directly.

Even though this application uses very little on-chip memory (less than 512 bytes of ROM and less than 20 bytes of RAM), the on-chip memory capabilities of the H8/330 provide enough room for code and data storage required by most applications.

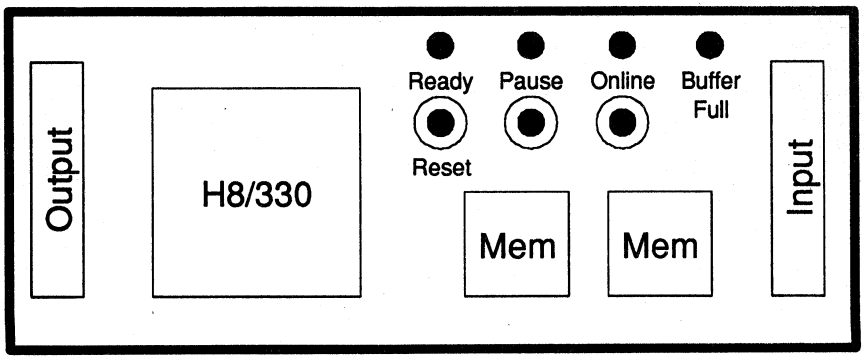


Figure 1: Buffer Block Diagram

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Listing 1: 16x16 Multiply Routine

```
*** H8/300 ASSEMBLER                    VER 1.1 ***   05/01/91 12:20:11                                    PAGE     1
PROGRAM NAME =

                                        ;H8/300 CPU 16x16 Multiply Routine

                                        ;This routine uses strictly registers to maintain all
                                        ;storage facilities and for calculation.

                                        ;Register Usage
                                        ;Entry:
                                        ;          R1 = Multiplier
                                        ;          R2 = Multiplicand
                                        ;       R3,R4 = Temporary Result
                                        ;          R5 = Temporary Storage
                                        ;Exit:
                                        ;          R1 = Result, LSW
                                        ;          R2 = Result, MSW

                                        ;Pictorial Description:
                                        ;               R2H     R2L
                                        ;               R1H     R1L
                                        ;               ----
                                        ;                   R2L*R1L
                                        ;               R2H*R1L
                                        ;           R2L*R1H
                                        ;       R2H*R1H
                                        ;       --------
                                        ;       - - R E S U L T - -

 P          C 0000                      mult16:
 P          C 0000 79040000             step1:  mov.w   #0,r4          ;clear result register
 P          C 0004 0D25                         mov.w   r2,r5          ;save multiplicand

 P          C 0006 5092                 step2:  mulxu   r1l,r2         ;multiplier(L) x multiplicand(L)
 P          C 0008 0D23                         mov.w   r2,r3          ;1.     R3  <- R2L*R1L

 P          C 000A 0C5A                 step3:  mov.b   r5h,r2l        ;retrieve multiplicand(H)

 P          C 000C 5092                 step4:  mulxu   r1l,r2         ;multiplier(L) x multiplicand(H)
 P          C 000E 08A3                         add.b   r2l,r3h        ;2.     R3H <- R3H + (R2H*R1L)L
 P          C 0010 0E2C                         addx    r2h,r4l        ;       R4L <- (R2H*R1L)H + CY

 P          C 0012 0D52                 step5:  mov.w   r5,r2          ;retrieve multiplicand

 P          C 0014 5012                 step6:  mulxu   r1h,r2         ;multiplier(H) x multiplicand(L)
 P          C 0016 08A3                         add.b   r2l,r3h        ;3.     R3H <- R3H + (R2L*R1H)L
 P          C 0018 0E2C                         addx    r2h,r4l        ;       R4L <- R4L + (R2L*R1H)H + CY
 P          C 001A 9400                         addx    #0,r4h         ;       R4H <- CY

 P          C 001C 0C5A                 step7:  mov.b   r5h,r2l        ;retrieve multiplicand(H)

 P          C 001E 5012                 step8:  mulxu   r1h,r2
 P          C 0020 0942                         add.w   r4,r2          ;4.     R2  <- R4 + (R2H*R1H)
 P          C 0022 0D31                         mov.w   r3,r1          ;setup return results

 P          C 0024 5470                         rts                    ;return

                                                .end
*****TOTAL ERRORS        0
*****TOTAL WARNINGS      0
```

SECTION 5

Again our original multiplicand that was in R2 is no longer valid, so we must retrieve it from storage (see Figure 10). We must retrieve the high-byte of the saved multiplicand and place it into the lower half of register R2. Remember that the destination operand must exist in the lower half of the register in order for the multiply instruction to execute. This is performed with the following instruction:

```
mov.b        r5h,r2l
```

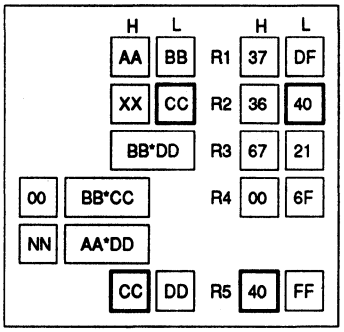


Figure 10 Step 7

In the eigth step (the final one of our multiplcation itself) we multiply the upper byte of the multiplicand by the upper byte of the multiplier (see Figure 11). This result (H'0DC0) is then added to the upper word (R4) of the previous results to provide our final answer, H'0E2F6721 (H'006F6721 + H'0DC00000). At this time we also move the result to registers R1 and R2 for return to the calling program (see Figure 12). This is performed by the following instructions:

```
mulxu        r1h,r2
add.w        r4,r2
mov.w        r3,r1
rts
```

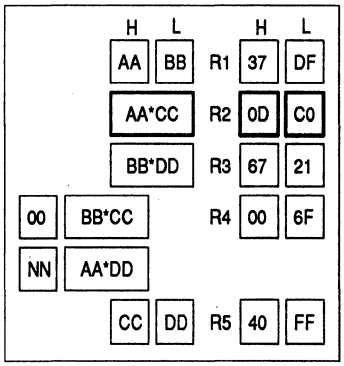


Figure 11: Step 8a

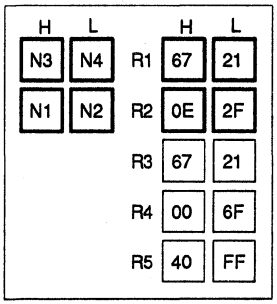


Figure 12: Step 8b

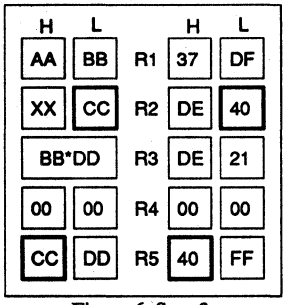


Figure 6: Step 3

The fourth step now multiplies the upper half of the multiplicand by the lower half of the multiplier. This result (H'37C0) is added to the previous result (see Figure 7), but not directly to it since some shifting of the results must be performed. We must add the low byte of this result (R2L) with the high byte of the previous result (R3H). We must then add the high byte of this result with the carry-over from the previous addition and place the result in R4L. This generates a 32-bit result of H'00389E21 (H'37C000 + H'DE21). This is performed with the following instructions:

```
mulxu     r1h,r2
add.b     r2l,r3h
addx      r2h,r4l
```

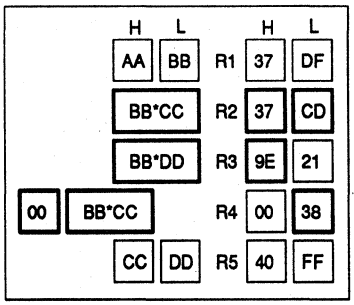


Figure 7: Step 4

We have now performed an 8x16 multiply function, but that is not what we wanted to do, but we are halfway through. Again our original multiplicand that was in R2 is no longer valid, so we must retrieve it from storage (see Figure 8). This is performed with the following instruction:

```
mov.w     r5,r2
```

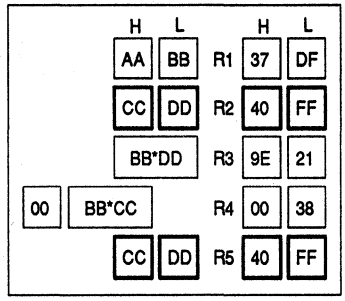


Figure 8: Step 5

The sixth step now multiplies the upper half of the multiplier with the lower half of the multiplicand. The result of this operation (H'36C9) is also added to the previous result, again with some shifting performed (see Figure 9). We must add the low byte of this result (R2L) with the high byte of the previous result (R3H). We must then add the high byte of this result with the carry-over from the previous addition and the current value in R4L, and place the result in R4L. The carry-over from this addition is placed into R4H. This alters our previous 32-bit result to be H'006F6721 (H'00389E21 + H'36C900). This is performed with the following instructions:

```
mulxu     r1h,r2
add.b     r2l,r3h
addx      r2h,r4l
addx      #0,r4h
```

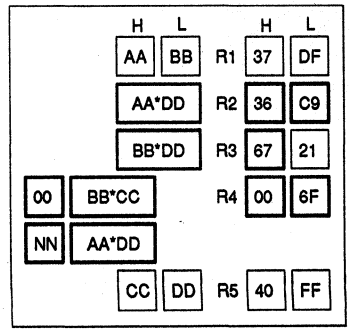


Figure 9: Step 6

SECTION 5

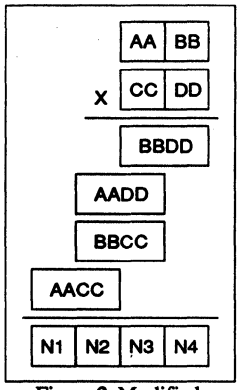


Figure 2: Modified Multiplication Procedure

used for performing the function. If the user decides it is important to save the current state of the working registers, then it is easy to add "push" instructions at the beginning of the routine to save the data, and "pop" instructions at the end of the routine to restore the data. As we discuss this routine, we will examine the modified procedure in detail as well as an example of data used in the execution of the routine.

Before the routine is called, the user must place the two 16-bit operands to be multiplied in registers R1 and R2 (see Figure 3). For this discussion, R1 will contain the "multiplier" and R2 will contain the "multiplicand." The result will be returned in these same registers, so if the original operands are to be used later, it is also up to the user to save them elsewhere. In our example, we will use the data H'37DF and H'40FF for the multiply routine.

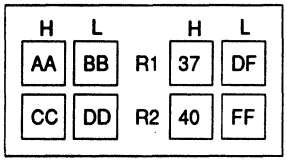


Figure 3: Parameter Passing

The first step we must perform in this routine is to prepare the destination working registers (R3 and R4, only R4 requires clearing) and also to save the multiplicand into a temporary register (R5) because we will need it again later (see Figure 4). This is performed with the following instructions:

```
mov.w    #0,r4
mov.w    r2,r5
```

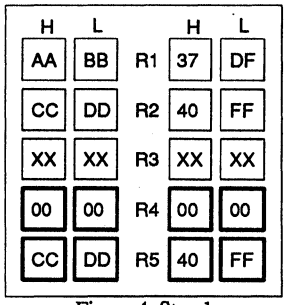


Figure 4: Step 1

In the next step, we perform the multiplication of the two low-bytes of the multiplier and multiplicand (see Figure 5). This result (H'DE21), which exists in R2, is then placed into our destination registers (R3 and R4, only R3 is required at this time). This is performed with the following instructions:

```
mulxu    r1l,r2
mov.w    r2,r3
```

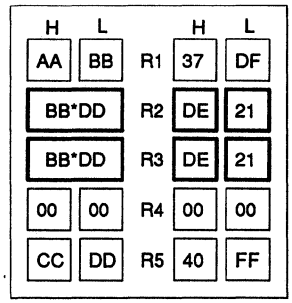


Figure 5: Step 2

Our original multiplicand that used to be in R2 is no longer valid since R2 has been corrupted because of the multiply instruction. In the third step (see Figure 6), we must retrieve the high-byte of the saved multiplicand and place it into the lower half of register R2. Remember that the destination operand must exist in the lower half of the register in order for the multiply instruction to execute. This is performed with the following instruction:

```
mov.w    r5,r2
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H8/300 Family

Application Note

16 × 16 Multiply

Tom Hampton

INTRODUCTION

The H8/300 CPU core is very powerful considering that it is an 8-bit architecture. Part of its power comes from the flexible instruction set, which allows for many byte and word operations. Part of its power also comes from the ability of the architecture to allow use of the general purpose register as either 16-bit or 8-bit registers as needed. While the instruction set is extremely powerful in its ability to handle bit-wide and byte-wide data, it only has a small number of instructions (other than data transfer) that allow operations on word-wide data.

One of the instructions that is noticeably missing is in the arithmetic area. While the CPU has the capability of doing an 8x8 unsigned multiply, it lacks the capability of doing a 16x16 unsigned multiply. Even though this instruction does not exist, the function can be easily implemented using the instructions currently available.

In this application note, we will examine a routine that provides the user with a 16x16 unsigned multiply function as well as perform it in a very short amount of time. In performing this operation, we will make use of the flexible instruction set as well as the architecture's flexibility to be used as either byte-wide or word-wide registers. Only five general purpose registers are used in this routine, including the two that pass the parameters. No fancy tricks were used, but it was important to pay special attention to the manner in which the multiply instruction operated on the registers.

This instruction requires that the destination operand be contained in the lower half of a 16-bit register even though the entire register will be used to hold the results. For this reason, it was necessary to use other register for working registers and temporary storage.

MULTIPLICATION PROCEDURES

Standard Procedure

If one examines the normal procedure of a 16-bit multiply operation (see Figure 1), it would be easy to see the steps that would be taken if the H8/300 CPU had a 16-bit unsigned multiply instruction. The first step would be to multiply the 16-bits of one operand by the lower "digit" of the second operand, thus potentially yielding a 24-bit response. The second step would be to multiply the same 16-bit one operand by the higher "digit" of the second operand, which could yield yet another 24-bit response. The final step would be to add these 24-bit responses together in the proper sequence (with required shifting) to form a 32-bit result. This is the procedure that we are all used to for multiplication.

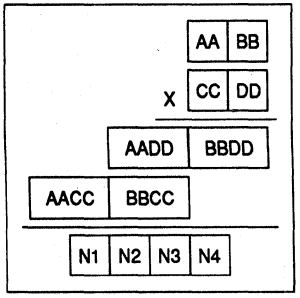


Figure 1: Standard Multiplication Procedure

Modified Procedure

Since the H8/300 CPU does not have a 16-bit multiply instruction, the normal procedure cannot be used. Instead we must modify the procedure to account for the creation of only 16-bit intermediate results (see Figure 2). In this procedure, we must multiply the individual 8-bit "pieces" of the operand to form intermediate results. This requires four steps since we actually have four bytes of operand data that we must multiply together. The results of these pieces of intermediate data must then be added together in the proper sequence (with shifting) to form the final 32-bit result.

SOFTWARE DESCRIPTION

The routine written to perform the 16x16 unsigned multiply function is shown in Listing 1. You may wish to refer to this listing during the following discussions. The routine occupies only 38 bytes of code space while executing in 8.6 µsec. One of the first things to note in this routine is that no registers are saved even though some of the general purpose registers are

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-027A
Topic	Debug information		

Question

1. If I link a program with the /DEBUG option and store it as an absolute module, then use the converter to convert it from SYSROF type format to S type or HEX type format, will debug information be included?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
○	Software
	Development tools
	Miscellaneous

Answer

1. Debug information is included only in the SYSROF type format.

   No debug information is included in the S type or HEX type format.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-026A
Topic	Omitting :3, :8, :16, and @		

Question

1. In using the H8/300 cross assembler, when can you do the following?

(1) Omit :3

(2) Omit :8

(3) Omit :16

(4) Omit @

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
○	Software
	Development tools
	Miscellaneous

Answer

1. These can be omitted in the following cases.

(1) You can omit :3 when coding an immediate value in a bit manipulation instruction.

(2) You can omit :8 when coding an immediate value in a byte operand, or an address value in the absolute addressing mode.

(3) You can omit :16 when coding an immediate value in a word operand, an address value in the absolute addressing mode, or a displacement in the register indirect with displacement addressing mode.

(4) You can never omit @.

@ indicates the addressing mode. You must code this symbol whenever using the absolute address, register indirect, memory indirect, or register indirect with displacement addressing mode.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-025A
Topic	Setting a symbol with .EQU $		

Question

In a program coded like this:

```
LBL1    .EQU    $    ........................ (1)
LBL2                 ........................ (2)
```

What values are assigned to LBL1 and LBL2?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
○	Software
	Development tools
	Miscellaneous

Answer

1. In both lines (1) and (2), the current location counter value is assigned to LBL1 or LBL2.

   The presence or absence of .EQU $ does not make any difference.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-024A
Topic	Moving data specified by a label		

Question

How do you program the following operations?

1. Transfer data defined by an EQU directive to a general register.
2. Transfer data defined by a DATA directive to a general register
3. Transfer data in an area defined by an RES directive to a general register

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
○	Software
	Development tools
	Miscellaneous

Answer

Code your software as follows:

1. MOV.B #[label-name], Rn
   (W)
2. MOV.B @[label-name], Rn
   (W)
3. MOV.B @[label-name], Rn
   (W)

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-023A
Topic	BRA and BRN instructions		

Question

1. What does the "True" condition mean in the BRA (or BT) instruction?
2. What does the "False" condition mean in the BRN (or BF) instruction?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
○	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The BRA instruction is equivalent to a JMP instruction, performing the operation PC + disp → PC. Unlike the JMP instruction, however, it can branch only in the range from +255 to –256 bytes.
2. The BRN instruction is equivalent to two NOP instructions.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-022A
Topic	Notes on stack usage		

Question

1. Are there any points to note about stack usage?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
○	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. When used for exception handling, the stack is always accessed a word at a time, regardless of the data size.

   When accessing the stack using the post-increment or pre-decrement addressing mode, you should always use word access.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-021A
Topic	Support of DAA and DAS instructions for INC and DEC		

Question

1. The DAA instruction is intended for use with add (ADD) instructions, but what happens if DAA is executed after an INC instruction?
2. The DAS instruction is intended for use with subtract (SUB) instructions, but what happens if DAS is executed after a DEC instruction?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
○	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Basically, the DAA instruction is not supported after execution of an INC instruction.
2. Basically, the DAS instruction is not supported after execution of a DEC instruction.

Related Manuals

Manual Title

H8/300 Series Programming Manual

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

The actual operation performed is determined by the flag states.

SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-020A
Topic	Sampling and acceptance of interrupts during sleep mode		

Question

1. When are interrupts sampled during sleep mode?
2. If an interrupt is sampled, how many system clock cycles later does the chip wake up?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
○	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Interrupts are sampled on the falling edge of the system clock, just as in active mode.
2. The chip wakes up from sleep mode 0.5 system clock cycle after the interrupt is sampled.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

QA8300-004A

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-019A
Topic	Interrupts during fetching and execution of SLEEP instruction		

Question

1. How does the H8/300 CPU operate if it receives an interrupt while fetching a SLEEP instruction?
2. How does the H8/300 CPU operate if it receives an interrupt while executing a SLEEP instruction?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
○	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The SLEEP instruction is executed after the end of interrupt handling, without passing through sleep mode.
2. Sleep mode is released to handle the interrupt. At the end of interrupt handling, the next instruction after the SLEEP instruction is executed.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-018A
Topic	Notes on entering sleep mode		

Question

1. Are there any points to note when the H8/300 CPU enters sleep mode by executing the SLEEP instruction?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
○	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The points listed below should be noted, depending on the method used to recover from sleep mode.

	Recovery Method	
	NMI Interrupt	**IRQn Interrupt**
Necessary register settings and other conditions	Set I bit in CCR to 1, or clear all interrupt enable bits to 0.	Clear I bit in CCR to 0, and clear interrupt enable bits to 0, except for interrupts used for recovery, and make sure NMI is not requested.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

QA8300-015A

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-017A
Topic	Entering sleep mode		

Question

1. Does the H8/300 CPU always enter sleep mode when it executes the SLEEP instruction?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
○	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Executing the SLEEP instruction does not always put the H8/300 to sleep. The SLEEP instruction enters sleep mode in the following cases:

(1) The software standby bit (SSBY) in the system control register (SYSCR) is cleared to 0, the $\overline{RES}$ and $\overline{STBY}$ lines are high, there is no NMI request, and all other interrupt requests are disabled (or no other interrupt is requested).

(2) The SSBY bit in the SYSCR is cleared to 0, the $\overline{RES}$ and $\overline{STBY}$ lines are high, there is no NMI request, and the I bit in CCR is set to 1.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

QA8300-014A

Additional Information

SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-016A
Topic	Acceptance of external IRQ1 after recovery from software standby mode		

Question

1. Suppose the chip enters software standby mode with the IRQ sense control register (ISCR) cleared to H'00 (selecting level triggering) and the I bit in CCR cleared to 0. During software standby mode the $\overline{IRQ1}$ line is driven low, then the chip recovers to active mode. If the $\overline{IRQ1}$ line continues to be held low after that, will the $\overline{IRQ1}$ interrupt be accepted?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
○	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Yes, it will be accepted.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-015A
Topic	Notes on entering software standby mode		

Question

1. Are there any points to note when the H8/300 CPU enters software standby mode by executing the SLEEP instruction after setting the software standby bit (SSBY) in the system control register (SYSCR) to 1?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
○	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The points listed below should be noted, depending on the method used to recover from software standby mode.

	Recovery Method	
	NMI Interrupt	**IRQ_0–IRQ_2 Interrupt**
Necessary register settings and other conditions	Set I bit in CCR to 1, or clear bits IRQ_0E–IRQ_2E to 0.	Clear I bit in CCR to 0, and clear interrupt enable bits to 0, except for interrupts IRQ_0–IRQ_2 if used for recovery, and make sure NMI is not requested.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

QA8300-018A

Additional Information

SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-014A
Topic	Entering software standby mode		

Question

1. Does the H8/300 CPU always enter software standby mode when the SLEEP instruction is executed after the software standby bit (SSBY) in the system control register (SYSCR) is set to 1?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
○	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The H8/300 CPU does not always enter software standby mode when the SLEEP instruction is executed after the SSBY bit in SYSCR is set to 1. It enters software standby mode when the SLEEP instruction is executed after the SSBY bit in SYSCR is set to 1 in the following cases:

(1) If $\overline{RES}$ and $\overline{STBY}$ are both high, there is no NMI request, and IRQ_0 and IRQ_2 are disabled (or neither interrupt is requested)

(2) If $\overline{RES}$ and $\overline{STBY}$ are both high, there is no NMI request, and the I bit in CCR is set to 1.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

QA8300-017A

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-013A
Topic	Recovery from hardware standby mode		

Question

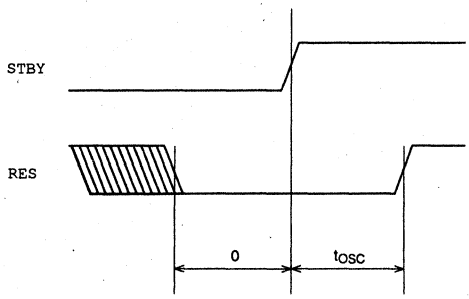
1. The chip can be recovered from hardware standby mode by driving $\overline{RES}$ low, then $\overline{STBY}$ high. How long before $\overline{STBY}$ goes high does $\overline{RES}$ have to go low?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
○	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. When the chip is recovered from hardware standby mode, the setup time of $\overline{RES}$ to $\overline{STBY}$ becomes 0.



Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

QA8300-009A

Additional Information

SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-012A
Topic	Acceptance of external IRQ1 after recovery from hardware standby mode		

Question

1. Suppose $\overline{IRQ1}$ is driven low during hardware standby mode, and is still low when the chip is recovered from hardware standby mode?

   If the $\overline{IRQ1}$ line is kept low after recovery, will the interrupt be accepted?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
○	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The chip recovers from hardware standby mode via the reset state.

   IER (the IRQ enable register) is therefore initialized and IRQ1 is disabled. The IRQ1E bit (IRQ1 enable) in IER is cleared to 0. The interrupt is not accepted immediately after recovery.

   If the IRQ1E bit in IER is later set to 1, the I bit in CCR is cleared to 0, and the $\overline{IRQ1}$ line is still low, then the interrupt will be accepted.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-011A
Topic	Instruction execution at changeover to hardware standby mode		

Question

1. When a low $\overline{\text{STBY}}$ input drives the H8/300 CPU into hardware standby mode, what happens to the instruction currently being executed?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
○	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The instruction being executed is aborted, without waiting for the instruction to end. Normal execution of the instruction is not assured.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-010A
Topic	Entering hardware standby mode		

Question

1. Will the chip enter hardware standby mode if $\overline{STBY}$ is driven low while $\overline{RES}$ is high?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
○	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Yes. Regardless of its current state, the chip enters hardware standby mode whenever $\overline{STBY}$ is driven low.

Related Manuals

Manual Title

H8/300 Series Programming Manual

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

QA8300-009A

Additional Information

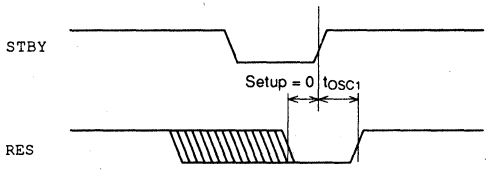
HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-009A–2
Topic	Hardware standby mode entry timing		

Answer

(2) When it is not necessary to hold RAM contents, the setup time of RES to $\overline{STBY}$ becomes 0.



Additional Information

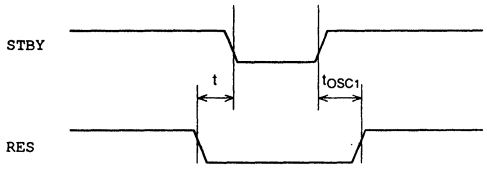
SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-009A–1
Topic	Hardware standby mode entry timing		

Question

1. Are there any restrictions on the time t in the drawing below for entering the hardware standby mode?



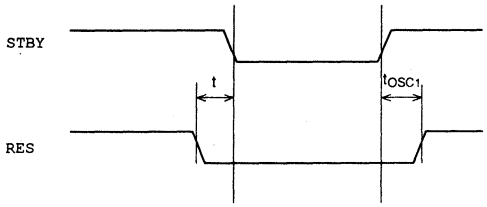
Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
○	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The following restrictions apply.

(1) To hold RAM contents, t must be at least 10 system clock cycles.



(Continued on next page)

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

QA8300-013A

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-008A
Topic	Stack pointer initialization immediately after a reset		

Question

1. Why is it necessary to initialize the stack pointer immediately after a reset, even though all interrupts are inhibited immediately after a reset?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
○	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Sampling of the $\overline{\text{NMI}}$ signal starts one system clock after the reset. If the $\overline{\text{NMI}}$ signal is active at this time, the NMI interrupt will be accepted as soon as the first instruction has been executed after the reset.

   To prevent program crashes, you should therefore initialize the stack pointer immediately after the reset.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

It is only for one system clock cycle that all interrupts are inhibited immediately after a reset.

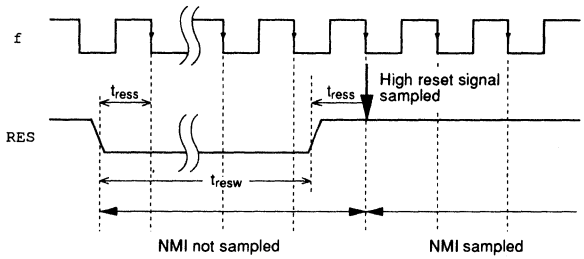
SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-007A–2
Topic	NMI sampling and acceptance immediately after a reset		

Answer

[Example]



Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-007A–1
Topic	NMI sampling and acceptance immediately after a reset		

Question

1. When is the reset signal sampled?
2. When is the $\overline{\text{NMI}}$ signal sampled?
3. After a reset, when is the $\overline{\text{NMI}}$ signal first sampled, and when can it first be accepted?

Classification—H8/300

	Registers
	Read timing
	Write timing
	Interrupts
○	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The reset signal is sampled on the falling edge of the system clock.
2. The $\overline{\text{NMI}}$ signal is also sampled on the falling edge of the system clock.

   The $\overline{\text{NMI}}$ signal is not sampled during the reset period, however.
3. Sampling of the $\overline{\text{NMI}}$ signal starts from the first system clock cycle in which the high reset signal is sampled. The NMI interrupt becomes acceptable when the first instruction has been executed after the chip comes out of reset.

(See next page)

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-006A
Topic	Pending interrupts		

Question

IRQn (external interrupts) can be disabled in the two ways listed below. What happens to pending interrupts in these two cases?

1. Disabled in IER (IRQ enable register)
2. Disabled by I bit in CCR

Classification—H8/300

	Registers
	Read timing
	Write timing
○	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

Pending interrupts are handled as follows

1. If an interrupt is disabled in IER (IRQnE bit = 0), it is not held pending

   In the disabled state, the $\overline{IRQn}$ signal input is ignored. Interrupt requests made in the disabled state are not sampled. The interrupt will not be handled even if it is later enabled in IER.

2. An interrupt that is requested while enabled in IER (IRQnE bit = 1) but masked because the I bit in CCR is set to 1 is held pending in the CPU's interrupt controller. If the I bit is later cleared to 0, the interrupt will be handled.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-005A
Topic	NMI requirements		

Question

1. What are the requirements for NMI?

Classification—H8/300

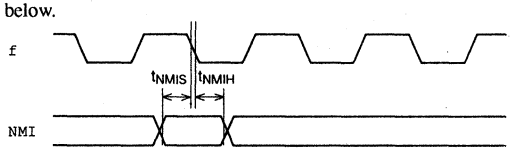
	Registers
	Read timing
	Write timing
○	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The following requirements apply to NMI.

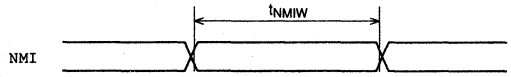
(1) During normal operation

There are requirements for setup and hold times. See the drawing below.



(2) To recover from software standby mode

The requirement is given in terms of NMI width t_{NMIW}.



Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

QA8300-004A

Additional Information

H8/330 (with 10-MHz system clock):
t_{NMIS} = 110 ns (Min), t_{NMIH} = 10 ns (Min), t_{NMIW} = 200 ns (Min)

SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-004A
Topic	Interrupt sampling and acceptance		

Question

1. When are interrupts sampled?
2. In cases where instruction execution may last several hundred states or more, as with the block data transfer instruction, when is an interrupt accepted?

Classification—H8/300

	Registers
	Read timing
	Write timing
○	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Interrupts are sampled on every falling edge of the system clock.
2. Interrupts are accepted at the end of the instruction (but not at the end of some CCR control instructions).

Related Manuals

Manual Title

H8/300 Series Programming Manual

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

QA8300-005A
QA8300-020A

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-003A
Topic	Usage of general registers		

Question

1. Is it possible to use both 8-bit and 16-bit general registers at the same time?

Classification—H8/300

○	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Yes. For example:

[Example]

R0H	R0L
R1	
R2H	R2L
R3	
R4	
R5	
R6H	R6L
R7 (SP)	

```
MOV.B #H'FF:8,R0H
MOV.B #H'E0:8,R0L
MOV.B R0H,R2H
MOV.B #H'00:8,R2L
MOV.W #H'4000:16,R1
MOV.W R1,R3
MOV.W @H'4000:16,R4
MOV.W R4,R5
MOV.B @H'4000,R6H
MOV.B R6H,R6L
```

Note that R7 is implicitly used as the stack pointer.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

SECTION 5

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-002A
Topic	Definition of V flag in CCR		

Question

Is there a difference in the way the V flag is modified by:

1. Instructions that operate on byte data, and
2. Instructions that operate on word data?

Classification—H8/300

○	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

The V flag in CCR is modified as follows:

1. Byte add and subtract instructions set the V flag to 1 if a register value changes from H'7F or less to H'80 or greater, or from H'80 or greater to H'7F or less.
2. Word add and subtract instructions set the V flag to 1 if a register value changes from H'7FFF or less to H'8000 or greater, or from H'8000 or greater to H'7FFF or less.

Related Manuals

Manual Title

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Questions and Answers

Product	H8/300 CPU	Q&A No.	QA8300-001A
Topic	Register contents after power-up reset		

Question

1. What are the register contents after a power-up reset?

Classification—H8/300

○	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Register contents are as follows:

(1) Registers of on-chip supporting modules

Initialized. (In some cases the initial values are undetermined. See the User's Manual for details.)

(2) CPU registers

The program counter is loaded from the vector table. The I bit in CCR is set to 1. Registers R0 to R7 and the other CCR bits have undetermined contents.

Note: RAM contents are also undetermined.

Related Manuals

Manual Title

H8/300 Series Hardware Manuals

Other Technical Documentation

Document Name

Related Microcomputer Technical Q&A

Title

Additional Information

SECTION 5

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Contents

How to Use Microcomputer Technical Questions and Answers

Technical Questions and Answers has been created by arranging technical questions actually asked by users of Hitachi microcomputers in a question-and-answer format. It should be read for technical reference in conjunction with the User's Manual.

Technical Questions and Answers can be read before beginning a microcomputer application design project to gain a more thorough understanding of the microcomputer, or during the design process to check up on difficult points.

This document gives concrete explanations of points that are not completely covered in the User's Manual. Most of the contents is user-inspired. Future editions of this document will contain further information, and efforts will also be made to improve the User's Manual.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H8/300 CPU

Application Note

Technical Q & A

Preface

The H8/300 CPU is a high-speed central processing unit with an original Hitachi architecture.

The main features of the H8/300 CPU are listed below.

- General-register architecture
  - — Two-way register configuration
    - — Sixteen 8-bit general registers, or
    - — Eight 16-bit general registers

- High-speed operation
  - — Maximum clock rate is 10 MHz (internal system clock)
  - — High-speed arithmetic operations
    - — 8- or 16-bit register-register add or subtract: 0.2 µs (at 10 MHz)
    - — 8 × 8-bit multiply: 1.4 µs (at 10 MHz)
    - — 16 ÷ 8-bit divide: 1.4 µs (at 10 MHz)

- Concise instruction set
  - — 57 Basic instruction types

— Data transfer instructions	3
— Arithmetic instructions	14
— Logic operation instructions	4
— Shift instructions	8
— Bit manipulation instructions	14
— Branch instructions	5
— System control instructions	4
— Block data transfer/EEPROM write instruction	1

- Maximum 64-kbyte address space

- Three operating modes
  - — Mode 1: expanded mode, on-chip ROM disabled
  - — Mode 2: expanded mode, on-chip ROM enabled
  - — Mode 3: single-chip mode

- Three power-down modes
  - — Sleep mode
  - — Software standby mode
  - — Hardware standby mode

- Eight addressing modes

— Register direct	Rn
— Register indirect	@Rn
— Register indirect with displacement	@(d:16, Rn)
— Register indirect with post-increment or pre-decrement	@Rn+ or @–Rn
— Absolute address	@aa:8 or @aa:16
— Immediate	#xx:8 or #xx:16
— Program-counter relative	@(d:8, PC)
— Memory indirect	@@aa:8

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

7. CONCLUSION

Smart card applications do require a high-level of performance and memory requirements on a microcontroller. With their high-performance CPU core and large memory capacities, the H8/310 and H8/3101 devices provide the necessary system requirements. With their lower power consumption and physical size, these devices also meet the necessary physical requirements for smart card applications.

APPENDIX A: H8/310 PUBLICATIONS

Further information on the H8/310 can be found in the documentation (available from Hitachi America, Ltd.) listed below.

Title	Hitachi Order Number
H8/310 Architectural Overview	M21T031
H8/300 Programming Manual	M21T004
H8/310 Hardware User's Manual	ADE-602-024

By setting control register bits, EEPMOV can also be used to erase and overwrite pages. The control registers also provide an EEPMOV disable bit. Specific EEPROM data may be protected by writing into the protect area of the EEPROM. This is done by setting the EEPROM protect bit in EPR register, and then overwriting the page the user wishes to protect.

Data in the H8/310's EEPROM is guaranteed for a minimum of ten years. Each page may accept up to ten thousand writes.

4. INPUT/OUTPUT

The H8/310 has a simple I/O structure, adapted to its role as a smart card IC. There is a single, one-bit bidirectional port. The port is accessed as the MSB of memory location H'FFFE. The data direction register for the port is the MSB of the adjacent memory location. The H8/3101 has two of these one-bit ports, with a memory mapped location for each. The high-speed operation of the H8/300 series CPU core allows the creation of a software UART that is capable of asynchronous operations up to 9600 baud.

5. PACKAGING/DIE CHARACTERISTICS

5.1 H8/310 DIE

The H8/310 is available as a die or as a prefabricated button. The die drawing is shown in Figure 3.

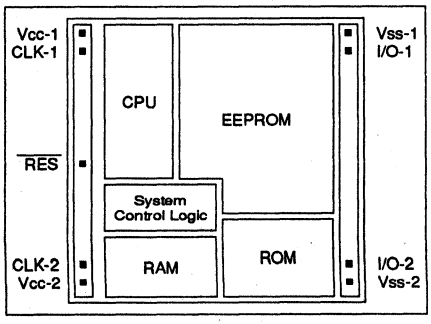


Figure 3: H8/310 Die Diagram

5.2 H8/310 Chip On Board (COB) Module

The H8/310 is also available in a button assembly, which offers all the electrical components of an ISO smart card, assembled and tested by Hitachi. Custom contact metallization patterns are also available.

5.3 SMALL OUTLINE PACKAGE (SOP)

For applications other than smart cards, where a standard package might be required, the H8/3101 is available in a SOP-10 surface mount package.

6. SUPPORT TOOLS

6.1 SOFTWARE

6.1.1 Cross Tools

Code development for the H8/310 is possible using our Cross-Assembler for the IBM® PC® and compatibles, or using our optimizing, ANSI compliant C Compiler. The tools work with the Librarian/Linker, so C routines can be easily integrated into assembly language applications.

6.1.2 Simulators/Debuggers

Our new XRAY simulator and debugger is an interactive, windowed environment for source level symbolic debugging for both C language and assembly code. With XRAY, users can step through code. Concurrent windows can display the full C source statements, the compiler output in assembly, and the status of any registers in the processor. XRAY works with a software simulator, or, using the same user interface, with the ASE emulator.

6.2 HARDWARE

The Hitachi ASE emulator, a full featured hardware development system, is available for the H8/310. The ASE provides real-time emulation, with software and hardware breakpoints, as well as software trace triggering. The trace buffers can monitor both instruction accesses as well as data reads and writes to all memory and I/O locations. Other ASE functions include performance and memory coverage analysis. Parallel mode allows the examination of processor registers while code is running.

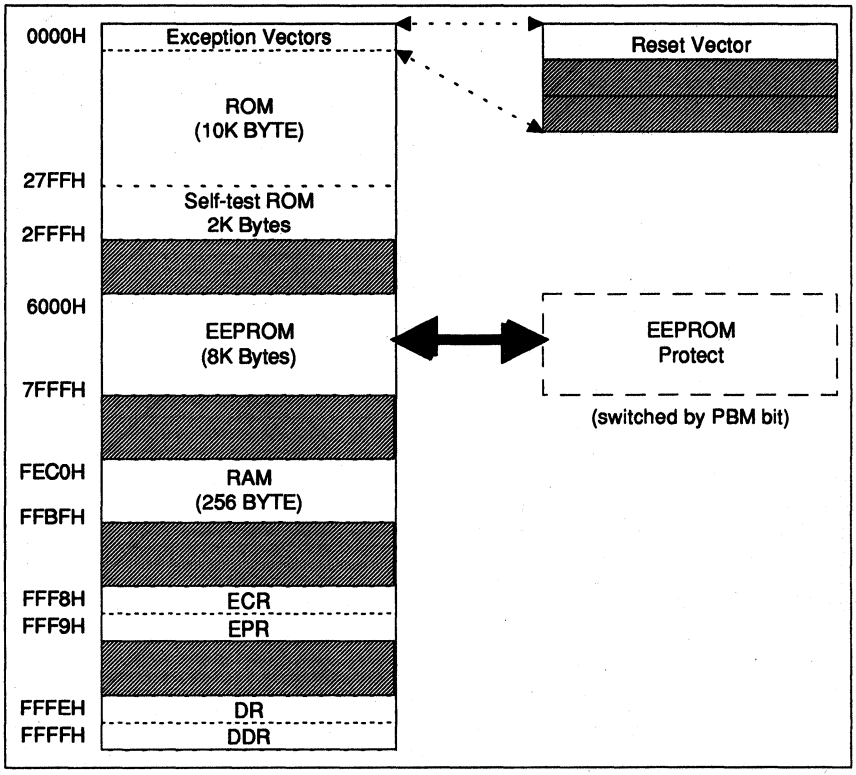


Figure 2: Memory Block Diagram

3.1 ROM

The 10K of usable ROM (2K is reserved for testing) on the H8/310 is linked to the CPU via a 16-bit wide bus. Two bytes of data thus can be transferred in the same amount of time as one byte takes, without any performance penalties. This type of access takes only two CPU clock cycles.

3.2 RAM

Like the masked ROM memory, the 256 bytes of RAM are also arranged in 16-bit words. It too can be accessed in two CPU clock cycles. This RAM memory can also be used for more than just data storage as the CPU allows programs to be executed from RAM memory.

3.3 EEPROM

The H8/310's 8K x 8 EEPROM can be used for storing data or for program code. Reading the EEPROM is the same as ROM reads except that the data is byte wide.

The EEPROM on the H8/310 is organized as 256 pages of 32 bytes. Writes into a page of EEPROM are accomplished using the EEPMOV instruction. EEPMOV uses the contents of three general registers to set up a 32 byte transfer from RAM into EEPROM. The execution of this instruction makes use of an on-chip timer and high voltage generator to perform the write operation in approximately 10 msec. A status bit is available that suggests if any power supply fluctuation occurred during the write time. This would allow the software to perform EEPROM re-writes when voltage drops may cause data corruption.

Rn	Register Direct	Rn for 16-bit, RnL or RnH for 8-bit
@Rn	Register Indirect	Contents of register are used as a pointer
@Rn,disp	Register Indirect w/(16-bit displacement)	Contents of register are used as a pointer with displacement
@Rn+ @-Rn	Register Indirect w/Post-Increment or Pre-Decrement	Contents of register are used as a pointer with automatic adjustment during instruction execution
#nn:8 #nn:16	Immediate (8-bit or 16-bit data)	
@aa:8 @aa:16	Absolute Address (8-bit or 16-bit)	16-bit addresses cover entire memory range; 8-bit addresses cover H'FF00 through H'FFFF
@aa:8	PC relative (8-bit displacement)	Branch instructions
@@aa:8	Memory Indirect	Addressed memory contents are used as a pointer

Table 1: H8/300 CPU Addressing Modes

While the architecture is register oriented, many addressing modes are supported by the CPU that allow the user easy access to both memory and register contents. The addressing modes are listed in Table 1.

2.1.2 Operating Modes

The H8/310 has only two states of operation, program execution and reset. To this, the H8/3101 adds the "sleep" mode and exception handling states. The sleep mode of operation reduces the H8/3101's current requirement to less than 100 microamperes. Recovery from sleep mode may be accomplished by processor reset, or by external interrupt (INT1). Both events trigger a transition through the exception handling state to the program execution state.

2.1.3 INTERRUPTS

The H8/300 series interrupt features are not used in the H8/310. The H8/3101 however, uses an interrupt on one of the I/O ports to wake up the processor from sleep mode.

2.2 PERFORMANCE

The maximum internal clock rate is 5 MHz, obtained from a 10 MHz external input. At this clock rate, register oriented instructions are remarkably fast. Examples of some instruction execution times are listed in Table 2.

8- or 16-bit register add	0.4 µS
8 x 8 multiply	2.8 µS
16 / 8 divide	2.8 µS

Table 2: H8/300 Execution Times

3. MEMORY

The H8/310 provides a variety of memory types to support smart card applications. Shown in Figure 2, the memory map consists of a single 64K address space containing 10K of ROM, 8K of EEPROM, and 256 bytes of RAM.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H8/310

Application Note

A Microcontroller for Smart Card Application

W. Stan Ayers
Tom Hampton

1. INTRODUCTION

Smart cards are the next generation of transaction and information exchange vehicles. They promise to transform the way we carry data and buy services and products in the future. For smart cards to fulfill their broad potential, the on board LSIs must meet a certain baseline of performance.

True general purpose smart cards require sufficient user storage for complex (perhaps multiple) applications. This implies that large program storage and user data storage areas are required. Also needed is a powerful CPU to support encryption algorithms (such as DES, RSA, and FEAL8) and still give fast response.

The H8/300 core processor from Hitachi answers the processing power issue very capably. In adapting it to smart card use, Hitachi added generous ROM and EEPROM areas, and I/O specifically designed for smart card use. We will review these special purpose devices, the H8/310 and the new H8/3101, in this paper. Since the H8/3101 is similar to the H8/310, we will refer only to the H8/310 throughout the paper, except where the differences are important.

2. ARCHITECTURAL OVERVIEW

2.1 H8/300 SERIES CPU

The H8/300 series CPU is a general register machine with sixteen 8-bit registers (or eight 16-bit registers), which support a speed-oriented instruction set. The Program Counter and the Condition Code Register are the only other registers that are part of the CPU core. Five of the eight bits in the CCR are used by the CPU to hold the status of the last operation. The general purpose registers and Condition Code Register are shown in Figure 1.

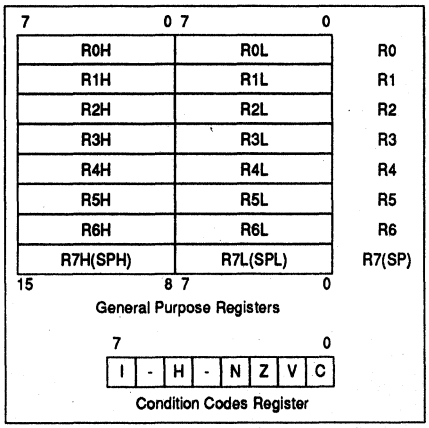


Figure 1: H8/300 CPU Registers

2.1.1 Instruction Set

The instruction set comprises fifty-four types, including powerful bit manipulation and accumulation, multiply and divide, and data transfer. Each instruction has optimum addressing modes designed to enhance speed or save code (arithmetic instructions, for example, are register oriented).

SECTION 5

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Section 5

H8 Family
H8/3XX Series

HITACHI®

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Section 5

H8 Family

- H8/3XX Series
- H8/5XX Series

HITACHI®

HITACHI

Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD66780 (LCD-II A)

Tech Notes

Application Engineering

Kash Yajnik

MPU READ Operation (Data Sheet Changes)

When a Micro processor reads the HD66780 registers or requests the eight bits of pripheral data, the associated interface signals are shown in timing diagram below in Figure 53. The read operation parameters with their older values (Incorrect) and the revised values (Correct) are also listed accordingly :

Read Operation (Reading Data from HD66780 to MPU)

Item	symbol	Min	Max	Unit	Test Cndition
Incorrect					
Enable Pulse Width(High level)	PWEH	220	–	ns	Fig.53
Data Delay Time	tDDR	–	120	ns	Fig.53
Data Hold Time	tDHR	20	100	ns	Fig.53
Correct					
Enable Pulse Width(High level)	PWEH	250	–	ns	Fig.53
Data Delay Time	tDDR	–	250	ns	Fig.53
Data Hold Time	tDHR	20	–	ns	Fig.53

Read Operation

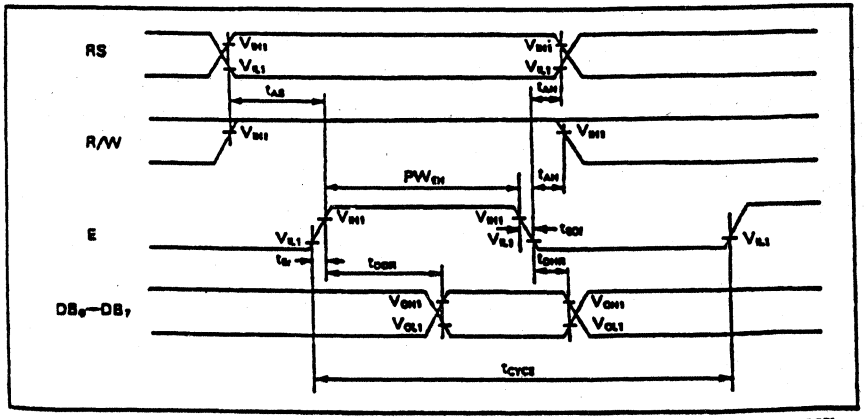


Figure 53. Bus Read Operation Sequence (Reading Data from HD66780 to MPU)

SECTION 4

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD66108T 165 Channel Driver with RAM

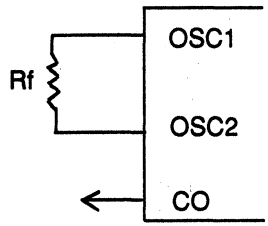
Tech Notes

Application Engineering

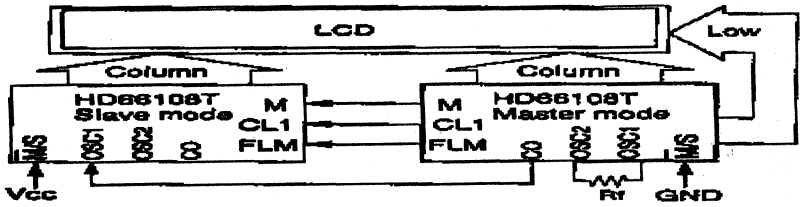
Kash Yajnik

Internal Oscillator

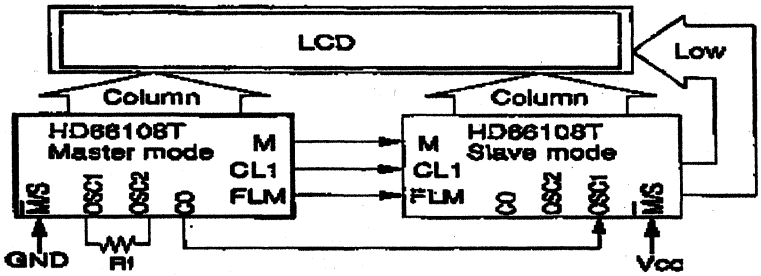
To use the on chip oscillator circuit, connect the resistor " Rf " between the terminals " OSC1" and " OSC2 ". For synchronous slave operation, the internally generated waveform is output on the " CO " terminal. The " Rf " resistor tolerance should be +2% , -2% or better. The resisitor wiring length should be minimized since the oscillation frequency is affected by the terminal capacitance. Refer to the Figure shown below:



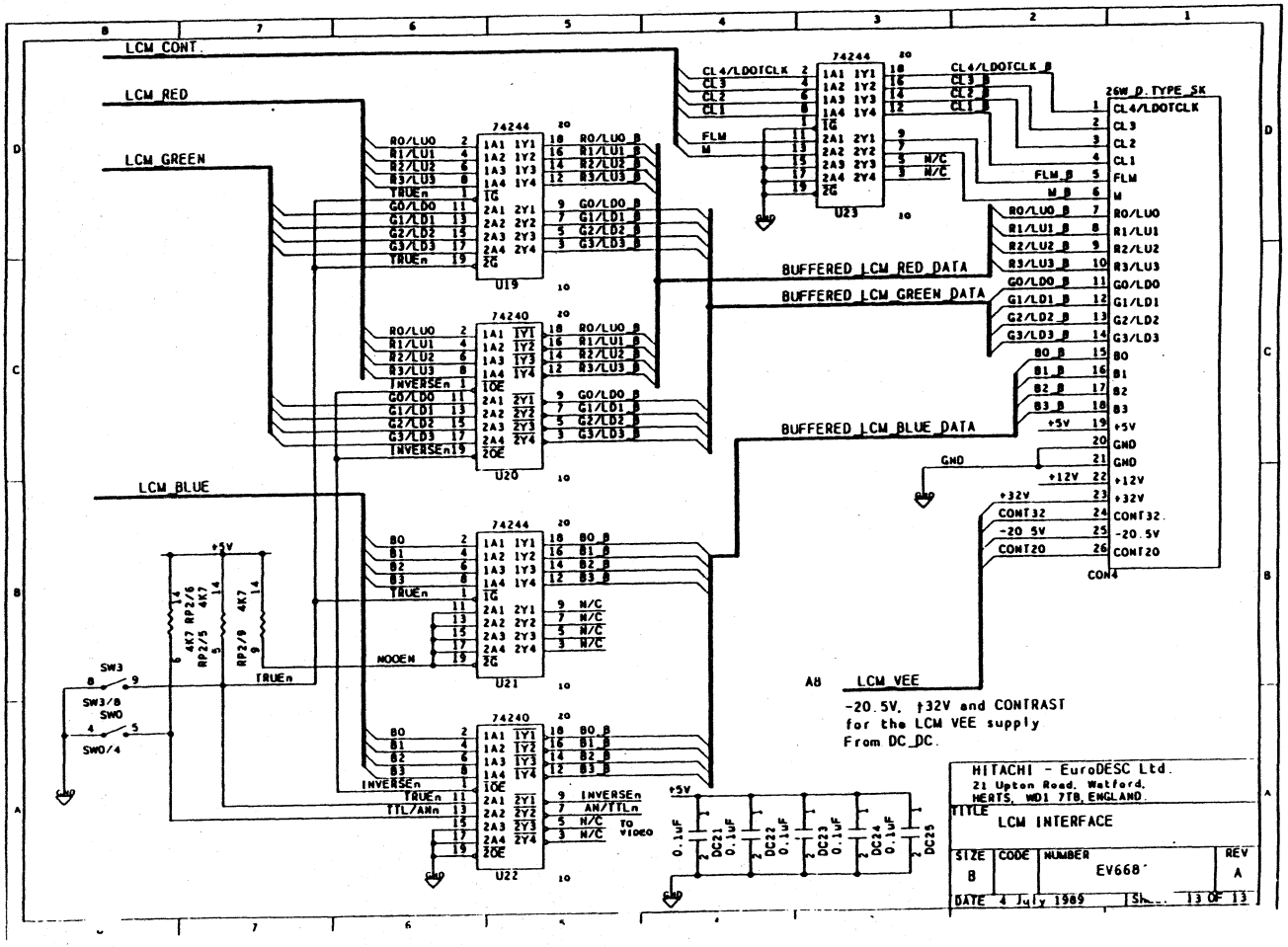
The Figure 24 on the Page 34 is revised to show the internal oscillator usage :

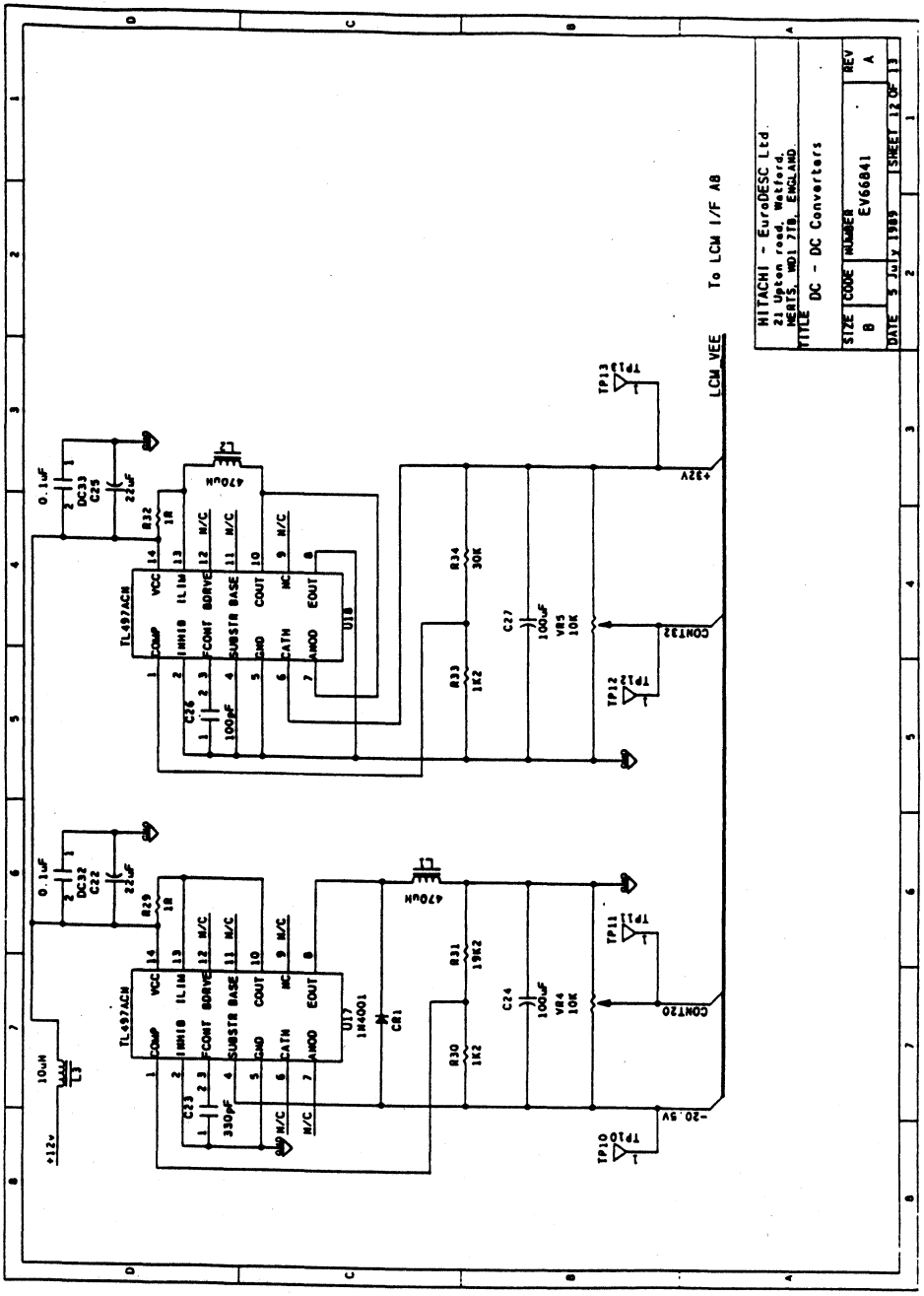


Also, the Figure 25 on the Page 35 is corrected to show the internal oscillator usage :



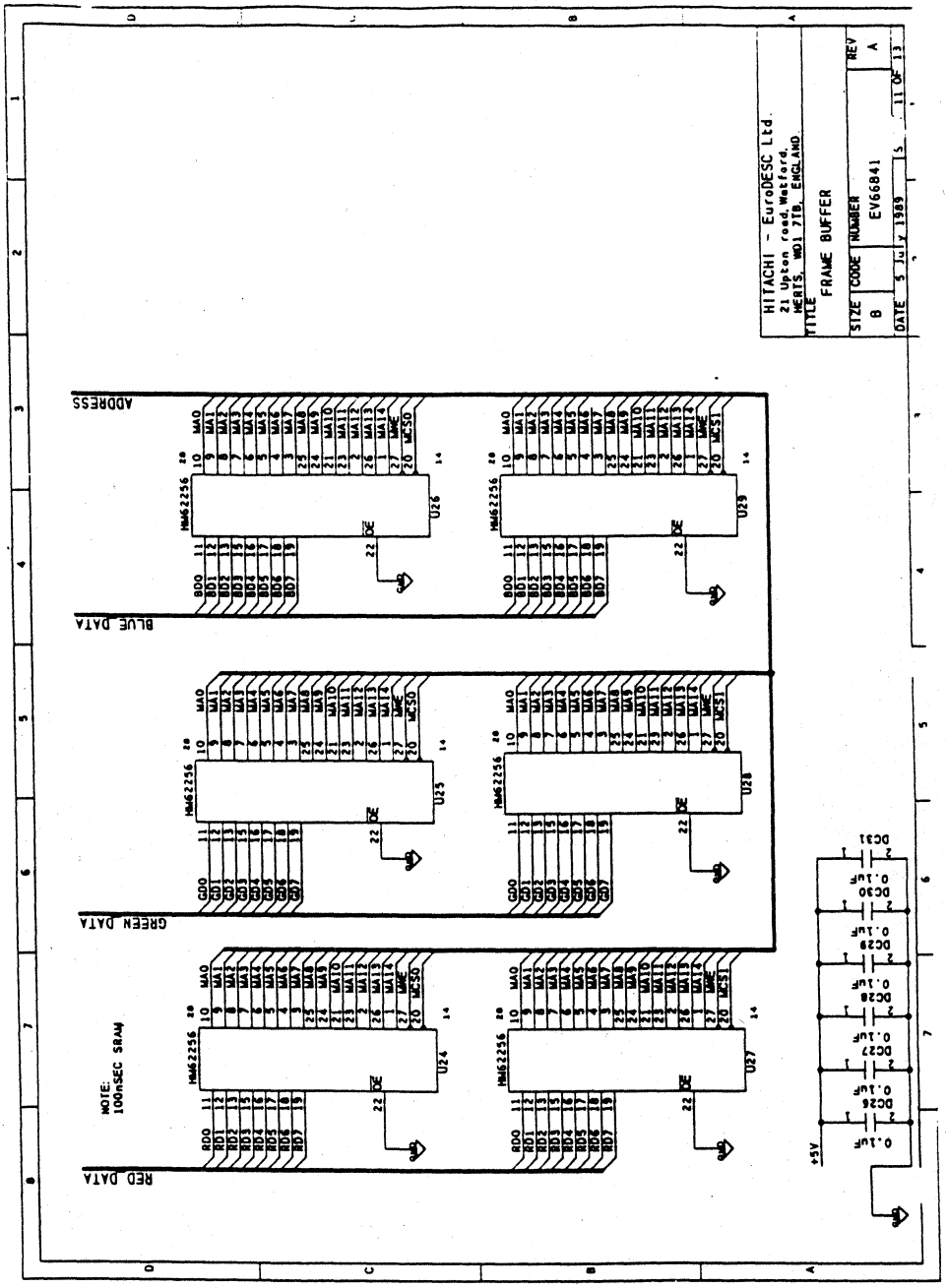
HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

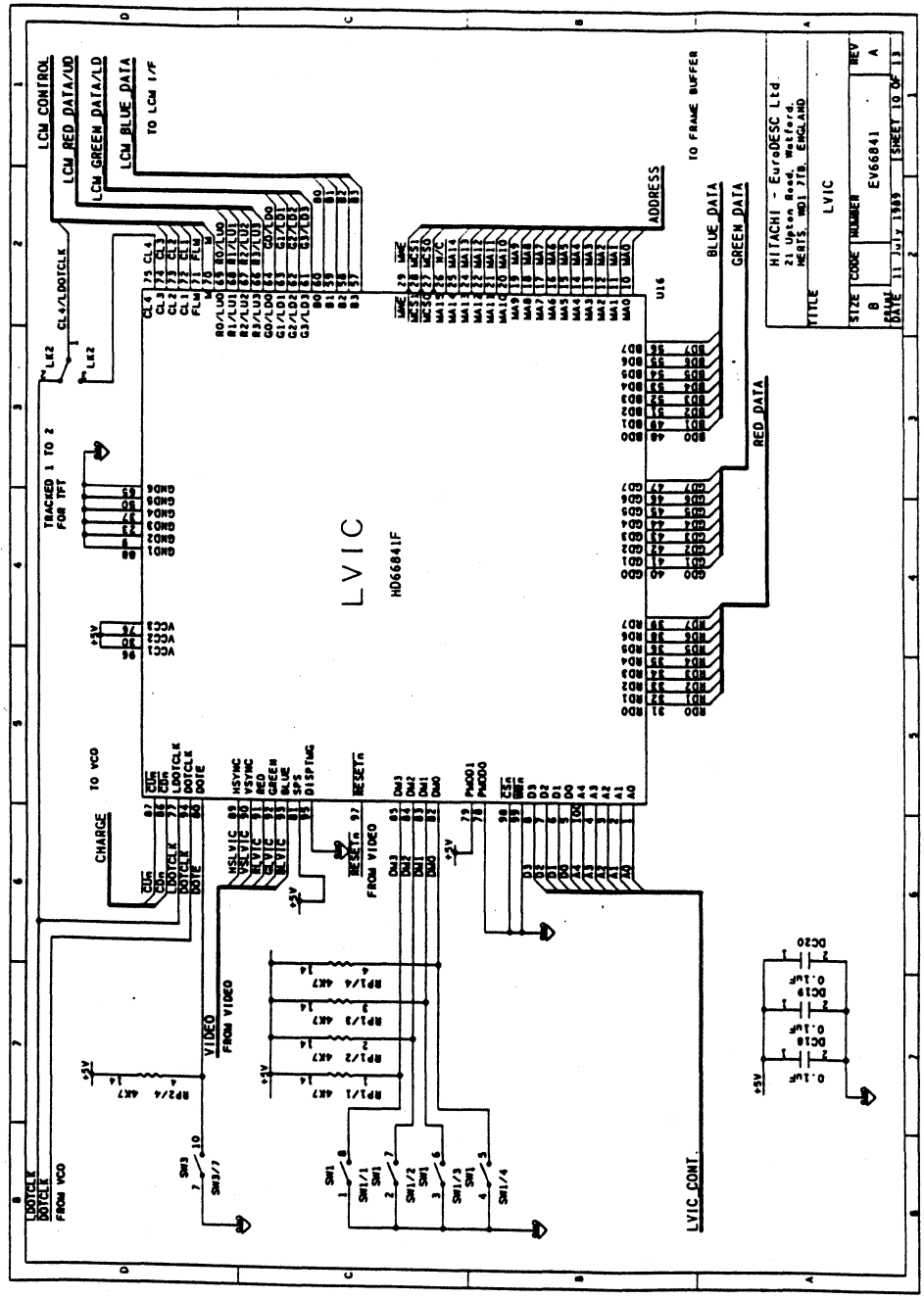

LCM_CONT
LCM_RED
LCM_GREEN
LCM_BLUE
74244
U19
74240
U20
U21
U22
U23
BUFFERED_LCM_RED_DATA
BUFFERED_LCM_GREEN_DATA
BUFFERED_LCM_BLUE_DATA
26W_D_TYPE_SK
CON4
LCM_VEE
-20.5V, +32V and CONTRAST
for the LCM VEE supply
From DC_DC.
SW3
SW3/8
SW0
SW0/4
4K7 RP2/6
RP2/5 4K7
RP2/9 4K7
TRUEn
NOOEN
INVERSEn
TTL/ANn
AN/TTLn
TO VIDEO
DC21 0.1uF
DC22 0.1uF
DC23 0.1uF
DC24 0.1uF
DC25
HITACHI - EuroDESC Ltd.
21 Upton Road, Watford,
HERTS, WD1 7TB, ENGLAND
TITLE
LCM INTERFACE
SIZE B
CODE
NUMBER EV668
REV A
DATE 4 July 1989
13 OF 13

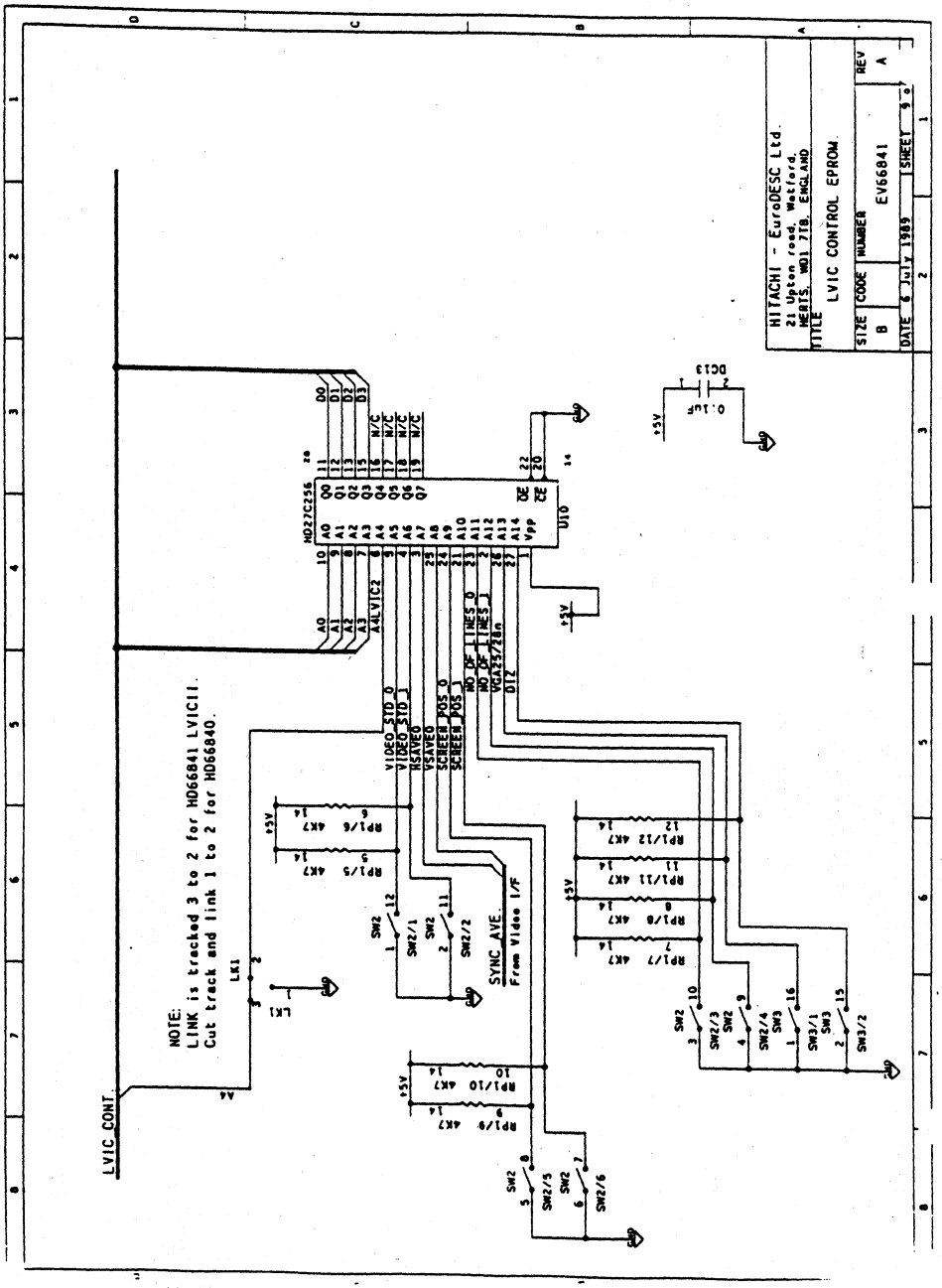


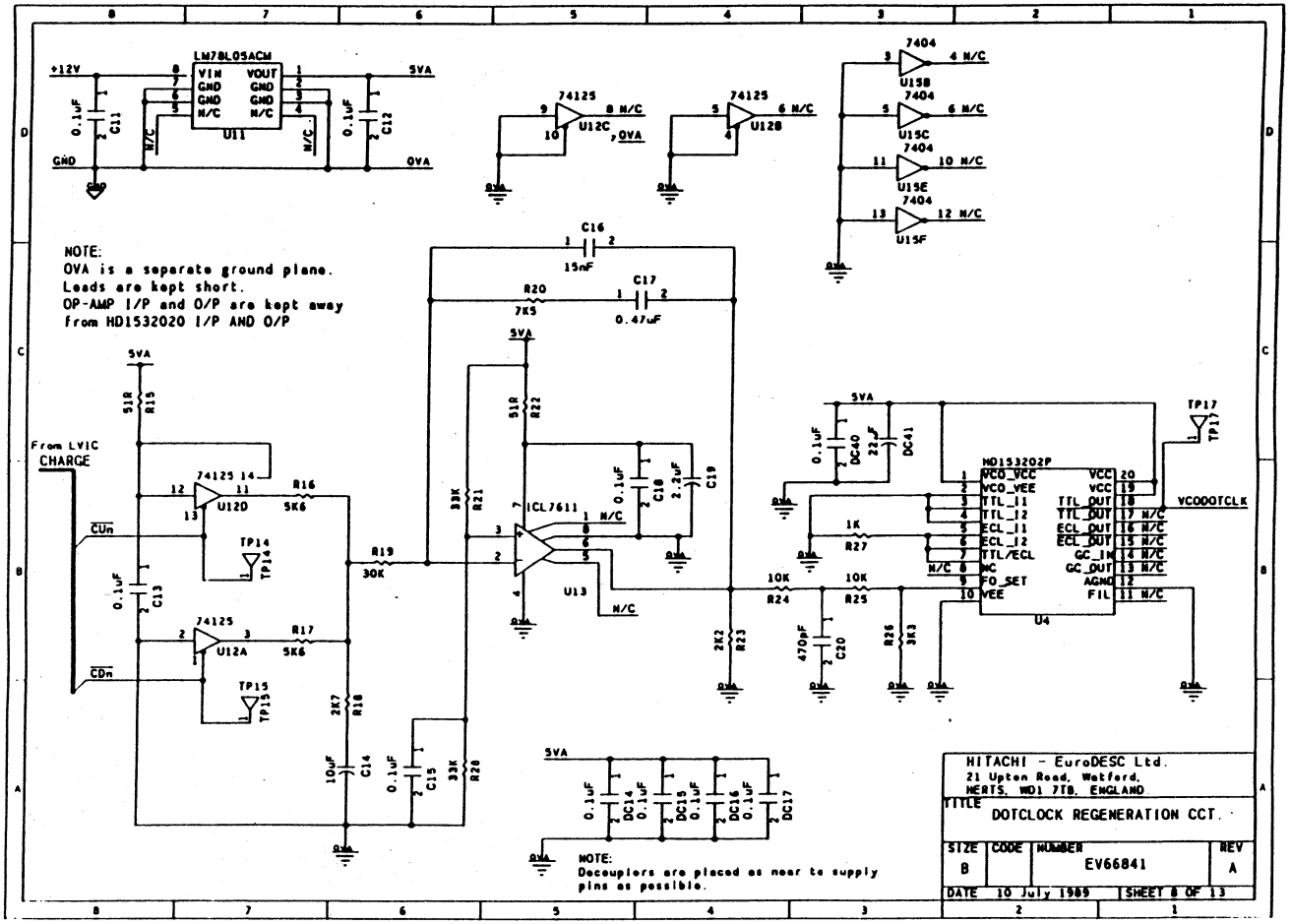
HITACHI

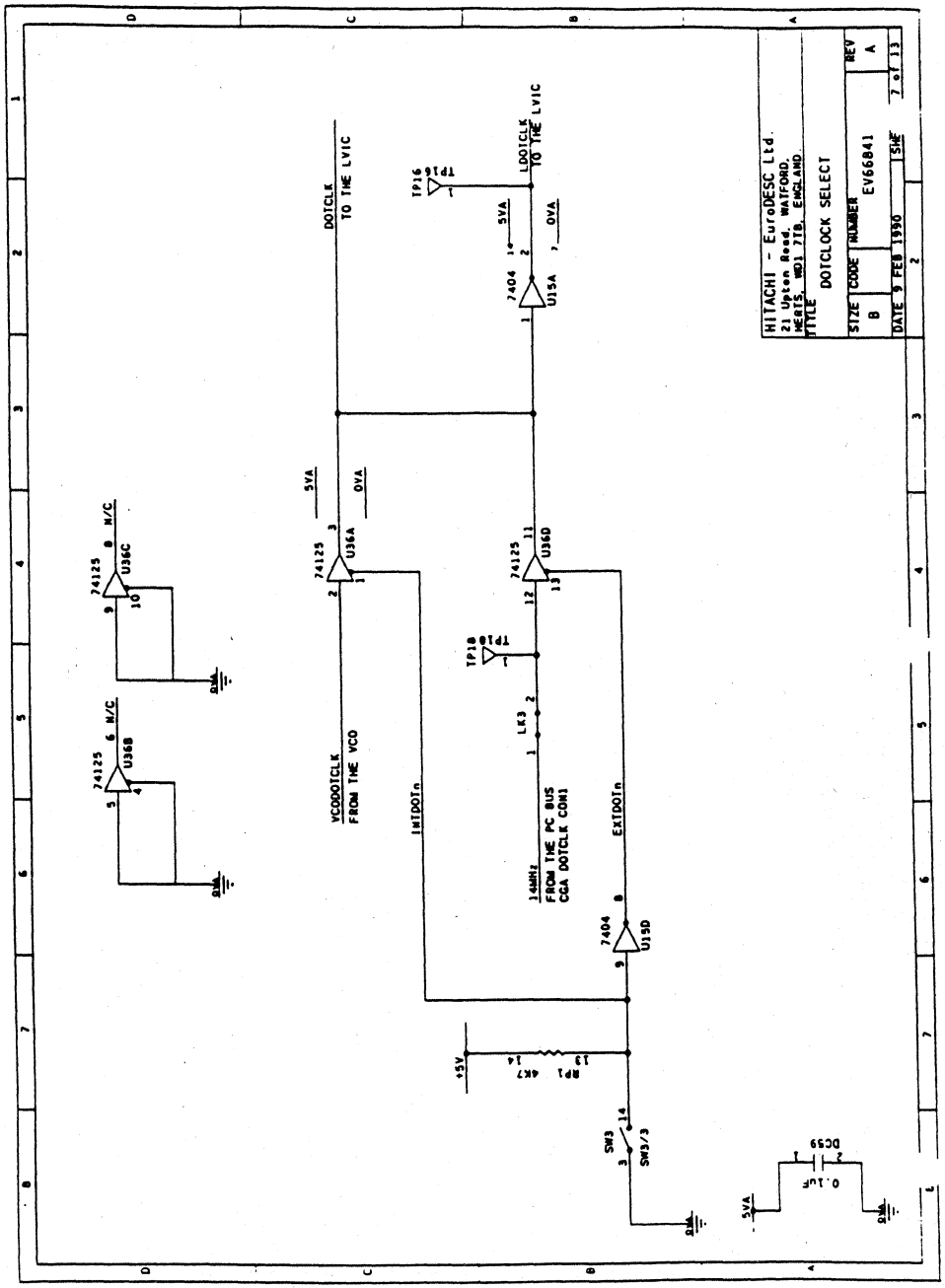
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300


HITACHI - EuroDESC Ltd.
21 Upton road,Watford,
HERTS, WD1 7TB, ENGLAND
TITLE
FRAME BUFFER
SIZE B
CODE NUMBER EV66841
REV A
DATE 5 July 1989
11 OF 13
ADDRESS
BLUE DATA
GREEN DATA
RED DATA
NOTE: 100nSEC SRAM
HM62256
U24
U25
U26
U27
U28
U29
DC26 0.1uF
DC27 0.1uF
DC28 0.1uF
DC29 0.1uF
DC30 0.1uF
DC31 0.1uF
+5V

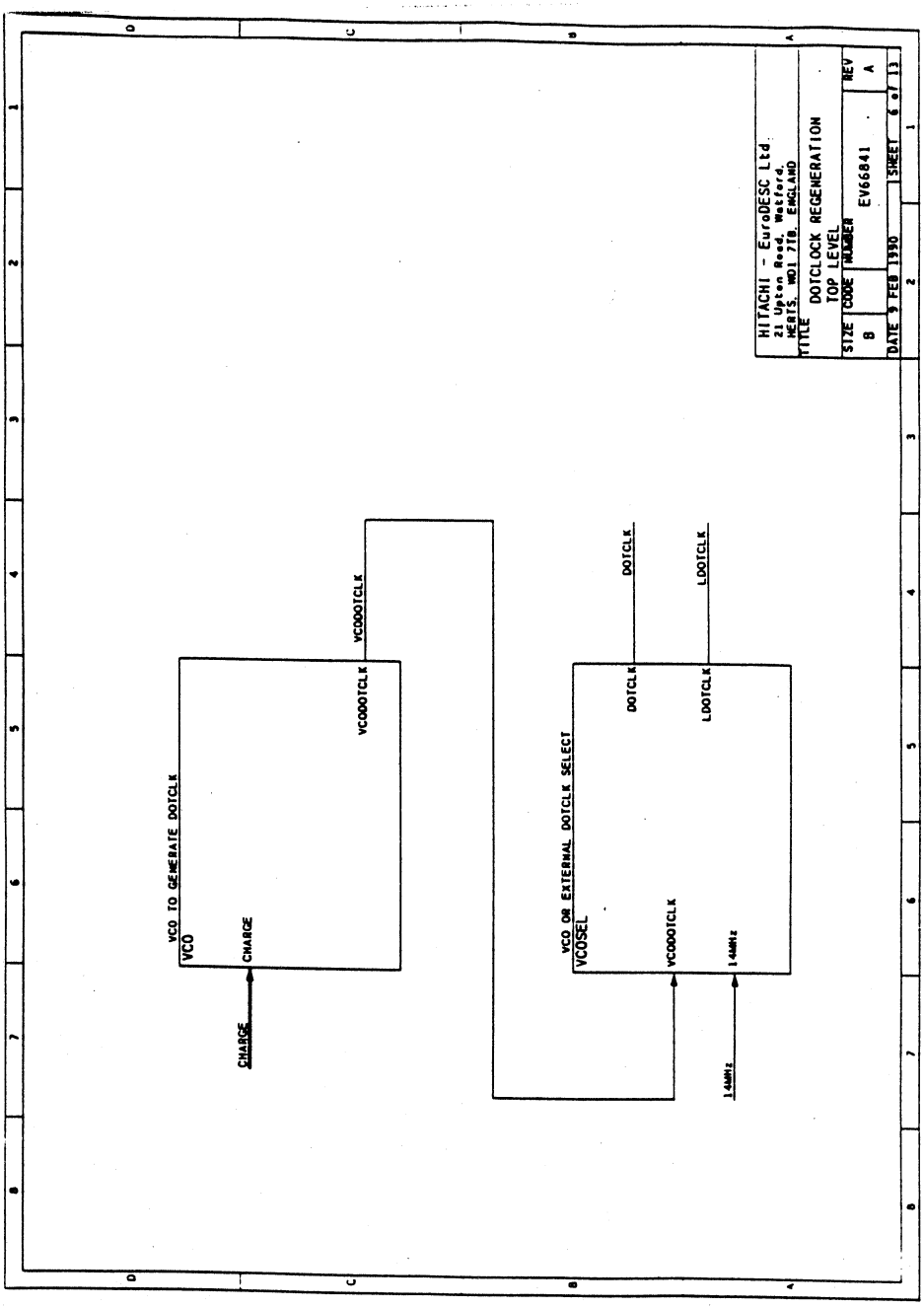

LCM CONTROL
LCM RED DATA/LD
LCM GREEN DATA/LD
LCM BLUE DATA
TO LCM I/F
CL4/LDOTCLK
TRACKED 1 TO 2 FOR TFT
LVIC
HD66841F
U16
ADDRESS
TO FRAME BUFFER
BLUE DATA
GREEN DATA
RED DATA
TO VCO
CHARGE
VIDEO
FROM VIDEO
LDOTCLK
DOTCLK
FROM VCO
LVIC CONT.
HITACHI - EuroDESC Ltd.
21 Upton Road, Watford,
HERTS. WD1 7TB. ENGLAND
TITLE
LVIC
SIZE
B
CODE NUMBER
EV66841
REV
A
DATE 11 July 1989
SHEET 10 OF 13


HITACHI - EuroDESC Ltd.
21 Upton road, Watford,
HERTS, WD1 7TB, ENGLAND
TITLE LVIC CONTROL EPROM
SIZE B
CODE
NUMBER EV66841
REV A
DATE 6 July 1989
SHEET 1 of 1
HD27C256
U10
DC13
0.1uF
+5V
LVIC CONT.
NOTE:
LINK is tracked 3 to 2 for HD66841 LVIC11.
Cut track and link 1 to 2 for HD66840.
LK1
VIDEO_STD_0
VIDEO_STD_1
VSAVEO
HSAVEO
SCREEN_POS_0
SCREEN_POS_1
NO_OF_LINES_0
NO_OF_LINES_1
VGA25/28m
DIZ
A4LVIC2
SYNC_AVE.
From Video I/F
RPL/5 4K7
RPL/6 4K7
RPL/7 4K7
RPL/8 4K7
RPL/9 4K7
RPL/10 4K7
RPL/11 4K7
RPL/12 4K7
SW2/1
SW2/2
SW2/3
SW2/4
SW2/5
SW2/6
SW3/1
SW3/2

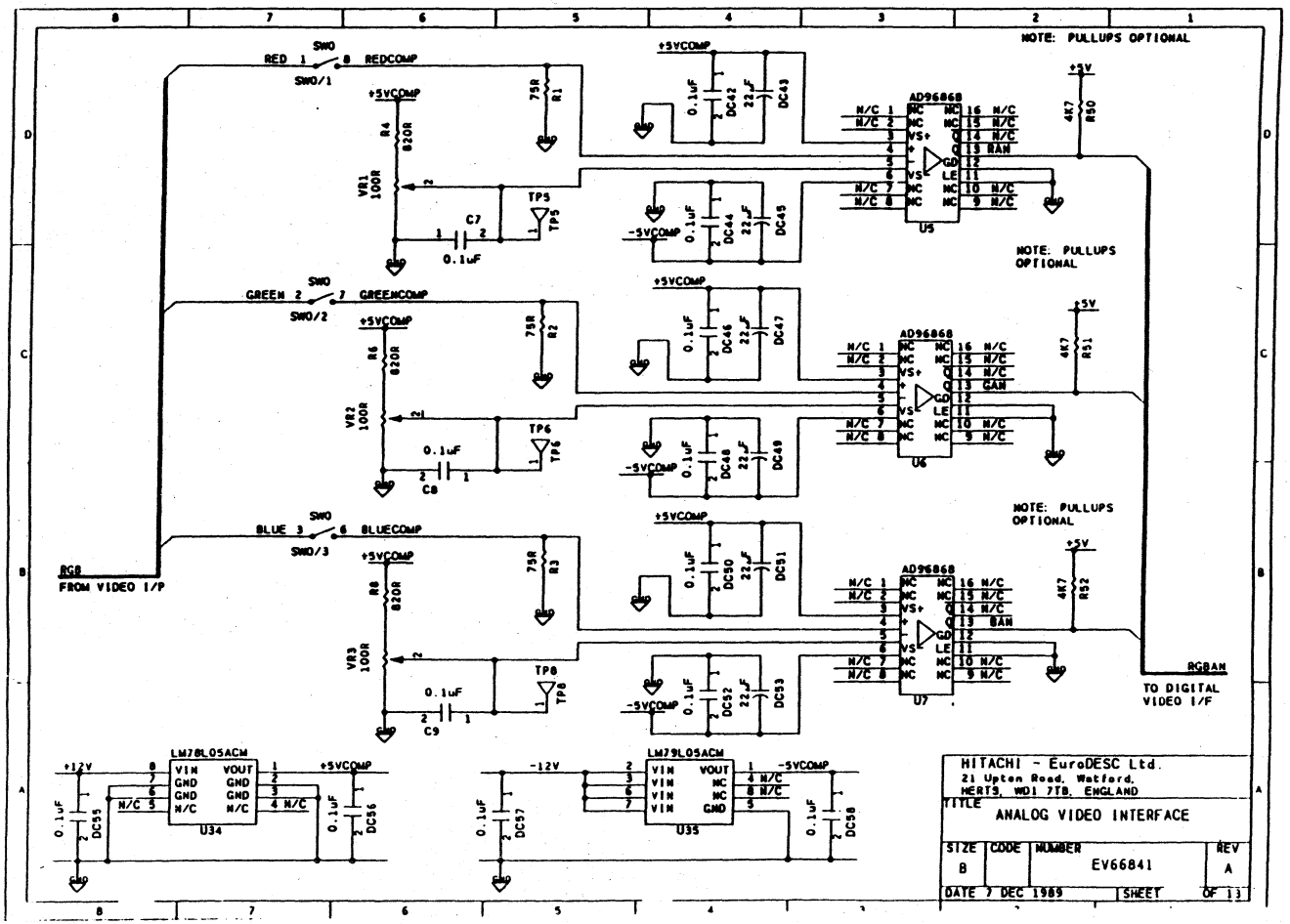

LM78L05ACM
VIN
GND
GND
N/C
VOUT
GND
GND
N/C
U11
+12V
GND
5VA
0VA
0.1uF
C11
C12
74125
U12C
U12B
U12D
U12A
7404
U15B
U15C
U15E
U15F
NOTE:
0VA is a separate ground plane.
Leads are kept short.
OP-AMP I/P and O/P are kept away
from HD1532020 I/P AND O/P
C16
15nF
R20
7K5
C17
0.47uF
From LVIC
CHARGE
CUn
CDn
R15
51R
R16
5K6
R17
5K6
R18
2K7
R19
30K
C13
C14
10uF
TP14
TP15
TP17
ICL7611
U13
R21
33K
R22
51R
C18
2.2uF
C19
R23
2K2
R24
10K
R25
10K
R27
1K
R26
3K3
C20
470pF
DC40
DC41
HD153202P
VCO_VCC
VCO_VEE
TTL_I1
TTL_I2
ECL_I1
ECL_I2
TTL/ECL
NC
FO_SET
VEE
VCC
VCC
TTL_OUT
TTL_OUT
ECL_OUT
ECL_OUT
GC_IN
GC_OUT
AGND
FIL
U4
VCODOTCLK
R28
33K
C15
DC14
DC15
DC16
DC17
NOTE:
Decouplers are placed as near to supply
pins as possible.
HITACHI - EuroDESC Ltd.
21 Upton Road, Watford,
HERTS, WD1 7TB, ENGLAND
TITLE
DOTCLOCK REGENERATION CCT.
SIZE
B
CODE
NUMBER
EV66841
REV
A
DATE 10 July 1989
SHEET 8 OF 13

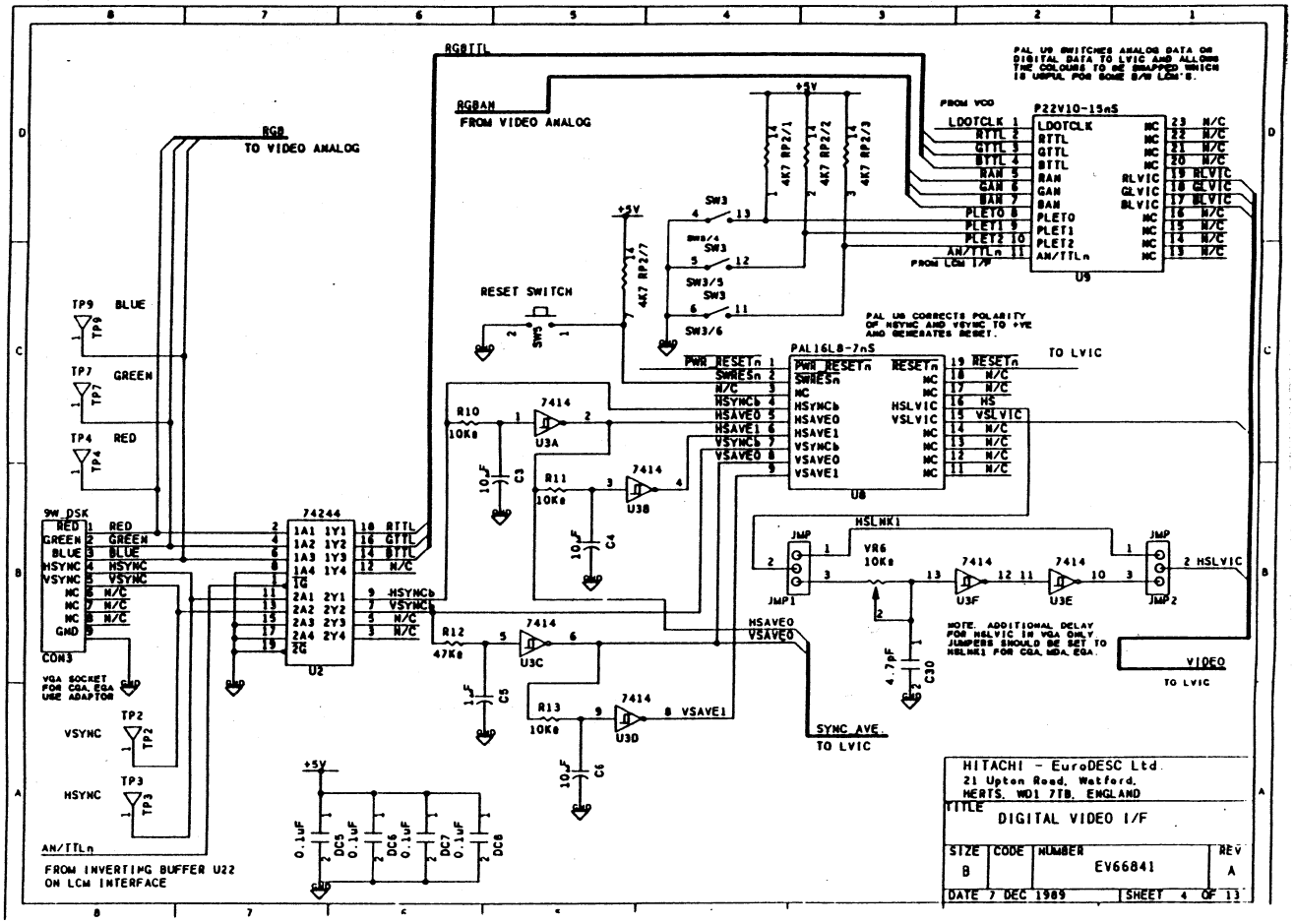

HITACHI - EuroDESC Ltd.
21 Upton Road, WATFORD,
HERTS, WD1 7TB, ENGLAND.
TITLE DOTCLOCK SELECT
SIZE B
CODE
NUMBER EV66841
REV A
DATE 9 FEB 1990
SHT 2
7 of 13
DOTCLK
TO THE LV1C
LDOTCLK
TO THE LV1C
TP16
TP18
5VA
0VA
7404
U15A
74125
U36A
U36B
U36C
U36D
N/C
VCODOTCLK
FROM THE VCO
INTDOTn
EXTDOTn
LK3
140MHz
FROM THE PC BUS
CGA DOTCLK CON1
U15D
+5V
RP1 4K7
SW3
SW3/3
DC39
0.1uF

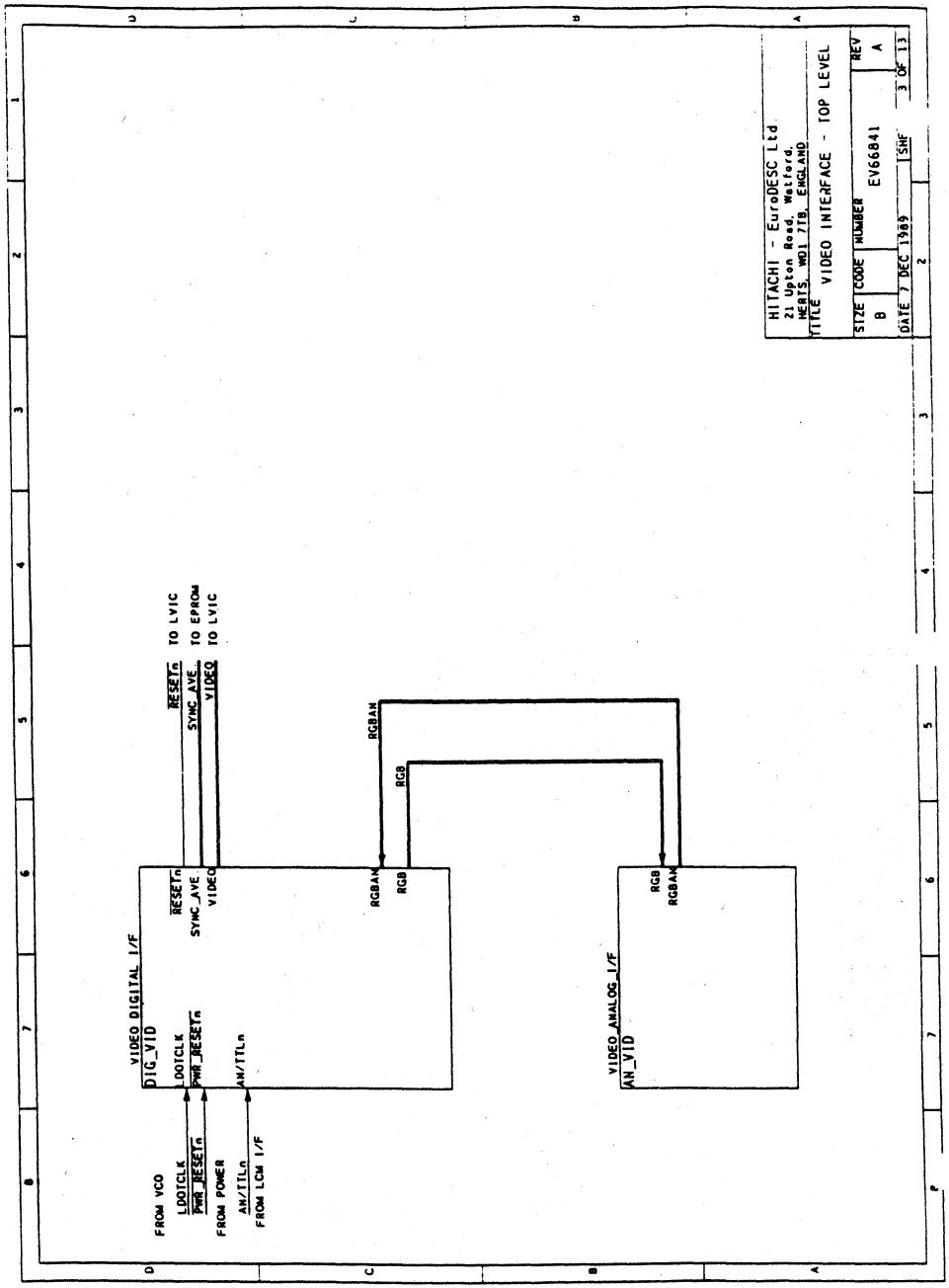
SECTION 4



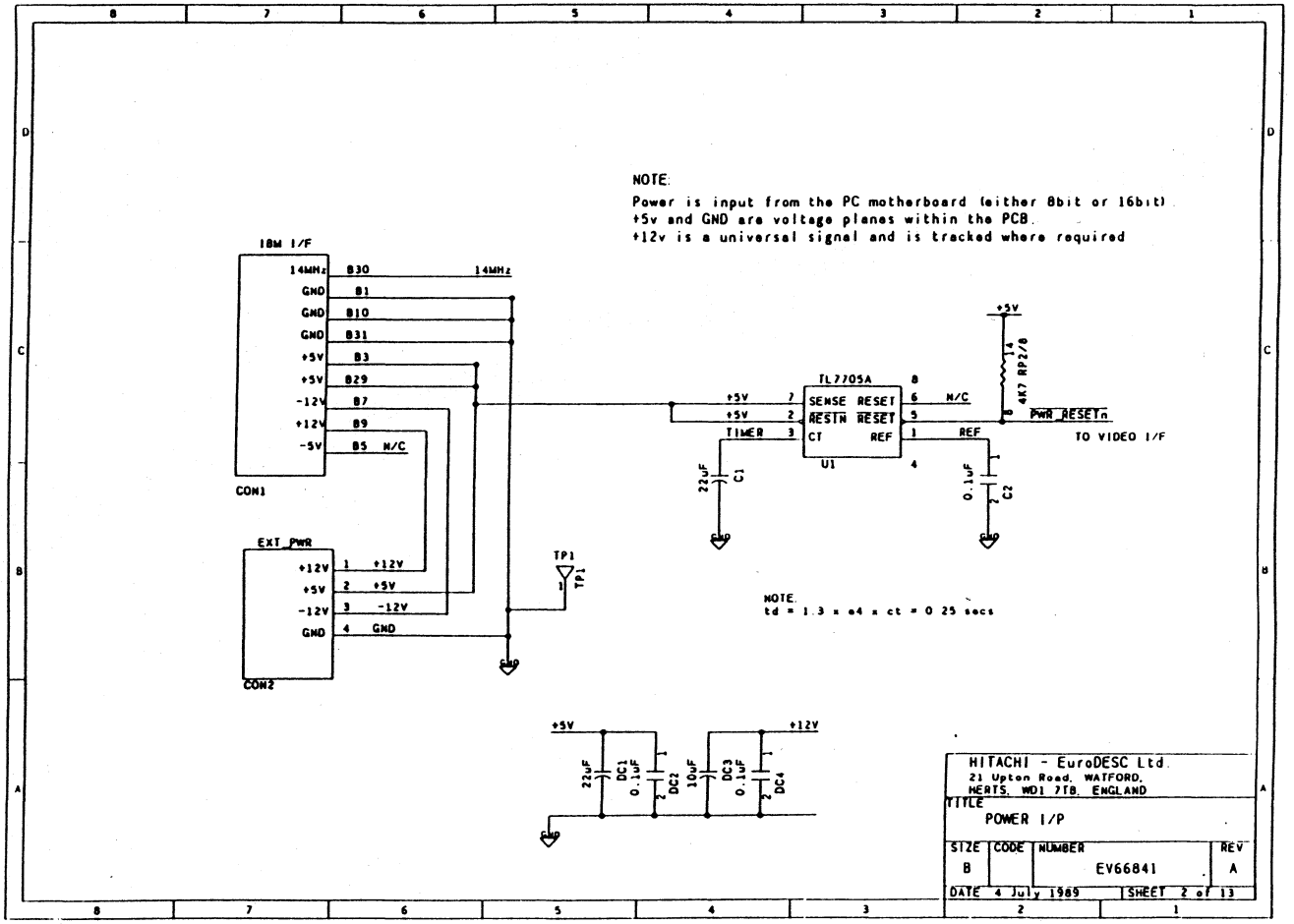
HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300


NOTE: PULLUPS OPTIONAL
SW0
RED 1
8 REDCOMP
SW0/1
+5VCOMP
R4 820R
VR1 100R
R1 75R
TP5
C7 0.1uF
GREEN 2
7 GREENCOMP
SW0/2
R6 820R
VR2 100R
R2 75R
TP6
C8 0.1uF
BLUE 3
6 BLUECOMP
SW0/3
R8 820R
VR3 100R
R3 75R
TP8
C9 0.1uF
RGB
FROM VIDEO I/P
AD96868
U5
U6
U7
+5V
R50 4K7
R51 4K7
R52 4K7
-5VCOMP
DC42 0.1uF
DC43 22uF
DC44 0.1uF
DC45 22uF
DC46 0.1uF
DC47 22uF
DC48 0.1uF
DC49 22uF
DC50 0.1uF
DC51 22uF
DC52 0.1uF
DC53 22uF
RGBAN
TO DIGITAL VIDEO I/F
LM78L05ACM
U34
+12V
DC55 0.1uF
DC56 0.1uF
LM79L05ACM
U35
-12V
DC57 0.1uF
DC58 0.1uF
HITACHI - EuroDESC Ltd.
21 Upton Road, Watford,
HERTS, WD1 7TB, ENGLAND
TITLE
ANALOG VIDEO INTERFACE
SIZE B
CODE
NUMBER EV66841
REV A
DATE 7 DEC 1989
SHEET
OF 13

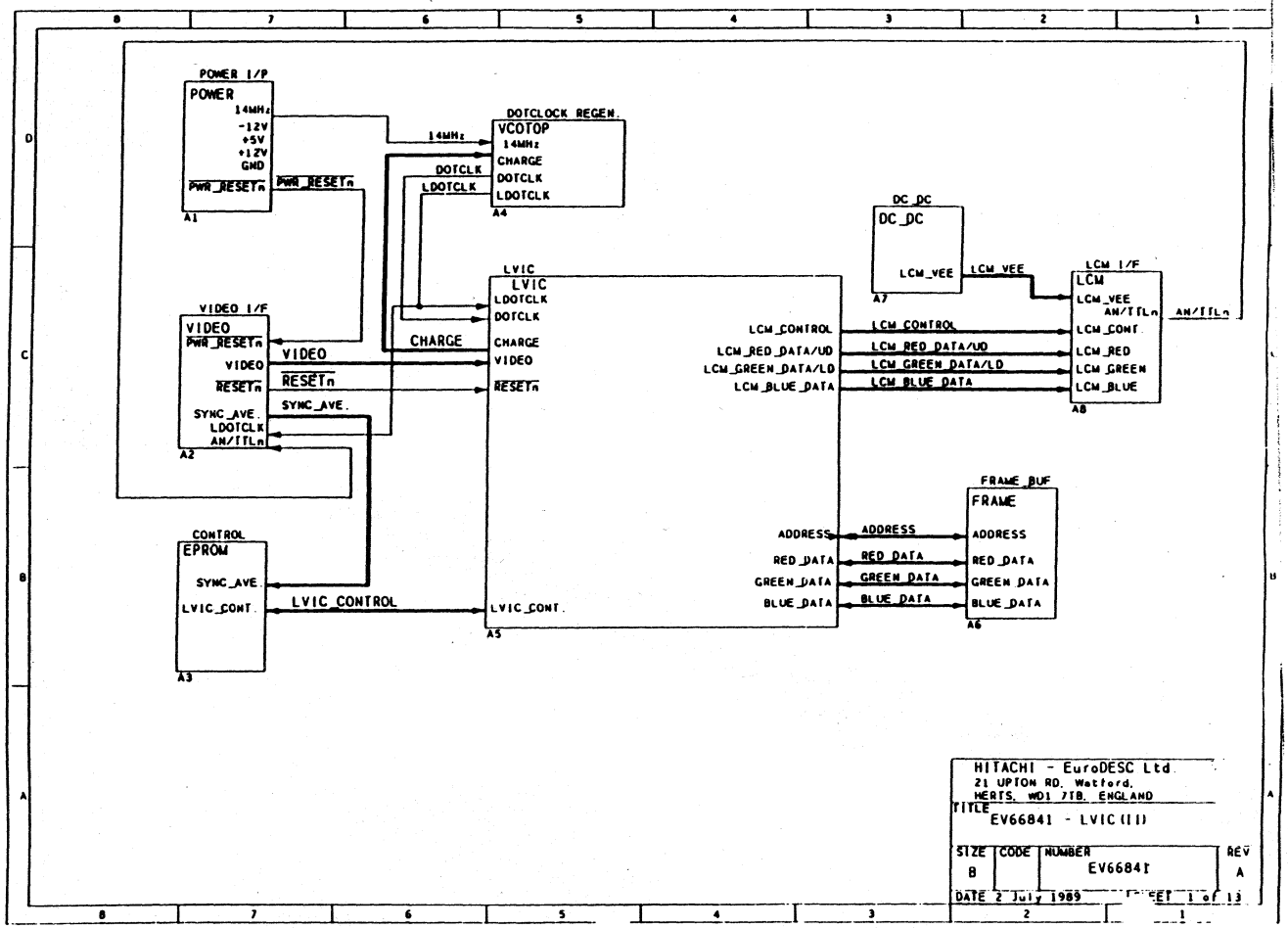

RGBTTL
RGBAN
FROM VIDEO ANALOG
RGB
TO VIDEO ANALOG
PAL U9 SWITCHES ANALOG DATA OR DIGITAL DATA TO LVIC AND ALLOWS THE COLOURS TO BE SWAPPED WHICH IS USFUL FOR SOME B/W LCM'S.
FROM VCO
P22V10-15nS
U9
FROM LCM I/F
RESET SWITCH
PAL U8 CORRECTS POLARITY OF HSYNC AND VSYNC TO +VE AND GENERATES RESET.
PAL16L8-7nS
U8
TO LVIC
TP9 BLUE
TP7 GREEN
TP4 RED
9W DSK
CON3
74244
U2
VGA SOCKET FOR CGA, EGA USE ADAPTOR
TP2 VSYNC
TP3 HSYNC
AN/TTLn
FROM INVERTING BUFFER U22 ON LCM INTERFACE
HSLNK1
JMP1
JMP2
2 HSLVIC
VR6 10K8
NOTE. ADDITIONAL DELAY FOR HSLVIC IN VGA ONLY. JUMPERS SHOULD BE SET TO HSLNK1 FOR CGA, MDA, EGA.
VIDEO
TO LVIC
HSAVEO
VSAVEO
SYNC AVE.
TO LVIC
HITACHI - EuroDESC Ltd.
21 Upton Road, Watford.
HERTS. WD1 7TB. ENGLAND
TITLE
DIGITAL VIDEO I/F
SIZE B
CODE
NUMBER EV66841
REV A
DATE 7 DEC 1989
SHEET 4 OF 13


HITACHI - EuroDESC Ltd
21 Upton Road, Watford,
HERTS, WD1 7TB, ENGLAND
TITLE VIDEO INTERFACE - TOP LEVEL
SIZE B
CODE
NUMBER EV66841
REV A
DATE 7 DEC 1989
SHT 3 OF 13
VIDEO DIGITAL I/F
DIG_VID
LDOTCLK
PWR_RESETn
AN/TTLn
RESETn
SYNC_AVE
VIDEO
RGBAN
RGB
VIDEO ANALOG I/F
AN_VID
FROM VCO
FROM POWER
FROM LCM I/F
TO LVIC
TO EPROM
TO LVIC

SECTION 4

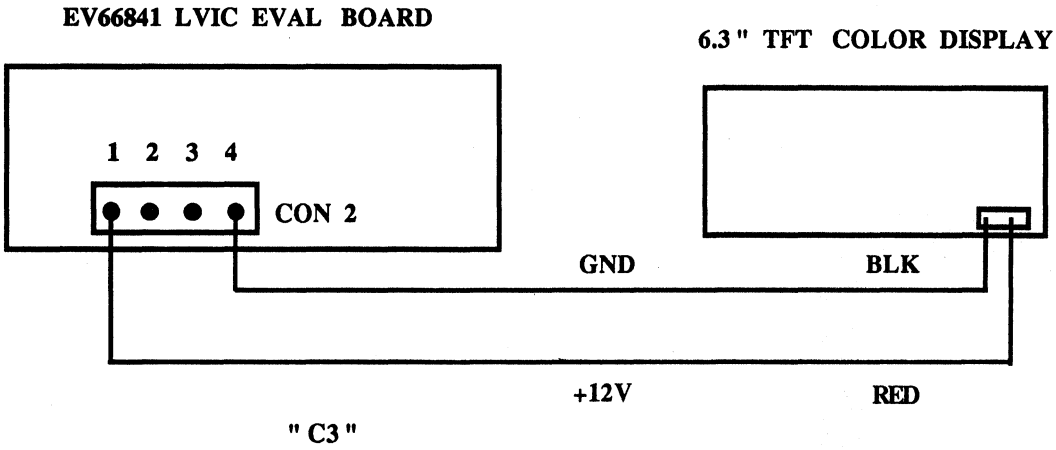

NOTE:
Power is input from the PC motherboard (either 8bit or 16bit).
+5v and GND are voltage planes within the PCB.
+12v is a universal signal and is tracked where required
IBM I/F
14MHz B30 14MHz
GND B1
GND B10
GND B31
+5V B3
+5V B29
-12V B7
+12V B9
-5V B5 N/C
CON1
EXT PWR
+12V 1 +12V
+5V 2 +5V
-12V 3 -12V
GND 4 GND
CON2
TP1
TL7705A
SENSE RESET
RESTN RESET
CT REF
U1
TIMER
N/C
REF
PWR_RESETn
TO VIDEO I/F
+5V
4K7 RP2/8
22uF C1
0.1uF C2
NOTE:
td = 1.3 x e4 x ct = 0.25 secs
22uF DC1
0.1uF DC2
10uF DC3
0.1uF DC4
HITACHI - EuroDESC Ltd.
21 Upton Road, WATFORD,
HERTS, WD1 7TB, ENGLAND
TITLE
POWER I/P
SIZE B
CODE
NUMBER EV66841
REV A
DATE 4 July 1989
SHEET 2 of 13

This section shows a copy of the schematic supplied by "Eurodesc", Hitachi Europe Ltd., United Kingdom. It is merely included for reference.



APPENDIX " B "

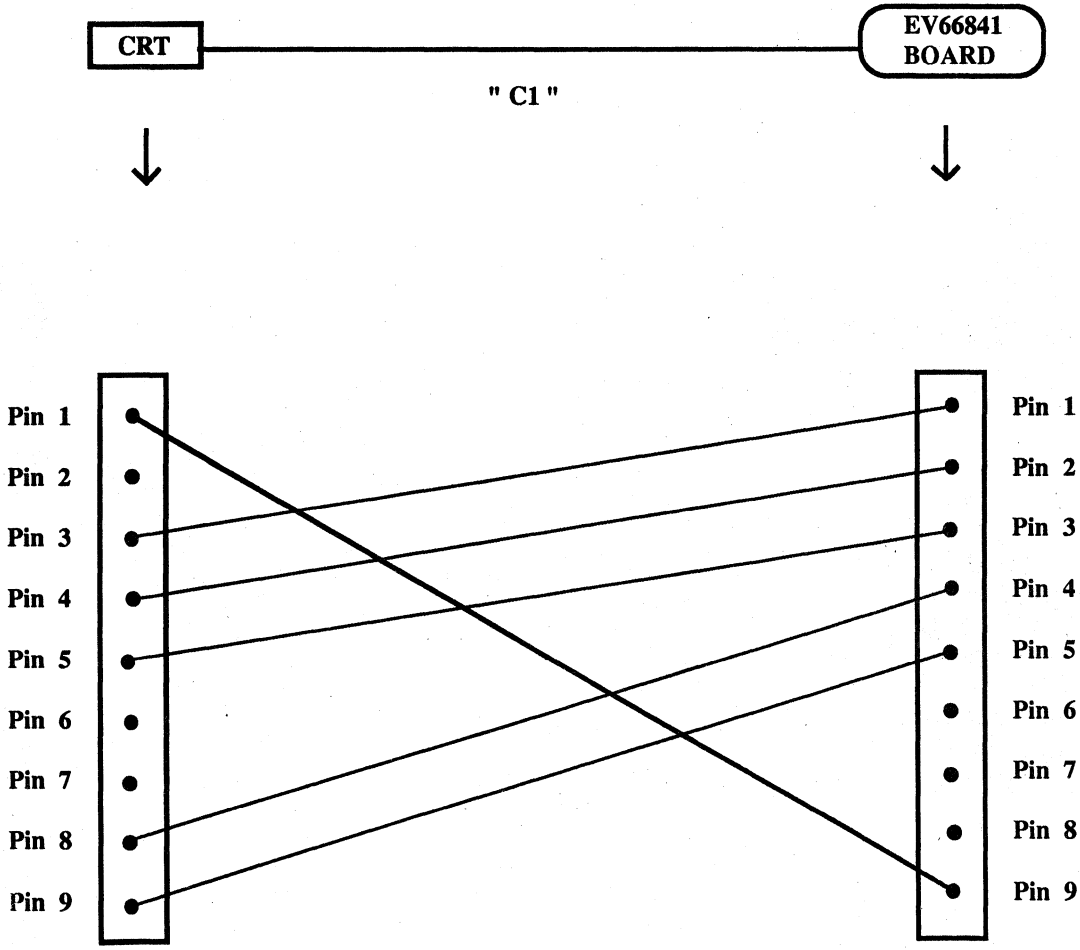
Back light power cable " C3 " is shown in this Appendix :



NOTE : Back light power is to be externally supplied.

APPENDIX " A "

The 9 pin video cable " C1 " translation is shown below :



HITACHI COLOR LCD TFT MODULE (TM16D01HC) :

Refer to the display data sheet for detail. The page 14 of it shows how the sub-pixels are designated for LVIC HD66841 interface with 160 dots (H) and 200 dots (V) resolution. The cable " C2 " provides the signals to the display while cable " C3 " provides the back light power. The dispaly tilt and swivel angles provide different contrast ratios in the ambient light, so it should be adjusted for the most desirable viewing angle.

SYSTEM DEBUG

First power up the system in CGA mode using the AMDEK color monitor and the EGA board effectively disconnecting the 6.3" TFT LCD display and reconnecting the cable " C1 " to the monitor. After the system is up in CGA mode, varify that it works correctly. Then, disconnect the cable " C1 " and connect it to the EV66841 board input. Also, varify that the cable " C2 " is properly connected to the display, since there is no key in the connector. If cable " C3 " is properly connected, back florescent light should come on and it is clearly visible.

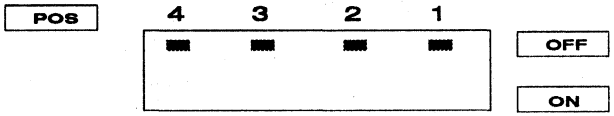
If every thing is working correctly, one can execute all the DOS commands when appropriate prompts are displayed on the LCD screen.

NOTE :
If the LCD screen is split and shows unreadable data, then disconnect and reconnect the cable " C2 " to the color TFT display when the power is on. This dynamic reset should cause the correct data to be displayed.

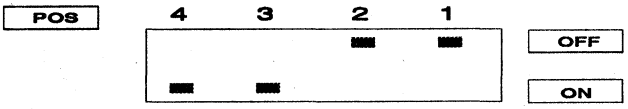
DEMONSTRATION SOFTWARE

After DOS 3.2 or later is installed, load any CGA graphics package such as PRINTSHOP or file management package XTREE for the video color display. With EV66841 evaluation board in the system, the color video display will be replaced by the color TFT LCD display. Both, XTREE and PRINTSHOP were used for the system display demonstration. By running Kaleidoscope 1, differrent colored dot patterns can be shown on the LCD screen. When Kaleidoscope 2 is run, various colored geometric patterns are displayed on the screen. For scanned color image files (*.GIF) for display, call Hitachi America Ltd., office at Sierra Point , CA.

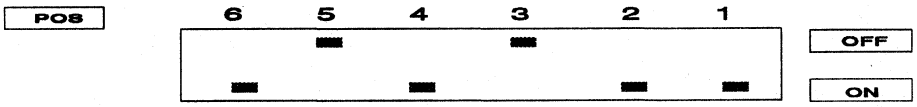
1.0 SWITCH " 0 " : It is set for digital input.



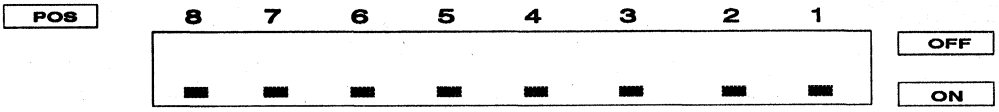
2.0 SWITCH " 1 " : This is set for LVIC mode " 13H ".



3.0 SWITCH " 2 " : Set for CGA mode and 200 vertical lines.



4.0 SWITCH " 3 " : Dynamic fuctions saettings - 25 MHz and regenerated dot clock.



NOTE : 1.0 Make sure that cable " C1 " is corectly connected at the display side. There is **no** cable key.

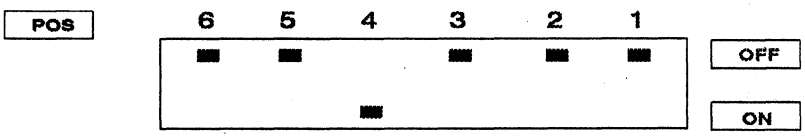
2.0 The attached schematic is only for reference so do **not** copy it for your design.

SYSTEM COMPONENTS

The hardware components are described in this section while the " C1 ", and " C2 " cable wiring diagrams are shown in the Appendices.

PC-AT : AST Premium 286 model 70 was operating at 10 MHZ, with 512KB memory, 20MB hard disk drive, and 1.2 MB, 5.25 " floppy drive. It was also running DOS version 3.2.

VIDEO CONTROLLER : Paradise Autoswitch EGA is card used in the CGA mode at (640H x 200V) resolution providing TTL level signals to the CRT monitor. The switch settings for 80 column, RGB monitor in CGA mode are listed below :



NOTE : 1.0 For more details refer to the Paradise CRT controller manual.
2.0 Make sure this switch is correctly set.

EV66841 LVIC EVALUATION BOARD

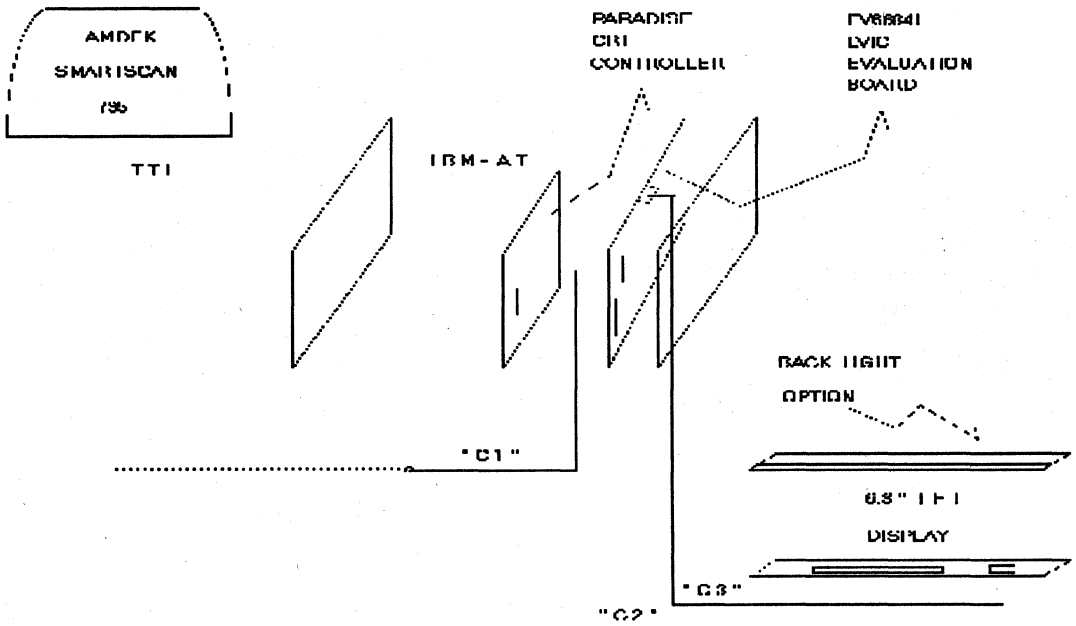
This board has numerous switches and its settings are complex. So, please refer to the EV66841 User's Guide for details. Only the 6.3" TFT LCD switch settings are addressed in this section.

The EV66841 board accepts TTL level input signals carried by the 9 pin cable " C1 " from the video controller board. The R,G,B, HSYNC, and VSYNC signals are used to regenerate the CRT dot clock, and sample the incoming video data. The output signals are sent over the cable " C2 " to the 6.3" TFT, 8 colors, Hitachi display. This board also provides +12 Volts required by the back light through the cable " C3 ". The switch settings of this board are shown on the following page:

SYSTEM CONFIGURATION

The development system was configured with IBM PC-AT or compatible machine, Paradise Autoswitch EGA 480 card, EV66841 LVIC Evaluation Board, Smartscan Amdek 735 digital color monitor, and Hitachi TFT active matrix, 8 color, 6.3", LCD display TM16D01HC from the ELT division. A custom cable is required to provide TTL level input video signals to the EV66841 board and is **not** provided. The LVIC Evaluation board output connector to the 6.3" TFT display **is** provided to make the display connection task easier. A separate +12V DC cable is also required for the back light option (#BLS-006M). The back light is easily mounted with the four corner screws of the 6.3" TFT display.

The system diagram is shown below :



NOTE : 1.0 " C1 " = " C3 " = Cables **not** provided.
2.0 " C2 " Cable provided.

EV66841

March, 1991

Technical Brief

LVIC Evaluation Board

Kash Yajnik

The EV66841 LVIC Evaluation Board was designed by Eurodesc, Hitachi Europe Ltd., and can be ordered through Hitachi America Ltd. in U.S.A. The board is shipped with cables for **multiple** Liquid Crystal Modules from Hitachi.

Black and white as well as color information can be displayed depending upon the selected LCD panel from Hitachi's ELT Division. The EV66841 LVIC Evaluation Board can reside inside IBM PC-AT or a compatible system running later than DOS version 2.0. It is also possible to run the EV66841 Board with external power supply. A User's Guide is also provided to customize the board for many applications.

This technical brief is written to complement the EV66841 User's Guide for **one** specific application using the 6.3" color TFT module (TM16D01HC) from Hitachi's Electron Tube Division (ELT). A copy of the schematic is also included to provide the design implementation detail. A system diagram is also included to high light the laboratory environment. Similarly, each user may tailor display subsystem requirements for the desired application.

The scope of this document is to help make the customization task easier and quicker. The circuit minimization tasks are left to each user and are **not** attempted. This is intended as an illustrative example for the Hitachi field and technical staff, and their customers.

The following pages cover system configuration and componenents, EV66841 Board set up, System debug, and Demonstration software. The Appendix "A" covers 9 pin Video cable translation and the Appendix "B" shows the Back light power connections. The Appendix "C" lists the schematic.

Refer to the subsequent pages for more detail.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

LVIC Proto Type Board

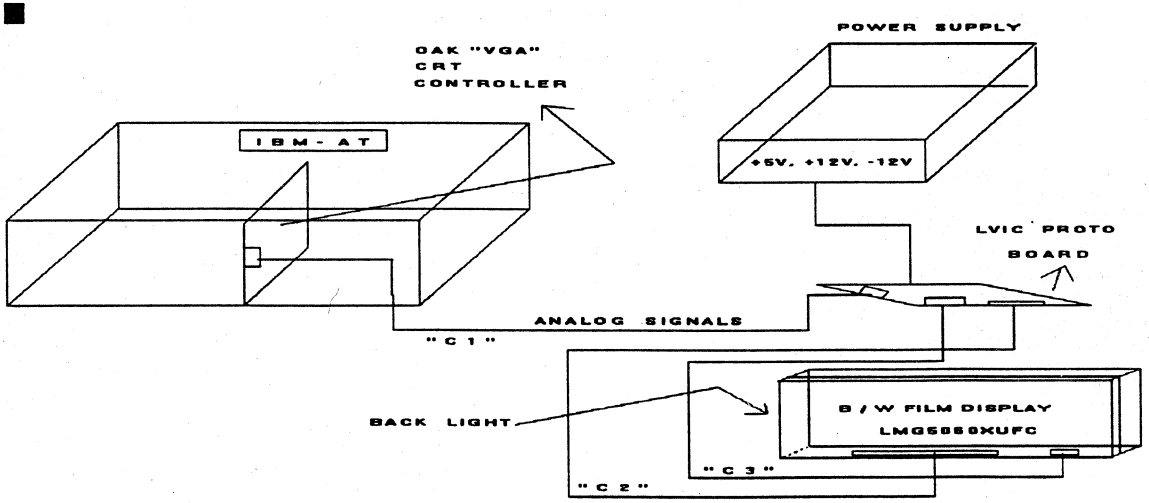
This document presents information for a 6.3" color active matrix or black and white LCD subsystem implementation using Hitachi controller HD66841. Its major components include IBM PC-AT, LVIC Proto Type Board, Paradise or Oak Video Controller Board, and the LCD display (TM16D01HC or LMG5060XUFC) from Hitachi's ELT Division. It can be further enhanced by adding demonstration software that runs on the IBM PC-AT or a compatible machine.

■ FEATURES

--Hardware--
(1) IBM PC-AT or compatible machine
(2) HD66841 LVIC Proto Type Board from Hitachi
(3) Color LCD Active Matrix or Black and White display from Hitachi with Back Light
(4) Paradise or Oak Video Controller Board

--Software--
(1) DOS 3.2 Version or later
(2) "XTREE", WINDOWS, or PRINTSHOP package
(3) Any CGA color or VGA demonstration package
(4) *.GIF files for color LCD or VGA type panel display

■

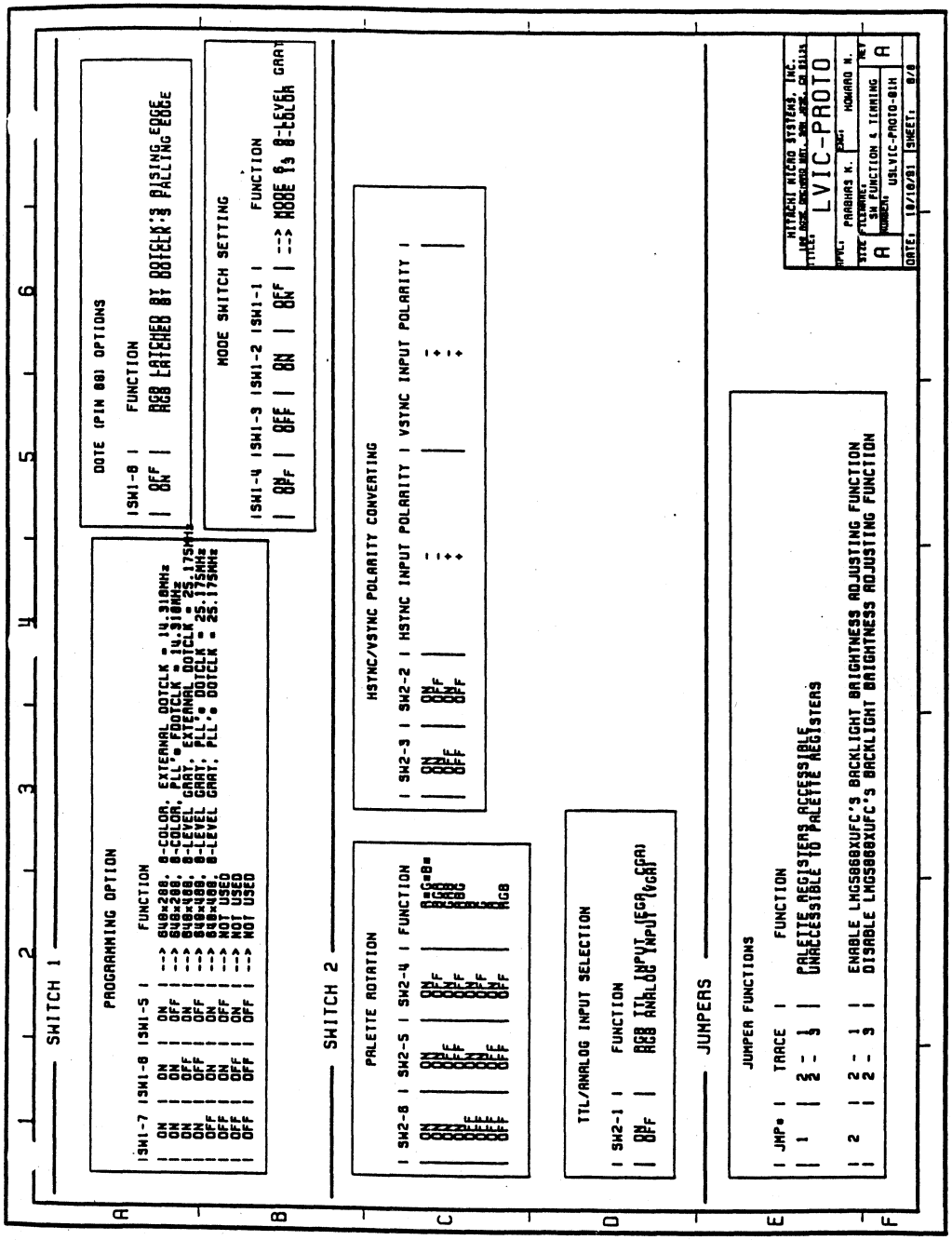


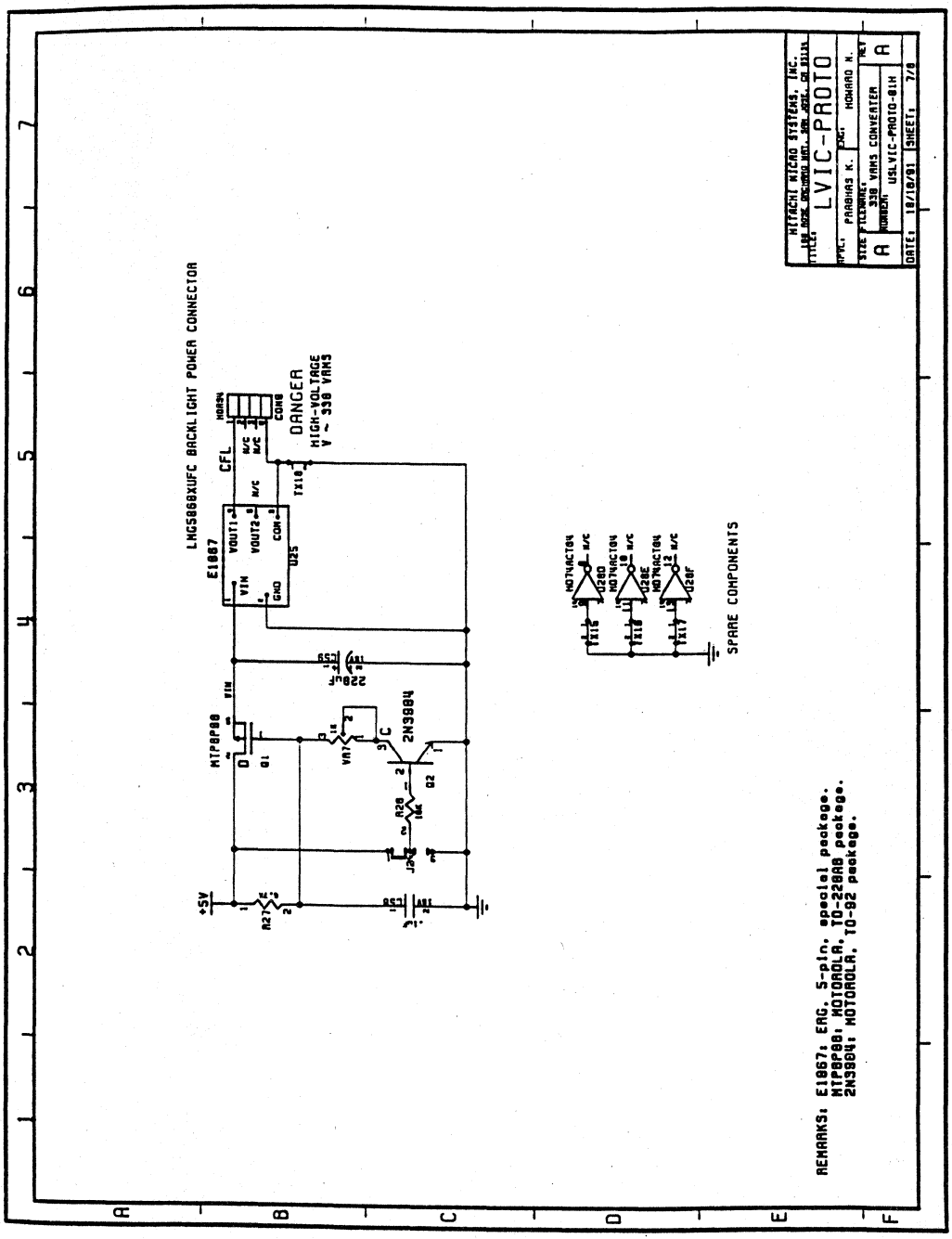
■ OBJECTIVES

(1) To display HD66841 LVIC Proto Type Board
(2) To demonstrate 8 colors or 8 shades of grey on LCD panel
(3) To show application software running on the HD66841 LVIC Proto Type Board
(4) To high light PC-AT Bus Interface

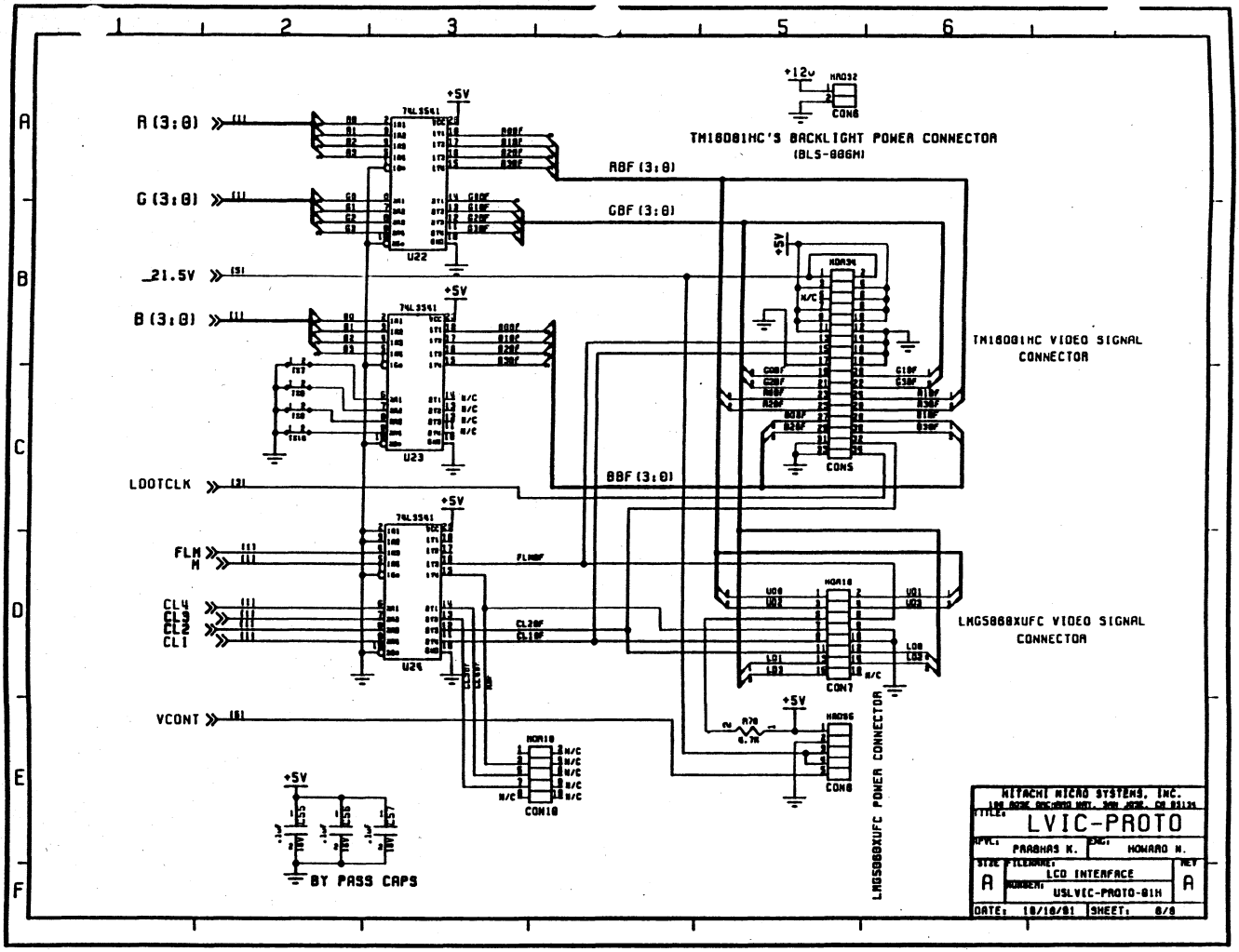
■ ADDITIONAL INFORMATION

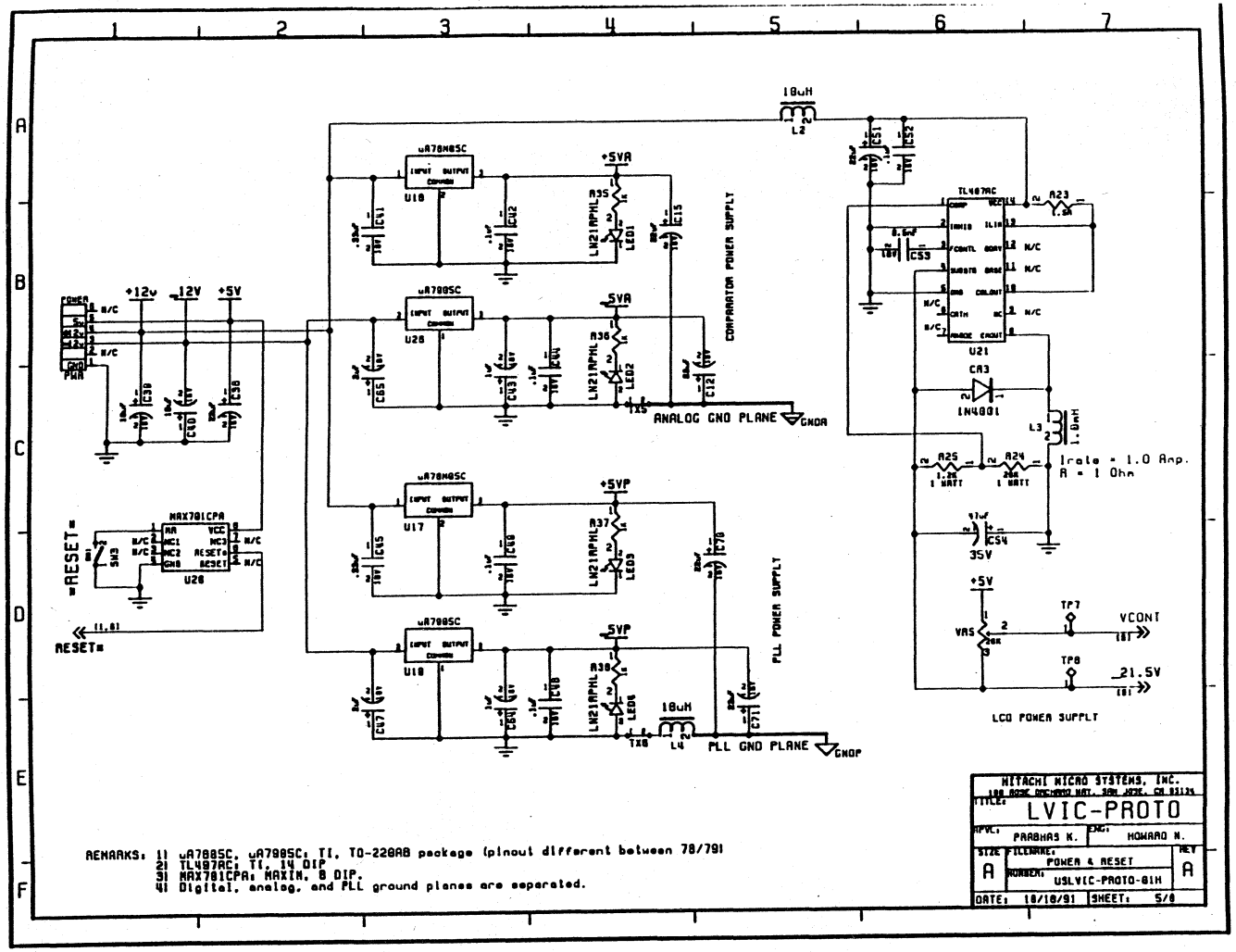
The details of the system configuration and its design along with the associated software, are available in the Hitachi America Ltd. Technical Brief #TB0103.

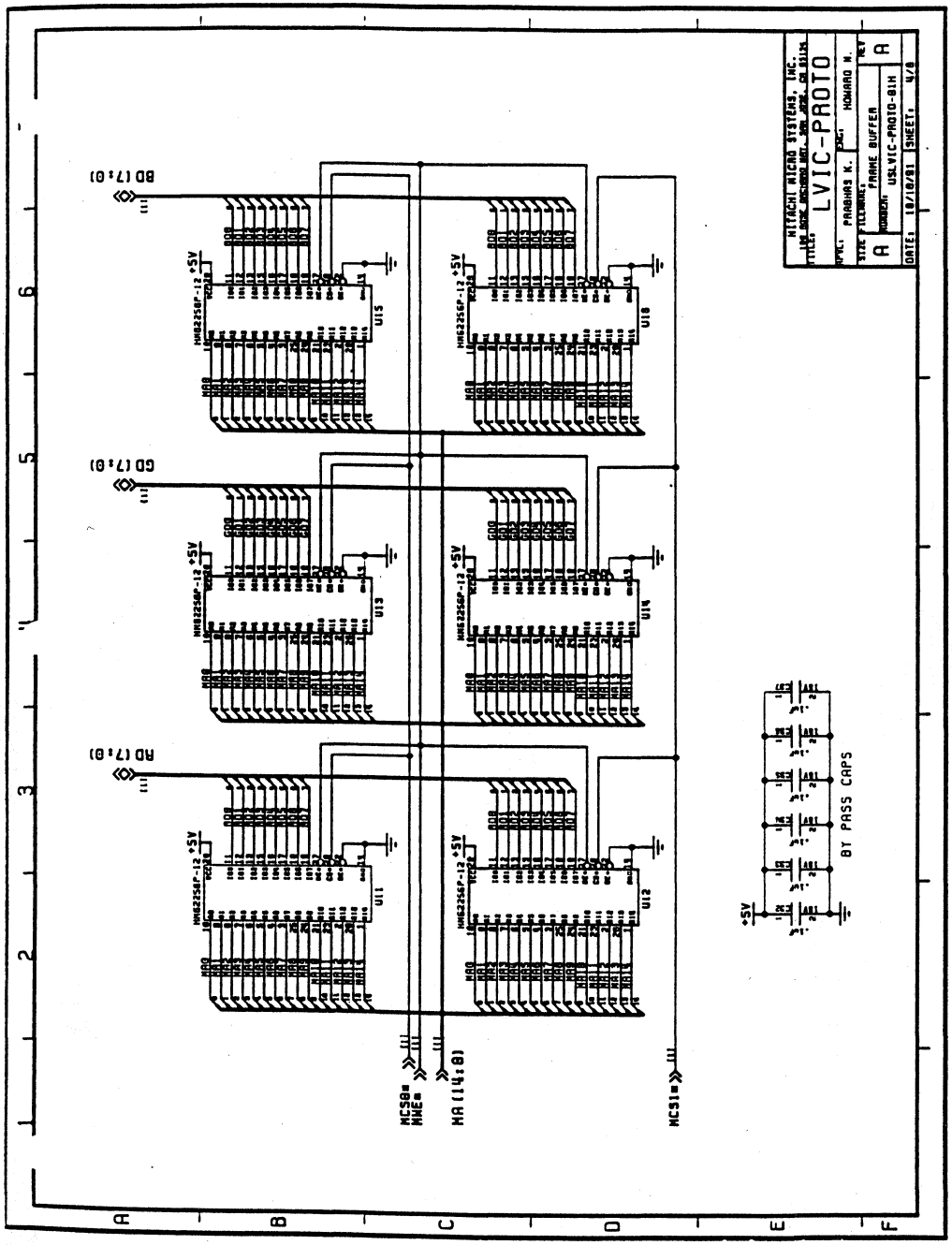

SWITCH 1
PROGRAMMING OPTION
ISW1-7 ISW1-8 ISW1-5 | FUNCTION
ON | ON | ON | --> 640x200, 8-COLOR, EXTERNAL DOTCLK = 14.318MHz
ON | OFF | OFF | --> 640x200, 8-COLOR, FULL* DOTCLK = 14.318MHz
ON | OFF | OFF | --> 640x400, 8-LEVEL GRAY, EXTERNAL DOTCLK = 25.175MHz
OFF | ON | ON | --> 640x400, 8-LEVEL GRAY, FULL* DOTCLK = 25.175MHz
OFF | OFF | OFF | --> NOT USED
OFF | OFF | ON | --> NOT USED
OFF | OFF | OFF | --> NOT USED
DOTE (PIN 88) OPTIONS
ISW1-8 | FUNCTION
| OFF | RGB LATCHED BY DOTCLK'S RISING EDGE
| ON | RGB LATCHED BY DOTCLK'S FALLING EDGE
MODE SWITCH SETTING
ISW1-4 ISW1-3 ISW1-2 ISW1-1 | FUNCTION
| OFF | OFF | ON | OFF | --> MODE 6 8-LEVEL GRAY
SWITCH 2
PALETTE ROTATION
SW2-8 | SW2-5 | SW2-4 | FUNCTION
ON | ON | ON | RGB
ON | ON | OFF | BGR
ON | OFF | ON | GRB
ON | OFF | OFF | GBR
OFF | ON | ON | RBG
OFF | ON | OFF | BRG
OFF | OFF | ON | GRG
OFF | OFF | OFF | RGB
HSYNC/VSYNC POLARITY CONVERTING
SW2-3 | SW2-2 | HSYNC INPUT POLARITY | VSYNC INPUT POLARITY
ON | ON | - | -
ON | OFF | - | +
OFF | ON | + | -
OFF | OFF | + | +
TTL/ANALOG INPUT SELECTION
SW2-1 | FUNCTION
OFF | RGB TTL INPUT (EGA)
ON | RGB ANALOG INPUT (VGA)
JUMPERS
JUMPER FUNCTIONS
JMP# | TRACE | FUNCTION
1 | 2 - 3 | PALETTE REGISTERS ACCESSIBLE
| | | UNACCESSIBLE TO PALETTE REGISTERS
2 | 2 - 1 | ENABLE LMG5060XUFC'S BACKLIGHT BRIGHTNESS ADJUSTING FUNCTION
| 2 - 3 | DISABLE LMG5060XUFC'S BACKLIGHT BRIGHTNESS ADJUSTING FUNCTION
HITACHI MICRO SYSTEMS, INC.
LVIC-PROTO
PARABHAS K.
HOWARD N.
SW FUNCTION & TIMMING
USLVIC-PROTO-01H
A
10/10/91
SHEET: 0/0

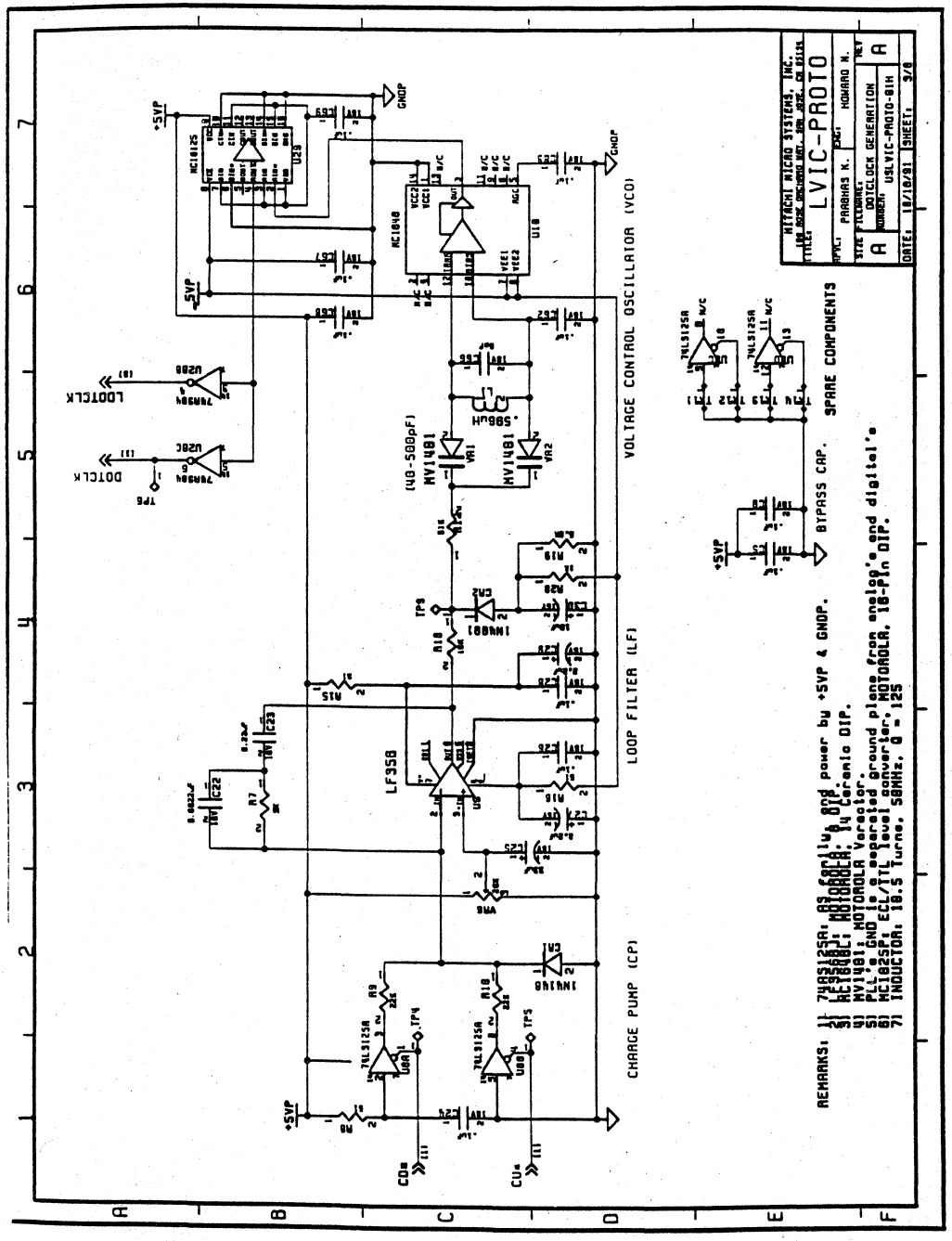

LMG5060XUFC BACKLIGHT POWER CONNECTOR
E1867
VIN
VOUT1
VOUT2
COM
GND
U25
CFL
HOT
N/C
N/C
COMB
DANGER
HIGH-VOLTAGE
V ~ 330 VRMS
TX10
MTP8P08
Q1
VR7
2N3904
Q2
R28
10K
R27
+5V
22uF
C25
C27
MOTMACT04
U28D
U28E
U28F
SPARE COMPONENTS
REMARKS: E1867: ERG, 5-pin, special package.
MTP8P08: MOTOROLA, TO-220AB package.
2N3904: MOTOROLA, TO-92 package.
HITACHI MICRO SYSTEMS, INC.
LVIC-PROTO
PARBHAS K.
HOWARD K.
330 VRMS CONVERTER
USLVIC-PROTO-01H
A
DATE: 10/16/91
SHEET: 7/8

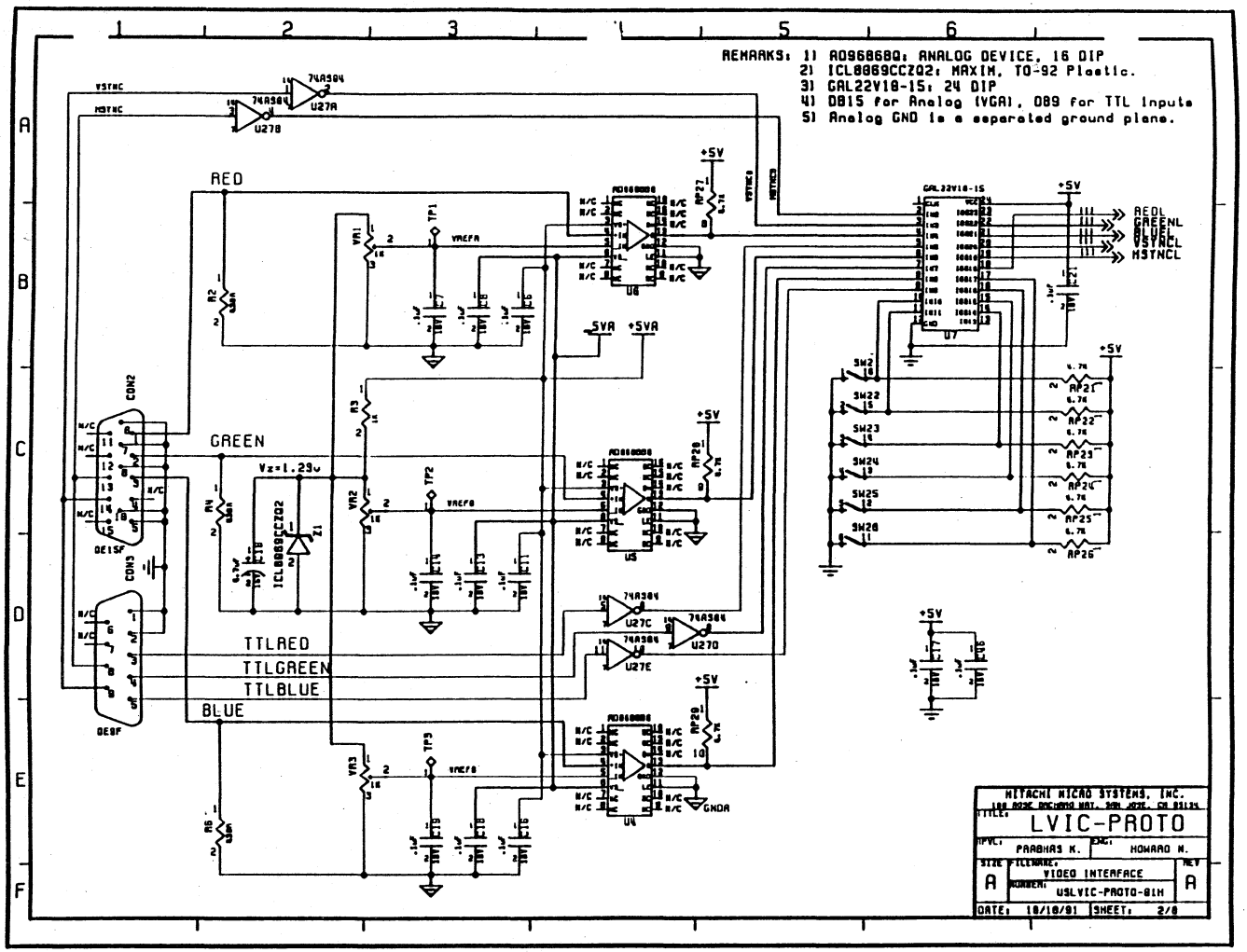
SECTION 4


R (3:0)
G (3:0)
-21.5V
B (3:0)
LOOTCLK
FLM
M
CL4
CL3
CL2
CL1
VCONT
+5V
74LS541
U22
U23
U24
R8F (3:0)
G8F (3:0)
B8F (3:0)
FLM8F
CL28F
CL18F
+12v
CON6
TM18081HC'S BACKLIGHT POWER CONNECTOR
(BLS-806M)
TM18081HC VIDEO SIGNAL
CONNECTOR
CON5
LMG5868XUFC VIDEO SIGNAL
CONNECTOR
CON7
CON8
LMG5868XUFC POWER CONNECTOR
CON10
BY PASS CAPS
HITACHI MICRO SYSTEMS, INC.
TITLE: LVIC-PROTO
DRVN: PRABHAS K.
ENG: HOWARD N.
SIZE A
FILENAME: LCD INTERFACE
REV A
NUMBER: USLVEC-PROTO-01H
DATE: 10/10/91
SHEET: 0/0


uA7805SC
U18
U28
U17
U19
uA7905SC
+5VA
+5VP
-5VA
-5VP
LN21RPHL
LED1
LED2
LED3
LED4
COMPARATOR POWER SUPPLY
PLL POWER SUPPLY
ANALOG GND PLANE
PLL GND PLANE
GNDA
GNDP
18uH
L2
L4
TL497AC
U21
CR3
1N4001
L3
1.0uH
Irate = 1.0 Amp.
R = 1 Ohm
LCD POWER SUPPLY
VCONT
21.5V
TP7
TP8
+12v
12V
+5V
POWER
PWR
MAX781CPA
U20
RESET
REMARKS: 1) uA7805C, uA7905C: TI, TO-220AB package (pinout different between 78/79)
2) TL497AC: TI, 14 DIP
3) MAX781CPA: MAXIM, 8 DIP.
4) Digital, analog, and PLL ground planes are seperated.
HITACHI MICRO SYSTEMS, INC.
LVIC-PROTO
PRABHAS K.
HOWARD N.
POWER & RESET
USLVIC-PROTO-01H
DATE: 10/10/91
SHEET: 5/8

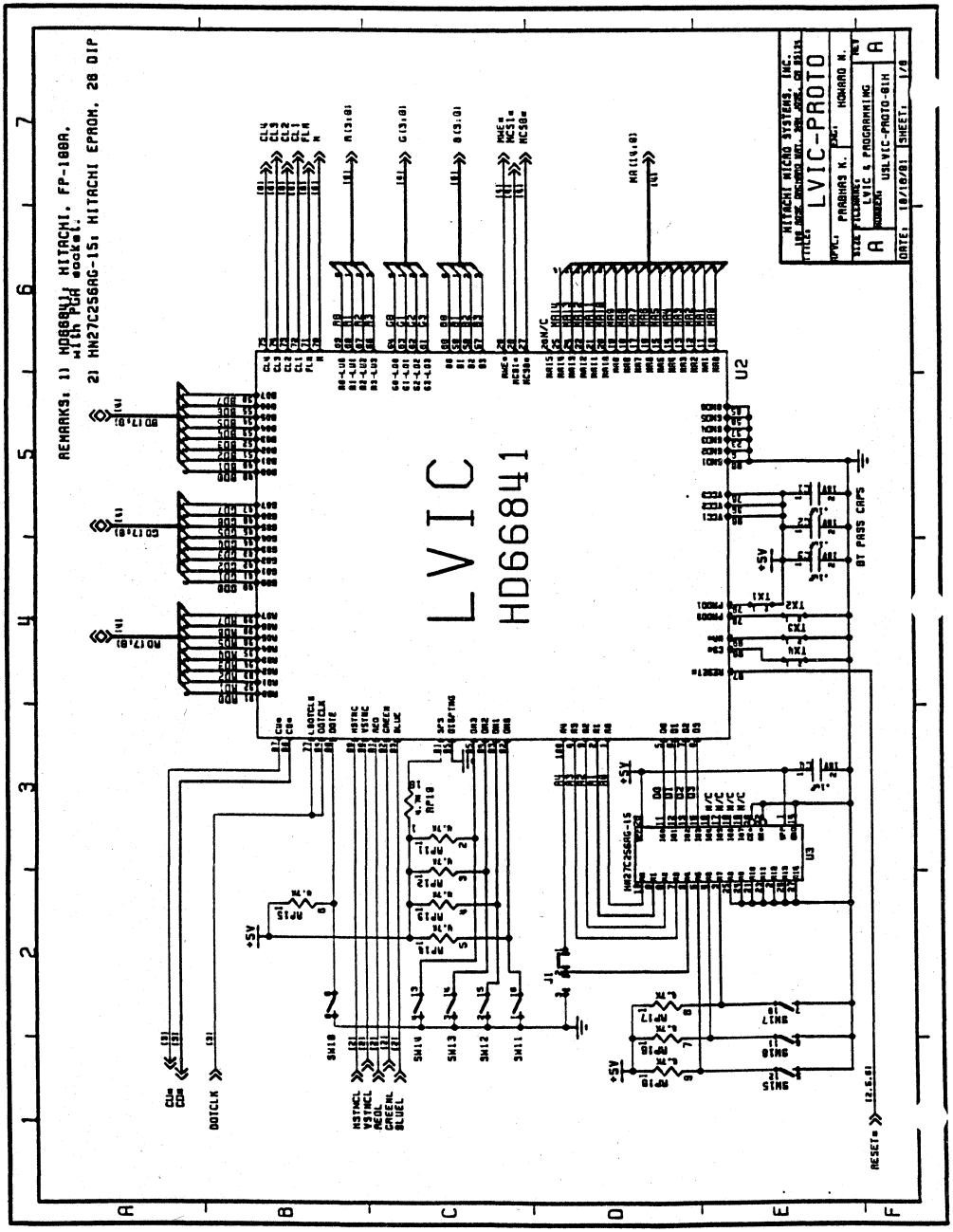

HITACHI MICRO SYSTEMS, INC.
LVIC-PROTO
PARBHAS K.
HOWARD N.
A
FRAME BUFFER
USLVIC-PROTO-01H
10/10/91
BD[7:0]
GD[7:0]
RD[7:0]
HMB2256P-12
+5V
U11
U12
U13
U14
U15
U16
MCS0=
MWE=
HA[14:0]
MCS1=
BY PASS CAPS


HITACHI MICRO SYSTEMS, INC.
LVIC-PROTO
DOTCLOCK GENERATION
USLVIC-PROTO-01H
VOLTAGE CONTROL OSCILLATOR (VCO)
LOOP FILTER (LF)
CHARGE PUMP (CP)
SPARE COMPONENTS
BYPASS CAP.
LOTCLK
DOTCLK
REMARKS:



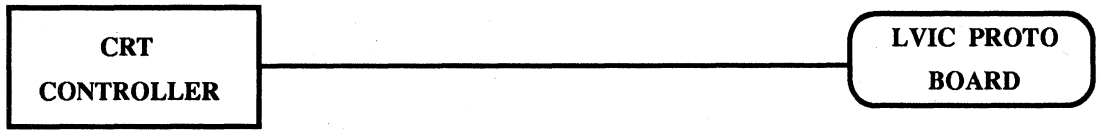
HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

This section shows a copy of the schematic supplied by Hitachi Micro System Inc., San Jose, California. It is merely included for reference and is not intended to be copied.



APPENDIX " A "

1.0 The 9 or 15 pin male to male video cable " C1 " is shown below :



" C1 "

2.0 The cables " C2 " and " C3 " are provided with as a part of the panel **inter connect kit.**

3.0 The LCD panel displays are to be ordered from Hitachi's Electron Tube Division.

HITACHI COLOR LCD TFT MODULE (TM16D01HC) :

Refer to the display data sheet for detail. The page 14 of it shows how the sub-pixels are designated for LVIC HD66841 interface with 160 dots (H) and 200 dots (V) resolution. The cable " C2 " provides the signals to the display while cable " C3 " provides the back light power. The display tilt and swivel angles provide different view angles in the ambient light, so it should be adjusted for the most desirable viewing angle.

HITACHI B/W FILM LCD MODULE (LMG5060XUFC) :

The mechanical, electrical, and optical specifications of this panel are stated in its data sheet, so , please refer to it. Its resolution is 640 dots (W) and 480 dots (H), with 1/240 duty cycle. Cold Cathode Flourescent back light is built inside the display.

SYSTEM DEBUG

First power up the system in CGA mode using a CRT color monitor and the Paradise EGA board effectively disconnecting the 6.3" TFT LCD display and reconnecting the cable " C1 " to the monitor. After the system is up in CGA mode, verify that it works correctly. Then, disconnect the cable " C1 " and reconnect it to the LVIC Proto Board 9 pin input. Also, verify that the cable " C2 " is properly connected to the display, since there is no key in the connector. If cable " C3 " is properly connected, florescent back light should come on and it is clearly visible. Also, verify the " SW1 " and " SW2 " settings on the LVIC Proto Board. The system must come up with " C " prompt.

Similarly, change the " C1 " cable to analog 15 pin male cable and test the VGA panel LMG5060. The "SW1" and "SW2" switch settings change as shown earlier. Create VGA directory under the C:\ Then, copy VGAMODE.EXE file from the Oak software diskette in it. Run the VGAMODE.EXE file for different screen resolutions. However, for LMG5060 panel select 640 x 480 resolution.

If every thing is working correctly, one can execute all the DOS commands when appropriate prompts are displayed on the LCD screen.

DEMONSTRATION SOFTWARE

After DOS 3.2 or later is installed, load any CGA or VGA graphics package such as PRINTSHOP, WINDOWS file management package XTREE for the LCD display demonstration. With the LVIC proto Board in the system, the color video display will be replaced by the color TFT LCD display. The black and white display will show shades of grey. By running Kaleidoscope 1 and 2, under PRINT SHOP, different dot and line patterns can be shown on either LCD screen.

To display scanned image files (*.GIF), call Hitachi America Ltd., office at Sierra Point , CA.

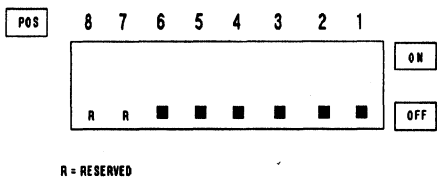
LVIC PROTO BOARD

The User Manual for this board describes all switches and their settings along with the PAL equations for LMG5060XUFC and TM16D01HC panels. In this section, only the switches are addressed. So, please refer to the LVIC Proto Board User's Guide for more details.

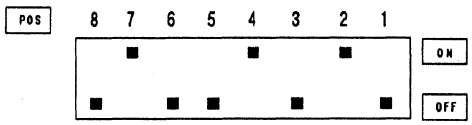
The CN1 and CN4 connector artwork is on the PCB, but they are not to be poppulated or used by the customer.
If the power cables are short, they may be extended. The external power connector and bench power supplies are to be used as shown in the system block diagram. The nominal power consumption of this board is stated below so that adequate power can be externally provided:

+5V @ 1 A , +12V @ 0.17 A, -12V @ 0.13 A

The LVIC Proto Board accepts analog level input signals carried by the 15 pin male cable " C1 " from theOak VGA video controller board inside the PC-AT. The R,G,B,HSYNC, and VSYNC signals are used to regenerate the CRT dot clock, and sample the incoming video data. The output signals are sent over the cable " C2 " to the black and white LMG5060 Hitachi display. This board also provides 330 Volts RMS required by the back light through the cable " C3 ". The switch settings of this board are shown below in Figure 4 :



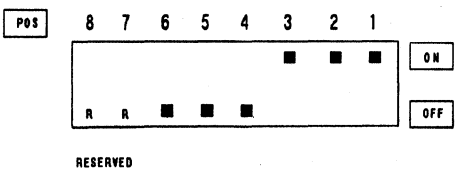
SW2



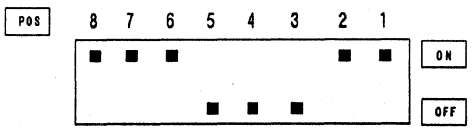
SW1

FIGURE 4

The LVIC Proto Board also accepts TTL level input signals carried by the 9 pin cable " C1 " from the Paradise video controller board. The R,G,B, HSYNC, and VSYNC signals are used to regenerate the CRT dot clock, and sample the incoming video data. The output signals are sent over the cable " C2 " to the 6.3" TFT, 8 colors, Hitachi display TM16D01HC. This board also provides +12 Volts required by the back light through the cable " C3 ". The switch settings of this board are shown under in Figure 5 :



SW2



SW1

NOTE: The attached schematic is only for reference, so do not copy it for your design.

FIGURE 5

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

SYSTEM COMPONENTS

The hardware components are described in this section while the " C1 ", and " C2 " cable details are shown in the Appendix " A ".

PC-AT : AST Premium 286 model 70 was operating at 10 MHZ, with 512KB memory; 20MB hard disk drive, and 1.2 MB, 5.25 " floppy drive. It was also running DOS version 3.2.

VIDEO CONTROLLERS : For OAK VGA cottroller board refer to its User's manual. The settings for the the six jumpers on the board are shown below in Figure 2 :

JUMPERS

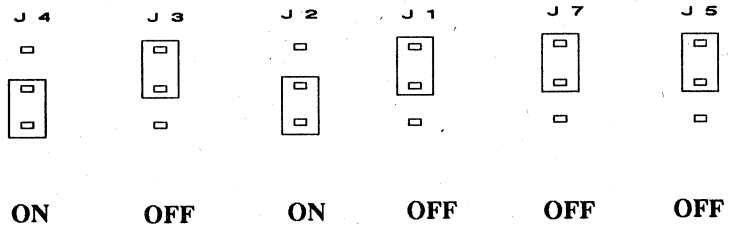
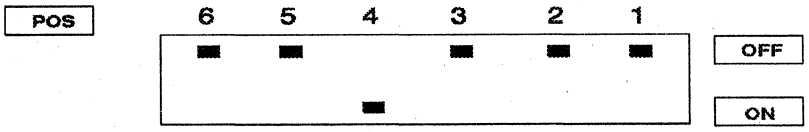


FIGURE 2

Paradise Autoswitch EGA is card used in the CGA mode at (640H x 200V) resolution providing TTL level signals to the CRT monitor. The switch settings for 80 column, RGB monitor in CGA mode are listed below in Figure 3 :



NOTE :

1.0 For more details refer to the Paradise CRT controller manual.

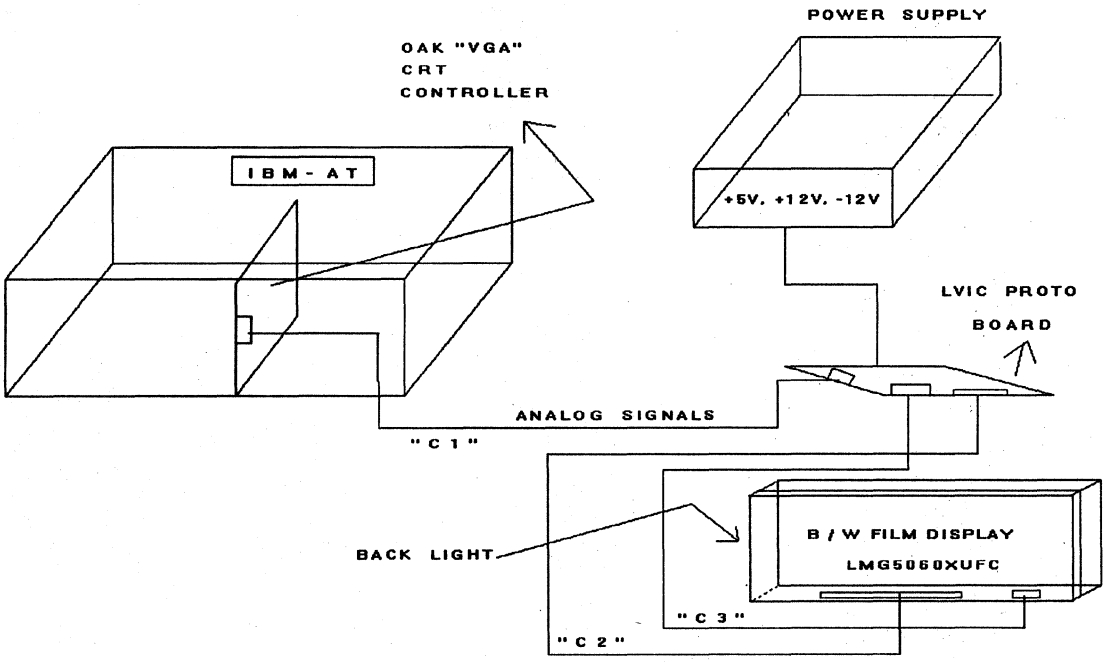
2.0 Make sure this switch is correctly set.

FIGURE 3

SYSTEM CONFIGURATION

The development system was configured with IBM PC-AT or compatible machine, OAK VGA or Paradise Autoswitch EGA 480 card, LVIC Proto Board, and Hitachi B/W LCD panel LMG5060 or TFT active matrix, 8 color, 6.3", LCD display TM16D01HC from the ELT division. The cables required to provide TTL or analog level input video signals to the LVIC Proto board are not provided. The LVIC Proto board output connectors to the 6.3" TFT or Black and white display are provided to make the display connection task easier. A separate AC high voltage cable is also required for the back light of each type of display. The back light is easily mounted with the four corner screws of the 6.3" TFT display. The LMG 5060 display has built in back light.

The system diagram for the LMG5060 LCD panel is shown below :



NOTE : " C2 " = " C3 " Cables are provided.

FIGURE 1

The diagram shown above in Figure 1 is to be modified for a TFT color display. The OAK VGA controller is replaced by the Paradise Autoswitch EGA 480 Board and the LCD panel LMG5060 is replaced by the Color panel TM16D01HC. The analog level input and CMOS level output cables for the LVIC proto board will be likewise changed to drive the color module.

HD66841 / LMG5060

Technical Brief

LVIC Proto Board

Kash Yajnik

The HD66841 LVIC Proto Board was designed by Hitachi Micro Systems Inc., San Jose, and can be ordered through Hitachi America Ltd., in U.S.A. The board is shipped with cables kit for **multiple** Liquid Crystal Modules from Hitachi.

Black and white as well as color information can be displayed depending upon the selected LCD panel from Hitachi's ELT Division. This board is designed to display a black and white image with eight shades of grey using LCD panel LMG5060XUFC having VGA (640Hx480V) resolution. It can also display an eight colors CGA (640Hx240V) image when used with color TFT LCD display panel TM16D01HC.

The LVIC Proto Board resides outside the IBM PC-AT or a compatible system running later than DOS version 2.0. It requires external power supply. The back light power is also provided by this board.

A special prototype space is reserved on this board for customer circuit design and development in the critical areas of LVIC HD66841 based implementation.

LVIC Proto Board User Manual is also provided to customize this board for many applications.

This technical brief is written to complement the LVIC Proto Board User Manual for **one** specific application using the 6.3" color TFT module (TM16D01HC) or VGA module (LMG5060XUFC) from Hitachi's Electron Tube Division (ELT). A copy of the schematic is also included to provide the design implementation detail. A system diagram is also included to high light the laboratory environment. Similarly, each user may tailor display subsystem requirements for the desired application.

The scope of this document is to help make the customization task easier and quicker. The circuit minimization tasks are left to each user and are **not** attempted. This is intended as an illustrative example for the Hitachi field and technical staff, and their customers.

The following pages cover system configuration and components, the LVIC Proto Board set up, system debug, and demonstration software. The Appendix "A" covers analog / digital cable connection and the Appendix "B" shows lists the schematic.

Refer to the subsequent pages for more detail.

APPENDIX " C "

LM016XML

- 16 character x 2 lines
- Controller LSI HD44780 is built-in (See page 97).
- +5V single power supply
- Color tone . . . . . New gray

MECHANICAL DATA (Nominal dimensions)

Module size . . . . . 84W x 44H x 12T (max.) mm
Effective display area . . . . . 61W x 15.8H mm
Character size (5 x 7 dots) . . . . . 2.96W x 4.86H mm
Character pitch . . . . . 3.55 mm
Dot size . . . . . 0.56W x 0.66H mm
Weight . . . . . about 35 g

ABSOLUTE MAXIMUM RATINGS

	min.	max.
Power supply for logic ($V_{DD}-V_{SS}$)	0	6.5 V
Power supply for LCD drive ($V_{DD}-V_O$)	0	6.5 V
Input voltage (Vi)	V_{SS}	V_{DD} V
Operating temeprature (Ta)	0	50°C
Storage temperature (Tstg)	−20	70°C

ELECTRICAL CHARACTERISTICS

Ta = 25°C, V_{DD} = 5.0 V ± 0.25 V

Input "high" voltage (Vi_H) . . . . . 2.2 V min.
Input "low" voltage (Vi_L) . . . . . 0.6 Vmax.
Output high voltage (V_{OH}) ($-I_{OH}$ = 0.2 mA) . . 2.4 V min.
Output low voltage (V_{OL}) (I_{OL} = 1.2 mA) . . . . 0.4 Vmax.
Power supply current (I_{DD}) (V_{DD} = 5.0 V) . . 1.0 mA typ.
3.0 mA max.
Power supply for LCD drive (Recommended) ($V_{DD}-V_O$)
Duty = 1/16
Range of $V_{DD} - V_O$ . . . . . 1.5 ~ 5.25 V
Ta = 0°C . . . . . 4.6 V typ.
Ta = 25°C . . . . . 4.4 V typ.
Ta = 50°C . . . . . 4.2 V typ.

Notes:

In the HD44780, the data can be sent in either 4-bit 2-operation or 8-bit 1-operation so that it can interface to both 4 and 8 bit MPU's.

(1) When interface data is 4 bits long, data is transferred using only 4 buses of DB_4~DB_7 and DB_0~DB_3 are not used. Data transfer between the HD44780 and the MPU completes when 4-bit data is transferred twice. Data of the higher order 4 bits (contents of DB_4~DB_7 when interface data is 8 bits long) is transferred first and then lower order 4 bits (contents of DB_0~DB_3 when interface data is 8 bits long).

(2) When interface data is 8 bits long, data is transferred using 8 data buses of DB_0~DB_7.

INTERNAL PIN CONNECTION

Pin No.	Symbol	Level	Function	
1	V_{SS}	–	0V	Power supply
2	V_{DD}	–	+5V	
3	V_O	–	–	
4	RS	H/L	L: Instruction code input H: Data input	
5	R/W	H/L	H: Data read (LCD module→MPU) L: Data write (LCD module←MPU)	
6	E	H, H→L	Enable signal	
7	DB0	H/L	Data bus line Note (1), (2)	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		

Unit: mm

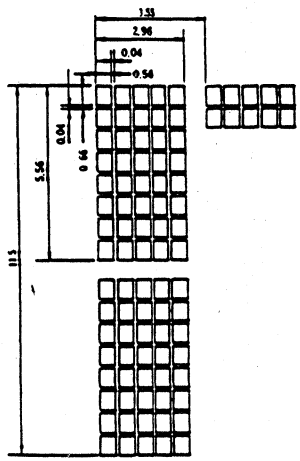
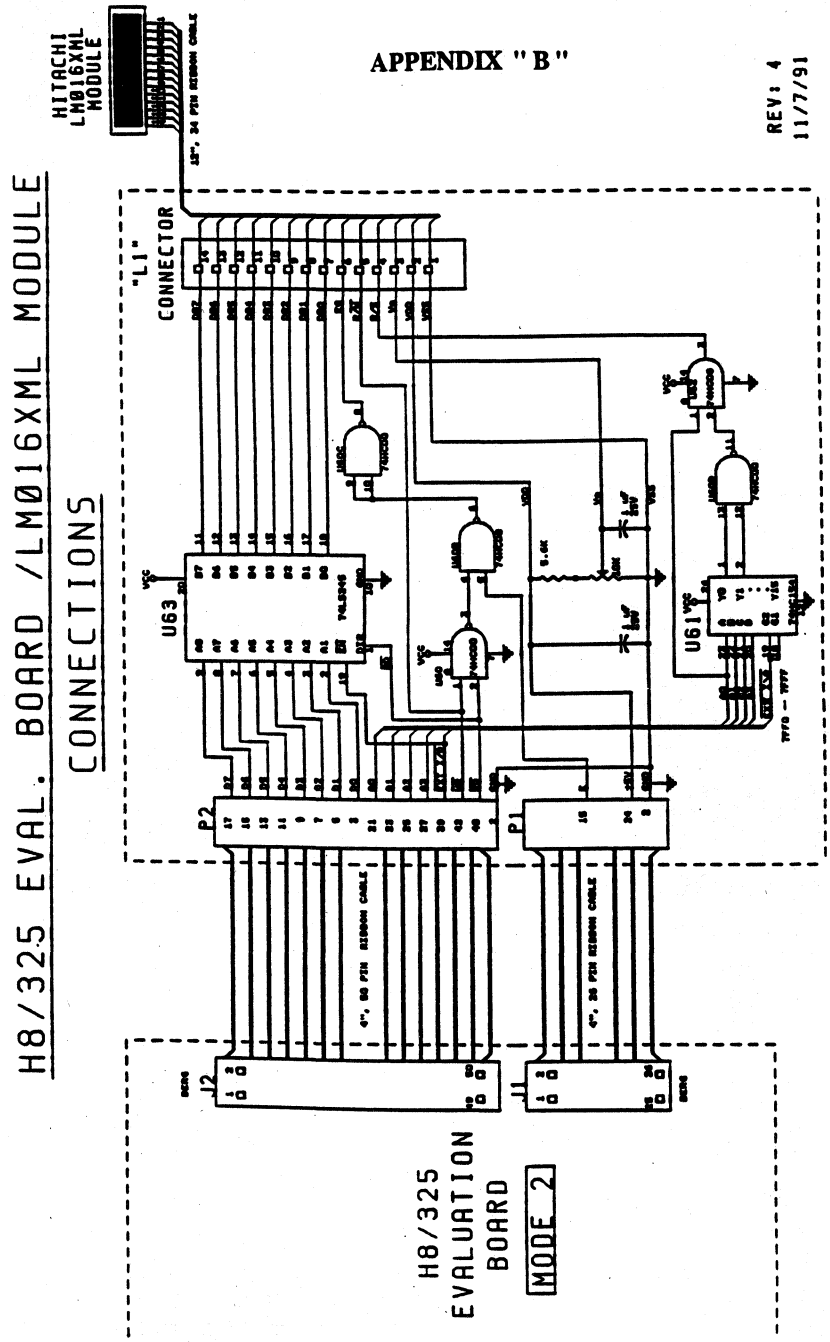


Fig. 1 Display pattern

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDIX "B"

REV: 4
11/7/91



APPENDIX " A "

The H8/325 Evaluation Board Kit (US328EVB01H) includes the following items:

- o H8 / 325 Evaluation Board
- o Power cable for the board
- o Board Stand Offs (Q=4)
- o Five HMSI Demonstration Programs Diskette for PC-AT
- o Software Agreement Copy
- o Hardware Manual (M21T133) from HMSI
- o Software User Manual (HSM325EMSI1SE) from HMSI

The board factory jumpers, switch settings, and other details are shown in the hardware manual. They may be changed for this application. The associated steps are listed below for clarity:

1.0 The 20 MHz crystal is to be replaced by 16 MHz crystal.
2.0 The " SW1 " is set for ROM position.
3.0 Jumper " J5 " is set for 32KB ROM space.
4.0 Jumper " J6 " for RAM space is not installed.
5.0 Jumpers " J3 " and " J4 " for mode selection are set for mode " 2 " i.e. " J3 " is installed while the jumper ' J4 " is removed.

These are shown below in the Figure 2 :

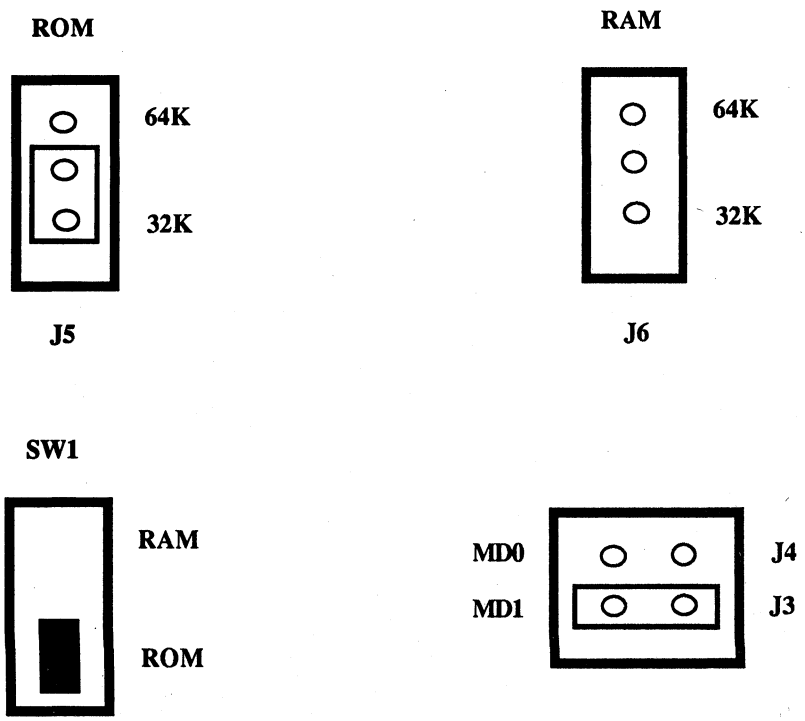


FIGURE 2

SUB SYSTEM COMPONENTS

The LCD display subsystem components such as H8 / 325 Evaluation Board, LM016XML display, LCD Interconnect Board, Hitachi Laptop Computer, External Power Supply and the related software are described in this section.

H8 / 325 Evaluation Board : This board was designed by Hitachi Micro Systems, San Jose, CA. It is provided as a demonstration and development tool. On-board EPROM contains the Hitachi Monitor firmware used for single line assembly, disassembly, line editing, and debug purposes. Of the two serial ports, only the **Terminal** port is used to down load, up load, and run the programs. The I/O extention connectors "J1" and "J2" are used to connect to the LCD Interconnect Board. The partially decoded, extented I/O space is further decoded on the LCD Interface Board. This board is designed to run at 10 MHz and uses a 20 MHz crystal for that purpose. However, in this application a 16 MHz crystal is used to provide 1 MHz "E" clock to the LCD Controller HD44780 located on the LCD panel. All the jumpers on this board are not set to the factory default states. Refer, to the Appendix "A" for the H8 / 325 Evaluation Board details including the switch and jumper settings.

LCD Panel Display (LM016XML) : This display is provided by the Hitachi's ELT Division. It is capable of displaying 2 lines of eight 5x7 alpha numeric characters. It is 40 dots wide and 16 dots high. It has 1/16 duty cycle. The parallel data is clocked in at 1 MHz " E " clock rate. It runs from +5V power supply. The customer has to solder 14 pins on LM016XML panel for the appropriate connector used on the LCD Interconnect Board. The LM016XML LCD panel mounting and the proper viewing angles are critical to a strain free LCD display. Please, handle the panels according to the care recommended by the LCD display manufacturer. The logic signals sent to the LCD panel are at CMOS levels. Refer to the Appendix " C " for more information on the panel.

LCD Interconnect Board : A wire wrap board was built to send parallel data, control signals, and power to the LCD panel over the "L1" cable. The I/O extention cables "J1" and "J2" were connected to the H8 / 325 Evaluation Board. The LM016XML LCD panel contrast adjust potentiometer was also put on this board. The data bus buffer and gating logic were also located on this board. The power on reset pulse was provided by the H8 / 325 Evaluation Board. Refer to the Appendix " B " for its schematic.

Hitachi Laptop Personal Computer "HL320" : It is connected to the serial terminal port of the H8/ 325 Evaluation Board. The connector RJ-12 is attached to the Terminal port while a male to female 25 pin adapter cable is required at the Laptop PC end. The Hitachi " HL320 " PC provides the software development tools for the user programs. The demonstration program up load and down load capability is also provided by the laptop PC. The communication link is full duplex, 9600 baud, 8 bits, 1 stop bit, and no parity check.

Power Supply : Open frame switcher power supply from Kepco, Model # ECM-021K-CB is used to power up the H8 / 325 Evaluation Board. Its rating is +5V @ 2A, +12V @ 0.3A, and -12V @ 0.2A. The Interconnect Board as well as the LCD display are also powered by it.

Software : The laptop PC resident software development tools, packages, and utilities are described very briefly:

H8 / 325 Cross Assembler : It is designed for DOS environment inside the laptop Personal Computer.
When the user program is submitted as the source file, it assembles the code. Consequently, it produces Object and List files of the source program.

H8 / 325 Linker : To link various object code segments (" *.OBJ " extention) developed in parallel for a larger program. The linked file has " *.ABS " extention. Motorola " S " record conversion utility is also included with the linker, and is used as output file with " S " record format.

Upload : To up Load " S " Record file, push " EDIT SHIFT " Key down. Depress the " PG UP "key when using "PROCOMM " package for communications. Also, select ASCII format.

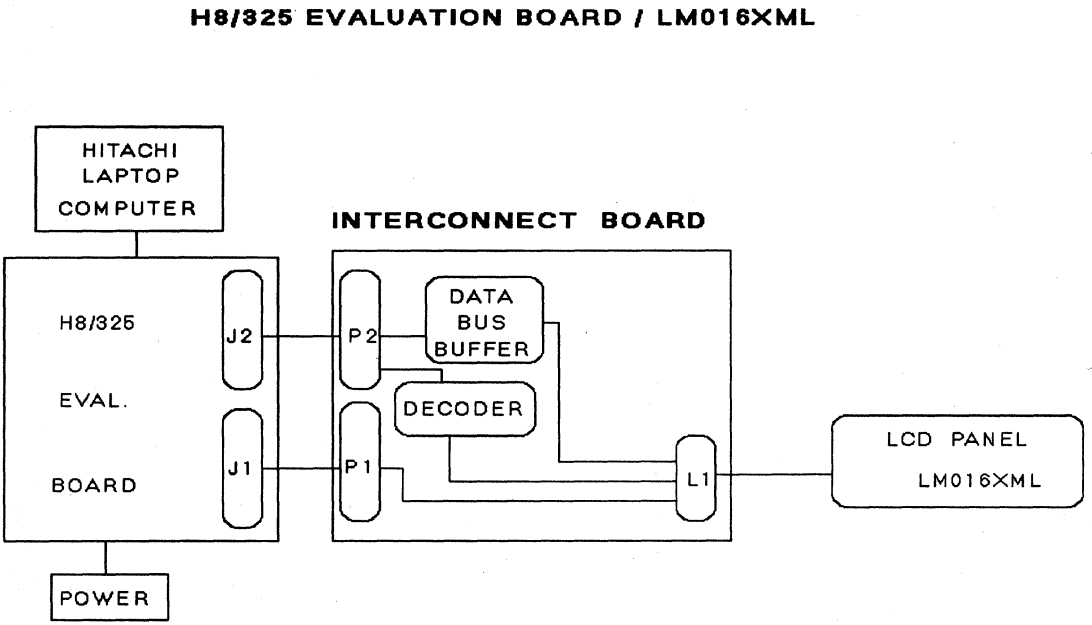
Demonstration File : Motorola " S " record file " INIT780C.ABS " is uploaded to the H8 / 325 Evaluation Board. The uploaded file i.e. " INIT780C.ABS " is run for display demonstration.

Screen Editor : Any word processing package is acceptable. In this application, Microsoft "WORD" package is used. The source programs are created and edited with this package. The source program files have " *.SRC " extensions.

SUB SYSTEM CONFIGURATION

The display subsystem was configured with H8/325 Evaluation Board, LCD display Interconnect Board, and LM015XML panel from the ELT division. The required cable lengths are shown in the schematic for CMOS signal levels. The LCD power pins are a part of the 14 pin panel cable, so a separate power cable is not required.

The subsystem block diagram for the Interconnect Board is shown below in Figure 1 :



BLOCK DIAGRAM

NOTE : The required cables cables may be built or purchased by the user, from other vendors.

FIGURE 1

HD6473258 / LM016XML

Technical Brief

H8/325 Evaluation Board & LCD Display

Kash Yajnik

The H8/325 Evaluation Board(US328EVB01H) was designed by Hitachi Micro Systems Inc., San Jose, and can be ordered through Hitachi America Ltd., in U.S.A. The board is shipped with power cable header, stand off hardware, demonstration programs, and associated manuals.

Black and white character information can be displayed depending upon the selected LCD panel from Hitachi's ELT Division. Among the many products offered by the Hitachi's ELT Division, for this application, LCD panel LM016XML was selected.

An Inter Connect Board is required to enable the H8/325 Evaluation Board to talk to the LCD display LM016XML. The character data is sent to the LCD panel for processing as well as display. The HD44780 LCD Controller Driver from Hitachi, SICD, located on the LM016XML panel, processes the data sent by the H8/325 Evaluation Board for display.

The H8/325 Evaluation Board resides on a bench connected to the LCD interface board. The other end of the interface board is connected to the panel. It requires external power supply. After power on, a demonstration program is down loaded and run, to display a character message.

This technical brief is written to complement the H8/325 Evaluation Board Manuals for one specific application i.e. interfacing to a peripheral. In this case, LCD panel LM016XML from Hitachi's Electron Tube Division (ELT).

A copy of the schematic is included to provide the design implementation detail. A system diagram is also included to high light the laboratory environment. Similarly, each user may tailor other subsystem requirements for the desired application.

The scope of this document is to help make the customizing task easier and quicker. The circuit minimization tasks are left to each user and are not attempted. This is intended as an illustrative example for the Hitachi field and technical staff, and their customers.

The following pages cover sub system configuration and components, H8/325 Evaluation Board set up, System Debug, and Demonstration Software. The Appendix " A " covers H8/325 Evaluation Board details, and the Appendix " B " shows the Interconnect Board schematic. Also, Appendix " C " lists the LCD Panel data sheet.

Refer to the subsequent pages for more detail.

5.0 For more details, refer to the lines for addresses **60H** and **70H** in the code sample is shown below:

ADDRESS	HEXADECIMAL															
	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
00000000	02	0E	0C	07	01	0B	02	07	04	00	0B	05	02	03	0B	00
00000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000020	03	0E	0C	07	01	0B	02	07	04	00	0B	05	02	03	0B	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	03	06	0D	0F	05	0B	02	07	04	00	04	05	02	00	09	00
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000060	03	06	0D	0F	05	0B	02	07	04	00	04	05	02	00	09	00
00000070	07	0C	02	05	06	07	08	0A	00	00	00	00	00	00	00	00
00000080	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00000090	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000000A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000000B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000000C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000000D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000000E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000000F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD66841 LVIC-II ROM Programming Mode

Tech Notes

Application Engineering

Kash Yajnik

Palette Registers Access

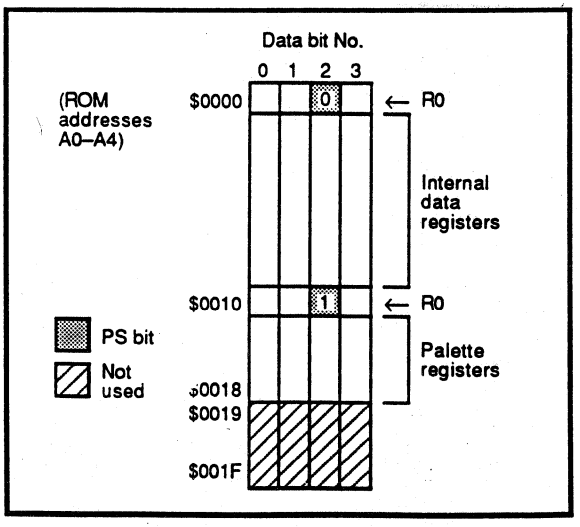
The palette registers inside the HD66841 (P1-P8) are provided for different shades of 13 level grey scale. The palette registers are **not** to be used for color LCD display. To use these palette registers, the following procedure is suggested along with sample code for it :

1.0 Connect HD66841 address A4 (Pin 100) to the EPROM / ROM address A4.

2.0 After power on, the HD66841 will continuously cycle the addresses A0 - A4. The contents of the EPROM / ROM where the programming information is stored will be continuously read by the HD66841. However, the EPROM / ROM contents will **only** be loaded, when the power on reset pulse is applied.

3.0 Therefore, if the LCD display register settings are to be changed **dynamically**, a power on reset pulse is required to reload the new EPROM / ROM data in the LVIC II.

4.0 The details of the palette register select (PS) bit i.e. Register R0 bit 2 for the ROM Programming method are shown in the diagram below:



HD44780 - ROM MASK CHANGE FOR CHARACTER GENERATION

CHARACTER ADDRESS MAP FOR 32 (5X10) CODES

UPPER NIBBLE (4 BITS)

LOWER NIBBLE (4 BITS)

	EH	FH	← NOT USED →																	
0H																				
1H																				
2H																				
3H																				
4H																				
5H																				
6H																				
7H																				
8H																				
9H																				
AH																				
BH																				
CH																				
DH																				
EH																				
FH																				

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD44780 - ROM MASK CHANGE FOR CHARACTER GENERATION

2.0 Sixteen EPROM addresses for one 5x10 character. Therefore, for 32 character codes: EPROM addresses used = 16 x 32 = 512. Character font or matrix for letter"y" is shown by the EPROM dot pattern listed below:

EPROM ADDRESS								LINE POSITION			EPROM OUTPUT						
DD RAM DATA CHARACTER CODES								CG RAM ADDRESS									
A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	+	+	O4	O3	O2	O1	O0
1	1	1	1	1	0	0	1	0	0	0			0	0	0	0	0
								0	0	1			0	0	0	0	0
								0	1	0			1	0	0	0	1
								0	1	1			1	0	0	0	1
								1	0	0			1	0	0	0	1
								1	0	1			1	0	0	0	1
								1	1	0			0	1	1	1	1
								1	1	1			0	0	0	0	1
	0	0						0	0	0			0	0	0	0	1
								0	0	1			0	1	1	1	0
								0	1	0			0	0	0	0	0 *
								0	1	1							
								1	0	0							
								1	0	1							
								1	1	0							
								1	1	1							

NOTE :
1.0 "++" = Unused and unprogrammed EPROM outputs.
2.0 "*" = Cursor 'OFF" code line 11. Fill with "0".
3.0 "1" = LCD display dot "ON".

HD44780 - ROM MASK CHANGE FOR CHARACTER GENERATION

CHARACTER ADDRESS MAP FOR 160 (5X7) CODES

UPPER NIBBLE (4 BITS)

LOWER NIBBLE (4 BITS)

	2H	3H	4H	5H	6H	7H	AH	BH	CH	DH	← NOT USED →									
0H																				
1H																				
2H																				
3H																				
4H																				
5H																				
6H																				
7H																				
8H																				
9H																				
AH																				
BH																				
CH																				
DH																				
EH																				
FH																				

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD44780 - ROM MASK CHANGE FOR CHARACTER GENERATION

1.0 Eight EPROM addresses for one 5x7 character. Therefore, for 160 character codes: EPROM addresses used = 8 x 160 = 1440. Character font or matrix for "P" is shown by the EPROM dot pattern listed below:

EPROM ADDRESS									LINE POSITION			EPROM OUTPUT						
DD RAM DATA CHARACTER CODES									CG RAM ADDRESS									
		A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	+	+	04	03	02	01	00
		0	1	0	1	0	0	0	0	0	0	0			1	1	1	1	0
		↓	↓	↓	↓	↓	↓	↓	↓	0	0	1			1	0	0	0	1
										0	1	0			1	0	0	0	1
										0	1	1			1	1	1	1	0
										1	0	0			1	0	0	0	0
										1	0	1			1	0	0	0	0
										1	1	0			1	0	0	0	0
										1	1	1			0	0	0	0	0 *

NOTE :
1.0 "++" = Unused and unprogrammed EPROM outputs.
2.0 "*" = Cursor 'OFF" code line 8. Fill with "0".
3.0 "1" = LCD display dot "ON".

HD44780 - ROM MASK CHANGE FOR CHARACTER GENERATION

HD44780 and HD 44780A have internal character generator ROM equivalent to Hitachi part # HD44780A00. This character set has English as well as other symbols and is shown in the data sheet. It can display 160 characters which are formed on 5X7 dot matrix with eighth row assigned for the cursor. Additionally, 32 different character patterns are possible with 5x10 character boxes at pre-assigned character addresses with eleventh row for the cursor display. In all, 160 plus 32 i.e. 192 different character codes can reside in the internal ROM.

When a customer wishes to display special characters, the HD44780 / HD44780A masked ROM inside the part **has** to be changed. The data sheet pages150 through 155 describe Hitachi 's procedure for modifying character patterns. The character patterns are provided to Hitachi inside a 2Kx8 or larger EPROM. After pattern varification by Hitachi and the customer, **trial** sample parts are given to the customer for display and evaluation. Subsequently, the custom part with CG ROM change is made for volume production. NRE charge may be normally required for this change. A customer develops Character patterns using DATA I/O or other programming tools. If the EPROM is bigger than 2Kx8, only the first 2Kx8 partition is to be used. The unused locations may be programmed as 0.

The page 2 shows, character code for "P" on a 5x7 character box. EPROM outputs O5, O6, and O7 are unused and can be treated as don't cares. The DD RAM data provides the 8 bit character codes while the CG RAM address supplies (lower 3 address) bits for the line positions inside the character box. A logic "1" corresponds to LCD display dot "ON" condition. Since, the unused bits in an EPROM are logic high, they may turn the undesired display dots "ON". Therefore, when in doubt turn the unused character dots "OFF" i.e. program logic "0".

The page 3 shows the pre-assigned character address map for the 160 (5x7) character codes.

Similarly, page 4 shows the example of 5x10 character box for the character "y". Note that the eleventh line is programmed "0" for the cursor. EPROM outputs - the 12 th row address and beyond are programmed "0". Also, note that for 9th, 10th, and 11th row address, the A9, and A8 bits are programmed "0".

The page 5 shows the character address map for the 32 (5x10) characters. No more than 32 characters can be accommodated. The 32 characters will require the number of EPROM addresses shown below: With 16 addresses per character, the 32 characters will require 16x32 = 512 EPROM addresses. Therefore, an EPROM with 1440 + 512 = 1952 i.e. 2K bytes will be enough to contain the desired number of character patterns for this part.

Since, the character address space inside the HD44780/HD44780A is pre-assigned only the character dot patterns can be changed. The character address space is **not** changeable. If a customer desires the flexibility of using an external EPROM for chareacter generation, please, recommend that HD66840 or other LCD controller or LCD module using HD66840 may be considered in the design. The Hitachi ELT Division, Schaumburg, Illinois, will be happy to provide RFQ for custom LCD modules.

The details discussed are expanded on the following pages.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD44780 LCD CONTROLLER

ROM Mask Change

Character Generation

Kash Yajnik

This technical brief covers custom character generation using LCD Controller HD44780. The data sheet specifies a standard character set using the C.G. ROM built inside the HD44780. Refer to the data sheet or the LCD Controller Manual # U74 for more information on the standard character set.

For character sets that require other special characters such as Arabic, Hebrew, Katakana, or Kanji, to name a few, a **mask change** is required for the internal CG ROM. This information is provided to augment the character development procedure inside the HD44780 data sheet. The following pages describe this in greater detail.

It is suggested that the customer build a target subsystem using the HD44780 and the desired LCD display panel. The internal C.G. can then be used to display and develop a working character pattern tester. The custom HD44780 **prototype** parts can be tested on this tester to check out the special character set.

Normally, EPROM resident character patterns are used to transfer the information between Hitachi and the customer. Any EPROM which can store larger than 2K bytes, may be used to transport the character dot patterns.

Commercially available EPROM programmers may be used to program the character dot patterns inside the EPROM for **submission** to Hitachi America, Ltd., (HAL).

For more information, consult your nearest **HAL**, sales office or call the address listed below.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

The literature and other documents used in this design are summarized below :

- o H8/532 Cross Assembler Manual #S085CPC and " C " compiler for IBM PC
- o H8/532 Evaluation Board User's Manual # US538EVB21H
- o H8/532 Software User's Manual # HS538EMSS1E
- o MS " WORD " User Manual and other reference manuals
- o " PROCOMM " User Manual and other reference manuals
- o LCD Data Book #M24T013 from Hitachi America Ltd.
- o Memory Data Books from Hitachi America Ltd.
- o Hitachi Graphic Module Catalog # XX-E139 from ELT Division
- o H8/532 Hardware User's Manual #M21T002 from Hitachi
- o H8/500 Programming Manual #M21T001 from Hitachi
- o H8/500 Software Application Note #M21T003 from Hitachi
- o H8/532 Overview #M21T173 from Hitachi
- o Hitachi Laptop Personal Computer HL320 - Operator Manual
- o Hitachi Laptop Personal Computer HL320 - MSDOS V3.2 User's Manual

HD61830B / LM200 LCD PANEL DESIGN

APPENDIX " C "

REFERENCE LITERATURE

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

*** SECTION DATA LIST

SECTION	ATTRIBUTE	SIZE	START
GRA	REL-CODE	0E198	

*** CROSS REFERENCE LIST

NAME	SECTION	ATTR	VALUE	SEQUENCE
C1	GRA		0000E096	68* 70
C2	GRA		0000E0AF	77* 79
C3	GRA		0000E0CB	87* 97
C4	GRA		0000E0CC	88* 90
GRA	GRA	SCT	00000000	2*
X	GRA	EXPT	0000E000	3 8*
X1	GRA		0000E01A	21* 23
X10	GRA		0000E137	131* 133
X11	GRA		0000E14E	139* 141
X12	GRA		0000E165	147* 149
X13	GRA		0000E17C	155* 157
X2	GRA		0000E033	30* 32
X3	GRA		0000E04C	39* 41
X4	GRA		0000E065	48* 50
X6	GRA		0000E07C	56* 58
X7	GRA		0000E0EB	102* 104
X8	GRA		0000E107	114* 116
X9	GRA		0000E11E	122* 124

```
*****TOTAL ERRORS    0
*****TOTAL WARNINGS  0
```

```
115 GRA   C E10C ACF7                BTST     #7,R4          ;BIT TEST #7 OF R4
116 GRA   C E10E 26F7                BNE      X8             ;IF B/FLAG =Z=1 GO TO X8
117 GRA   C E110 00        NOP                          ;B/FLAG NOT SET
118 GRA   C E111 500B                MOV:E    #H'B,R0        ;R0=BH
119 GRA   C E113 157FF10090          MOVTPE   R0,@H'7FF1     ;BH TO 7FF1
120 GRA   C E118 5204                MOV:E    #H'4,R2        ;R2=04H
121 GRA   C E11A 157FF00092          MOVTPE   R2,@H'7FF0     ;04H TO 7FF0
122 GRA   C E11F 00        NOP
123 GRA   C E120 157FF10084    X9:   MOVFPE   @H'7FF1,R4     ;READ 7FF1 DATA TO R4
124 GRA   C E125 ACF7                BTST     #7,R4          ;BIT TEST #7 OF R4
125 GRA   C E127 26F7                BNE      X9             ;IF B/FLAG =Z=1 GO TO X9
126 GRA   C E129 00        NOP                          ;B/FLAG NOT SET
127 GRA   C E12A 500C                MOV:E    #H'C,R0        ;R0=CH
128 GRA   C E12C 157FF10090          MOVTPE   R0,@H'7FF1     ;CH TO 7FF1
129 GRA   C E131 5100                MOV:E    #H'0,R1        ;R1=00H=GRAPHIC BYTE #1
130 GRA   C E133 157FF00091          MOVTPE   R1,@H'7FF0     ;0H TO 7FF0
131 GRA   C E138 00        NOP
132 GRA   C E139 157FF10084    X10:  MOVFPE   @H'7FF1,R4     ;READ 7FF1 DATA TO R4
133 GRA   C E13E ACF7                BTST     #7,R4          ;BIT TEST #7 OF R4
134 GRA   C E140 26F7                BNE      X10            ;IF B/FLAG =Z=1 GO TO X10
135 GRA   C E142 00        NOP                          ;B/FLAG NOT SET
136 GRA   C E143 157FF10090          MOVTPE   R0,@H'7FF1     ;CH TO 7FF1
137 GRA   C E148 51FF                MOV:E    #H'FF,R1   ;R1=FF=GRAPHIC BYTE #2
138 GRA   C E14A 157FF00091          MOVTPE   R1,@H'7FF0     ;FF TO 7FF0
139 GRA   C E14F 00        NOP
140 GRA   C E150 157FF10084    X11:  MOVFPE   @H'7FF1,R4     ;READ 7FF1 DATA TO R4
141 GRA   C E155 ACF7                BTST     #7,R4          ;BIT TEST #7 OF R4
142 GRA   C E157 26F7                BNE      X11            ;IF B/FLAG =Z=1 GO TO X11
143 GRA   C E159 00        NOP                          ;B/FLAG NOT SET
144 GRA   C E15A 157FF10090          MOVTPE   R0,@H'7FF1     ;CH TO 7FF1
145 GRA   C E15F 5166                MOV:E    #H'66,R1   ;R1=66=GRAPHIC BYTE #3
146 GRA   C E161 157FF00091          MOVTPE   R1,@H'7FF0     ;66 TO 7FF0
147 GRA   C E166 00        NOP
148 GRA   C E167 157FF10084    X12:  MOVFPE   @H'7FF1,R4     ;READ 7FF1 DATA TO R4
149 GRA   C E16C ACF7                BTST     #7,R4          ;BIT TEST #7 OF R4
150 GRA   C E16E 26F7                BNE      X12            ;IF B/FLAG =Z=1 GO TO X12
151 GRA   C E170 00        NOP                          ;B/FLAG NOT SET
152 GRA   C E171 157FF10090          MOVTPE   R0,@H'7FF1     ;CH TO 7FF1
153 GRA   C E176 5177                MOV:E    #H'77,R1   ;R1=77=GRAPHIC BYTE #4
154 GRA   C E178 157FF00091          MOVTPE   R1,@H'7FF0     ;77 TO 7FF0
155 GRA   C E17D 00        NOP
156 GRA   C E17E 157FF10084    X13:  MOVFPE   @H'7FF1,R4     ;READ 7FF1 DATA TO R4
157 GRA   C E183 ACF7                BTST     #7,R4          ;BIT TEST #7 OF R4
158 GRA   C E185 26F7                BNE      X13            ;IF B/FLAG =Z=1 GO TO X13
159 GRA   C E187 00        NOP                          ;B/FLAG NOT SET
160 GRA   C E188 157FF10093          MOVTPE   R3,@H'7FF1     ;0H TO 7FF1
161 GRA   C E18D 5132                MOV:E    #H'32,R1   ;LOAD R1=32H=DISP=ON
162 GRA   C E18F 157FF00091          MOVTPE   R1,@H'7FF0     ;32H TO 7FF0
163
164 GRA   C E194 00        NOP                  ; DISPLAY DOT LIGHT = LOGIC "0"
165 GRA   C E195 00        NOP                  ; DISPLAY DOT DARK  = LOGIC "1"
166
167 GRA   C E196 00        NOP
168 GRA   C E197 00        NOP
169 GRA   C E198 1A        SLEEP                          ;H8/532 ASLEEP
170 GRA   C E199 00        NOP
171
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

```
58 GRA   C E083 26F7                BNE       X6                ;IF B/FLAG =Z=1 GO TO X6
59 GRA   C E085 00        NOP                                   ;B/FLAG NOT SET
60 GRA   C E086 5009                MOV:E     #H'9,R0           ;LOAD R0=9H
61 GRA   C E088 157FF10090          MOVTPE    R0,@H'7FF1        ;9H TO 7FF1
62 GRA   C E08D 157FF00092          MOVTPE    R2,@H'7FF0        ;0H TO 7FF0
63 GRA   C E092 00        NOP
64                                  ; SCREEN CLEAR ROUTINE START
65
66 GRA   C E093 AD13                CLR.W     R5                ; CLEAR R5
67 GRA   C E095 00        NOP
68 GRA   C E096 157FF10084    C1:   MOVFPE    @H'7FF1,R4        ;READ 7FF1 DATA TO R4
69 GRA   C E09B ACF7                BTST      #7,R4             ;BIT TEST #7 OF R4
70 GRA   C E09D 26F7                BNE       C1                ;IF B/FLAG =Z=1 GO TO C1
71 GRA   C E09F 00        NOP                                   ;B/FLAG NOT SET
72 GRA   C E0A0 500A                MOV:E     #H'A,R0           ;R0=AH
73 GRA   C E0A2 157FF10090          MOVTPE    R0,@H'7FF1        ;AH TO 7FF1
74 GRA   C E0A7 5100                MOV:E     #H'0,R1           ;R1=0H
75 GRA   C E0A9 157FF00091          MOVTPE    R1,@H'7FF0        ;0H TO 7FF0-CUR L/B=0H
76 GRA   C E0AE 00        NOP
77 GRA   C E0AF 157FF10084    C2:   MOVFPE    @H'7FF1,R4        ;READ 7FF1 DATA TO R4
78 GRA   C E0B4 ACF7                BTST      #7,R4             ;BIT TEST #7 OF R4
79 GRA   C E0B6 26F7                BNE       C2                ;IF B/FLAG =Z=1 GO TO C2
80 GRA   C E0B8 00        NOP                                   ;B/FLAG NOT SET
81 GRA   C E0B9 500B                MOV:E     #H'B,R0           ;R0=BH
82 GRA   C E0BB 157FF10090          MOVTPE    R0,@H'7FF1        ;BH TO 7FF1
83 GRA   C E0C0 5100                MOV:E     #H'0,R1           ;R1=0H
84 GRA   C E0C2 157FF00091          MOVTPE    R1,@H'7FF0        ;0H TO 7FF0-CUR H/B=0H
85 GRA   C E0C7 00        NOP
86 GRA   C E0C8 5DFFFF              MOV:I     #H'FFFF,R5;COUNT=R5=FFFFH
87 GRA   C E0CB 00        C3:       NOP
88 GRA   C E0CC 157FF10084    C4:   MOVFPE    @H'7FF1,R4        ;READ 7FF1 DATA TO R4
89 GRA   C E0D1 ACF7                BTST      #7,R4             ;BIT TEST #7 OF R4
90 GRA   C E0D3 26F7                BNE       C4                ;IF B/FLAG =Z=1 GO TO C4
91 GRA   C E0D5 00        NOP                                   ;B/FLAG NOT SET
92 GRA   C E0D6 500C                MOV:E     #H'C,R0           ;R0=CH
93 GRA   C E0D8 157FF10090          MOVTPE    R0,@H'7FF1        ;CH TO 7FF1
94 GRA   C E0DD 5100                MOV:E     #H'0,R1           ;R1=0H=CODE FOR "DOT OFF"
95 GRA   C E0DF 157FF00091          MOVTPE    R1,@H'7FF0        ;0H TO 7FF0
96 GRA   C E0E4 00        NOP
97 GRA   C E0E5 01BDE3              SCB/F     R5,C3
98 GRA   C E0E8 00        NOP
99                                  ; SCREEN CLEAR ROUTINE COMPLETED
100 GRA  C E0E9 00        NOP
101 GRA  C E0EA 00                  NOP
102 GRA  C E0EB 157FF10084    X7:   MOVFPE    @H'7FF1,R4        ;READ 7FF1 DATA TO R4
103 GRA  C E0F0 ACF7                BTST      #7,R4             ;BIT TEST #7 OF R4
104 GRA  C E0F2 26F7                BNE       X7                ;IF B/FLAG =Z=1 GO TO X7
105 GRA  C E0F4 00        NOP                                   ;B/FLAG NOT SET
106 GRA  C E0F5 00        NOP
107 GRA  C E0F6 00        NOP                                   ;INITIALIZATION DONE
108 GRA  C E0F7 00        NOP
109 GRA  C E0F8 500A                MOV:E     #H'A,R0           ;R0=AH
110 GRA  C E0FA 157FF10090          MOVTPE    R0,@H'7FF1        ;AH TO 7FF1
111 GRA  C E0FF 51B0                MOV:E     #H'B0,R1  ;R1=B0H
112 GRA  C E101 157FF00091          MOVTPE    R1,@H'7FF0        ;0H TO 7FF0
113 GRA  C E106 00        NOP
114 GRA  C E107 157FF10084    X8:   MOVFPE    @H'7FF1,R4        ;READ 7FF1 DATA TO R4
```

SECTION 4

```
1                       .HEADING  "GRA-BCS"
2 GRA   C 0000          .SECTION  GRA,CODE,ALIGN=2
3                       .EXPORT   X
4 GRA   C E000          .ORG      H'E000          ;LOC CNTR =E000H
5
6                                 ; BUSY FLAG CHECKED
7
8 GRA   C   0000E000    X:        .EQU            $               ;X = E000H
9 GRA   C E000 A013     CLR.B     R0              ;CLEAR R0
10 GRA  C E002 A113     CLR.B     R1              ;CLEAR R1
11 GRA  C E004 A213     CLR.B     R2              ;CLEAR R2
12 GRA  C E006 A313     CLR.B     R3              ;CLEAR R3
13 GRA  C E008 A413     CLR.B     R4              ;CLEAR R4
14 GRA  C E00A 00       NOP
15 GRA  C E00B 00       NOP       ; INITIALIZATION START
16 GRA  C E00C 00       NOP
17 GRA  C E00D 157FF10090       MOVTPE  R0,@H'7FF1      ;0H TO 7FF1
18 GRA  C E012 5112             MOV:E   #H'12,R1  ;LOAD R1 =12H
19 GRA  C E014 157FF00091       MOVTPE  R1,@H'7FF0      ;12H TO 7FF0
20 GRA  C E019 00       NOP
21 GRA  C E01A 157FF10084  X1:  MOVFPE  @H'7FF1,R4      ;READ 7FF1 DATA TO R4
22 GRA  C E01F ACF7             BTST    #7,R4           ;BIT TEST #7 OF R4
23 GRA  C E021 26F7             BNE     X1              ;IF B/FLAG =Z=1 GO TO X1
24 GRA  C E023 00       NOP                     ;B/FLAG NOT SET
25 GRA  C E024 5001             MOV:E   #H'1,R0         ;LOAD R0=1H
26 GRA  C E026 157FF10090       MOVTPE  R0,@H'7FF1      ;1H TO 7FF1
27 GRA  C E02B 5107             MOV:E   #H'7,R1         ;LOAD R1=7H
28 GRA  C E02D 157FF00091       MOVTPE  R1,@H'7FF0      ;7H TO 7FF0
29 GRA  C E032 00       NOP
30 GRA  C E033 157FF10084  X2:  MOVFPE  @H'7FF1,R4      ;READ 7FF1 DATA TO R4
31 GRA  C E038 ACF7             BTST    #7,R4           ;BIT TEST #7 OF R4
32 GRA  C E03A 26F7             BNE     X2              ;IF B/FLAG =Z=1 GO TO X2
33 GRA  C E03C 00       NOP                     ;B/FLAG NOT SET
34 GRA  C E03D 5002             MOV:E   #H'2,R0         ;LOAD R0=2H
35 GRA  C E03F 157FF10090       MOVTPE  R0,@H'7FF1      ;2H TO 7FF1
36 GRA  C E044 511D             MOV:E   #H'1D,R1  ;LOAD R1=1DH
37 GRA  C E046 157FF00091       MOVTPE  R1,@H'7FF0      ;1DH TO 7FF0
38 GRA  C E04B 00       NOP
39 GRA  C E04C 157FF10084  X3:  MOVFPE  @H'7FF1,R4      ;READ 7FF1 DATA TO R4
40 GRA  C E051 ACF7             BTST    #7,R4           ;BIT TEST #7 OF R4
41 GRA  C E053 26F7             BNE     X3              ;IF B/FLAG =Z=1 GO TO X3
42 GRA  C E055 00       NOP                     ;B/FLAG NOT SET
43 GRA  C E056 5003             MOV:E   #H'3,R0         ;LOAD R0=3H
44 GRA  C E058 157FF10090       MOVTPE  R0,@H'7FF1      ;3H TO 7FF1
45 GRA  C E05D 511F             MOV:E   #H'1F,R1  ;LOAD R1=1FH
46 GRA  C E05F 157FF00091       MOVTPE  R1,@H'7FF0      ;1FH TO 7FF0
47 GRA  C E064 00       NOP
48 GRA  C E065 157FF10084  X4:  MOVFPE  @H'7FF1,R4      ;READ 7FF1 DATA TO R4
49 GRA  C E06A ACF7             BTST    #7,R4           ;BIT TEST #7 OF R4
50 GRA  C E06C 26F7             BNE     X4              ;IF B/FLAG =Z=1 GO TO X4
51 GRA  C E06E 00       NOP                     ;B/FLAG NOT SET
52 GRA  C E06F 5008             MOV:E   #H'8,R0         ;R0=8H
53 GRA  C E071 157FF10090       MOVTPE  R0,@H'7FF1      ;8H TO 7FF1
54 GRA  C E076 157FF00092       MOVTPE  R2,@H'7FF0      ;0H TO 7FF0
55 GRA  C E07B 00       NOP
56 GRA  C E07C 157FF10084  X6:  MOVFPE  @H'7FF1,R4      ;READ 7FF1 DATA TO R4
57 GRA  C E081 ACF7             BTST    #7,R4           ;BIT TEST #7 OF R4
```

APPENDIX " B "

1.0 PROGRAM NAME - " GRA-BCS.MOT "

2.0 ADDRESS RANGE - " E000H - E199H "

3.0 PROGRAM DESCRIPTION - CLEARS SCREEN, CHECKS BUSY FLAG, AND DIS PLAYS 4 GRAPHIC BYTES ON THE LCD LM200 PANEL STARTING AT THE 1200TH CURSOR POSITION.

*** SECTION DATA LIST

SECTION	ATTRIBUTE	SIZE	START
CHR	REL-CODE	0C1AF	

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

*** CROSS REFERENCE LIST

NAME	SECTION	ATTR	VALUE	SEQUENCE
A	CHR	EXPT	0000C000	3 8*
C1	CHR		0000C0B0	79* 81
C2	CHR		0000C0C9	88* 90
C3	CHR		0000C0E5	98* 108
C4	CHR		0000C0E6	99* 101
CHR	CHR	SCT	00000000	2*
X1	CHR		0000C01B	23* 25
X10	CHR		0000C151	143* 145
X11	CHR		0000C168	151* 153
X12	CHR		0000C17F	159* 161
X13	CHR		0000C196	167* 169
X2	CHR		0000C034	32* 34
X3	CHR		0000C04D	41* 43
X4	CHR		0000C066	50* 52
X5	CHR		0000C07F	59* 61
X6	CHR		0000C096	67* 69
X7	CHR		0000C104	113* 115
X8	CHR		0000C121	126* 128
X9	CHR		0000C138	134* 136

SECTION 4

```
172 CHR   C C1A5 513C          MOV.E      #H'3C,R1   ;LOAD R1=3CH=DISP=ON
173 CHR   C C1A7 157FF00091    MOVTPE     R1,@H'7FF0        ;3CH TO 7FF0
174
175 CHR   C C1AC 00            NOP
176 CHR   C C1AD 00            NOP
177
178 CHR   C C1AE 1A            SLEEP                        ; PROCESSOR H8/532
        ; SLEEP
179
180                    .END
*****TOTAL ERRORS    0
*****TOTAL WARNINGS  0
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

```
115 CHR   C C10B 26F7                BNE       X7                ;IF B/FLAG =Z=1 GO TO X7
116 CHR   C C10D 00                  NOP                         ;B/FLAG NOT SET
117 CHR   C C10E 00          NOP
118 CHR   C C10F 00          NOP                 ; INITIALIZATION END
119 CHR   C C110 00          NOP
120 CHR   C C111 00          NOP
121 CHR   C C112 500A                MOV:E     #H'A,R0           ;R0=AH
122 CHR   C C114 157FF10090          MOVTPE    R0,@H'7FF1        ;AH TO 7FF1
123 CHR   C C119 5108                MOV:E     #H'8,R1           ;R1=8H
124 CHR   C C11B 157FF00091          MOVTPE    R1,@H'7FF0        ;8H TO 7FF0
125 CHR   C C120 00          NOP
126 CHR   C C121 157FF10084    X8:   MOVFPE    @H'7FF1,R4        ;READ 7FF1 DATA TO R4
127 CHR   C C126 ACF7                BTST      #7,R4             ;BIT TEST #7 OF R4
128 CHR   C C128 26F7                BNE       X8                ;IF B/FLAG =Z=1 GO TO X8
129 CHR   C C12A 00          NOP                         ;B/FLAG NOT SET
130 CHR   C C12B 500B                MOV:E     #H'B,R0           ;R0=BH
131 CHR   C C12D 157FF10090          MOVTPE    R0,@H'7FF1        ;BH TO 7FF1
132 CHR   C C132 157FF00092          MOVTPE    R2,@H'7FF0        ;0H TO 7FF0
133 CHR   C C137 00          NOP
134 CHR   C C138 157FF10084    X9:   MOVFPE    @H'7FF1,R4        ;READ 7FF1 DATA TO R4
135 CHR   C C13D ACF7                BTST      #7,R4             ;BIT TEST #7 OF R4
136 CHR   C C13F 26F7                BNE       X9                ;IF B/FLAG =Z=1 GO TO X9
137 CHR   C C141 00          NOP                         ;B/FLAG NOT SET
138 CHR   C C142 500C                MOV:E     #H'C,R0           ;R0=CH
139 CHR   C C144 157FF10090          MOVTPE    R0,@H'7FF1        ;CH TO 7FF1
140 CHR   C C149 514B                MOV:E     #H'4B,R1    ;R1=4B=CODE FOR "K"
141 CHR   C C14B 157FF00091          MOVTPE    R1,@H'7FF0        ;4B TO 7FF0
142 CHR   C C150 00          NOP
143 CHR   C C151 157FF10084    X10:  MOVFPE    @H'7FF1,R4        ;READ 7FF1 DATA TO R4
144 CHR   C C156 ACF7                BTST      #7,R4             ;BIT TEST #7 OF R4
145 CHR   C C158 26F7                BNE       X10               ;IF B/FLAG =Z=1 GO TO X10
146 CHR   C C15A 00          NOP                         ;B/FLAG NOT SET
147 CHR   C C15B 157FF10090          MOVTPE    R0,@H'7FF1        ;CH TO 7FF1
148 CHR   C C160 5141                MOV:E     #H'41,R1    ;R1=41=CODE FOR "A"
149 CHR   C C162 157FF00091          MOVTPE    R1,@H'7FF0        ;41 TO 7FF0
150 CHR   C C167 00          NOP
151 CHR   C C168 157FF10084    X11:  MOVFPE    @H'7FF1,R4        ;READ 7FF1 DATA TO R4
152 CHR   C C16D ACF7                BTST      #7,R4             ;BIT TEST #7 OF R4
153 CHR   C C16F 26F7                BNE       X11               ;IF B/FLAG =Z=1 GO TO X11
154 CHR   C C171 00          NOP                         ;B/FLAG NOT SET
155 CHR   C C172 157FF10090          MOVTPE    R0,@H'7FF1        ;CH TO 7FF1
156 CHR   C C177 5153                MOV:E     #H'53,R1    ;R1=53=CODE FOR "S"
157 CHR   C C179 157FF00091          MOVTPE    R1,@H'7FF0        ;53 TO 7FF0
158 CHR   C C17E 00          NOP
159 CHR   C C17F 157FF10084    X12:  MOVFPE    @H'7FF1,R4        ;READ 7FF1 DATA TO R4
160 CHR   C C184 ACF7                BTST      #7,R4             ;BIT TEST #7 OF R4
161 CHR   C C186 26F7                BNE       X12               ;IF B/FLAG =Z=1 GO TO X12
162 CHR   C C188 00          NOP                         ;B/FLAG NOT SET
163 CHR   C C189 157FF10090          MOVTPE    R0,@H'7FF1        ;CH TO 7FF1
164 CHR   C C18E 5148                MOV:E     #H'48,R1    ;R1=48=CODE FOR "H"
165 CHR   C C190 157FF00091          MOVTPE    R1,@H'7FF0        ;48 TO 7FF0
166 CHR   C C195 00          NOP
167 CHR   C C196 157FF10084    X13:  MOVFPE    @H'7FF1,R4        ;READ 7FF1 DATA TO R4
168 CHR   C C19B ACF7                BTST      #7,R4             ;BIT TEST #7 OF R4
169 CHR   C C19D 26F7                BNE       X13               ;IF B/FLAG =Z=1 GO TO X13
170 CHR   C C19F 00          NOP                         ;B/FLAG NOT SET
171 CHR   C C1A0 157FF10093          MOVTPE    R3,@H'7FF1        ;0H TO 7FF1
```

SECTION 4

```
58 CHR   C 07E 00              NOP
59 CHR   C 07F 157FF10084   X5:  MOVFPE    @H'7FF1,R4     ;READ 7FF1 DATA TO R4
60 CHR   C 084 ACF7             BTST      #7,R4          ;BIT TEST #7 OF R4
61 CHR   C 086 26F7             BNE       X5             ;IF B/FLAG =Z=1 GO TO X5
62 CHR   C 088 00              NOP                       ;B/FLAG NOT SET
63 CHR   C 089 5008             MOV:E     #H'8,R0        ;R0=8H
64 CHR   C 08B 157FF10090       MOVTPE    R0,@H'7FF1     ;8H TO 7FF1
65 CHR   C 090 157FF00092       MOVTPE    R2,@H'7FF0     ;0H TO 7FF0
66 CHR   C 095 00              NOP
67 CHR   C 096 157FF10084   X6:  MOVFPE    @H'7FF1,R4     ;READ 7FF1 DATA TO R4
68 CHR   C 09B ACF7             BTST      #7,R4          ;BIT TEST #7 OF R4
69 CHR   C 09D 26F7             BNE       X6             ;IF B/FLAG =Z=1 GO TO X6
70 CHR   C 09F 00              NOP                       ;B/FLAG NOT SET
71 CHR   C 0A0 5009             MOV:E     #H'9,R0        ;LOAD R0=9H
72 CHR   C 0A2 157FF10090       MOVTPE    R0,@H'7FF1     ;9H TO 7FF1
73 CHR   C 0A7 157FF00092       MOVTPE    R2,@H'7FF0     ;0H TO 7FF0
74 CHR   C 0AC 00              NOP
75                                        ; SCREEN CLEAR ROUTINE START
76
77 CHR   C 0AD AD13             CLR.W     R5             ; CLEAR R5
78 CHR   C 0AF 00              NOP
79 CHR   C 0B0 157FF10084   C1:  MOVFPE    @H'7FF1,R4     ;READ 7FF1 DATA TO R4
80 CHR   C 0B5 ACF7             BTST      #7,R4          ;BIT TEST #7 OF R4
81 CHR   C 0B7 26F7             BNE       C1             ;IF B/FLAG =Z=1 GO TO C1
82 CHR   C 0B9 00              NOP                       ;B/FLAG NOT SET
83 CHR   C 0BA 500A             MOV:E     #H'A,R0        ;R0=AH
84 CHR   C 0BC 157FF10090       MOVTPE    R0,@H'7FF1     ;AH TO 7FF1
85 CHR   C 0C1 5100             MOV:E     #H'0,R1        ;R1=0H
86 CHR   C 0C3 157FF00091       MOVTPE    R1,@H'7FF0     ;0H TO 7FF0-CUR L/B=0H
87 CHR   C 0C8 00              NOP
88 CHR   C 0C9 157FF10084   C2:  MOVFPE    @H'7FF1,R4     ;READ 7FF1 DATA TO R4
89 CHR   C 0CE ACF7             BTST      #7,R4          ;BIT TEST #7 OF R4
90 CHR   C 0D0 26F7             BNE       C2             ;IF B/FLAG =Z=1 GO TO C2
91 CHR   C 0D2 00              NOP                       ;B/FLAG NOT SET
92 CHR   C 0D3 500B             MOV:E     #H'B,R0        ;R0=BH
93 CHR   C 0D5 157FF10090       MOVTPE    R0,@H'7FF1     ;BH TO 7FF1
94 CHR   C 0DA 5100             MOV:E     #H'0,R1        ;R1=0H
95 CHR   C 0DC 157FF00091       MOVTPE    R1,@H'7FF0     ;0H TO 7FF0-CUR H/B=0H
96 CHR   C 0E1 00              NOP
97 CHR   C 0E2 5D01FF           MOV:I     #H'1FF,R5  ;COUNT=R5=1FFH
98 CHR   C 0E5 00          C3:  NOP
99 CHR   C 0E6 157FF10084   C4:  MOVFPE    @H'7FF1,R4     ;READ 7FF1 DATA TO R4
100 CHR  C 0EB ACF7             BTST      #7,R4          ;BIT TEST #7 OF R4
101 CHR  C 0ED 26F7             BNE       C4             ;IF B/FLAG =Z=1 GO TO C4
102 CHR  C 0EF 00               NOP                      ;B/FLAG NOT SET
103 CHR  C 0F0 500C             MOV:E     #H'C,R0        ;R0=CH
104 CHR  C 0F2 157FF10090       MOVTPE    R0,@H'7FF1     ;CH TO 7FF1
105 CHR  C 0F7 5120             MOV:E     #H'20,R1   ;R1=20H=CODE FOR "BLANK"
106 CHR  C 0F9 157FF00091       MOVTPE    R1,@H'7FF0     ;20H TO 7FF0
107 CHR  C 0FE 00               NOP
108 CHR  C 0FF 01BDE3           SCB/F     R5,C3
109 CHR  C 102 00              NOP
110                                        ; SCREEN CLEAR ROUTINE COMPLETED
111 CHR  C 103 00              NOP
112
113 CHR  C 104 157FF10084   X7:  MOVFPE    @H'7FF1,R4     ;READ 7FF1 DATA TO R4
114 CHR  C 109 ACF7             BTST      #7,R4          ;BIT TEST #7 OF R4
```

```
1                       .HEADING  "CHR-BCS"
2 CHR   C 0000          .SECTION  CHR,CODE,ALIGN=2
3                       .EXPORT           A
4 CHR   C C000          .ORG              H'C000              ;LOC CNTR =C000H
5
6                                         ; BUSY FLAG CHECKED
7
8 CHR   C    0000C000   A:       .EQU            $                   ;A = C000H
9 CHR   C C000 A013     CLR.B             R0                  ;CLEAR R0
10 CHR  C C002 A113             CLR.B           R1                  ;CLEAR R1
11 CHR  C C004 A213             CLR.B           R2                  ;CLEAR R2
12 CHR  C C006 A313             CLR.B           R3                  ;CLEAR R3
13 CHR  C C008 A413             CLR.B           R4                  ;CLEAR R4
14
15 CHR  C C00A 00       NOP
16 CHR  C C00B 00       NOP               ; INITIALIATION START
17 CHR  C C00C 00       NOP
18 CHR  C C00D 00       NOP
19 CHR  C C00E 157FF10090       MOVTPE          R0,@H'7FF1          ;0H TO 7FF1
20 CHR  C C013 511C             MOV.E           #H'1C,R1    ;LOAD R1 = CH
21 CHR  C C015 157FF00091       MOVTPE          R1,@H'7FF0          ;1CH TO 7FF0
22 CHR  C C01A 00       NOP
23 CHR  C C01B 157FF10084       X1:     MOVFPE          @H'7FF1,R4          ;READ 7FF1 DATA TO R4
24 CHR  C C020 ACF7             BTST            #7,R4               ;BIT TEST #7 OF R4
25 CHR  C C022 26F7             BNE             X1                  ;IF B/FLAG =Z=1 GO TO X1
26 CHR  C C024 00       NOP                                     ;B/FLAG NOT SET
27 CHR  C C025 5001             MOV.E           #H'1,R0             ;LOAD R0=1H
28 CHR  C C027 157FF10090       MOVTPE          R0,@H'7FF1          ;1H TO 7FF1
29 CHR  C C02C 5195             MOV.E           #H'95,R1    ;LOAD R1=95H
30 CHR  C C02E 157FF00091       MOVTPE          R1,@H'7FF0          ;95H TO 7FF0
31 CHR  C C033 00       NOP
32 CHR  C C034 157FF10084       X2:     MOVFPE          @H'7FF1,R4          ;READ 7FF1 DATA TO R4
33 CHR  C C039 ACF7             BTST            #7,R4               ;BIT TEST #7 OF R4
34 CHR  C C03B 26F7             BNE             X2                  ;IF B/FLAG =Z=1 GO TO X2
35 CHR  C C03D 00       NOP                                     ;B/FLAG NOT SET
36 CHR  C C03E 5002             MOV.E           #H'2,R0             ;LOAD R0=2H
37 CHR  C C040 157FF10090       MOVTPE          R0,@H'7FF1          ;2H TO 7FF1
38 CHR  C C045 5127             MOV.E           #H'27,R1    ;LOAD R1=27H
39 CHR  C C047 157FF00091       MOVTPE          R1,@H'7FF0          ;27H TO 7FF0
40 CHR  C C04C 00       NOP
41 CHR  C C04D 157FF10084       X3:     MOVFPE          @H'7FF1,R4          ;READ 7FF1 DATA TO R4
42 CHR  C C052 ACF7             BTST            #7,R4               ;BIT TEST #7 OF R4
43 CHR  C C054 26F7             BNE             X3                  ;IF B/FLAG =Z=1 GO TO X3
44 CHR  C C056 00       NOP                                     ;B/FLAG NOT SET
45 CHR  C C057 5003             MOV.E           #H'3,R0             ;LOAD R0=3H
46 CHR  C C059 157FF10090       MOVTPE          R0,@H'7FF1          ;3H TO 7FF1
47 CHR  C C05E 511F             MOV.E           #H'1F,R1    ;LOAD R1=1FH
48 CHR  C C060 157FF00091       MOVTPE          R1,@H'7FF0          ;1FH TO 7FF0
49 CHR  C C065 00       NOP
50 CHR  C C068 157FF10084       X4:     MOVFPE          @H'7FF1,R4          ;READ 7FF1 DATA TO R4
51 CHR  C C06B ACF7             BTST            #7,R4               ;BIT TEST #7 OF R4
52 CHR  C C06D 26F7             BNE             X4                  ;IF B/FLAG =Z=1 GO TO X4
53 CHR  C C06F 00       NOP                                     ;B/FLAG NOT SET
54 CHR  C C070 5004             MOV.E           #H'4,R0             ;LOAD R0=4H
55 CHR  C C072 157FF10090       MOVTPE          R0,@H'7FF1          ;4H TO 7FF1
56 CHR  C C077 5108             MOV.E           #H'8,R1             ;LOAD R1=8H
57 CHR  C C079 157FF00091       MOVTPE          R1,@H'7FF0          ;8H TO 7FF0
```

SECTION 4

APPENDIX " A "

1.0 PROGRAM NAME - " CHR-BCS.MOT "

2.0 ADDRESS RANGE - " C000H - C1AEH "

3.0 PROGRAM DESCRIPTION - CLEARS SCREEN, CHECKS BUSY FLAG, AND DIS PLAYS 4 LETTERS " KASH " ON THE LCD LM200 PANEL STARTING AT THE 8TH CURSOR POSITION.

7.4.4 "S" Record Conversion : To convert a file to "S" record format execute the command shown ;

o Enter CNVS.EXE KY.ABS (The extentions are not necessary)

As a result of the conversion process an ASCII coded file is created with "*.MOT" extention. In this case, it will be KY.MOT.

7.4.5 To Up-load a File : Change directory to "PROCOMM" i.e. C:\PROCOMM. Then execute the following commands ;

- o Enter - " PROCOMM " (Load the ' PROCOMM ' package)
- o Set the serial communication line to 9600 Baud, NPTY, 8 Bits, 1 STOP Bit, Full Duplex, with ASCII code.
- o At the HMS > prompt, enter - TL (For Terminal Load)
- o PUSH down " Shift/Edit " key. Press " Page Up" key.
- o The list of upload protocols appears.
- o PUSH up the " Shift/Edit " key.
- o Choose " 7 " for " ASCII " protocol.
- o Enter the file name that should be up-loaded in the window e.g . C:\H8-500\ASM\KY.MOT
- o As the data transmission continues, oserve the increasing line numbers in the status line along with other line settings.
- o At the end of data transmission, Address Range, followed by the HMS > prompt will appear on the screen, if there are no line errors. e.g. . . . . .

  Address Range C000 - C1AE
  HMS >

NOTE:

1.0 If there are any transmission errors, hit reset switch "SW3".
2.0 At the HMS > Prompt, Enter "TL" and repeat the upload process.

7.4.6 To run the " KY.MOT " program from the address range shown above, using the Hitachi Monitor System (HMS) on the H8/532 Evaluation Board, execute the following command ;

HMS > G C000 (Return) . . . [Refer to H8/532 Eval. Board Software Manual for more details].

7.4.7 To run another program, push NMI switch "SW2" on the H8/532Eval. Board , and at the HMS > prompt, enter "TL" and upload a new ASCII file.

7.4 Code Assembly Procedure :

Software code development procedure for this application is described in greater detail in this sub-section. The development tools and other commercially available packages used in this project are briefly addressed. **No** attempt is made to describe these packages in detail. Please, refer to their User manuals when in doubt. A working knowledge of the MS "WORD", "PROCOMM", and MSDOS in the PC environment is assumed in describing this procedure. Multiple code development stations were built with **identical** tool environment and allow the execution of this procedure.

7.4.1 User program :

The source code is written in assembly language for the H8/532 micro processor. The data is entered at location counter H'C000 and a Microsoft "WORD" file is created with " *.DOC " extension under MSDOS operat ing system for the laptop PC. This file is copied to the C:\H8-500\ASM directory with "*.SRC" extension e.g. KY.SRC.

Note : 1.0 The "*.DOC " file should be **unformatted** .
2.0 User program code space may vary from H'8000 to H'F000.

7.4.2 Invoke Assembler :

To invoke the assembler the following steps are recommended ;

o Change to the directory - C:\H8-500\ASM
o Enter - H8ASM KY.SRC (The extension is not necessary)

As a result of the assembly process, " KY.LIS and KY.OBJ " files are created. Also, the number and types of assembly errors are indicated.
If the number of errors, exeeds 0, then go back to "WORD" and examine the "KY.LIS" file to see where the errors were made. Find the correspond ing errors in the source code file, and correct them. This process may have to be repeated many times until all the assembly errors are removed.

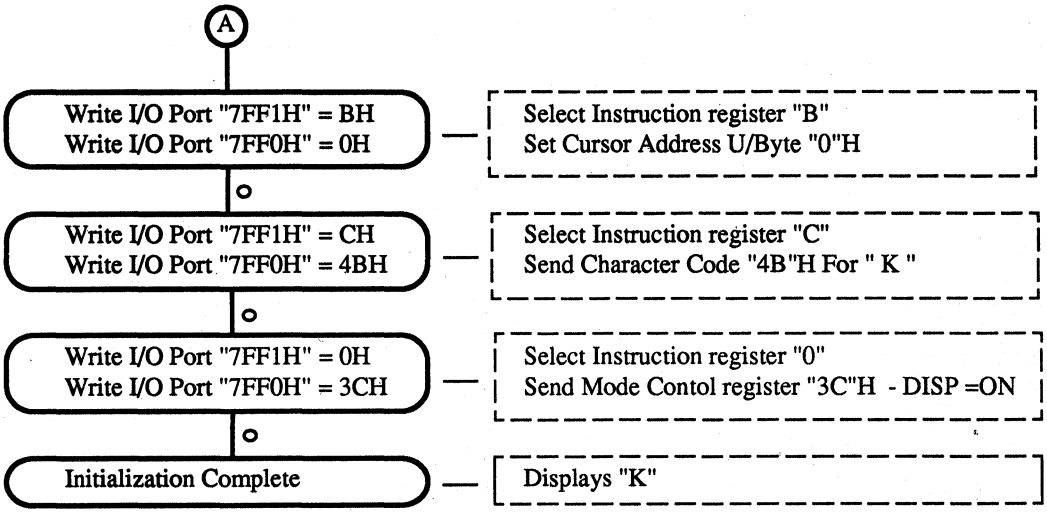
7.4.3 Linker :

When there are no errors in the assembly process, generally, a hard copy of the "*.LIS" file is made for software documentation process. By providing adequate comments in the source file, software debug process is made easier.
The multiple object files are then linked using the linker. To link a file execute the folowing command in the C:\H8-500\ASM directory ;

o Enter - LNK KY.OBJ (The exension is not necessary)

The linking process generates the file " KY.ABS ". It is to be converted to the Motorola "S" record format for up-loading to the H8/532 Evaluation Board for execution.

7.3 Initialization Flow Chart : (CNT'D.)



SECTION 4

NOTE :

1.0 Busy Flag Check code sequence is indicated by " o "

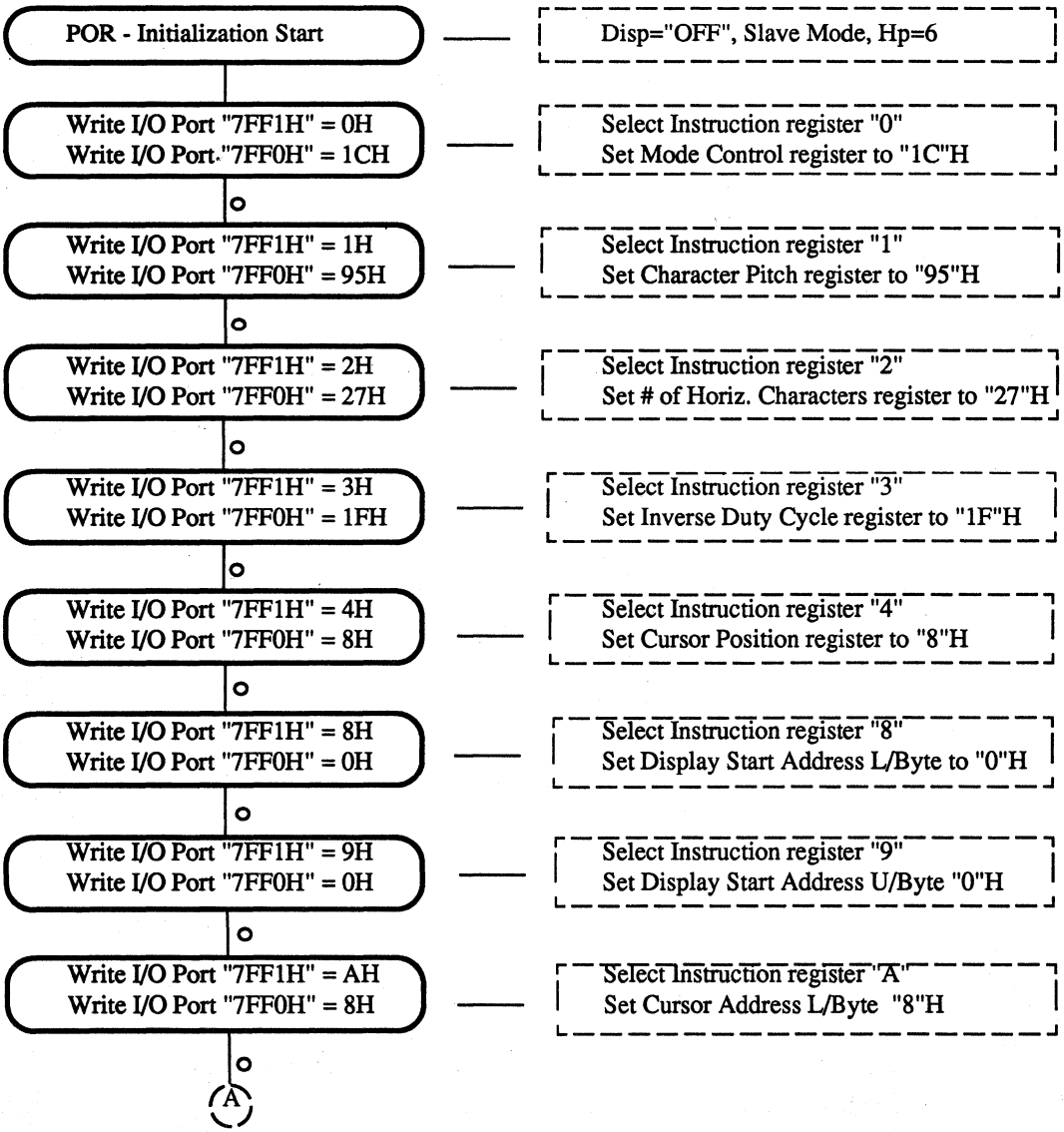
2.0 For code details in Graphics or Character mode, see the Appendices.

3.0 After Initialization, normally the display is cleared by writing character code for a blank "20 H" in the display buffer memory or by writing "0" in the graphics memory space. If the LCD screen is not cleared, power on random memory data will be displayed.

4.0 For programming details, refer to the HD61830B data sheet.

7.3 Initialization Flow Chart :

Power on initialization flow chart is shown in this sub-section. The consecutive I/O instructions and their brief description is also listed.



7.0 SOFTWARE :

Software section covers I/O addressing, Busy Flag Varification, Initialization Flow Chart, Code Assembly Procedure while Appendices show the program listings.

7.1 I/O Addresss :

Referring to the H8/532 Evaluation Board Hardware Manual From Hitachi Micro Systems Inc. (HMSI), the Expansion Bus I/O space is located from H'7FF0 to H'7FFF for expanded minimum mode 1 memory (64 KBytes) space. Read the H8/532 documentation for more details. On the LCD Interface Board this space is further decoded. "MOVTPE" and "MOVFPE" instructions to or from the I/O addresses H'7FF0 or H'7FF1 are used for data transfers.
These I/O addresses are memory mapped. The first write data to address H'7FF1 specifies one of the 13 instruction registers inside the HD61830B. It is followed by a second write to the I/O address H'7FF0 which sends the data to the data input regiser inside the HD61830B. Therefore, two sequential peripheral write commands are required. Fore more details, see the HD61830B data sheet.
Similarly, reading the I/O address H'7FF1, allows the programer to check the Busy Flag before sending a a second instruction when the first instruction is being processed by the HD61830B LCD Controller. In the same manner, reading the I/O address H'7FF0, provides the programer output data at the current cursor address. The I/O address (Hex) table for read or write operation is shown below :

I/O ADD.	OPERATION READ/WRITE	DATA BUS D7	D6	D5	D4	D3	D2	D1	D0
7FF1	Write Instruction Reg. Bits (I3-I0)	0	0	0	0	I3	I2	I1	I0
7FF1	Read Busy Flag (B/F))	B/F	D	D	D	D	D	D	D
7FF0	Write Data,Character Code, or Graphics Byte(W7-W0)	W7	W6	W5	W4	W3	W2	W1	W0
7FF0	Read Data, Character Code,or Graphics Byte (R7-R0)	R7	R6	R5	R4	R3	R2	R1	R0

NOTE: "D" implies Don't Care.

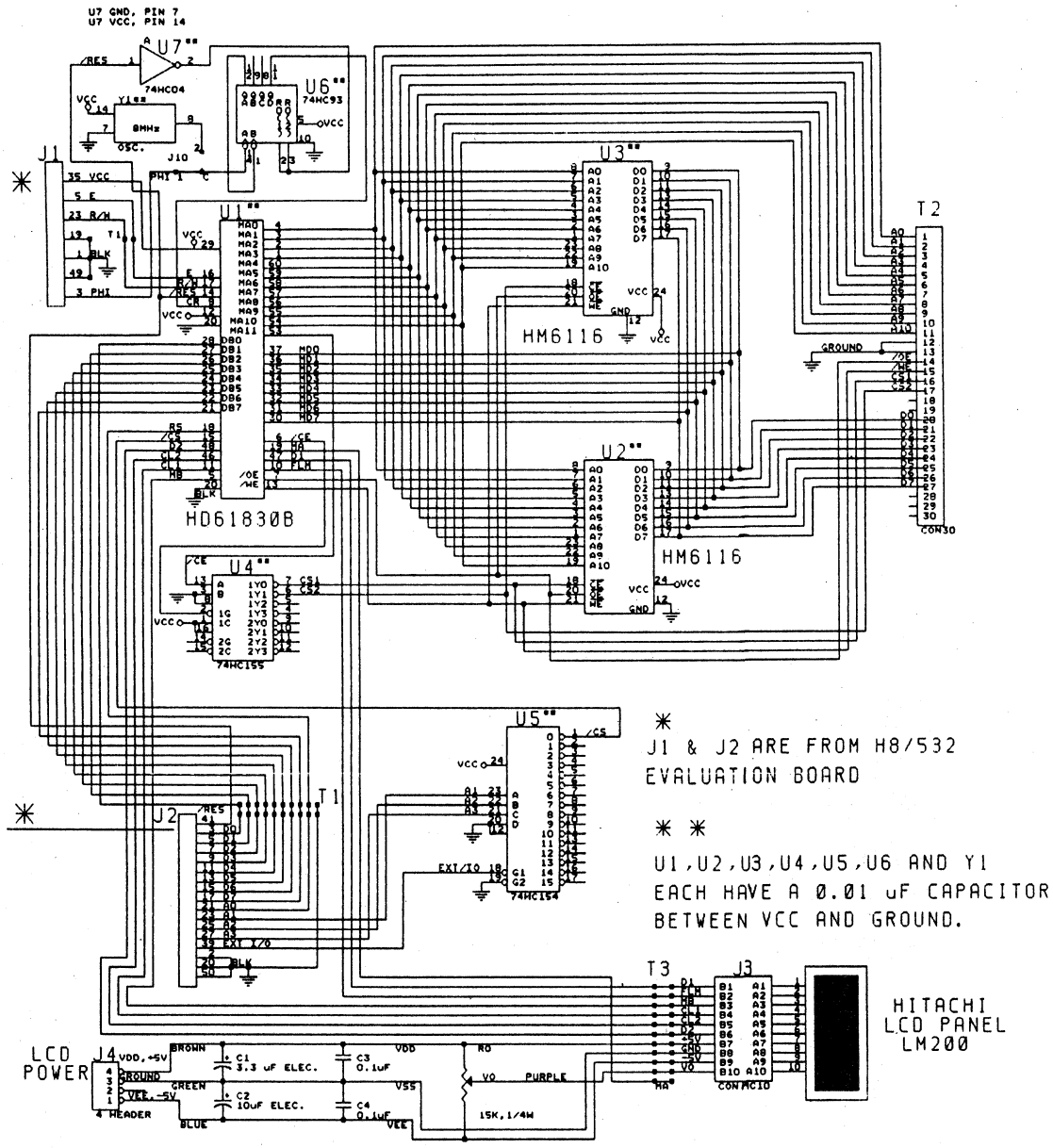
7.2 Busy Flag Check :

Emperically, it is determined that if the MPU processes consecutive I/O data instructions faster than FOUR "CL2" cycle times, the Busy Flag Check is required. In this application, f_{CL2} is 500KHz i.e. 2us cycle time. The four cycle times would make the HD61830B instruction execution time of 2x4 = 8us.

The H8/532 MPU operating with 16 MHz crystal provides 8MHz (T=125ns) "Phi" clock. From the Instruction Execution Table, "MOVFPE" or "MOVTPE" instruction requires 13 to 20 "Phi" clocks. With, faster time, i.e. 13 "Phi" clocks, it will take 13x 125ns = 1.625 micro-seconds. The two consecutive I/O Instructions will require at least 1.625 us x 2 = 3.25 micro-seconds.

Since, MPU Instruction time (3.25us) is substantially faster than 61830B instruction execution time (8us) the **Busy Flag Check is required** in this application. This was also varified in the laboratory.

HD61830B/LM200 LCD Panel Design—Schematic



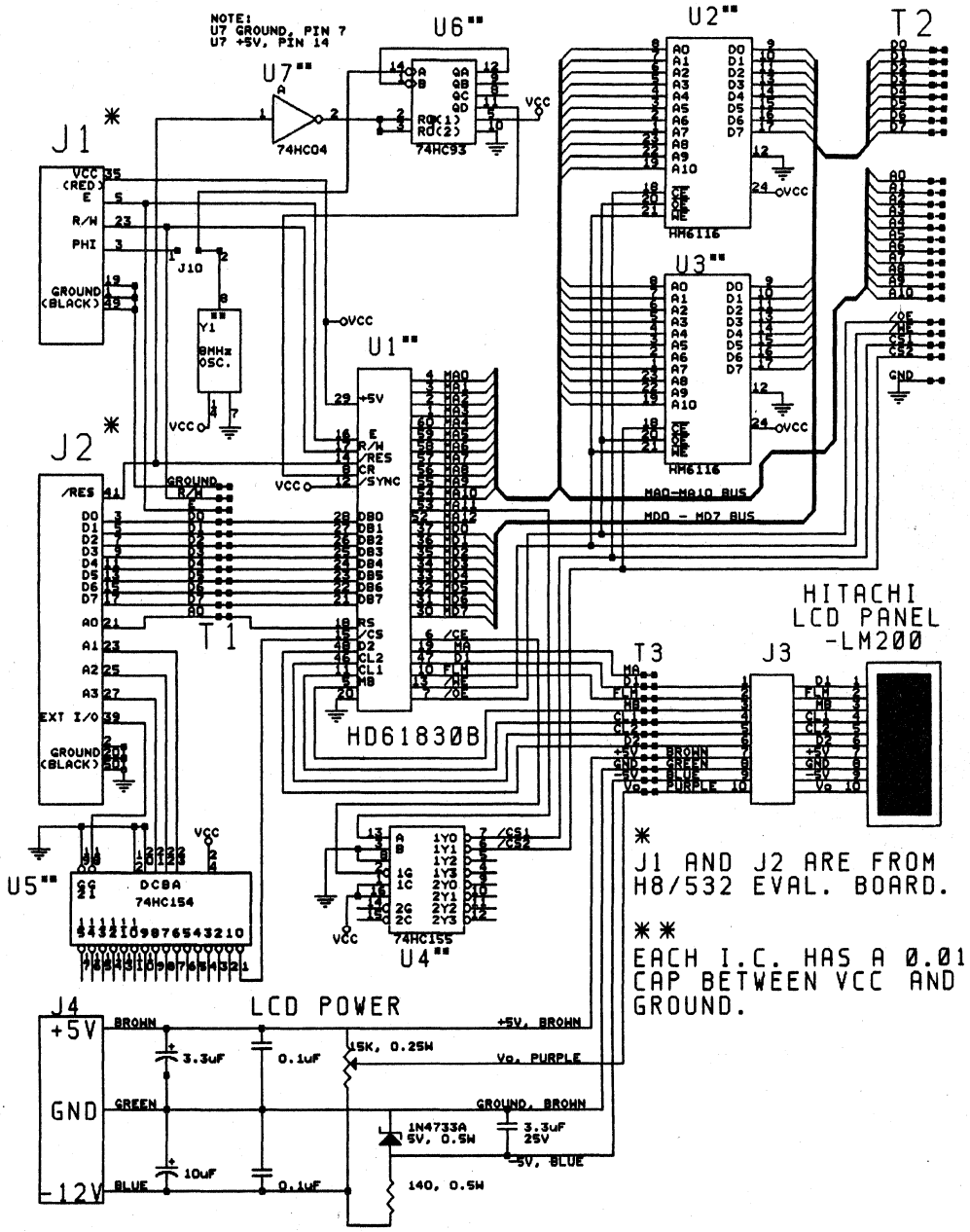
NOTE:

1.0 Test Connectors T1,T2, and T3 are for test and debug.

2.0 After power on reset, Display is "OFF", Slave Mode "ON", and Hp = 6.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

LCD Interface Board—Schematic



5.2 LCD Display Timing :

LCD panel LM200 feature highlights are summarized along with some of the timing calculations for 1/32 duty cycle.

5.2.1 LM200 Features : Its pertinent features are listed below ;

5.2.1.1 Power Supply - (V_{DD} - V_{SS}) = 7V Max
- (V_{DD} -V_{EE}) = 13.5 V Max

5.2.1.2 Input Signals : 0.7 x5 = 3.5 V (High), 0.3x5 = 1.5V (Low)

5.2.1.3 f_{CL2} Dot Clock Frequency - 390KHz (Min.), 460KHz (Typ.), and 520KHz (Max.)

5.2.1.4 Duty Cycle - 1/32

5.2.1.5 Power Supply For LCD Drivers - 8.1V (Ta=0°C), 7.4V (Ta=25°C), and 6.5V (Ta=50°C)

5.2.1.6 Scanning - Split with top half panel during "D1" and bottom half panel during "D2".

5.2.3 LCD Interface Board Design : To meet the features shown above, the design data is presented in the following description :

- o Power Supply - V_{DD} = +5V, V_{SS}=0, and V_{EE} = -5V
- o Input Signals - CMOS levels
- o f_{CLK2} - 500KHz
- o Duty Cycle - 1/32
- o Scanning - Split as provided by the LCD Controller HD61830B
- o LCD Driver Voltage - 4.75V (V_O) , variable through contrast adjustment pot.

5.2.4 LCD Timing : All the calculations are based upon the data provided above. They are summa rized as follows :

- o C_R = F_{CLK2} = 500 KHZ implies that if a 8 MHz crystal oscillator used, a divide by16 counter is required to produce a square waveform signal. Therefore, the dot clock time is 2us. This is an external oscillator and the jumper "J10" is set to "C-2" position. See the schematic in section 6.0.

- o T_{CL1}= Row scan time for 240 dots horizontally, is 240x2 = 480 us = 0.48 ms

- o With Duty Cycle = 1/32, LCD AC drive = M_A = 32 x 0.48 ms = 15.36 ms

- o M_B = 2M = 2x15.36 ms = 30.72 ms. Therefore, M_B Frequency = 32.55 Hz. Since, it is not a h. onic of the line frequency, there will not be any visible flickering of the LCD display.

5.0 LM200 LCD PANEL DISPLAY CRITERIA :

This section describes the display buffer memory capacity calculations and the LCD Display Timing related information for the LM200 LCD display panel. For the LM200 panel specifications, refer to the Graphics Panel Catalog from the Hitachi, ELT division.

5.1 Display Buffer Capacity :

LM200 panel can display 64 dots vertically, and 240 dots horizontally.
This implies a display of 64 x240 = 15,360 bits in graphics mode where a dot represents one LCD pixel on the panel. This would be lesser than 16,000 pixels that would be provided by a 2K by 8 SRAM in one bit per pixel mode. Allowing for scrolling, and other software overhead, this space was increased to 4K bytes. Therefore, the LCD Interface Board was designed for 4K bytes using two HM6116ALP-12 SRAM parts.

Character Mode :

The built in character ROM inside the HD61830B Controller is used. 5 dots(W) by 7 dots(H) character matrix is used with 6th column, 8th, and 10th rows as inter-character space. The cursor is set for the 9 th row.
With this data, 240/6 = 40 characters can be displayed per line. There can be 64 /10 = 6.4 i.e. 6 lines of character display resulting in 40x6 = 240 characters per panel. By changing the charac ter definition matrix , different numbers of characters can be displayed. Also, note that the HD61830B LCD Controller allows display of 32 different 5dots(w) x 10dots(h) characters. By using an external EPROM, special characters can also be displayed.

Graphics Mode :

By defining horizontal pitch (Hp) to be 8 dots, and 1 bit per pixel, 1 byte in the display memory would represent 8 LCD dots in a row. Note that, the horizontal pitch can be 6,7, or 8 dots per byte to be displayed.
With Hp=8, 240 / 8 = 30 bytes of graphics data can be displayed per a row of dots. Since, there are 64 rows, a total of 64 x 30 = 1,920 bytes of memory can be displayed on the LM200 panel. Note that even though LM200 panel is a split panel scanned as "D1", and "D2" halves, with 1/32 duty cycle, the display memory space is contiguous.
Observe, the buffer data inversion when it is displayed on the LM200 panel. For example, buffer data "33H" is displayed as "CCH" on the panel. Therefore, desired display data has to be inverted and then written to the display buffer.

4.3 Display Memory Read Timing : Refer to the SRAM data sheet for Read Cycle (1) timing diagram. The timing comparision between the HD61803B and the HM6116ALP-12 SRAM parameters is done below. Only the critical parameters are addressed :

	PARAMETERS	Symbol	HM6116ALP	HD61830B	UNITS
4.3.1.	Read Access Time	t_{AC}	120	650(Max.)	ns
4.3.2	Data Setup Time	t_{SMD}	65(Max.)	50	ns
4.3.3	Data Hold Time	t_{HMD}	10(Min.)	40	ns

NOTE: Any EPROM or SRAM with access time faster than 450 ns and meeting the above parameters would be sufficient for memory read in this application. EPROM may be used as a external a custom character generator for special characters.

4.4 Conclusion :

Since, the external clock frequency "CR" in this design is only 500KHz, no critical memory parameters are voilated. However, for **faster** panels, this analysis is very important.

4.0 DISPLAY MEMORY TIMING :

This section describes the display buffer memory read and write timing . The Hitachi HM6116ALP-12 SRAMS (Q=2) with 120 ns access time make up the 4096 byte buffer. For the detailed read / write timing diagrams and their parameters refer to the HD61830B data sheet. To see the SRAM data sheet refer to the Hitachi Memory Data Book #M11.

4.1 Timing Data :

- o If the External Clock "CR" (Refer to the schematic in section 6.0) on the LCD Interface Board is set to 2 MHz, then T_{CR} = 500 ns .
- o T1 = Memory Data Refresh Time for Upper Screen = $4T_{CR}$ (For Horizontal H_P = 8)
  = 4 x 500 = 2,000 ns = 2 us.
  Note: For H_P = 7, T1= 1.5 us, while For H_P=6, T1 = 1.0 us.
- o T2 = Memory Data Refresh Time for Lower Screen = $2T_{CR}$
  = 2 x 500 = 1,000 =1 us.
- o T3 = Memory Read / Write Time = 2 x T_{CR} = 2 x 500 ns = 1us.

4.2 Display Memory Write Timing : In the SRAM data sheet, use the Write Cycle (1) timing diagram. The comparision of the HM6116ALP-12 parameters and the Hitachi HD61830B memory write timing is listed below:

	PARAMETERS	Symbol	HM6116ALP	HD61830B	UNITS
4.2.1.	Write Cycle Time	t_{WC}	120(Min.)	1000	ns
4.2.2	/CS To Write End	t_{CW}	70(Min.)	600	ns
4.2.3	Write Recovery Time	t_{WR}	0(Min.)	350	ns
4.2.4	Write Pulse Width	t_{WP}	70(Min)	150	ns
4.2.5	Data To Write Overlap	t_{DW}	35(Min)	150	ns
4.2.6	Data hold From Write Time	t_{DH}	0(Min)	10	ns
4.2.7	Address Setup Time	t_{AS}	0(Min)	350	ns
4.2.8	Address Valid To Write End	t_{AW}	105(Min)	450	ns

3.0 MPU READ / WRITE TIMING :

This section describes the various H8 / 532 Evaluation Board and the HD61830B Hitachi LCD Controller specifications and arrives at the design trade-offs. Refer to the timing diagrams in the HD61830B and the H8 / 532 single chip MPU data sheets for more details.

	PARAMETERS	Symbol	H8 / 532 EVAL. BOARD	HD61830B	UNITS
3.1	"E" Clock Cycle Time	t_{CYC}	800 *	1000 (Min.)	ns
3.2	"E" High Pulse Width	t_{WEH}	370 *	450 (Min.)	ns
3.3	"E" Low Pulse Width	t_{WEL}	370 *	450 (Min.)	ns
3.4	Address Hold time	t_{AH}	20	10 (Min.)	ns
3.5	Address Setup Time	t_{AS}	180	140 (Min.)	ns
3.6	Write Data Hold Time	t_{DHW}	30	10 (Min.)	ns
3.7	Write Data Setup Time	t_{DSW}	440	225 (Min.)	ns
3.8	Read Data Hold Time	t_{DH}	0 (Min.)	20	ns
3.9	Read Data Setup Time	t_{DDR}	40(Min.)	225	ns

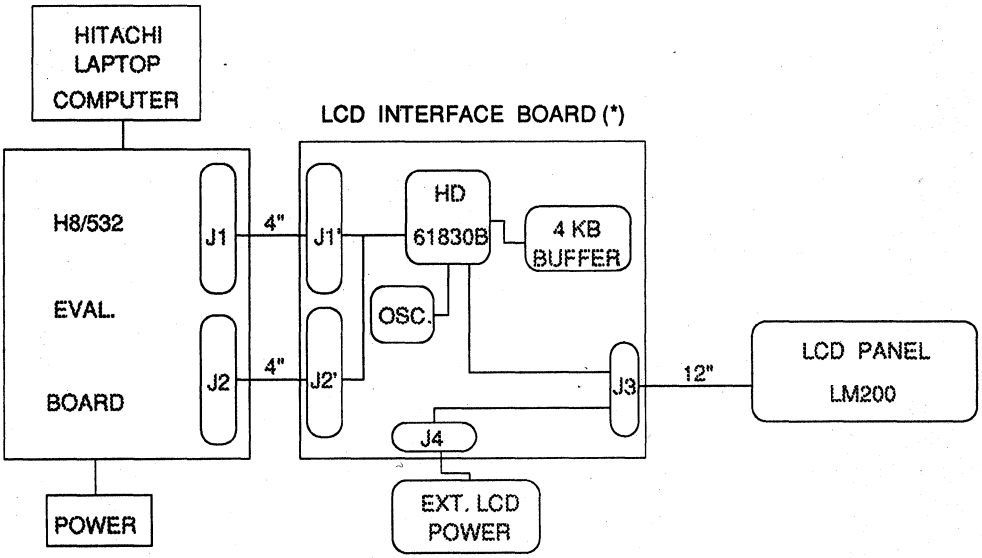
NOTE :

* Timing specifications of the HD61830B are violated .

Problem : The H8 / 532 Evaluation Board running with 20 MHz crystal produces a 10 MHZ "PHI" clock. It is further divided down and results in 1.25 MHz " E " clock with 50% duty cycle.

Solution : Run the H8 / 532 Evaluation Board with a 16 MHz crystal. This results in 1 MHz " E " clock. In this manner, by slowing down the " PHI " clock, the problem mentioned above is resolved. This is the reason for installing the 16 MHz crystal on the H8 /532 Evaluation Board.

HD61830B / LM200 SOFTWARE DEVELOPMENT STATION



*NOTE : 1.0 8 MHZ OSC. DIVIDED DOWN.
2.0 SET "J1" JUMPER TO "C-2" POSITION.

BLOCK DIAGRAM

SECTION 4

2.0 DESIGN OVERVIEW : (CNTD.)

2.5 Software Tools : The laptop PC resident software development tools, packages, and utilities are described very breifly.

H8 / 532 Cross Assembler : It is designed for DOS environment inside the laptop Personal Computer. When the user program is submitted as the source file, it assembles the code. Consequently, it produces Object and List files of the source program. The list files with " *.LIS " extentions are reproduced in the appendices for the programs developed on the software work station.

H8/ 532 Linker : To link various object code segments (" *.OBJ " extention) developed in parallel for a larger program. The linked file has " *.ABS " extention.

Motorola " S " record Conversion Utility : It is used to convert the machine code into Motorola " S " record format for uploading it to the H8 / 532 Evaluation Board. The converted file has " *.MOT " extention.

Up Loading Of Laptop PC " S " Record file : Push " EDIT SHIFT " Key down. Depress the " PG UP " key when using " PROCOMM " package for communications. Also, select ASCII format.

Screen Editor : Any word processing package is acceptable. In this application, Microsoft "WORD" package is used. The source programs are created and edited with this package. The source program files have " *.SRC " extentions.

File Management Utilities : To help aid the program development, packages such as " XTREE ", or " TREE86 " may also be used.

Back -Up Utility : It is a good practice to back up program files. Such packages as " FASTBACK ", OR " FASTBACK PLUS " can also be used.

The software development station block diagram is shown on the next page.

2.0 DESIGN OVERVIEW :

The LCD display subsystem components such as H8 / 532 Evaluation Board, LM200 display, LCD Interface Board, Hitachi Laptop Computer, and the related software are described in this section. At the end, a subsystem block diagram is also presented. For the HD61830B LCD Controller, and the LM200 LCD panel data sheets, as well as other related documentation refer, to the Appendix "C".

2.1 H8/532 Evaluation Board : This board was designed by Hitachi Micro Systems. It is provided as a training and development tool. On-board EPROM contains the Hitachi Monitor firmware used for single line assembly, disassembly, line editing, and debug purposes. Of the two serial ports, only the Terminal port is used to down load, up load, and run the programs. The I/O extention connectors "J1" and "J2" are used to connect to the LCD Interface Board. The partially decoded extented I/O space is further decoded on the LCD Interface Board. This board is designed to run at 10MHz and uses a 20 MHz crystal for that purpose. **However, in this application a 16 MHz crystal is used to provide 1MHz "E" clock to the LCD Controller HD61830B.** All the jumpers on this board are set at the factory according to their default states.

2.2 LM200 LCD Panel display : This display is provided by the Hitachi ELT Division. It is capable of displaying alpha-numeric characters as well as the graphics data. It is 240 dots wide and 64 dots high. It has 1 /32 duty cycle. The serial data is clocked in at 500KHz. It runs from +5V, and -5V power supply. The customer has to solder the pins on LM200 for the appropriate connector used on the LCD Interface Board. The LM200 LCD panel mounting and the proper viewing angles are critical to a strain free LCD display. Please, handle the panels according to the care recommended by the LCD display manufacturer. The logic signals sent to the LCD panel are at CMOS levels. External power supply was used for the LCD panel.

2.3 LCD Interface Board : A wire wrap board was built to control the LCD panel LM200. It also exchanged data with the H8/532 Evaluation Board over the I/O extention cables "J1" and "J2". The Hitachi LCD controller HD61830B was used on the LCD Interface Board. A 4,096 byte display buffer memory was also designed to store the character or graphics data. The 500KHz dot clock required by the display was also provided on this board. The LM200 LCD panel contrast adjust potentiometer was also put on this board. Set the jumper "J10" on this board to the "C-2" position. Test connectors were also provided to help debug this board.

2.4 Hitachi Laptop Personal Computer "HL320" : It is connected to the serial terminal port of the H8/532 Evaluation Board. The connector RJ-12 is attached to the Terminal port while a male to female 25 pin adapter cable is required at the Laptop PC end. The Hitachi "HL320" PC provides the software development tools for the user programs. The program up load and down load capability is also provided by the laptop PC. The communication link is full duplex, 9600 baud, 8 bits, 1 stop bit, and no parity check.

1.0 INTRODUCTION :

This section describes the design goals, LCD display subsystem with its components, provides a general overview of this presentation, along with a software development station block diagram, and the organization of the other sections in this document.

The **design goals** established for this project are briefly listed below:

1.1 To use H8/532 Evaluation Board with Monitor Software.
1.2 To provide LCD display with LM200 panel from Hitachi.
1.2 Alpha-Numeric and Graphic display capability.
1.4 To design Interface Board for the LM200 LCD panel.
1.5 To write programs for debug and test.
1.6 To Use Hitachi Laptop Personal Computer "HL320".
1.7 To use readily available software at Hitachi Field Offices for development.
1.8 To build multiple HD61830B programming stations.
1.9 To generate HD61830B / LM200 panel design tutorial.

A brief description of the LCD display subsystem components listed above is provided in the next section as general overview. To complete the overview, a subsystem block diagram is also presented. The rest of the sections described in the Table Of Contents are expanded in greater details along with their technical data. The Appendices give the program listings, and also list the referenced literature. A copy of the LCD Interface Board schematic is also provided to illustrate the implementation details of this application.

TABLE OF CONTENTS

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD61830B / LM200

Panel Design

TUTORIAL

Kash Yajnik

This tutorial presents in depth design process for a LCD subsystem. Its major components include H8/532 Evaluation board as the local processor, LCD Controller HD61830B, and the display panel LM200 from Hitachi ELT Division.

The HD61830B controller is designed to run in the character or graphics mode. The H8/532 Evaluation Board is designed by Hitachi Microsystems. The LM200 LCD panel can display 240 Dots(W) by 64 Dots(H) character or graphics data. Hitachi Monitor firmware resident on the H8/532 Evaluation Board provides the program debugging and host computer communication facilities.

By adding a laptop computer to down load the programs to the Evaluation Board, a program development station can be readily built. The H8/532 Cross Assembler, Linker, any word processor package e.g. "WORD" as screen editor, and Motorola "S" record conversion utility inside the Hitachi laptop PC complete the software development environment. The "PROCOMM" communication package is used to facilitate down load or up load of programs to the H8/532 Evaluation board.
In this manner, a number of software development stations were built to debug HD61830B / LM200 display programs.

These programs are listed in the Appendices "A" and "B". **No effort is spent in either code or logic minimization.**

This tutorial is intended for the technical staff at customer sites and other Hitachi employees who are fairly familiar with LCD design guide lines. Therefore, basic LCD design principles are **not** covered.

The HD61830B LCD Controller design tutorial includes Introduction, Design Overview, MPU Read/ Write Timing, Display Memory Timing, LM200 Panel Criteria, LCD Interface Board Schematic, and the associated Software.

While a lot of programs were developed, **only** two are listed as examples in their respective Appendices. The Appendix "A" shows the listing of the Character Mode display while the Appendix "B" shows the Graphics Mode listing. The Appendix "C" covers the reference literature.

Only the details not available in the reference literature are explained at greater length in this article. The page 2 shows the Table Of Contents.

Refer to the susequent pages for more details of this design.

The literature and other documents used in this design are summarized below :

- o H8/532 Cross Assembler Manual #S085CPC and " C " compiler for IBM PC
- o H8/532 Evaluation Board User's Manual # US538EVB21H
- o H8/532 Software User's Manual # HS538EMSS1E
- o MS " WORD " User Manual and other reference manuals
- o " PROCOMM " User Manual and other reference manuals
- o LCD Data Book #M24T013 from Hitachi America Ltd.
- o Memory Data Books from Hitachi America Ltd.
- o Hitachi Graphic Module Catalog # XX-E139 from ELT Division
- o H8/532 Hardware User's Manual #M21T002 from Hitachi America, Ltd.
- o H8/500 Programming Manual #M21T001 from Hitachi America, Ltd.
- o H8/500 Software Application Note #M21T003 from Hitachi America, Ltd.
- o H8/532 Overview #M21T173 from Hitachi America, Ltd.
- o Hitachi Laptop Personal Computer HL320 - Operator Manual
- o Hitachi Laptop Personal Computer HL320 - MSDOS V3.2 User's Manual
- o Hitachi HD61830B / LM200 Panel Design Tutorial Part I (in this manual)

APPENDIX " C "

REFERENCE LITERATURE

HD61830B / LM200 DESIGN - CUSTOM CHARACTER GENERATION

CHARACTER GENERATOR EPROM HN27C256AG-15

ADDRESS	DATA	CHARACTER
053FH	0H	" S "
0480H	82H	
1H	82H	
2H	82H	
3H	82H	
4H	82H	
5H	82H	
6H	82H	
7H	FEH	" H "
8H	82H	
9H	82H	
AH	82H	
BH	82H	
CH	82H	
DH	82H	
EH	82H	
FH	0H	

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD61830B / LM200 DESIGN - CUSTOM CHARACTER GENERATION

CHARACTER GENERATOR EPROM HN27C256AG-15

ADDRESS	DATA	CHARACTER
020AH	0H	
BH	0H	
CH	0H	" BLANK "
DH	0H	
EH	0H	
FH	0H	
0530H	FEH	
1H	02H	
2H	02H	
3H	02H	
4H	02H	
5H	02H	
6H	02H	" S "
7H	FEH	
8H	80H	
9H	80H	
AH	80H	
BH	80H	
CH	80H	
DH	80H	
EH	FEH	

HD61830B / LM200 DESIGN - CUSTOM CHARACTER GENERATION

CHARACTER GENERATOR EPROM HN27C256AG-15

ADDRESS	DATA	CHARACTER
0415H	82H	
6H	82H	
7H	82H	
8H	FEH	
9H	82H	
AH	82H	" A "
BH	82H	
CH	82H	
DH	82H	
EH	82H	
FH	0H	
0200H	0H	
1H	0H	
2H	0H	
3H	0H	
4H	0H	
5H	0H	" BLANK "
6H	0H	
7H	0H	
8H	0H	
9H	0H	

HD61830B / LM200 DESIGN - CUSTOM CHARACTER GENERATION

CHARACTER GENERATOR EPROM HN27C256AG-15

ADDRESS	DATA	CHARACTER
04B0H	82H	
1H	42H	
2H	22H	
3H	12H	
4H	0AH	
5H	06H	
6H	06H	
7H	06H	" K "
8H	06H	
9H	06H	
AH	0AH	
BH	12H	
CH	22H	
DH	42H	
EH	82H	
FH	0H	
0410H	10H	
1H	28H	
2H	44H	" A "
3H	82H	
4H	82H	

APPENDIX " B "

1.0 EPROM FONT DATA

2.0 ADDRESS RANGE - " 000H - 3FFH "

3.0 PROGRAM DESCRIPTION - CUSTOM CHARACTER FONT PATTERN LISTING FOR "K","A","S","H", AND "BLANK".

4.0 CHECK SUM - 7F3CAD

HD61830B / LM200 DESIGN - CUSTOM CHARACTER GENERATION

```
172 CHR   C 81A1 157FF10093        MOVTPE     R3,@H'7FF1      ;0H TO 7FF1
173 CHR   C 81A6 513D              MOV.E      #H'3D,R1        ;LOAD R1=3DH=DISP=ON
174 CHR   C 81A8 157FF00091        MOVTPE     R1,@H'7FF0      ;3DH TO 7FF0
175
176 CHR   C 81AD 00          NOP
177 CHR   C 81AE 00          NOP
178
179 CHR   C 81AF 1A          SLEEP                          ; PROCESSOR H8/532
; SLEEP
180
181                          .END
*****TOTAL ERRORS    0
*****TOTAL WARNINGS   0
```

HD61830B / LM200 DESIGN - CUSTOM CHARACTER GENERATION

```
115 CHR  C 810A ACF7          BTST    #7,R4          ;BIT TEST #7 OF R4
116 CHR  C 810C 26F7          BNE     X7             ;IF B/FLAG =Z=1 GO TO X7
117 CHR  C 810E 00     NOP                           ;B/FLAG NOT SET
118 CHR  C 810F 00     NOP
119 CHR  C 8110 00     NOP                  ; INITIALIZATION END
120 CHR  C 8111 00     NOP
121 CHR  C 8112 00     NOP
122 CHR  C 8113 500A          MOV:E   #H'A,R0        ;R0=AH
123 CHR  C 8115 157FF10090    MOVTPE  R0,@H'7FF1     ;AH TO 7FF1
124 CHR  C 811A 510F          MOV:E   #H'F,R1        ;R1=FH
125 CHR  C 811C 157FF00091    MOVTPE  R1,@H'7FF0     ;FH TO 7FF0
126 CHR  C 8121 00     NOP
127 CHR  C 8122 157FF10084    X8:     MOVFPE  @H'7FF1,R4     ;READ 7FF1 DATA TO R4
128 CHR  C 8127 ACF7          BTST    #7,R4          ;BIT TEST #7 OF R4
129 CHR  C 8129 26F7          BNE     X8             ;IF B/FLAG =Z=1 GO TO X8
130 CHR  C 812B 00     NOP                           ;B/FLAG NOT SET
131 CHR  C 812C 500B          MOV:E   #H'B,R0        ;R0=BH
132 CHR  C 812E 157FF10090    MOVTPE  R0,@H'7FF1     ;BH TO 7FF1
133 CHR  C 8133 157FF00092    MOVTPE  R2,@H'7FF0     ;0H TO 7FF0
134 CHR  C 8138 00     NOP
135 CHR  C 8139 157FF10084    X9:     MOVFPE  @H'7FF1,R4     ;READ 7FF1 DATA TO R4
136 CHR  C 813E ACF7          BTST    #7,R4          ;BIT TEST #7 OF R4
137 CHR  C 8140 26F7          BNE     X9             ;IF B/FLAG =Z=1 GO TO X9
138 CHR  C 8142 00     NOP                           ;B/FLAG NOT SET
139 CHR  C 8143 500C          MOV:E   #H'C,R0        ;R0=CH
140 CHR  C 8145 157FF10090    MOVTPE  R0,@H'7FF1     ;CH TO 7FF1
141 CHR  C 814A 514B          MOV:E   #H'4B,R1       ;R1=4B=CODE FOR "K"
142 CHR  C 814C 157FF00091    MOVTPE  R1,@H'7FF0     ;4B TO 7FF0
143 CHR  C 8151 00     NOP
144 CHR  C 8152 157FF10084    X10:    MOVFPE  @H'7FF1,R4     ;READ 7FF1 DATA TO R4
145 CHR  C 8157 ACF7          BTST    #7,R4          ;BIT TEST #7 OF R4
146 CHR  C 8159 26F7          BNE     X10            ;IF B/FLAG =Z=1 GO TO X10
147 CHR  C 815B 00     NOP                           ;B/FLAG NOT SET
148 CHR  C 815C 157FF10090    MOVTPE  R0,@H'7FF1     ;CH TO 7FF1
149 CHR  C 8161 5141          MOV:E   #H'41,R1       ;R1=41=CODE FOR "A"
150 CHR  C 8163 157FF00091    MOVTPE  R1,@H'7FF0     ;41 TO 7FF0
151 CHR  C 8168 00     NOP
152 CHR  C 8169 157FF10084    X11:    MOVFPE  @H'7FF1,R4     ;READ 7FF1 DATA TO R4
153 CHR  C 816E ACF7          BTST    #7,R4          ;BIT TEST #7 OF R4
154 CHR  C 8170 26F7          BNE     X11            ;IF B/FLAG =Z=1 GO TO X11
155 CHR  C 8172 00     NOP                           ;B/FLAG NOT SET
156 CHR  C 8173 157FF10090    MOVTPE  R0,@H'7FF1     ;CH TO 7FF1
157 CHR  C 8178 5153          MOV:E   #H'53,R1       ;R1=53=CODE FOR "S"
158 CHR  C 817A 157FF00091    MOVTPE  R1,@H'7FF0     ;53 TO 7FF0
159 CHR  C 817F 00     NOP
160 CHR  C 8180 157FF10084    X12:    MOVFPE  @H'7FF1,R4     ;READ 7FF1 DATA TO R4
161 CHR  C 8185 ACF7          BTST    #7,R4          ;BIT TEST #7 OF R4
162 CHR  C 8187 26F7          BNE     X12            ;IF B/FLAG =Z=1 GO TO X12
163 CHR  C 8189 00     NOP                           ;B/FLAG NOT SET
164 CHR  C 818A 157FF10090    MOVTPE  R0,@H'7FF1     ;CH TO 7FF1
165 CHR  C 818F 5148          MOV:E   #H'48,R1       ;R1=48=CODE FOR "H"
166 CHR  C 8191 157FF00091    MOVTPE  R1,@H'7FF0     ;48 TO 7FF0
167 CHR  C 8196 00     NOP
168 CHR  C 8197 157FF10084    X13:    MOVFPE  @H'7FF1,R4     ;READ 7FF1 DATA TO R4
169 CHR  C 819C ACF7          BTST    #7,R4          ;BIT TEST #7 OF R4
170 CHR  C 819E 26F7          BNE     X13            ;IF B/FLAG =Z=1 GO TO X13
171 CHR  C 81A0 00     NOP                           ;B/FLAG NOT SET
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD61830B / LM200 DESIGN - CUSTOM CHARACTER GENERATION

```
58 CHR   C 807A 157FF00091          MOVTPE    R1,@H'7FF0      ;EH TO 7FF0
59 CHR   C 807F 00        NOP
60 CHR   C 8080 157FF10084    X5:   MOVFPE    @H'7FF1,R4      ;READ 7FF1 DATA TO R4
61 CHR   C 8085 ACF7                BTST      #7,R4           ;BIT TEST #7 OF R4
62 CHR   C 8087 26F7                BNE       X5              ;IF B/FLAG =Z=1 GO TO X5
63 CHR   C 8089 00        NOP                                 ;B/FLAG NOT SET
64 CHR   C 808A 5008                MOV:E     #H'8,R0         ;R0=8H
65 CHR   C 808C 157FF10090          MOVTPE    R0,@H'7FF1      ;8H TO 7FF1
66 CHR   C 8091 157FF00092          MOVTPE    R2,@H'7FF0      ;0H TO 7FF0
67 CHR   C 8096 00        NOP
68 CHR   C 8097 157FF10084    X6:   MOVFPE    @H'7FF1,R4      ;READ 7FF1 DATA TO R4
69 CHR   C 809C ACF7                BTST      #7,R4           ;BIT TEST #7 OF R4
70 CHR   C 809E 26F7                BNE       X6              ;IF B/FLAG =Z=1 GO TO X6
71 CHR   C 80A0 00        NOP                                 ;B/FLAG NOT SET
72 CHR   C 80A1 5009                MOV:E     #H'9,R0         ;LOAD R0=9H
73 CHR   C 80A3 157FF10090          MOVTPE    R0,@H'7FF1      ;9H TO 7FF1
74 CHR   C 80A6 157FF00092          MOVTPE    R2,@H'7FF0      ;0H TO 7FF0
75 CHR   C 80AD 00        NOP
76                                  ; SCREEN CLEAR ROUTINE START
77
78 CHR   C 80AE AD13                CLR.W     R5              ; CLEAR R5
79 CHR   C 80B0 00        NOP
80 CHR   C 80B1 157FF10084    C1:   MOVFPE    @H'7FF1,R4      ;READ 7FF1 DATA TO R4
81 CHR   C 80B6 ACF7                BTST      #7,R4           ;BIT TEST #7 OF R4
82 CHR   C 80B8 26F7                BNE       C1              ;IF B/FLAG =Z=1 GO TO C1
83 CHR   C 80BA 00        NOP                                 ;B/FLAG NOT SET
84 CHR   C 80BB 500A                MOV:E     #H'A,R0         ;R0=AH
85 CHR   C 80BD 157FF10090          MOVTPE    R0,@H'7FF1      ;AH TO 7FF1
86 CHR   C 80C2 5100                MOV:E     #H'0,R1         ;R1=0H
87 CHR   C 80C4 157FF00091          MOVTPE    R1,@H'7FF0      ;0H TO 7FF0-CUR L/B=0H
88 CHR   C 80C9 00        NOP
89 CHR   C 80CA 157FF10084    C2:   MOVFPE    @H'7FF1,R4      ;READ 7FF1 DATA TO R4
90 CHR   C 80CF ACF7                BTST      #7,R4           ;BIT TEST #7 OF R4
91 CHR   C 80D1 26F7                BNE       C2              ;IF B/FLAG =Z=1 GO TO C2
92 CHR   C 80D3 00        NOP                                 ;B/FLAG NOT SET
93 CHR   C 80D4 5008                MOV:E     #H'8,R0         ;R0=8H
94 CHR   C 80D6 157FF10090          MOVTPE    R0,@H'7FF1      ;8H TO 7FF1
95 CHR   C 80DB 5100                MOV:E     #H'0,R1         ;R1=0H
96 CHR   C 80DD 157FF00091          MOVTPE    R1,@H'7FF0      ;0H TO 7FF0-CUR H/B=0H
97 CHR   C 80E2 00        NOP
98 CHR   C 80E3 5D01FF              MOV:I     #H'1FF,R5       ;COUNT=R5=1FFH
99 CHR   C 80E6 00        C3:       NOP
100 CHR  C 80E7 157FF10084    C4:   MOVFPE    @H'7FF1,R4      ;READ 7FF1 DATA TO R4
101 CHR  C 80EC ACF7                BTST      #7,R4           ;BIT TEST #7 OF R4
102 CHR  C 80EE 26F7                BNE       C4              ;IF B/FLAG =Z=1 GO TO C4
103 CHR  C 80F0 00        NOP                                 ;B/FLAG NOT SET
104 CHR  C 80F1 500C                MOV:E     #H'C,R0         ;R0=CH
105 CHR  C 80F3 157FF10090          MOVTPE    R0,@H'7FF1      ;CH TO 7FF1
106 CHR  C 80F8 5120                MOV:E     #H'20,R1        ;R1=20H-CODE FOR "BLANK"
107 CHR  C 80FA 157FF00091          MOVTPE    R1,@H'7FF0      ;20H TO 7FF0
108 CHR  C 80FF 00        NOP
109 CHR  C 8100 01BDE3              SCB/F     R5,C3
110 CHR  C 8103 00        NOP
111                                 ; SCREEN CLEAR ROUTINE COMPLETED
112 CHR  C 8104 00        NOP
113
114 CHR  C 8105 157FF10084    X7:   MOVFPE    @H'7FF1,R4      ;READ 7FF1 DATA TO R4
```

HD61830B / LM200 DESIGN - CUSTOM CHARACTER GENERATION

```
1                          .HEADING  "XCG"
2 CHR   C 0000             .SECTION  CHR,CODE,ALIGN=2
3                          .EXPORT                    A
4 CHR   C 8000             .ORG                       H'8000          ;LOC CNTR =8000H
5
6                                                                     ; BUSY FLAG CHECKED
7
8 CHR   C   00008000       A:        .EQU             $               ;A = 8000H
9 CHR   C 8000 A013        CLR.B                      R0              ;CLEAR R0
10 CHR  C 8002 A113                  CLR.B            R1              ;CLEAR R1
11 CHR  C 8004 A213                  CLR.B            R2              ;CLEAR R2
12 CHR  C 8006 A313                  CLR.B            R3              ;CLEAR R3
13 CHR  C 8008 A413                  CLR.B            R4              ;CLEAR R4
14
15 CHR  C 800A 00          NOP
16 CHR  C 800B 00          NOP                  ; INITIALIATION START
17 CHR  C 800C 00          NOP
18 CHR  C 800D 00          NOP                  ; EXTERNAL CG ENABLED
19 CHR  C 800E 00          NOP
20 CHR  C 800F 157FF10080            MOVTPE           R0,@H'7FF1      ;0H TO 7FF1
21 CHR  C 8014 511D                  MOV:E            #H'1D,R1        ;LOAD R1 =1DH
22 CHR  C 8016 157FF00081            MOVTPE           R1,@H'7FF0      ;1DH TO 7FF0
23 CHR  C 801B 00          NOP
24 CHR  C 801C 157FF10084     X1:    MOVFPE           @H'7FF1,R4      ;READ 7FF1 DATA TO R4
25 CHR  C 8021 ACF7                  BTST             #7,R4           ;BIT TEST #7 OF R4
26 CHR  C 8023 28F7                  BNE              X1              ;IF B/FLAG =Z=1 GO TO X1
27 CHR  C 8025 00          NOP                                        ;B/FLAG NOT SET
28 CHR  C 8026 5001        MOV:E                       #H'1,R0         ;LOAD R0=1H
29 CHR  C 8028 157FF10080            MOVTPE           R0,@H'7FF1      ;1H TO 7FF1
30 CHR  C 802D 51F7                  MOV:E            #H'F7,R1        ;LOAD R1=F7H
31 CHR  C 802F 157FF00081            MOVTPE           R1,@H'7FF0      ;F7H TO 7FF0
32 CHR  C 8034 00          NOP
33 CHR  C 8035 157FF10084     X2:    MOVFPE           @H'7FF1,R4      ;READ 7FF1 DATA TO R4
34 CHR  C 803A ACF7                  BTST             #7,R4           ;BIT TEST #7 OF R4
35 CHR  C 803C 28F7                  BNE              X2              ;IF B/FLAG =Z=1 GO TO X2
36 CHR  C 803E 00          NOP                                        ;B/FLAG NOT SET
37 CHR  C 803F 5002                  MOV:E            #H'2,R0         ;LOAD R0=2H
38 CHR  C 8041 157FF10080            MOVTPE           R0,@H'7FF1      ;2H TO 7FF1
39 CHR  C 8046 511D                  MOV:E            #H'1D,R1        ;LOAD R1=1DH
40 CHR  C 8048 157FF00081            MOVTPE           R1,@H'7FF0      ;1DH TO 7FF0
41 CHR  C 804D 00          NOP
42 CHR  C 804E 157FF10084     X3:    MOVFPE           @H'7FF1,R4      ;READ 7FF1 DATA TO R4
43 CHR  C 8053 ACF7                  BTST             #7,R4           ;BIT TEST #7 OF R4
44 CHR  C 8055 28F7                  BNE              X3              ;IF B/FLAG =Z=1 GO TO X3
45 CHR  C 8057 00          NOP                                        ;B/FLAG NOT SET
46 CHR  C 8058 5003        MOV:E                       #H'3,R0         ;LOAD R0=3H
47 CHR  C 805A 157FF10080            MOVTPE           R0,@H'7FF1      ;3H TO 7FF1
48 CHR  C 805F 511F                  MOV:E            #H'1F,R1        ;LOAD R1=1FH
49 CHR  C 8061 157FF00081            MOVTPE           R1,@H'7FF0      ;1FH TO 7FF0
50 CHR  C 8066 00          NOP
51 CHR  C 8067 157FF10084     X4:    MOVFPE           @H'7FF1,R4      ;READ 7FF1 DATA TO R4
52 CHR  C 806C ACF7                  BTST             #7,R4           ;BIT TEST #7 OF R4
53 CHR  C 806E 28F7                  BNE              X4              ;IF B/FLAG =Z=1 GO TO X4
54 CHR  C 8070 00          NOP                                        ;B/FLAG NOT SET
55 CHR  C 8071 5004        MOV:E                       #H'4,R0         ;LOAD R0=4H
56 CHR  C 8073 157FF10080            MOVTPE           R0,@H'7FF1      ;4H TO 7FF1
57 CHR  C 8078 510E                  MOV:E            #H'E,R1         ;LOAD R1=EH
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

APPENDIX " A "

1.0 PROGRAM NAME - " XCG.MOT "

2.0 ADDRESS RANGE - " 8000H - 81AFH "

3.0 PROGRAM DESCRIPTION - CLEARS SCREEN, CHECKS BUSY FLAG, AND DISPLAYS 4 CUSTOM CHARACTERS ON THE LCD LM200 PANEL STARTING AT THE 8TH CURSOR POSITION.

Listing 3: INPUT.LIS

```
*** H8/300 ASSEMBLER                    VER 1.1 ***    03/20/91 08:11:34                                    PAGE    1
PROGRAM NAME =                                    Date Input Routine

                                                  .heading        "Date Input Routine"

                                        ;H8/330 Print Buffer Routine
                                        ;version 2.0

                                        ;written by:
                                        ;       Tom Hampton
                                        ;       Hitachi America, Ltd.
                                        ;       Application Engineering

                                                  .output         dbg,obj
                                                  .print          nocref,nosct

                                                  .global         input_int

 program  C 0000                                  .section        program,code

                                        ;Input /STB Interrupt Routine

 program  C 0000                        input_int:
 program  C 0000 7FB27040                         bset.b  #ibusy_bit,@in_hs       ;set input busy signal

                                        ;get input data
 program  C 0004                        get_data:
 program  C 0004 20BE                             mov.b   @in_port,r0h    ;read data

                                        ;write data into memory buffer
 program  C 0006 F8FF                             mov.b   #write,r0l
 program  C 0008 38B1                             mov.b   r0l,@mem_dir    ;set port direction for write
 program  C 000A 30B3                             mov.b   r0h,@mem_data   ;set data
 program  C 000C 3DB6                             mov.b   r5l,@addr_lo
 program  C 000E 35BF                             mov.b   r5h,@addr_hi    ;output buffer address
 program  C 0010 4B04                             bmi     wr_cs1
 program  C 0012                        wr_cs0:
 program  C 0012 F802                             mov.b   #wrcs0,r0l      ;write to U3
 program  C 0014 4002                             bra     wr_cont
 program  C 0016                        wr_cs1:
 program  C 0016 F801                             mov.b   #wrcs1,r0l      ;write to U4
 program  C 0018                        wr_cont:
 program  C 0018 38BA                             mov.b   r0l,@mem_ctrl   ;activate write pulse
 program  C 001A F807                             mov.b   #7,r0l
 program  C 001C 38BA                             mov.b   r0l,@mem_ctrl   ;de-activate write pulse
 program  C 001E 0B05                             adds    #1,r5           ;increment input pointer

                                        ;test for buffer full
 program  C 0020 0D53                             mov.w   r5,r3
 program  C 0022 1963                             sub.w   r6,r3           ;is IDP = ODP ?
 program  C 0024 4708                             beq     buff_full       ;yes

                                        ;buffer is not full, but cannot be empty either
 program  C 0026 720C                             bclr.b  #buf_mt_flag,r4l        ;clear buffer empty flag
 program  C 0028 7FB27240                         bclr.b  #ibusy_bit,@in_hs       ;clear IBUSY signal
 program  C 002C 4006                             bra     clean_ret

 program  C 002E                        buff_full:
 program  C 002E 701C                             bset.b  #buf_fl_flag,r4l        ;set buffer full flag
                                                                                  ; IBUSY remains set
 program  C 0030 7FB27220                         bclr.b  #buf_ful_bit,@stat_port ;turn Buffer Full LED ON

                                        ;reset input delay timer
 program  C 0034                        clean_ret:
 program  C 0034 79000000                         mov.w   #0,r0
 program  C 0038 6B80FF92                         mov.w   r0,@frt_frc     ;reset delay timer

                                        ;enable delay timer interrupts
 program  C 003C 7F917230                         bclr.b  #3,@frt_tcsr    ;clear compare flag

 program  C 0040 5670                             rte

                                                  .end
*****TOTAL ERRORS        0
*****TOTAL WARNINGS      0
```

SECTION 5

Listing 4: OUTPUT.LIS

```
*** H8/300 ASSEMBLER                    VER 1.1 ***   03/20/91 08:11:43                                    PAGE    1
PROGRAM NAME -                                        Data Output Routine

                                                        .heading        "Data Output Routine"

                                                ;H8/330 Print Buffer Routine
                                                ;version 2.0

                                                ;written by:
                                                ;       Tom Hampton
                                                ;       Hitachi America, Ltd.
                                                ;       Application Engineering

                                                        .output         dbg,obj
                                                        .print          nocref,nosct

                                                        .global         output_int

program  C 0000                                         .section        program,code

                                                ;Data Output Service
                                                ;input delay timer interrupt

program  C 0000                                 output_int:
program  C 0000 7FB27040                                bset.b  #ibusy_bit,@in_hs       ;set input busy signal

                                                ;test for buffer empty
program  C 0004 730C                                    btst.b  #buf_mt_flag,r4l        ;is buffer empty ?
program  C 0006 464E                                    bne     ret1                    ;yes

                                                ;test for output busy
program  C 0008 7EB77300                                btst.b  #obusy_bit,@out_hs      ;output busy ?
program  C 000C 4648                                    bne     ret1                    ;yes

                                                ;test for output hold
program  C 000E 733C                                    btst.b  #ohold_flag,r4l         ;output on hold ?
program  C 0010 4644                                    bne     ret1                    ;yes

                                                ;get output buffer data
program  C 0012 F800                                    mov.b   #read,r0l
program  C 0014 38B1                                    mov.b   r0l,@mem_dir    ;set port direction for read

program  C 0016 3EB6                                    mov.b   r6l,@addr_lo
program  C 0018 36BF                                    mov.b   r6h,@addr_hi    ;output buffer address
program  C 001A 4B04                                    bmi     rd_cs1
program  C 001C                                 rd_cs0:
program  C 001C F806                                    mov.b   #rdcs0,r0l      ;read from U3
program  C 001E 4002                                    bra     rd_cont
program  C 0020                                 rd_cs1:
program  C 0020 F805                                    mov.b   #rdcs1,r0l      ;read from U4
program  C 0022                                 rd_cont:
program  C 0022 38BA                                    mov.b   r0l,@mem_ctrl   ;activate chip select pulse
program  C 0024 20B3                                    mov.b   @mem_data,r0h   ;get output data
program  C 0026 F807                                    mov.b   #7,r0l
program  C 0028 38BA                                    mov.b   r0l,@mem_ctrl   ;de-activate write pulse
program  C 002A 0B06                                    adds    #1,r6   ;increment output pointer

                                                ;output data to port
program  C 002C 30BB                                    mov.b   r0h,@out_port   ;output data

                                                ;generate output data strobe
program  C 002E F800                                    mov.b   #0,r0l
program  C 0030 38CC                                    mov.b   r0l,@tmr0_tcnt  ;clear counter
program  C 0032 28C9                                    mov.b   @tmr0_tcsr,r0l  ;read flags
program  C 0034 F816                                    mov.b   #h'16,r0l
program  C 0036 38C9                                    mov.b   r0l,@tmr0_tcsr  ;generate a negative strobe
                                                                                ; 2.4 usec wide and clear flags
program  C 0038 F841                                    mov.b   #h'41,r0l
program  C 003A 38C8                                    mov.b   r0l,@tmr0_tcr   ;enable compare A interrupt

                                                ;enable interrupts for strobe generation
program  C 003C 0700                                    ldc     #0,ccr          ;enable interrupts

                                                ;wait for output strobe interrupt
program  C 003E 0180                                    sleep

                                                ;disable interrupts
program  C 0040 0780                                    ldc     #h'80,ccr       ;mask interrupts

                                                ;test for buffer empty
program  C 0042 0D63                                    mov.w   r6,r3           ;temporary work register
program  C 0044 1953                                    sub.w   r5,r3           ;is ODP - IDP ?
```

Listing 4: OUTPUT.LIS (continued)

```
 317 program  C 0046 470C                        beq     buff_mt ;yes
 318
*** H8/300 ASSEMBLER                VER 1.1 ***   03/20/91 08:11:43                                    PAGE     2
PROGRAM NAME -                                  Data Output Routine

 319                                     ;test for in full area
 320 program  C 0048 084B                        add.b   r4h,r3l         ;is buffer still full ?
 321 program  C 004A 4516                        bcs     ret2            ;yes
 322
 323 program  C 004C 721C                        bclr.b  #buf_fl_flag,r4l        ;clear buffer full flag
 324 program  C 004E 7FB27020                    bset.b  #buf_ful_bit,@stat_port ;turn Buffer Full LED OFF
 325 program  C 0052 4002                        bra     ret1
 326
 327                                     ;set buffer empty flag
 328 program  C 0054                     buff_mt:
 329 program  C 0054 700C                        bset.b  #buf_mt_flag,r4l        ;set buffer empty flag
 330
 331                                     ;should IBUSY be cleared ?
 332 program  C 0056                     ret1:
 333 program  C 0056 734C                        btst.b  #online_flag,r4l        ;is buffer online ?
 334 program  C 0058 4708                        beq     ret2                    ;no
 335
 336 program  C 005A 731C                        btst.b  #buf_fl_flag,r4l        ;is buffer full ?
 337 program  C 005C 4604                        bne     ret2                    ;yes
 338
 339                                     ;clear IBUSY signal
 340 program  C 005E 7FB27240                    bclr.b  #ibusy_bit,@in_hs       ;set IBUSY inactive
 341
 342 program  C 0062                     ret2:
 343 program  C 0062 79000000                    mov.w   #0,r0
 344 program  C 0066 6B80FF92                    mov.w   r0,@frt_frc     ;reset delay timer
 345
 346                                     ;enable delay timer interrupts
 347 program  C 006A 7F917230                    bclr.b  #3,@frt_tcsr    ;clear compare flag
 348
 349 program  C 006E 5670                        rte
 350
 351                                             .end
 *****TOTAL ERRORS        0
 *****TOTAL WARNINGS      0
```

SECTION 5

Listing 5: OUT-STB.LIS

```
*** H8/300 ASSEMBLER                    VER 1.1 ***   03/20/91 08:11:54                                    PAGE     1
PROGRAM NAME =                                        Output STB Routine

  1                                                     .heading        "Output STB Routine"
  2
  3                                             ;H8/330 Print Buffer Routine
  4                                             ;version 2.0
  5
  6                                             ;written by:
  7                                             ;       Tom Hampton
  8                                             ;       Hitachi America, Ltd.
  9                                             ;       Application Engineering
 10
248                                                     .output         dbg,obj
249                                                     .print          nocref,nosct
250
251                                                     .global         ostb_int
252
253 program  C 0000                                     .section        program,code
254
255                                             ;Output Strobe interrupt
256 program  C 0000                             ostb_int:
257 program  C 0000 F801                                mov.b   #1,r01
258 program  C 0002 38C8                                mov.b   r01,@tmr0_tcr   ;disable further timer interrupts
259
260 program  C 0004 28C9                                mov.b   @tmr0_tcsr,r01  ;read flags
261 program  C 0006 F81A                                mov.b   #h'1a,r01
262 program  C 0008 38C9                                mov.b   r01,@tmr0_tcsr  ;clear flags, outputs to high level
263
264 program  C 000A 5670                                rte
265
266                                                     .end
*****TOTAL ERRORS        0
*****TOTAL WARNINGS      0
```

Listing 6: IN-INIT.LIS

```
*** H8/300 ASSEMBLER                    VER 1.1 ***   03/20/91 08:12:01                                    PAGE     1
PROGRAM NAME =                                        Input INIT Pulse Service Routine

  1                                                     .heading        "Input INIT Pulse Service Routine"
  2
  3                                             ;H8/330 Print Buffer Routine
  4                                             ;version 2.0
  5
  6                                             ;written by:
  7                                             ;       Tom Hampton
  8                                             ;       Hitachi America, Ltd.
  9                                             ;       Application Engineering
 10
248                                                     .output         dbg,obj
249                                                     .print          nocref,nosct
250
251                                                     .global         iinit_int,start
252
253 program  C 0000                                     .section        program,code
254
255                                             ;disable any timer interrupts
256 program  C 0000                             iinit_int:
257 program  C 0000 F801                                mov.b   #1,r01
258 program  C 0002 38C8                                mov.b   r01,@tmr0_tcr   ;disable output strobe interrupts
259 program  C 0004 38D0                                mov.b   r01,@tmr1_tcr   ;disable init pulse interrupts
260 program  C 0006 3890                                mov.b   r01,@frt_tier   ;disable input delay interrupts
261
262                                             ;jump to beginning
263 program  C 0008 5A000000                            jmp     @start          ;jump to initialization routine
264
265                                                     .end
*****TOTAL ERRORS        0
*****TOTAL WARNINGS      0
```

Listing 7: OUT-INIT.LIS

```
*** H8/300 ASSEMBLER                    VER 1.1 ***   03/20/91 08:12:08                                    PAGE     1
PROGRAM NAME =                                   INIT Pulse Output Routine

                                                 .heading        "INIT Pulse Output Routine"

                                        ;H8/330 Print Buffer Routine
                                        ;version 2.0

                                        ;written by:
                                        ;       Tom Hampton
                                        ;       Hitachi America, Ltd.
                                        ;       Application Engineering

                                                 .output         dbg,obj
                                                 .print          nocref,nosct

                                                 .global         oinit_int

                                                 .section        program,code

                                        ;output INIT signal interrupt
program  C 0000                         oinit_int:
                                        ;disable further timer interrupts
program  C 0000 F901                             mov.b   #1,r1l
program  C 0002 39D0                             mov.b   r1l,@tmr1_tcr   ;use phi/8, no interrupts

                                        ;clear match flag
program  C 0004 7FD17260                         bclr.b  #6,@tmr1_tcsr   ;clear compare match a flag

                                        ;clear OINIT\ signal
program  C 0008 F900                             mov.b   #0,r1l
program  C 000A 39D1                             mov.b   r1l,@tmr1_tcsr  ;no more strobes

                                        ;clear oinit flag
program  C 000C 722C                             bclr.b  #buf_init_flag,r4l      ;clear oinit flag

program  C 000E 5670                             rte

                                                 .end
*****TOTAL ERRORS        0
*****TOTAL WARNINGS      0
```

SECTION 5

Listing 8: ONLINE.LIS

```
*** H8/300 ASSEMBLER                    VER 1.1 ***   03/20/91 08:12:16                                          PAGE     1
PROGRAM NAME =                                   Online Pushbutton Service Routine

                                                 .heading        "Online Pushbutton Service Routine"

                                        ;H8/330 Print Buffer Routine
                                        ;version 2.0

                                        ;written by:
                                        ;       Tom Hampton
                                        ;       Hitachi America, Ltd.
                                        ;       Application Engineering

                                                 .output         dbg,obj
                                                 .print          nocref,nosct

                                                 .global         online_int

                                                 .section        program,code

                                        ;on line pushbutton test
program  C 0000                         online_int:

                                        ;set input port busy
program  C 0000 7FB27040                         bset.b  #ibusy_bit,@in_hs       ;set IBUSY active

                                        ;test online switch
program  C 0004                         test_sw:
program  C 0004 7EC17320                         btst.b  #online_sw_bit,@in_hs2  ;test online switch
program  C 0008 47FA                             beq     test_sw                 ;still low
                                                                                 ; will not go further
                                                                                 ; until released

                                        ;test online status
program  C 000A 734C                             btst.b  #online_flag,r4l        ;test online status
program  C 000C 4708                             beq     put_online              ;currently offline

                                        ;currently online
program  C 000E                         put_offline:
program  C 000E 7FB27010                         bset.b  #online_bit,@stat_port  ;clear Online LED
program  C 0012 724C                             bclr.b  #online_flag,r4l        ;clear online status

program  C 0014                         just_ret:
program  C 0014 5670                             rte

                                        ;currently offline
program  C 0016                         put_online:
program  C 0016 7FB27210                         bclr.b  #online_bit,@stat_port  ;set Online LED
program  C 001A 704C                             bset.b  #online_flag,r4l        ;set online status

                                        ;should IBUSY be cleared ?
program  C 001C 731C                             btst.b  #buf_fl_flag,r4l        ;is buffer full
program  C 001E 46F4                             bne     just_ret                ;yes, IBUSY should remain active

                                        ;clear input port busy
program  C 0020 7FB27240                         bclr.b  #ibusy_bit,@in_hs       ;set IBUSY inactive

program  C 0024 5670                             rte

                                                 .end
*****TOTAL ERRORS        0
*****TOTAL WARNINGS      0
```

Listing 9: PAUSE.LIS

```
*** H8/300 ASSEMBLER                    VER 1.1 ***   03/20/91 08:12:24                                    PAGE     1
PROGRAM NAME =                                  Pause Pushbutton Service Routine

                                                .heading        "Pause Pushbutton Service Routine"

                                        ;H8/330 Print Buffer Routine
                                        ;version 2.0

                                        ;written by:
                                        ;       Tom Hampton
                                        ;       Hitachi America, Ltd.
                                        ;       Application Engineering

                                                .output         dbg,obj
                                                .print          nocref,nosct

                                                .global         pause_int

program  C 0000                                 .section        program,code

                                        ;pause pushbutton test
program  C 0000                         pause_int:

                                        ;set input port busy
program  C 0000 7FB27040                        bset.b  #ibusy_bit,@in_hs       ;set IBUSY active

                                        ;test online switch
program  C 0004                         test_sw:
program  C 0004 7EC17310                        btst.b  #pause_sw_bit,@in_hs2   ;test pause switch
program  C 0008 47FA                            beq     test_sw                 ;still low
                                                                                ; will not go further
                                                                                ; until released

                                        ;test hold status
program  C 000A 733C                            btst.b  #ohold_flag,r4l ;test hold status
program  C 000C 470C                            beq     put_on_hold     ;currently not on hold

                                        ;currently on hold
program  C 000E                         put_off_hold:
program  C 000E 7FB27030                        bset.b  #ohold_bit,@stat_port   ;clear Output Hold LED
program  C 0012 723C                            bclr.b  #ohold_flag,r4l ;clear hold status

                                        ;enable delay timer interrupts
program  C 0014 F808                            mov.b   #8,r0l
program  C 0016 3890                            mov.b   r0l,@frt_tier   ;enable delay timer interrupts

program  C 0018 400E                            bra     pause_cont

                                        ;currently off hold
program  C 001A                         put_on_hold:
program  C 001A 7FB27230                        bclr.b  #ohold_bit,@stat_port   ;set Output Hold LED
program  C 001E 703C                            bset.b  #ohold_flag,r4l         ;set hold status

                                        ;disable delay timer interrupts
program  C 0020 F800                            mov.b   #0,r0l
program  C 0022 3890                            mov.b   r0l,@frt_tier   ;disable delay timer interrupts
program  C 0024 7F917230                        bclr.b  #3,@frt_tcsr    ;clear compare flag in case

program  C 0028                         pause_cont:
                                        ;should IBUSY be cleared ?
                                        ;test for online first
program  C 0028 734C                            btst.b  #online_flag,r4l        ;is buffer offline ?
program  C 002A 4708                            beq     pause_ret               ;yes, keep IBUSY set

                                        ;test for buffer full
program  C 002C 731C                            btst.b  #buf_fl_flag,r4l        ;is buffer full ?
program  C 002E 4604                            bne     pause_ret               ;yes, keep IBUSY set

                                        ;clear IBUSY
program  C 0030 7FB27240                        bclr.b  #ibusy_bit,@in_hs       ;set IBUSY inactive

program  C 0034                         pause_ret:
program  C 0034 5670                            rte

                                                .end
*****TOTAL ERRORS        0
*****TOTAL WARNINGS      0
```

Listing 10: BUFFER.INC

```
;H8/330 Printer Buffer Program
;revision 2.0

;Register Usage
;       R0H = Buffer Data
;       R4L = Status Flags
;       R4H = Buffer Margin (16 bytes)
;       R5  = Input Buffer Pointer
;       R6  = Output Buffer Pointer

;buffer control flags (R4L)
online_flag     .equ    4               ;       -4- On Line Flag
ohold_flag      .equ    3               ;       -3- Output Hold Flag
buf_init_flag   .equ    2               ;       -2- Buffer Init Flag
buf_fl_flag     .equ    1               ;       -1- Buffer Full Flag
buf_mt_flag     .equ    0               ;       -0- Buffer Empty Flag

;Port 1 Usage
in_hs           .equ    p1_dr           ;Input Port Control (output)
stat_port       .equ    p1_dr           ;Status Indicators
ibusy_bit       .equ    4               ;       -4- IBUSY (output)
ohold_bit       .equ    3               ;       -3- Output Hold LED (output)
buf_ful_bit     .equ    2               ;       -2- Buffer Full LED (output)
online_bit      .equ    1               ;       -1- On Line LED (output)
ready_bit       .equ    0               ;       -0- Ready LED (output)

;Port 2 Usage
mem_data        .equ    p2_dr           ;Data (input/output)
mem_dir         .equ    p2_ddr
write           .equ    h'ff            ;write direction
read            .equ    0               ;read direction

;Port 3 Usage
addr_lo         .equ    p3_dr           ;Address, Low Byte (output)

;Port 4 Usage
out_hs          .equ    p4_dr           ;Output Port Handshake
oinit_bit       .equ    4               ;       -4- OINIT\ (output)
                                        ;       -1- OSTB\ (TMO0)
obusy_bit       .equ    0               ;       -0- OBUSY (input)

;Port 5 Usage
mem_ctrl        .equ    p5_dr           ;Memory Buffer Control
                                        ;       -2- WE\ (output)
                                        ;       -1- CS1\ (output)
                                        ;       -0- CS0\ (output)
wrcs1           .equ    1               ;write to chip 1
wrcs0           .equ    2               ;write to chip 0
rdcs1           .equ    5               ;read from chip 1
rdcs0           .equ    6               ;read from chip 0

;Port 6 Usage
out_port        .equ    p6_dr           ;Output Port

;Port 7 Usage
in_port         .equ    p7_dr           ;Input Port

;Port 8 Usage
addr_hi         .equ    p8_dr           ;Address, High Byte (output)

;Port 9 Usage
in_hs2          .equ    p9_dr           ;Port Control (inputs)
online_sw_bit   .equ    2               ;       -2- ONLINE (input, IRQ0)
pause_sw_bit    .equ    1               ;       -1- HOLD\ (input, IRQ1)
istb_bit        .equ    0               ;       -0- ISTB\ (input, IRQ2)

;maskable interrupts
online          .equ    0               ;irq0
pause           .equ    1               ;irq1
istb            .equ    2               ;irq2
```

Listing 11: H8330.INC

```
;H8/330 Port Definitions

;I/O Port Address
p1_ddr          .equ            h'ffb0
p2_ddr          .equ            h'ffb1
p3_ddr          .equ            h'ffb4
p4_ddr          .equ            h'ffb5
p5_ddr          .equ            h'ffb8
p6_ddr          .equ            h'ffb9
p8_ddr          .equ            h'ffbd
p9_ddr          .equ            h'ffc0

p1_dr           .equ            h'ffb2
p2_dr           .equ            h'ffb3
p3_dr           .equ            h'ffb6
p4_dr           .equ            h'ffb7
p5_dr           .equ            h'ffba
p6_dr           .equ            h'ffbb
p7_dr           .equ            h'ffbe
p8_dr           .equ            h'ffbf
p9_dr           .equ            h'ffc1

;System Control Registers
syscr           .equ            h'ffc4
mdcr            .equ            h'ffc5
iscr            .equ            h'ffc6
ier             .equ            h'ffc7

;Free-Running Timer
frt             .equ            h'ff90
frt_tier        .equ            h'ff90
frt_tcsr        .equ            h'ff91
frt_frc         .equ            h'ff92
frt_frch        .equ            h'ff92
frt_frcl        .equ            h'ff93
frt_ocra        .equ            h'ff94
frt_ocrah       .equ            h'ff94
frt_ocral       .equ            h'ff95
frt_ocrb        .equ            h'ff94
frt_ocrbh       .equ            h'ff94
frt_ocrbl       .equ            h'ff95
frt_tcr         .equ            h'ff96
frt_tocr        .equ            h'ff97
frt_icra        .equ            h'ff98
frt_icrah       .equ            h'ff98
frt_icral       .equ            h'ff99
frt_icrb        .equ            h'ff9a
frt_icrbh       .equ            h'ff9a
frt_icrbl       .equ            h'ff9b
frt_icrc        .equ            h'ff9c
frt_icrch       .equ            h'ff9c
frt_icrcl       .equ            h'ff9d
frt_icrd        .equ            h'ff9e
frt_icrdh       .equ            h'ff9e
frt_icrdl       .equ            h'ff9f

;Pulse-Width Modulation Timers
pwm0            .equ            h'ffa0
pwm0_tcr        .equ            h'ffa0
pwm0_dtr        .equ            h'ffa1
pwm0_tcnt       .equ            h'ffa2
pwm1            .equ            h'ffa4
pwm1_tcr        .equ            h'ffa4
pwm1_dtr        .equ            h'ffa5
pwm1_tcnt       .equ            h'ffa6

;Multi-Function Timers
tmr0            .equ            h'ffc8
tmr0_tcr        .equ            h'ffc8
tmr0_tcsr       .equ            h'ffc9
tmr0_tcora      .equ            h'ffca
tmr0_tcorb      .equ            h'ffcb
tmr0_tcnt       .equ            h'ffcc
tmr1            .equ            h'ffd0
tmr1_tcr        .equ            h'ffd0
tmr1_tcsr       .equ            h'ffd1
tmr1_tcora      .equ            h'ffd2
tmr1_tcorb      .equ            h'ffd3
tmr1_tcnt       .equ            h'ffd4

;Serial Communications Interface
sci             .equ            h'ffd8
sci_smr         .equ            h'ffd8
sci_brr         .equ            h'ffd9
```

SECTION 5

Listing 11: H8330.INC (continued)

```
sci_scr      .equ        h'ffda
sci_tdr      .equ        h'ffdb
sci_ssr      .equ        h'ffdc
sci_rdr      .equ        h'ffdd

;A/D Converter
adc          .equ        h'ffe0
adc_a        .equ        h'ffe0
adc_b        .equ        h'ffe2
adc_c        .equ        h'ffe4
adc_d        .equ        h'ffe6
adc_adcsr    .equ        h'ffe8
adc_adcr     .equ        h'ffea

;Dual-Port RAM
dpram        .equ        h'fff0
pccsr        .equ        h'fff0
pcdr0        .equ        h'fff1
pcdr2        .equ        h'fff2
pcdr3        .equ        h'fff3
pcdr4        .equ        h'fff4
pcdr5        .equ        h'fff5
pcdr6        .equ        h'fff6
pcdr7        .equ        h'fff7
pcdr8        .equ        h'fff8
pcdr9        .equ        h'fff9
pcdr10       .equ        h'fffa
pcdr11       .equ        h'fffb
pcdr12       .equ        h'fffc
pcdr13       .equ        h'fffd
pcdr14       .equ        h'fffe
pcdr15       .equ        h'ffff

;Memory Definitions
code_start   .equ        h'0040
end_rom      .equ        h'3fff
ram_start    .equ        h'fd80
end_ram      .equ        h'ff7f
top_ram      .equ        h'ff80

;Interrupt Vector Locations
nmi_vec      .equ        h'0006                  ;nmi
irq0_vec     .equ        h'0008                  ;irq0
irq1_vec     .equ        h'000a                  ;irq1
irq2_vec     .equ        h'000c                  ;irq2
irq3_vec     .equ        h'000e                  ;irq3
irq4_vec     .equ        h'0010                  ;irq4
irq5_vec     .equ        h'0012                  ;irq5
irq6_vec     .equ        h'0014                  ;irq6
irq7_vec     .equ        h'0016                  ;irq7
icia_vec     .equ        h'0018                  ;frt input capture a
icib_vec     .equ        h'001a                  ;frt input capture b
icic_vec     .equ        h'001c                  ;frt input capture c
icid_vec     .equ        h'001e                  ;frt input capture d
ocia_vec     .equ        h'0020                  ;frt output compare a
ocib_vec     .equ        h'0022                  ;frt output compare b
fovi_vec     .equ        h'0024                  ;frt overflow
cmi0a_vec    .equ        h'0026                  ;mft0 output compare a
cmi0b_vec    .equ        h'0028                  ;mft0 output compare b
ovi0_vec     .equ        h'002a                  ;mft0 overflow
cmi1a_vec    .equ        h'002c                  ;mft1 output compare a
cmi1b_vec    .equ        h'002e                  ;mft1 output compare b
ovi1_vec     .equ        h'0030                  ;mft1 overflow
mrei_vec     .equ        h'0032                  ;dpram read end
mwei_vec     .equ        h'0034                  ;dpram write end
eri_vec      .equ        h'0036                  ;receive error
rxi_vec      .equ        h'0038                  ;receive data available
txi_vec      .equ        h'003a                  ;transmit buffer empty
adi_vec      .equ        h'003c                  ;a/d complete
```

H8/330

Application Note

Power-Down Operation

Tom Hampton

INTRODUCTION

The H8/330 devices have three different power-down states of operation that significantly reduce power consumption by stopping some (or all) of the on-chip functions. These three modes differ not only for power consumption reduction, but also in how the entry and exit methods. Figure 1 shows a simple flow chart of the processing states for the H8/300 CPU.

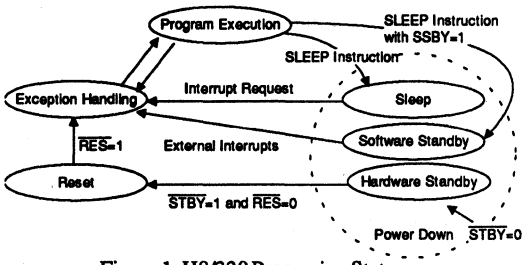


Figure 1: H8/330 Processing States

This flow chart describes the processing sequence for all exception processing as well as power-down modes of operation. Table 1 shows power consumption during all modes of operation while Table 2 shows an overview of the individual modes for power-down operation.

	Typical	Maximum		Frequency
Normal	12	25	mA	6MHz
	16	30	mA	8MHz
	20	40	mA	10MHz
Sleep Mode	8	15	mA	6MHz
	10	20	mA	8MHz
	12	25	mA	10MHz
Standby Modes	0.01	5.0	µA	

Table 1: H8/330 Power Consumption

Mode	Entrance	Clock	CPU	CPU Registers	On-Chip Peripherals	On-Chip RAM	I/O Ports	Exiting
Sleep Mode	Execute SLEEP Instruction	Run	Halt	Held	Run	Held	Held	Interrupt $\overline{RES}$ $\overline{STBY}$
Software Standby Mode	SSBY=1, Execute SLEEP Instruction	Halt	Halt	Held	Halt and Initialized	Held	Held	$\overline{NMI}$ $\overline{IRQ2}$-$\overline{IRQ0}$ $\overline{RES}$ $\overline{STBY}$
Hardware Standby Mode	$\overline{STBY}$ pin Active	Halt	Halt	Not Held	Halt and Initialized	Held	High Impedance	$\overline{STBY}$ high, then pulse $\overline{RES}$

Table 2: H8/330 Power-Down Modes

SECTION 5

HARDWARE STANDBY MODE

The "Hardware Standby" mode of power-down operation is controlled by an external input pin ($\overline{STBY}$) on the H8/330. When the input to the $\overline{STBY}$ pin is made active, the H8/330 enters the hardware standby mode after completion of the current instruction.

Operation of the CPU and all on-chip peripherals are stopped completely during this mode of operation. The system oscillator is also stopped, to reduce power consumption to its minimum, so that no clock is supplied to any of the parts (CPU or peripherals) of the H8/330. Not only is the system clock stopped, but all the I/O ports on the H8/330 are placed into a high-impedance state. This inhibits the I/O ports from dribing or sourcing any external devices.

Only the on-chip RAM of the H8/330 is maintained during this mode of operation. Whatever values are placed into this RAM area are retained while nothing else is saved. (For further power reduction savings while still maintaining RAM data, please refer to the Special Considerations section later in this document.)

The H8/330 device remains in this mode of operation since the $\overline{STBY}$ pin remains active, despite the state of any other inputs (including $\overline{RES}$). The only way to remove the H8/330 from this mode of operation is with the following sequence:

1. release the $\overline{STBY}$ pin to the inactive state,
2. reset the device by pulsing the $\overline{RES}$ pin (see Figure 2 for the timing relationships on performing this function).

Since you are resetting the H8/330, you probably will only use this mode of operation when the H8/330 is used to initialize some external devices and then go to sleep until the external devices requires operations from the H8/330. Because the on-chip RAM is maintained during this mode of operation, you can place software semaphores in the RAM that will allow the initialization routines of the H8/330 to decide whether to do a complete re-initialization (as from a power-on condition) or a re-initialization of only itself (as in waking up from the hardware standby mode).

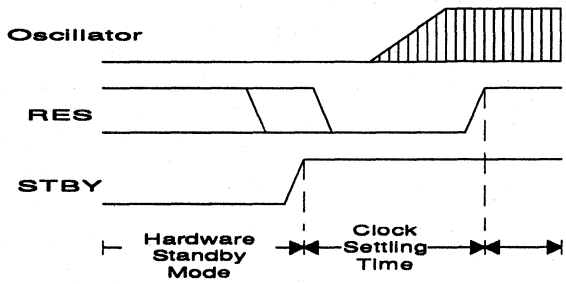


Figure 2: Exiting Hardware Standby Mode

SOFTWARE STANDBY MODE

The software standby mode of operation is very similar to the hardware standby mode, the same power consumption savings are available in either mode. Like the hardware standby mode of operation, the CPU and all on-chip peripherals are stopped completely during the software standby mode. The difference between the two modes is how they are entered and exited, and in how the CPU's registers and the I/O ports are handled.

The software standby mode of operation is controlled via software operation instead of hardware. There are two power-down functions controlled in software by programming the System Control Register (See Figure 3); the entering of the software standby mode and the time delay when leaving the software standby mode.

This mode of operation is entered by setting the "software standby bit" (SSBY) in the System Control Register (SYSCR) and then executing the SLEEP instruction. When the SLEEP instruction is executed, the SSBY bit is tested to find its value. If this bit is not set, then the H8/330 enters the "Sleep" mode of operation (discussed later in this document). If this bit is set, then the H8/330 enters the software standby mode of operation.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

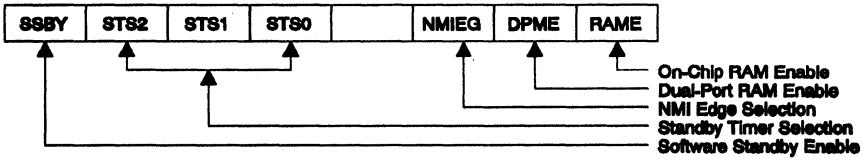


Figure 3: System Control Register

Before executing the SLEEP instruction, the user must program not only the SSBY bit in the SYSCR, but also the "Standby Timer Select" (STS_2-STS_0) bits. Since the on-chip oscillator is stopped during this mode of operation, enough time must be allowed to allow the oscillator to re-start (AC parameter t_{OSC2}). The user can control this time by programming these three bits. By setting them to different values, the user controls how many clock cycles the CPU delays between recognizing the external interrupt signal and starting the exception processing service routine (see Table 3).

Unlike the hardware standby mode, this mode of operation maintains the registers of the CPU. This allows program execution to continue at the location following the SLEEP instruction when the H8/330 is released from the software standby mode. Also during this mode of operation, the I/O ports are maintained in their current states instead of being re-initialized. But, the on-chip peripherals (such as timers, serial channel, etc.) are reset and must be re-initialized whenever the H8/330 is released from software standby mode.

Since the on-chip peripherals are not operating during the software standby mode, it is only external interrupts ($\overline{NMI}$ or $\overline{IRQ_2}$-$\overline{IRQ_0}$) that can awaken the H8/330 and return it to its normal operating sequence. This is handled just like any other exception sequence. The interrupting device is serviced after the oscillator settling time delay by the exception processing routine and operation is returned to the location following the SLEEP instruction.

This mode is probably the most useful of the power-down modes of operation because it offers the most power consumption savings. The CPU and on-chip peripherals are stopped while external devices (and on-chip I/O ports) are still allowed to function. This allows the user to have the rest of his system monitor external events while the CPU remains inactive.

Of course, you can always leave the software standby mode of operation by resetting the H8/330 or by entering the hardware standby mode.

STS2	STS	STS0	Settling Time	System Clock Frequency (MHz)						
				10	8	6	4	2	1	0.5
0	0	0	8192	0.8	1.0	1.3	2.0	4.1	8.2	**16.4**
0	0	1	16384	1.6	2.0	2.7	4.1	8.2	**16.4**	32.8
0	1	0	32768	3.3	4.1	5.5	8.2	**16.4**	32.8	65.5
0	1	1	65536	6.6	8.2	**10.9**	**16.4**	32.8	65.5	131.1
1	-	-	131072	**13.1**	**16.4**	21.8	32.8	65.5	131.1	262.1

Table 3: Standby Timer Select Values

SLEEP MODE

The "Sleep" mode of power-down operation is controlled by software. During this mode, operation of the H8/300 CPU core is halted while the rest of the on-chip functions remain active. Because of this, the "Sleep" mode offers the least amount of power consumption savings.

This mode of operation is controlled by executing the SLEEP instruction during the normal program operation. When this occurs, the H8/300 CPU is placed into a "halt" state with no further activity taking place. This halt state is similar to the situation where the CPU may be in an indefinite "wait" state except that no control signals are active.

Since the CPU is halted, no change in the I/O ports will occur (meaning that their current values will be held). Though the CPU is no longer running, all values in the registers are held in their current state. By doing this, the CPU is allowed to continue its operations directly from the location following the SLEEP instruction (after processing a return from the sleep mode).

Though the CPU is halted, the system clock is still allowed to run. This means that the on-chip peripherals can still function; the timers, the serial channel, the A/D converter, and the Dual-Port RAM can still do all their normal operations. In fact the H8/330 device gets out of the sleep mode of operation.

Whenever any of the on-chip peripherals generate an interrupt or an external interrupt is input to the device, the CPU is awakened from its sleep mode and processing continues as normal (see Figure 1 for flow details). The interrupting device is serviced during the exception processing routine and operation is returned to the location following the SLEEP instruction.

Like the Software Standby Mode, you can always leave the sleep mode of operation by resetting the H8/330 or by entering the hardware standby mode.

SPECIAL CONSIDERATIONS

RAM Retention

The H8/330 also offers the ability for the user to maintain the contents of the on-chip RAM and CPU registers with a low voltage input to the device.

During either of the standby modes (hardware or software) of operation, the user can drop his supply voltage to +2.0 volts DC and still be assured that the contents of the on-chip RAM will not be lost. To use this feature correctly, the user must ensure that he disables the on-chip RAM (by clearing the RAME bit in the SYSCR) just before entering the standby mode. While in the standby modes of operation, the user can now reduce his supply voltage (thus further reducing current consumption in his system). During the software standby mode of operation, the user cannot only maintain the RAM contents but also the contents of the CPU's registers while the low voltage is applied.

Before releasing the H8/330 from either standby mode of operation, it is the responsibility of the user to ensure that the proper operating voltage (V_{CC}=+5.0V ±10%) be available.

External Oscillator

In most systems (or microcontrollers), it is the oscillator that is the main concern when attempting to reduce power consumption. Though peripheral and CPU functions are stopped, since the oscillator continues to operate small power savings are observed. The H8/330 overcomes this concern by providing its own on-chip oscillator that is stopped during the standby mode of operation.

If your system uses an external oscillator to drive the H8/330 device and you still wish to enjoy the power consumption savings that the H8/330 offers, you still can. In instances such as this, the H8/330 would accept the external clock input and stop the internal clock from being provide to the on-chip peripherals during the power-down modes. Here the oscillator stabilization time (AC parameter t_{osc2}) becomes effectively 0 ms. You can now program the Standby Timer Select bits in the SYSCR to "000" to reduce the delay when coming out of the software standby mode to its absolute minimum.

APPLICATION EXAMPLE

SOFTWARE STANDBY MODE

In this example, we will use the $\overline{NMI}$ input to suggest when the H8/330 should be in a power-down state. Since the $\overline{NMI}$ input is high, we would like the H8/330 to continue normal operations. When the NMI input goes low, we want to enter the software standby mode. This is possible because we can sense both edges of the $\overline{NMI}$ input on the H8/330. For the sake of programming the Standby Timer Select bits, lets assume that the H8/330 is operating at a clock frequency of 6MHz. In discussion of the software, we will talk only about programming that is required and not discuss peripheral initialization at all (refer to Figure 4 for a flow chart of the operations sequence).

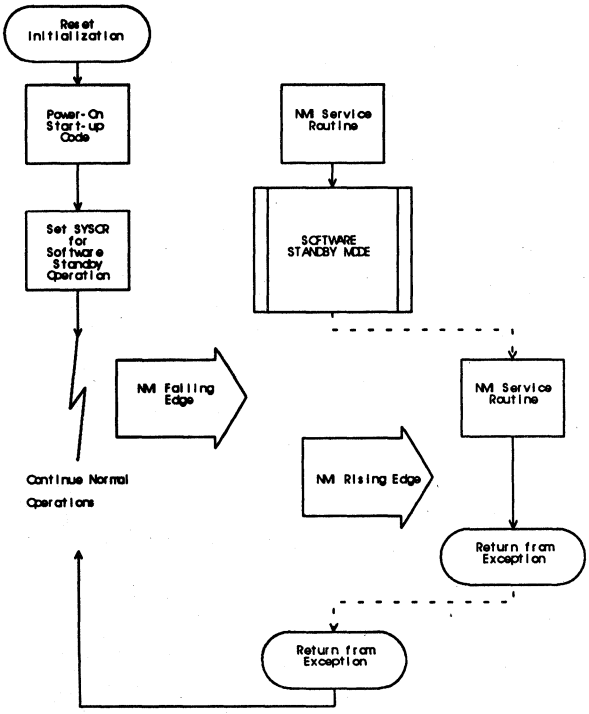


Figure 4: Application Example Processing Flowchart

During the normal operating sequence, the H8/330 would go through the process of initializing all its peripherals and other functions for normal operation. Since the System Control Register defaults to having the $\overline{NMI}$ edge selection for falling edge, no programming of that bit is necessary at this time. We will take this opportunity to program the SYSCR for the proper STS values. We know that the t_{OSC2} value is 10 msec from the AC characteristics of the H8/330. This calculates out to 60,000 t-states at 6MHz. To allow for this number of clock cycles, we must program STS_2-STS_0 to a value of "011." This will allows 10.9 msec to elapse for oscillator stabilization.

Whenever the falling edge of the $\overline{NMI}$ signal is recognized, the H8/330 will begin the processing of the NMI exception processing service routine. During this service routine we must do three basic operations; figure out whether we are going into or out of software standby mode, change the state of the $\overline{NMI}$ edge selection, and execute the SLEEP instruction (if we are going into the standby mode). Optionally we could also enable or disable the on-chip RAM if we were going to reduce voltage to the H8/330. After that we would return from this exception processing service routine to our normal operation (a flow chart of the NMI service routine is shown in Figure 5).

For our discussion of the software, please refer to Listing 1. In the main routine, the only thing we really need to do is to program the SYSCR with the values necessary for the $\overline{NMI}$ edge selection and the standby timer selection (for oscillator stabilization time). Initially we want to capture the falling edge of the $\overline{NMI}$ input and set the STS bits for a count of 65536. This requires the programming of "101110X1" into the SYSCR (refer to Figure 3 for a description). With this programmed into the SYSCR, we can continue with our normal processing.

Whenever the falling edge of the $\overline{NMI}$ signal is observed (see Figure 6), the H8/330 will begin processing the NMI exception processing service routine. Since this routine must handle both placing the H8/330 into the software standby mode as well as recovering from it, we must first decide which one it is. To do this we can test the state of the NMIEG bit. If this bit is a "0," then we can assume that we have detected the falling edge and that we are going to go

into the software standby mode. Before we execute the SLEEP instruction we would need to program the NMIEG bit to "1" so that we can now monitor for the rising edge of the $\overline{NMI}$ signal. Optionally, if we are going to reduce the V_{cc} level we would need to clear the RAME bit in the SYSCR now before we execute the SLEEP instruction.

After executing the SLEEP instruction the H8/330 is now in the software standby mode of operation awaiting the input of the rising edge on the $\overline{NMI}$ signal. When the rising edge is detected (see Figure 7), the H8/330 starts the internal counter for the standby timer and delays further processing until the counter has timed out. At this point the H8/330 begins processing the NMI exception processing service routine again.

We still need to test the NMIEG bit to decide whether we are going into the standby mode or coming out of it. If this bit is a "1," then we can assume that we are coming out of the standby mode. Here, we would want to change the NMI edge selection from rising edge to falling edge. If we had disabled the on-chip RAM, we would want to make sure that we re-enabled it for use. Afterward we merely return from the NMI service routine (which incidentally returns us to the NMI service routine that we were performing to go into standby mode).

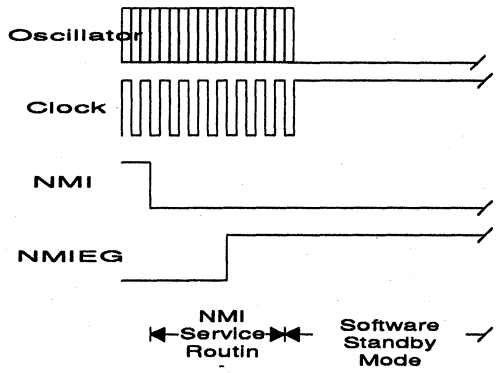


Figure 6: Application Example
NMI Falling Edge Timing

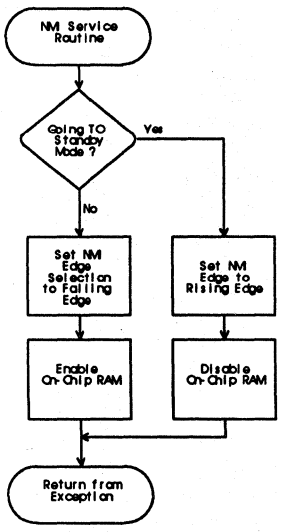


Figure 5: Application Software Flowchart
(NMI Service Routine)

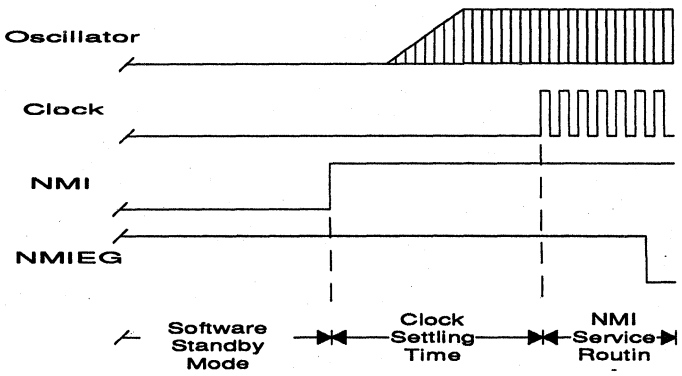


Figure 7: Application Example
NMI Rising Edge Timing

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Listing 1: Application Example NMI Service Routine

```
;H8/330 Power-Down Application Example

;NMI Service Routine

nmi_service:
            btst.b      #2,@h'ffc4          ;test nmieg bit in SYSCR
            beq         falling_edge        ;going into power-down

;coming out of power-down
rising_edge:
            bclr.b      #2,@h'ffc4          ;set nmieg for falling edge
            bset.b      #0,@h'ffc4          ;enable on-chip RAM
            rte                             ;return from processing
                                            ; to previous NMI routine

;going into power-down
falling_edge:
            bset.b      #2,@h'ffc4          ;set nmieg for rising edge
            bclr.b      #0,@h'ffc4          ;disable on-chip RAM
            sleep                           ;go to power-down mode
            rte                             ;return from processing
                                            ; to normal operation

            .end
```

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H8/3XX Instruction Timing

Tech Notes

Application Engineering

Carol Jacobson

While benchmarks can provide a good estimate of a controllers CPU performance, they seldom give us enough information to determine a part's suitability for a particular application or the relative performance of a peripheral. A good approximation of a controllers ability to execute a particular function, within an allotted time, can be obtained by adding module overhead (interrupt latency, A/D conversion, serial bit rates) to the time required to execute the modules driver routine. Using information shown in the H8/300 Series Programming Manual, instruction execution times for any combination of addressing modes and memory access types can be calculated.

Instruction Fetch

H8/3xx devices have three possible data paths: a 16-bit internal data bus for R0 to R7 and on-chip memory, an 8-bit internal bus for on-chip peripherals and an 8-bit external bus. The H8/300 CPU uses a 16-bit word instruction set. The number of cycles needed to fetch an instruction equals the number words in the instruction times the number of cycles needed to fetch each word. This later value depends upon the data path used. These numbers are given in Tables C1 and C2 of Appendix C in the Programming Manual. For Example:

From on-chip ROM:

	hex code	# words	x	#clocks / 16-bits =	instruction fetch time
mov.b r0l,@h'2000	6A88 2000	2		2	4 clks

From external memory: wait states (m) cam be included for access to slower peripherals requiring additional time to complete the transfer:

	hex code	# bytes	x	#clocks / 8-bits =	instruction fetch time
mov.b r0l,@h'2000	6A88 2000	4		3 + m	12 + 4m clks

Execution

Like the instruction fetch, the number of cycles needed to execute each part of the instruction depends upon the operation and memory location. Additional cycles, represented in table C2 as columns J through N, can be defined as follows:

H8/3XX Instruction Timing

Branch Address Read: Cycles needed to fetch the destination address during an 8-bit indirect jump or jump to subroutine (JMP,JSR @@aa:8).

Stack Operation: Additional cycles for incrementing or decrementing the stack pointer and storing the program counter on the stack.

Byte Data Access: Time required to obtain non-immediate (or indirect) 8-bit data or address locations.

Word data Access: Time required to obtain non-immediate (or indirect) 16-bit data or address locations.

Internal Operation: Additional cycles for arithmetic address or data calculations.

The total instruction cycle time is the sum of the instruction fetch time plus any additional cycle time needed to complete execution of the instruction.

Appendix B of the H8/300 Programming Manual gives the number of clock cycles for each instruction, for all supported addressing modes, when all operations are on-chip. For instructions fetched from off-chip memory, timing can be calculated from table C1 and C2 values.

Note:Table C1, On-chip Reg. Field, refers to on-chip I/O and module registers not to registers R0 - R7 or the CCR.

Table Calculations

From tables C1 & C2:

			(# of cycles)	(clks/cyc)	
		hex code	I	Si	instruction fetch time
int:	mov.b r0l,@h'2000	6A88 2000	2	2	4 clks
ext:	mov.b r0l,@h'2000	6A88 2000	4	3 + m	12 + 4m clks

Examples

1. a. MOV.B @R1+,R1H where the instruction resides in off-chip memory requiring no wait states and R1 contains an off-chip address value.

from C2	from C1	
I= 1	Si= 6 + 2 x 0 = 6	;instruction fetch
L= 1	Sl= 3 + 2 x 0 = 3	;indirect address access cycle
N= 1	Sn= 2	;time to increment R1l

Total = (1 x 6) + (1 x 3) + (1 x 2) = 11 clocks = 1.1 us @ 10MHz

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

b. If the same instruction had been stored on-chip and the address in R1 was on-chip RAM, from App. B we read:

6 states or clocks = 0.6 us @ 10MHz

or: I=L=N= 1 Si= Sl= Sn= 2

Total = (1 x 2) + (1 x 2) + (1 x 2) = 6 clocks

2. a. BCLR @R3L,@H'8031 where the instruction resides in external ROM requiring 1 wait state and the destination is off-chip RAM requiring one wait state.

from C2	from C1	
I= 2	Si= 6 + 2 x 1 = 8	;off chip access
L= 2	Sl= 3 + 2 x 1 = 5	;byte access to off chip memory

Total = (2 x 8) + (2 x 5) = 26 clocks = 2.6 us @ 10 MHz

b. BCLR #03,@FRT_TCR where the instruction resides in external ROM requiring 1 wait state and the destination is the on-chip register field.

from C2	from C1	
I= 2	Si= 6 + 2 x 1 = 8	;fetch from off chip memory
L= 2	Sl=3	;byte access to on chip register field

Total = (2 x 8) + (2 x 3) = 22 clocks = 2.2us @ 10 MH

H8/320 Family Device EPROM Security

Tech Notes

Application Engineering

Tom Hampton

EPROM Security

The H8/320 Family of microcontrollers (except the H8/324 which is a masked programmed device only) have an EPROM security feature that can be used by the customer. This feature allows the user of the microcontroller to protect parts (or all) of the code programmed into the on-chip EPROM of the H8/320 from being read by means other than his own program. This feature cannot be tested by Hitachi and, due to this, is unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

Memory Configuration

The memory matrix of the H8/320 Family of microcontrollers is configured as a dual matrix, one with even addresses and the other with odd addresses. The configuration of each matrix appears as lines of memory 32 bytes wide (32 x 8, 256 bits). This configuration allows an individual memory line to consist of 64 bytes of data (including both even and odd addresses). Each memory line has 1 security bit thus allowing every 64 byte segment to have the option of the security feature. The address of this security bit is the same as the starting address for the memory line.

Security Functions

The security function had two different operations depending upon the mode of operation that the H8/320 Family device is placed into, EPROM programming mode or CPU operation mode.

EPROM Programming Mode

In the EPROM programming mode, the ability of the EPROM programmer to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read. If the security bit is a "0" (programmed), then any read operation to the EPROM will result in a "00" being read. This indicates that once the security bit is programmed, the user will be unable to verify the contents of the EPROM.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

security bit 1 EPROM data can be read (normal)
security bit 0 "00" data is always read

CPU Operating Modes

In the CPU operating modes, the ability of any device to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read by the CPU. If the security bit is a "0" (programmed), then the read state of the EPROM (from the CPU), depends upon where instruction execution is occurring from.

security bit 1 EPROM data can be read by CPU (normal)
security bit 0 After RESET, the CPU can read EPROM data until it executes an instruction outside the internal EPROM area (either external memory or internal RAM). Once an instruction is executed outside the internal EPROM memory area, then the EPROM becomes disabled and cannot be accessed any further. This prohibits an external program from being able to "dump" the contents of the on-chip EPROM.

Programming the Security Bit

There exists two EPROM programming mode; Normal and Security. The normal EPROM programming mode is used to program the code/data area of the on-chip EPROM memory for the H8/320 device. The "security" programming mode is used to program the security bits of the EPROM's memory area. The security function is then implemented by programming a "0" into the address corresponding to the memory line location. Setting the programming mode is done by setting certain I/O port pins to the following states:

	H8/320 Device I/O Port Pin	
Programming Mode	$P7_0/\overline{IS}$	$P7_1/\overline{OS}$
Normal	1	1
Security	1	0

Again, this feature cannot be tested by Hitachi and thus remains unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

SECTION 5

H8/350 Device EPROM Security

Tech Notes

Application Engineering

Tom Hampton

EPROM Security

The H8/350 Microcontroller has an EPROM security feature that can be used by the customer. This feature allows the user of the microcontroller to protect parts (or all) of the code programmed into the on-chip EPROM of the H8/350 from being read by means other than his own program. This feature cannot be tested by Hitachi and, due to this, is unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

Memory Configuration

The memory matrix of the H8/350 Microcontroller is configured as a dual matrix, one with even addresses and the other with odd addresses. The configuration of each matrix appears as lines of memory 32 bytes wide (32 x 8, 256 bits). This configuration allows an individual memory line to consist of 64 bytes of data (including both even and odd addresses). Each memory line has 1 security bit thus allowing every 64 byte segment to have the option of the security feature. The address of this security bit is the same as the starting address for the memory line.

Security Functions

The security function had two different operations depending upon the mode of operation that the H8/350 device is placed into, EPROM programming mode or CPU operation mode.

EPROM Programming Mode

In the EPROM programming mode, the ability of the EPROM programmer to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read. If the security bit is a "0" (programmed), then any read operation to the EPROM will result in a "00" being read. This indicates that once the security bit is programmed, the user will be unable to verify the contents of the EPROM.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

security bit	1	EPROM data can be read (normal)
security bit	0	"00" data is always read

CPU Operating Modes

In the CPU operating modes, the ability of any device to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read by the CPU. If the security bit is a "0" (programmed), then the read state of the EPROM (from the CPU), depends upon where instruction execution is occurring from.

security bit	1	EPROM data can be read by CPU (normal)
security bit	0	After RESET, the CPU can read EPROM data until it executes an instruction outside the internal EPROM area (either external memory or internal RAM). Once an instruction is executed outside the internal EPROM memory area, then the EPROM becomes disabled and cannot be accessed any further. This prohibits an external program from being able to "dump" the contents of the on-chip EPROM.

Programming the Security Bit

There exists two EPROM programming mode; Normal and Security. The normal EPROM programming mode is used to program the code/data area of the on-chip EPROM memory for the H8/350 device. The "security" programming mode is used to program the security bits of the EPROM's memory area. The security function is then implemented by programming a "0" into the address corresponding to the memory line location. Setting the programming mode is done by setting certain I/O port pins to the following states:

	H8/350 Device I/O Port Pin	
Programming Mode	$P8_0/RS_0/E$	$P8_1/RS_1/\overline{IOS}$
Normal	1	1
Security	1	0

Again, this feature cannot be tested by Hitachi and thus remains unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

H8/330 Microcontroller EPROM Security

Tech Notes

Application Engineering

Tom Hampton

EPROM Security

The H8/330 Microcontroller has an EPROM security feature that can be used by the application programmer. This feature allows the user of the microcontroller to protect parts (or all) of the code programmed into the on-chip EPROM of the H8/330 from being read by means other than his or her own program. This feature cannot be tested by Hitachi and, due to this, is unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

Memory Configuration

The memory matrix of the H8/330 Microcontroller is configured as a dual matrix, one with even addresses and the other with odd addresses. The configuration of each matrix appears as lines of memory 32 bytes wide (32 x 8, 256 bits). This configuration allows an individual memory line to consist of 64 bytes of data (including both even and odd addresses). Each memory line has 1 security bit thus allowing every 64 byte segment to have the option of the security feature. The address of this security bit is the same as the starting address for the memory line.

Security Functions

The security function had two different operations depending upon the mode of operation that the H8/330 device is placed into, EPROM programming mode or CPU operation mode.

EPROM Programming Mode

In the EPROM programming mode, the ability of the EPROM programmer to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read. If the security bit is a "0" (programmed), then any read operation to the EPROM will result in a "00" being read. This indicates that once the security bit is programmed, the user will be unable to verify the contents of the EPROM.

security bit	1	EPROM data can be read (normal)
security bit	0	"00" data is always read

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

CPU Operating Modes

In the CPU operating modes, the ability of any device to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read by the CPU. If the security bit is a "0" (programmed), then the read state of the EPROM (from the CPU), depends upon where instruction execution is occurring from.

security bit 1 EPROM data can be read by CPU (normal)

security bit 0 After RESET, the CPU can read EPROM data until it executes an instruction outside the internal EPROM area (either external memory or internal RAM). Once an instruction is executed outside the internal EPROM memory area, then the EPROM becomes disabled and cannot be accessed any further. This prohibits an external program from being able to "dump" the contents of the on-chip EPROM.

Programming the Security Bit

There exists two EPROM programming mode; Normal and Security. The normal EPROM programming mode is used to program the code/data area of the on-chip EPROM memory for the H8/330. The "security" programming mode is used to program the security bits of the EPROM's memory area. The security function is then implemented by programming a "0" into the address corresponding to the memory line location. Setting the programming mode is done by setting certain I/O port pins to the following states:

	H8/330 I/O Port Pin	
Programming Mode	P80	P81
Normal	1	1
Security	1	0

Again, this feature cannot be tested by Hitachi and thus remains unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

H8/300 CPU DIVXU Instruction

Tech Notes

Application Engineering

Tom Hampton

The H8/300 CPU provides an instruction to perform a 16/8 divide operation to yield an 8-bit result. The H8/300 Programming Manual incorrectly describes the flag results during the execution of this instruction. The text of the instruction states the following:

> "Valid results are not assured if division by zero is attempted or an overflow occurs. Division by zero is indicated in the Zero flag. Overflow can be avoided by the coding shown on the next page."

This is in error. While it is true that valid results cannot be assured if the division by zero is attempted or an overflow should occur, it is incorrect in stating that the Zero flag will indicate that a divide by zero operation was attempted. The text should read:

> "Valid results (remainder, quotient, and flag operation) are not assured if division by zero is attempted or an overflow occurs. Overflow can be avoided by the coding shown on the next page."

The text for the flag description currently states:

> Z: Set to "1" if the divisor is zero; otherwise cleared to "0."

This text should read:

> Z: Unpredictable if the divisor is zero; otherwise cleared to "0."

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H8/300 CPU SUBX Instruction

Tech Notes

Application Engineering

Tom Hampton

The H8/300 CPU provides an instruction for subtracting two bytes from each other along with the value of the Carry flag. This instruction is useful when performing subtraction operations that are greater than 16-bits (an instruction is already available that can do either 8-bit or 16-bit subtractions with no problems). Lets take the example of a 32-bit subtraction as follows:

```
  H'40000000
- H'3F8F2356
  H'0070DCAA          (result)
```

If we look are each operation individually, the result is easily explained.

1. In subtracting the low order bytes from each other (56 from 00), we get a result of AA with a borrow from the next higher byte.
2. In subtracting the next higher order bytes from each other (23 from FF because of the borrow), we get a result of DC with the borrow continuing to the next higher byte.
3. In subtracting the next higher order bytes from each other (8F from FF because of the borrow), we get a result of 70 with the borrow continuing to the next higher byte.
4. In subtracting the highest order bytes from each other (3F from 3F because of the borrow), we get a result of 00 with no borrow.

If no borrow operations were never to occur, then we could code this very simply with two word subtract operations. But since this is not the case, we must code the sequence so as to keep track of the borrow operations. If we code this in the same sequence as the operation described above, it might look something like this:

```
mov.w    #h'4000,r1
mov.w    #0,r2
mov.w    #h'3f8f,r3
mov.w    #h'2356,r4
sub.b    r2l,r4l          ;00-56
subx     r2h,r4h          ;00-23-borrow
subx     r1l,r3l          ;00-8f-borrow
subx     r1h,r3l          ;40-3f-borrow
```

(We could also replace the first two subtraction instructions with a subtract word operation to reduce code size and execution time but this method makes it easier to read for now.)

```
sub.w    r2,r4            ;0000-2356
subx     r1l,r3l          ;00-8f-borrow
subx     r1h,r3l          ;40-3f-borrow
```

H8/300 CPU SUBX Instruction

The trick in using the SUBX instruction is to pay attention to the flag operations. During execution of a normal subtraction operation, the Zero flag is used to determine if the result of the operation is zero or not. However, the execution of the SUBX instruction is a little bit different. If the result of the operation is zero, then the Zero flag remains unchanged from the previous instruction. If the result is non-zero, then the Zero flag is cleared to correctly indicate a non-zero result. While this sounds a lot like what it is supposed to be, look at a scenario where the previous instruction would clear the zero flag (this may be something as simple as a MOV instruction). If the SUBX instruction were to follow this operation and the result were zero, then the Zero flag would remain at "0," clearly not indicating the result of the operation.

Because of this, it is extremely important that the SUBX instruction be used **IMMEDIATELY** following other SUB or SUBX instructions. This sequence allows the H8/300 CPU to properly keep track of borrows, and maintain the Zero flag in the correct state. To illustrate this problem, lets assume that our variables are stored in memory rather than registers. In this example, we have to move the data into our registers in order to perform the operation.

```
    var1    .equ    H'40000000
    var2    .equ    H'3f8f2356
            mov.b   @(var2+3),rlh       ;get 56
            mov.b   @(var1+3),rll       ;get 00
1.          sub.b   rlh,rll             ;00-56, Zero=0, Carry=1
            mov.b   rll,@(var1+3)       ;store 1st result (AA), Zero=0, Carry=1
            mov.b   @(var2+2),rlh       ;get 23, Zero=0, Carry=1
            mov.b   @(var1+2),rll       ;get 00, Zero=1, Carry=1
2.          subx    rlh,rll             ;00-23-borrow, Zero=1, Carry=1
            mov.b   rll,@(var1+2)       ;store 2nd result (DC), Zero=0, Carry=1
            mov.b   @(var2+1),rlh       ;get 8F, Zero=0, Carry=1
            mov.b   @(var1+1),rll       ;get 00, Zero=1, Carry=1
3.          subx    rlh,rll             ;00-8F-borrow, Zero=1, Carry=1
            mov.b   rll,@(var1+1)       ;store 2nd result (70), Zero=0, Carry=1
            mov.b   @(var2),rlh         ;get 3F, Zero=0, Carry=1
            mov.b   @(var1),rll         ;get 40, Zero=0, Carry=1
4.          subx    rlh,rll             ;40-3F-borrow, Zero=0, Carry=0
            mov.b   rll,@(var1)         ;store 2nd result (00), Zero=1, Carry=0
```

Since the result of the SUB and SUBX operations are not 00 (until number 4), the flags behave as we wish them to. During the 4th subtraction operation, the Zero flag remains clear even though the result of the subtraction operation was zero. While this is the correct flag value for the entire operation, it would not be correct if we test the flag after the MOV instruction.

Lets change our variables so that the result of the subtraction operation should be zero (H'40000000 - H'40000000) and again follow the code sequence. Since each of the substraction operations (SUBX) would result in a zero value, the contents of the Zero flag would remain as it was prior to the execution of the instruction. In the 4th subtraction operation, we need only look at the values transfered by the MOV instructions. Since both values are non-zero in nature, then the contents of the Zero flag would be cleared as a result of that instruction. This allows the final value of the Zero flag (before the MOV instruction) to be 0, and this would be incorrect for the entire operation. Of course it would be correct after the MOV operation, but our previous example showed it to be opposite of this example.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

```
        mov.b   rl1,@(var1+1)       ;store 2nd result (70), Zero=0, Carry=1
        mov.b   @(var2),rlh         ;get 40, Zero=0, Carry=1
        mov.b   @(var1),rl1         ;get 40, Zero=0, Carry=1
4.      subx    rlh,rl1             ;40-3F-borrow, Zero=0, Carry=0
        mov.b   rl1,@(var1)         ;store 2nd result (00), Zero=1, Carry=0
```

In the final analysis, to make things much simpler for the user, it is recommended that the SUBX instructions always follow other subtraction instructions **IMMEDIATELY**. The resulting Zero flag status would also the user to determine the status of his complete result.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Section 5

H8 Family
H8/5XX Series

SECTION 5

HITACHI®

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H8/500 CPU

Application Note

Technical Q & A

How to Use Microcomputer Technical Questions and Answers

Technical Questions and Answers has been created by arranging technical questions actually asked by users of Hitachi microcomputers in a question-and-answer format. It should be read for technical reference in conjunction with the User's Manual.

Technical Questions and Answers can be read before beginning a microcomputer application design project to gain a more thorough understanding of the microcomputer, or during the design process to check up on difficult points.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Contents

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 001B
Topic	Register contents after power-up reset		

Question

1. What are the CPU register contents after a power-up reset?

Classification—H8/500

O	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. In minimum mode, the program counter is loaded from the vector table. The interrupt mask bits (I_2, I_1, I_0) in the status register (SR) are set to 1, and the trace bit (T) is cleared to 0. Registers R0 to R7, the base register (BR), and the other SR bits have undetermined values.

   In maximum mode the code page register (CP) is loaded from the vector table. Other page registers have undetermined values. Registers other than the page registers are the same as in minimum mode.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 002B
Topic	Page registers in single-chip mode and expanded minimum modes		

Question

1. Can the DP, EP, and TP page registers be used as data registers in the single-chip mode and expanded minimum modes?

Classification—H8/500

○	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Yes, but since the page registers are control registers, they can only be accessed by system control instructions (LDC, STC).

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 036A
Topic	DP contents in unconditional jump within page		

Question

1. If the JMP @R0 unconditional in-page jump instruction is executed in expanded maximum mode, are the data page (DP) register contents used in calculating the effective address?

Classification—H8/500

O	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The DP contents are not used in calculating the effective address of an unconditional jump within the same page.

   If the JMP @R0 instruction is executed to jump within the same page, the R0 contents are loaded into the program counter (PC), but the code page (CP) register value does not change. The DP contents are therefore ignored.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 004B
Topic	Interrupt sampling and acceptance		

Question

1. When are external interrupts (NMI, IRQ_n) sampled?

Classification—H8/500

	Registers
	Read timing
	Write timing
O	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Level-sensitive interrupts (IRQ_0) are sampled on the rising edge of the system clock. Edge-sensitive interrupts (external interrupts other than IRQ_0) are sampled on the falling edge of the system clock.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 006B
Topic	Holding of disabled external interrupts		

Question

1. In the following two cases, are external interrupts (IRQ_n) held pending?

   (1) IRQ_n enable bit is cleared to 0 in on-chip register field

   (2) IRQ_n interrupt priority level ≤ interrupt mask level set in status register (SR)

Classification—H8/500

	Registers
	Read timing
	Write timing
O	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. (1) In this state, the interrupt request signal is not sampled and the interrupt is not held pending. Interrupt requests made in this state will be ignored even if the IRQ_n enable bit is later set to 1.

   (2) An interrupt that is requested in this state is held pending in the CPU's interrupt controller. If the interrupt request mask level is later reduced to a value lower than the external (IRQ_n) interrupt priority level, the interrupt will be accepted.

   IRQ_0 is level-sensitive, however, so it is not held pending.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

The interrupt request mask level is set in bits I_2 to I_0 in the status register (SR).

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 008A
Topic	Disabling of invalid instruction exceptions		

Question

1. Can exception handling of invalid instructions be disabled? How does the exception handling routine terminate?

Classification—H8/500

	Registers
	Read timing
	Write timing
❍	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. No, it cannot be disabled.

   The invalid instruction exception handler cannot be terminated by returning with an RTE instruction. Use some other software technique, such as jumping to the reset routine.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 028A – 1
Topic	Interrupt contention while waiting for instruction execution to end		

Question

1. Suppose an interrupt occurs during execution of an instruction, then during the waiting state before the instruction ends another, higher-priority interrupt occurs. Which interrupt does the CPU accept?

	Classification—H8/500
	Registers
	Read timing
	Write timing
O	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The CPU accepts the interrupt with the highest priority level four states before the time of acceptance. (See the next page.) The interrupt mask level in bits I_2 to I_0 is not changed until the status register (SR) has been saved onto the stack.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

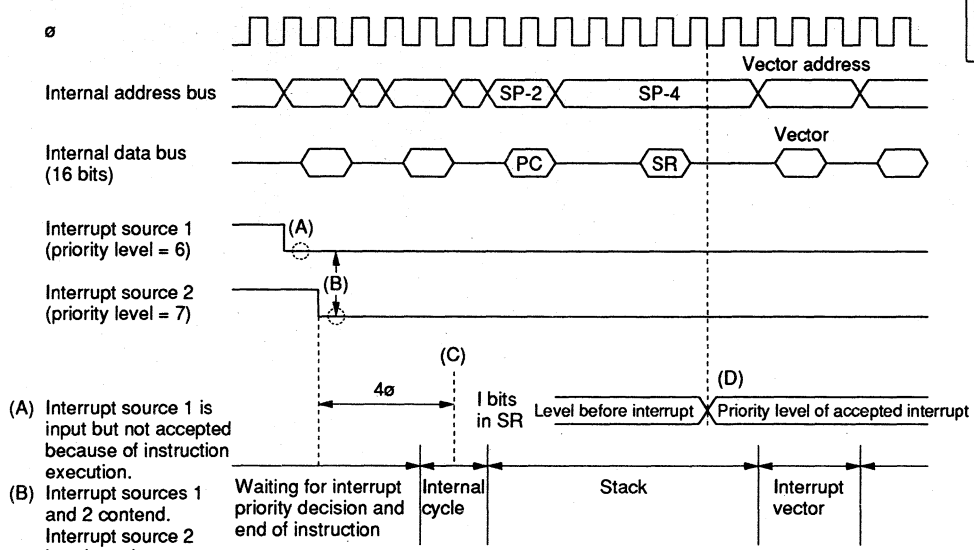
Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 028A – 2
Topic	Interrupt contention while waiting for instruction execution to end		

Answer



(A) Interrupt source 1 is input but not accepted because of instruction execution.
(B) Interrupt sources 1 and 2 contend. Interrupt source 2 is selected.
(C) The instruction being executed ends. The CPU accepts interrupt source 2.
(D) The I bits in the status register are changed to level 7.

Note: Conditions: minimum mode with the program and stack areas both in on-chip memory and interrupt handler starting at an even address.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 030A – 1
Topic	Time of clearing of IRQ_n interrupt request signal		

Question

1. There are no interrupt request flags for edge-sensitive external interrupts (IRQ_n). When are these requests cleared?

Classification—H8/500

	Registers
	Read timing
	Write timing
O	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The interrupt request is cleared during the internal cycle in which the interrupt is accepted, as indicated by the arrow in the diagram on the next page. If the same interrupt request signal (IRQ_n) occurs after this time, it will be latched again.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

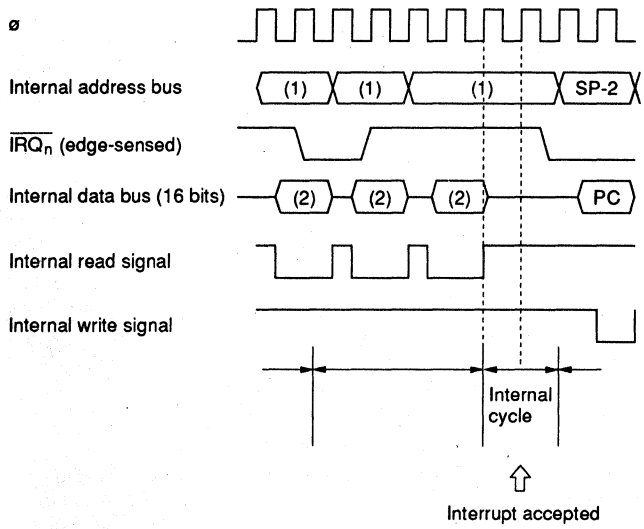
Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 030A – 2
Topic	Timing of clearing of IRQ_n interrupt request signal		
Answer			



(1) Instruction prefetch address (2) Instruction code

Taken from the User's Manual

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 031A
Topic	Requirements for enabling interrupts		

Question

1. Why do we fail to get an interrupt even though the interrupt request enable bit (IRQ_nE) is set to 1 and the interrupt request signal ($\overline{IRQ_n}$) is asserted?

Classification—H8/500

	Registers
	Read timing
	Write timing
O	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. To enable interrupts to be accepted, software must:

   (1) Set the interrupt enable bits for the desired interrupt sources to 1.

   (2) Set values in the interrupt priority registers (IPRs).

   (3) Set the desired interrupt request mask level in bits I_2 to I_0 in the status register (SR).

   Check the above points.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

A reset initializes all IPR values to 0 and sets bits I_2 to I_0 all to 1, masking all interrupts except NMI.

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 032A
Topic	Maximum wait after $\overline{\text{BREQ}}$		

Question

1. What is the maximum waiting time from input of an external bus request signal ($\overline{\text{BREQ}}$) until the CPU replies ($\overline{\text{BACK}}$)?

Classification—H8/500

	Registers
	Read timing
	Write timing
O	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The maximum waiting time is 10 to 17 states. This occurs if the CPU started executing the MOVFPE or MOVTPE instruction (which transfers data in synchronization with the E clock) just before $\overline{\text{BREQ}}$ was asserted. Because MOVTPE and MOVFPE execute in synchronization with the E clock, the number of states varies depending on the timing of the start of execution.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 034A
Topic	Clearing of interrupt request enable bits and pending interrupts		

Question

1. While an IRQ_n interrupt is being held pending because its priority is equal to or less than the interrupt request mask level in the status register (SR), does clearing the IRQ_n enable bit (IRQ_nE) also clear the IRQ_n interrupt request?

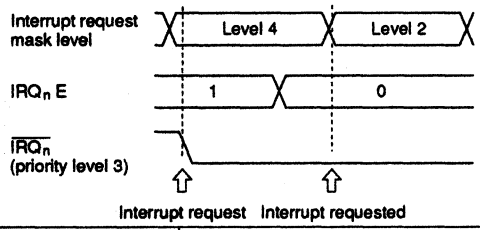
Classification—H8/500

	Registers
	Read timing
	Write timing
O	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. When an IRQ_n interrupt request is held pending because of the interrupt request mask level (I_2 to I_0), the request remains pending even if IRQ_nE is cleared to 0.

   The IRQ_n interrupt will be accepted later when the interrupt request mask level is reduced to a value less than the IRQ_n priority level.



Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

IRQ_0 is level-sensitive, so it is not held pending, regardless of whether IRQ_0E is set or cleared.

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 035A
Topic	Acceptance of NMI during NMI handling		

Question

1. NMI has the highest priority and is always accepted. During the NMI interrupt handling routine, if another NMI interrupt occurs will it also be accepted?

Classification—H8/500

	Registers
	Read timing
	Write timing
○	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. If another NMI request is made during the NMI interrupt handling routine, the second request will also be accepted.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 009B – 1
Topic	NMI sampling and acceptance immediately after a reset		

Question

1. When is the $\overline{\text{NMI}}$ signal first sampled after a reset?

Classification—H8/500	
	Registers
	Read timing
	Write timing
	Interrupts
❍	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Sampling of the $\overline{\text{NMI}}$ signal starts from the first falling edge of the system clock at which the reset signal is high. The NMI interrupt becomes acceptable when the first instruction has been executed after the chip comes out of reset.

(See next page)

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

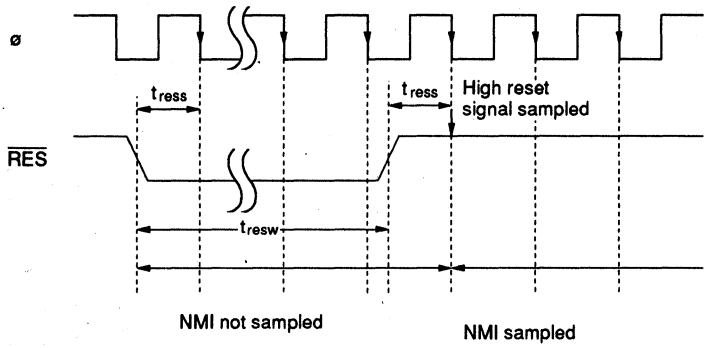
Additional Information

The reset and NMI signals are both sampled on the falling edge of the system clock.

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 009B – 2
Topic	NMI sampling and acceptance immediately after a reset		
Question			

[Example]



HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 010B
Topic	Stack pointer initialization immediately after a reset		

Question

1. Why is it necessary to initialize the stack pointer immediately after a reset?

Classification—H8/500

	Registers
	Read timing
	Write timing
	Interrupts
O	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. If the $\overline{\text{NMI}}$ request signal is active when the chip comes out of reset, the NMI interrupt will be accepted as soon as the first instruction has been executed. To prevent program crashes, you should therefore initialize the stack pointer immediately after the reset.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 037A
Topic	Pin states at power-up reset		

Question

1. What needs to be noted about pin states at a power-up reset?

Classification—H8/500

	Registers
	Read timing
	Write timing
	Interrupts
O	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. At a power-up reset, the mode pins (MD_2 to MD_0) must be tied to the desired mode setting and the $\overline{STBY}$ pin must be held high. Output from the ø and E pins is unpredictable until the clock oscillator settles into steady oscillation.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

When using a microcontroller that multiplexes the ø and E pins with general-purpose input ports, connect a resistor with a resistance of several kilohms in series with these pins.

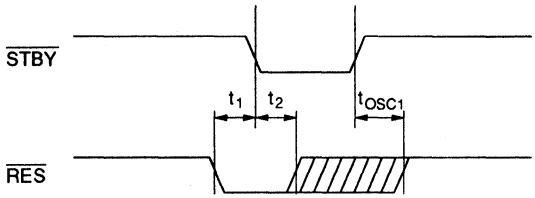
HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 011B
Topic	Hardware standby mode entry timing		

Question

1. Are there any restrictions on times t_1 and t_2 in the diagram below for entering hardware standby mode?



Classification—H8/500

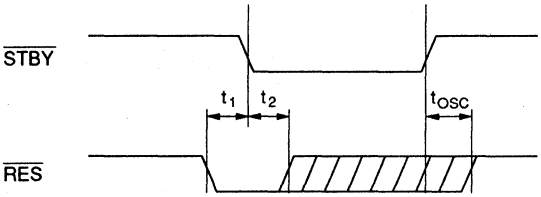
	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
O	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The following restrictions apply.

   (1) To hold RAM contents, t_1 must be at least 10 system clock cycles. The minimum value of t_2 is 0 ns.

   (2) When it is not necessary to hold RAM contents, there is no restriction on t_1 and t_2.



Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 013B
Topic	Instruction execution at changeover to hardware standby mode		

Question

1. When a low $\overline{STBY}$ input drives the chip into hardware standby mode, what happens to the instruction currently being executed?

Classification—H8/500

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
O	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The instruction being executed is aborted, without being completed. Normal execution of the instruction is not assured.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 014B
Topic	Mode pins in hardware standby mode		

Question

1. What happens if the states of the mode lines (MD_2 to MD_0) are changed during hardware standby mode?

Classification—H8/500

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
O	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Hardware standby mode will not operate correctly. Do not change the state of the mode lines during hardware standby mode.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 016B
Topic	Recovery from hardware standby mode		

Question

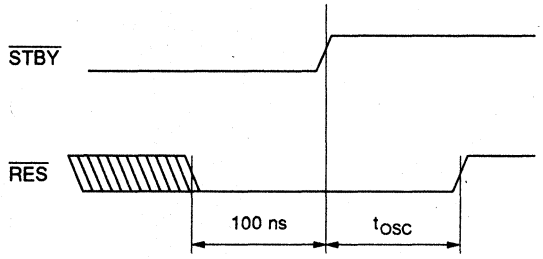
1. The chip must be recovered from hardware standby mode by holding $\overline{RES}$ low, then driving $\overline{STBY}$ high. How long before $\overline{STBY}$ goes high does $\overline{RES}$ have to go low?

Classification—H8/500

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
O	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. To recover from hardware standby mode, drive $\overline{RES}$ low at least 100 ns before driving $\overline{STBY}$ high.



Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 019B
Topic	Notes on entering sleep mode		

Question

1. Are there any points to note about entering sleep mode?

Classification—H8/500

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
O	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. The points listed below should be noted, depending on the method used to recover from sleep mode.

Recovery Method

NMI Interrupt	IRQ_n Interrupt
Clear all interrupt enable bits to 0, or set bits I_2 to I_0 in SR all to 1.	Set bits I_2 to I_0 in SR to a level lower than the priority level of the interrupt used for recovery, clear interrupt enable bits to 0 except for interrupts used for recovery, and make sure NMI is not requested.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 020B
Topic	Interrupts during fetching and execution of SLEEP instruction		

Question

1. What happens if an interrupt is accepted while the SLEEP instruction is being executed?

Classification—H8/500

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
O	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Sleep mode is released to handle the interrupt. At the end of interrupt handling, the next instruction after the SLEEP instruction is executed.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 021B
Topic	Sampling and acceptance of interrupts during sleep mode		

Question

1. When are external interrupts sampled during sleep mode?
2. If an interrupt is sampled, how many system clock cycles later does the chip wake up?

Classification—H8/500

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
O	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Level-sensitive interrupts (IRQ_0) are sampled on the rising edge of the system clock and edge-sensitive interrupts (external interrupts other than IRQ_0) are sampled on the falling edge of the system clock, just as in active mode.
2. The chip exits sleep mode six system clock cycles after the interrupt is sampled.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 027A
Topic	Execution time for entering software standby mode		

Question

1. How many states does it take to enter software standby mode by executing the SLEEP instruction?

Classification—H8/500

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
O	Power-down state
	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. Two states.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 023B
Topic	BRN instruction		

Question

1. What sort of instruction is BRN (or BF)?

Classification—H8/500

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
O	Instructions
	Software
	Development tools
	Miscellaneous

Answer

1. BRN is similar to a NOP instruction, but it has a different byte length and executes in a different number of states. See below.

		Byte Length	Number of States Required for Execution
BRN	d: 8	2	3*
	d: 16	3	3*
NOP		1	2*

* When instruction is fetched from on-chip ROM

BRN has the same byte length as Bcc, for example, which makes it useful in debugging.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 033A
Topic	Reserved addresses in interrupt vector area		

Question

1. Can the reserved addresses in the interrupt vector area be used to store program code?

Classification—H8/500

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
O	Software
	Development tools
	Miscellaneous

Answer

1. Yes, they can.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500 CPU	Q&A No.	QA8500 - 029A
Topic	Access to on-chip registers while bus is released		

Question

1. When the H8/500 CPU releases the bus to an external device, can the external device (bus master) access the H8/500's on-chip registers?

Classification—H8/500

	Registers
	Read timing
	Write timing
	Interrupts
	Reset
	External expansion
	Power-down state
	Instructions
	Software
	Development tools
O	Miscellaneous

Answer

1. No. On-chip registers cannot be accessed externally under any circumstances.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H8/500 Series

Application Note

Technical Q & A

Preface

The H8/500 Series is a series of highly integrated single-chip microcontrollers. Their CPU core has an internal 16-bit architecture, and each chip includes diverse high-performance peripheral hardware.

These technical questions and answers relate to the H8/510, H8/520, H8/532, H8/534, and H8/536.

H8/500 Family

Item			H8/510	H8/520	H8/532	H8/534	H8/536
CPU			H8/500	H8/500	H8/500	H8/500	H8/500
Memory	ROM	Masked ROM	—	16 kbytes	32 kbytes	32 kbytes	62 kbytes
		ZTAT™*2	No	Yes	Yes	Yes	Yes
	RAM		—	512 bytes	1 kbyte	2 kbytes	2 kbytes
Address space (bytes)			16 M	1 M	1 M	1 M	1 M
External data bus width (bits)			8/16	8	8	8	8
Timers	16-bit free-running timer		2 ch	2 ch	3 ch	3 ch	3 ch
	8-bit timer		1 ch	1 ch	1 ch	1 ch	1 ch
	Watchdog timer		1 ch	1 ch	1 ch	1 ch	1 ch
	PWM timer		—	—	3 ch	3 ch	3 ch
Serial communication interface (async/sync)			2 ch	2 ch	1 ch	2 ch	2 ch
A/D converter	External trigger input		10 bits, 4 channels, trigger	10 bits, 4 or 8* channels, trigger	10 bits, 8 channels, no trigger	10 bits, 8 channels, no trigger	10 bits, 8 channels, no trigger
Interrupts	External interrupts		5	9	3	7	7
	Internal interrupts		18	18	19	23	23
I/O ports			60	50/54*1	65	65	65
Packages			QFP-112	DILC-64S (windowed)	LCC-84 (windowed)	LCC-84 (windowed)	LCC-84 (windowed)
				DILP-64S	PLCC-84	PLCC-84	PLCC-84
				PLCC-68*1	QFP-80	QFP-80	QFP-80
				QFP-64			

Notes: 1. PLCC-68 package
2. ZTAT™ is a registered trademark of Hitachi, Ltd.

SECTION 5

How to Use These Technical Questions and Answers

Technical Questions and Answers has been created by arranging technical questions actually asked by users of Hitachi microcomputers in a question-and-answer format. It should be read for technical reference in conjunction with the User's Manual.

Technical Questions and Answers can be read before beginning a microcomputer application design project to gain a more thorough understanding of the microcomputer, or during the design process to check up on difficult points.

(For questions and answers about the H8/500 CPU, see *H8/500 CPU Microcomputer Technical Questions and Answers*.)

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Contents

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 001B
Topic	Address bus, data bus, and control line states during access to on-chip address space		

Question

1. What values are output on the following lines when on-chip memory or the on-chip register field is accessed?

   (1) Address bus

   (2) Data bus

   (3) Bus control signals

Classification—H8/500	
	Software
O	On-chip ROM
O	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

(1) The address bus carries the address data, regardless of whether the access is to an on-chip or off-chip address.

(2) The data bus is in the high-impedance state for both read and write access by the CPU to an on-chip address.

(3) The R/$\overline{W}$ signal is low for write access and high for read access. The other control signals ($\overline{AS}$, $\overline{DS}$, $\overline{RD}$, $\overline{WR}$) are high.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/536	Q&A No.	QA500 - 046A
Topic	Programming the H8/536 ZTAT		

Question

1. We are having trouble programming the ZTAT version of the H8/536. Are there any precautions we may be missing?

Classification—H8/536

	Software
O	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. When programming the H8/536, you must set your PROM writer to memory type HN27C101 and either write H'FF data in addresses H'F680 to H'1FFFF or set H'F67F as the end address.

   Be sure to use byte programming mode. The H8/536 does not support page programming.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

Some PROM writers do not support byte programming for the HN27C101.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 002B
Topic	EXTAL and system clock output line		

Question

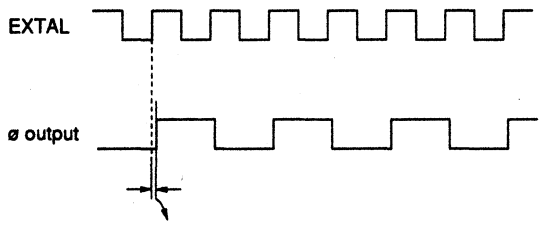
1. During external clock input, what is the phase relationship between EXTAL and the system clock output line (ø output)?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
O	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. During external clock input, the phase relationship between EXTAL and the system clock output line is as shown below.



Approx. 40 ns internal delay

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

The internal delay value is not guaranteed.

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 047A
Topic	External clock specifications		

Question

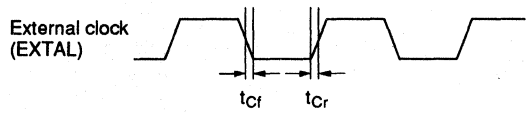
1. When an external clock is supplied to the EXTAL pin, what are the rise-time and fall-time requirements?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
O	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. For a 20-MHz clock, the rise time (t_{Cr}) and fall time (t_{Cf}) should both be approximately 5 ns.



Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

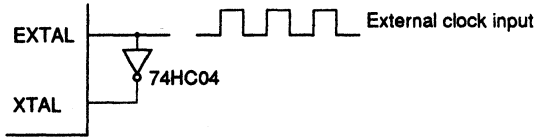
HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/520, 532, 534, 536	Q&A No.	QA500 - 003B
Topic	External clock input		

Question

1. For external clock input, the Hardware Manual shows an example of a circuit using a 74HC04 (see below). Why is the 74HC04 necessary?



Classification—H8/532

	Software
	On-chip ROM
	On-chip RAM
O	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. If the XTAL pin open is left open, operation may become unstable.

   The 74HC04 is necessary to assure stable operation at high clock rates.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

Note: The XTAL pin can be left open if the external clock rate is 16 MHz or less. For masked-ROM versions and the H8/510, the XTAL pin can be left open for external clock rates up to 20 MHz.

SECTION 5

Technical Question and Answer

Product	H8/520, 532, 534, 536	Q&A No.	QA500 - 048A
Topic	External clock input (2)		

Question

1. The H8/500 Series User's Manuals (except H8/510) show a circuit using a 74HC04 for external clock input. (See diagram on previous page.) Can an ALS-TTL, for example, be used instead?

Classification—H8/532

	Software
	On-chip ROM
	On-chip RAM
O	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. An ALS-TTL device can be used if its propagation delay time and drivability are equivalent to the 74HC04.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 006B
Topic	External clock input to 16-bit FRT		

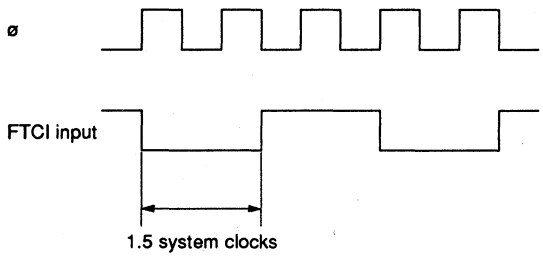
Question

1. When the external clock source is selected for the 16-bit free-running timer, what is the minimum pulse width of the external clock (FTCI)?

Classification—H8/500	
	Software
	On-chip ROM
	On-chip RAM
	Clock
O	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. The minimum pulse width of the external clock is 1.5 system clock cycles.



Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 007B
Topic	Input capture signal for 16-bit FRT		

Question

1. If an FRT input capture line (FTI) is multiplexed with a general-purpose input/output port that is used for output, will the rise and fall of the output data update the input capture register?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
O	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. Yes. The input capture register will be updated by output on the input/output line, on the edge selected by the input edge select bit (IEDG) in the timer control/status register (TCSR).

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 009B – 1
Topic	Access timing to FRC in 16-bit FRT		

Question

1. What is the read and write timing of the free-running counter (FRC) in the 16-bit free-running timer (FRT)?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
O	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. The access timing of the 16-bit timer's FRC is shown on the next page.

   Word access (or two successive byte accesses) should be used. The upper byte has to be accessed first.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

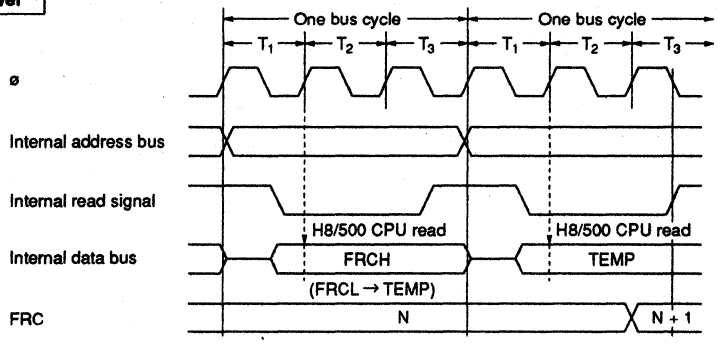
Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 009B – 2
Topic	Access timing to FRC in 16-bit FRT		

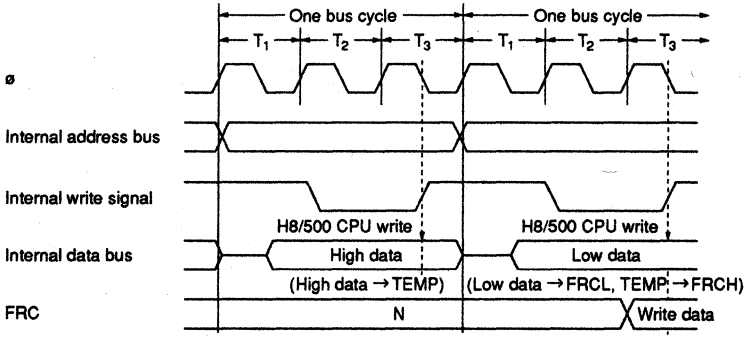
Answer



FRC Access Timing (read)

Operation when register is read

When the upper byte is read, the upper byte value is passed to the CPU and the lower byte value is transferred to TEMP. Next, when the lower byte is read, the lower byte value in TEMP is passed to the CPU.



FRC Access Timing (write)

Operation when register is written

When the upper byte is written, the upper byte value is stored in TEMP. Next, when the lower byte is written, it is combined with the upper byte value in TEMP and all 16 data bits are written in the register.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 011B
Topic	TCNT of 8-bit timer		

Question

1. When a compare-match signal clears the timer counter (TCNT) to H'00, does TCNT remain at H'00, or does it start counting up from H'00?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
O	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. TCNT starts counting up from H'00.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 012B
Topic	WDT when system clock stops		

Question

1. If the system clock stops, will the watchdog timer (WDT) detect anything wrong?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
O	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. If the system clock for the whole chip stops, the WDT count also stops, so the WDT cannot detect the failure.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/532	Q&A No.	QA500 - 013B
Topic	NMI requested by WDT		

Question

1. How can you distinguish between an NMI interrupt requested from the NMI pin and an NMI interrupt requested by the watchdog timer (WDT)?

Classification—H8/532

	Software
	On-chip ROM
	On-chip RAM
	Clock
O	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. When the WDT requests an NMI interrupt, it sets the overflow bit (OVF) in the WDT timer status/control register (TCSR) to 1. You can detect this by software.

	OVF Bit in TCSR
NMI requested by input signal from pin	0
NMI requested by WDT	1

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

When the WDT is used in interval timer mode, IRQ_0 interrupts can be discriminated in the same way. (H8/520, H8/532)

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 018B
Topic	Input/output designation of SCI clock pin		

Question

1. When the SCI is used, is the serial clock pin designated for input or output by writing a 0 or 1 in the data direction register (DDR) of the corresponding port?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
○	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. When you use the SCI, the input or output setting of the clock line depends on the communication mode bit (C/$\overline{A}$.) in the serial mode register (SMR) and the clock enable 1 and 0 bits (CKE1 and CKE0) in the serial control register (SCR). You don't have to set the DDR.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 019B
Topic	Serial I/O line status		

Question

1. After input/output ports multiplexed with TxD, RxD, and SCK lines have been used for serial communication, suppose they are redesignated as I/O ports by settings made in the serial control register (SCR) or serial mode register (SMR).

   What values will the corresponding data direction register (DDR) contain?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
O	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. SCI operations do not affect the contents of the DDR bits of input/output ports. Given the conditions you describe, the DDR bits will retain the values they had before the pins were used for serial communication.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 021B – 1
Topic	RDRF bit set timing		

Question

1. When data reception is completed, the receive data register full bit (RDRF) in the serial status register (SSR) is set to 1. At what timing does this occur in asynchronous mode?

2. At what timing does this occur in clocked synchronous mode?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
O	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

See the next page.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

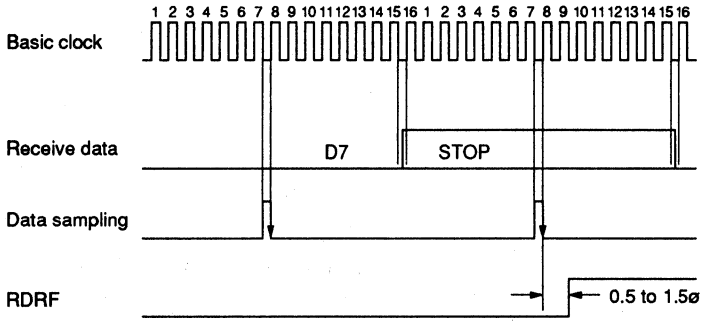
HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 021B – 2
Topic	RDRF bit set timing		

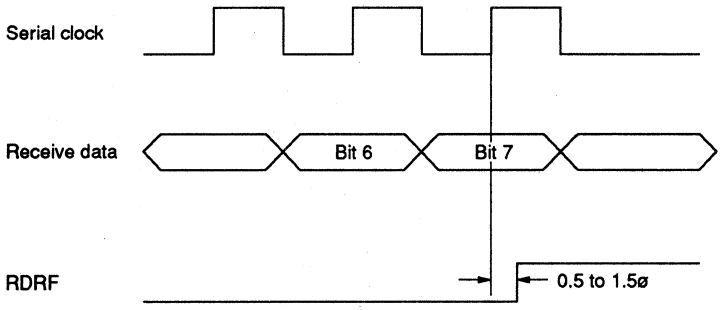
Answer

1. The RDRF bit is set to 1 after the fall of the next data sampling clock after the MSB of the data is received. (See the diagram below.)



8-Bit Data, 1 Stop Bit, Internal Clock

2. The RDRF bit is set to 1 after the rising edge of the serial clock cycle in which the MSB of the data is received. (See the diagram below.)



8-Bit Data

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 022B – 1
Topic	TDRE bit set timing		

Question

1. When eight data bits have been transmitted, the transmit data register empty bit (TDRE) in the serial status register (SSR) is set to 1. At what timing does this occur in asynchronous mode?

2. At what timing does this occur in clocked synchronous mode?

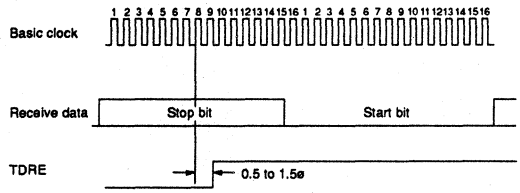
	Classification—H8/500
	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
O	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

The TDRE bit is set to 1 at different times depending on whether the transmit shift register (TSR) contains transmit data or not.

1. Asynchronous mode

   1.1 Transmit data present in TSR (see diagram below)



   The timing of the start of transmission after the transmit enable bit (TE) is set is similar.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

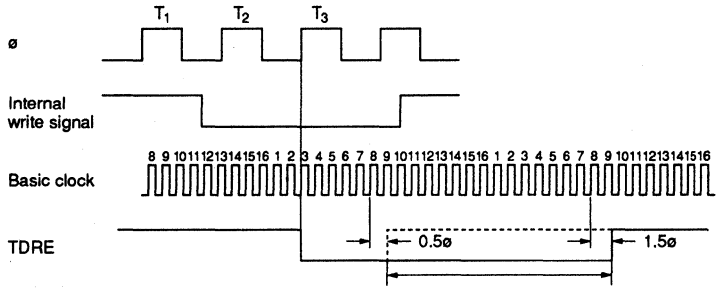
Continued on next page.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 022B – 2
Topic	TDRE bit set timing		
Answer			

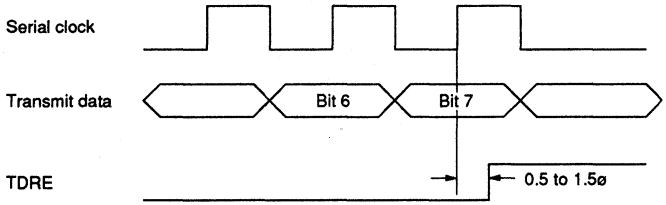
1.2 No transmit data in TSR (see diagram below)



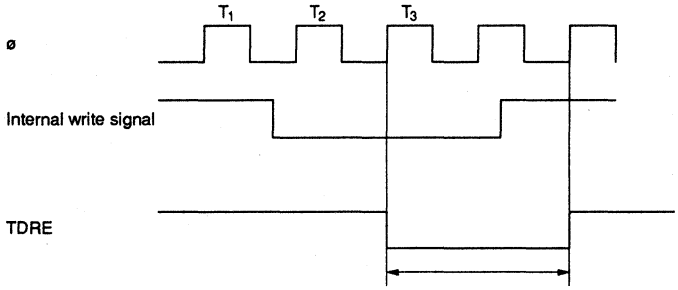
TDRE is set in interval from 8 basic clocks + 0.5ø to 24 basic clocks + 1.5ø

2. Clocked synchronous mode

2.1 Transmit data present in TSR (see diagram below)



2.2 No transmit data in TSR (see diagram below)



TDRE is set in interval from 2ø to 0.5 basic clock + 1.5ø

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 023B
Topic	RDR and DTR utilization when SCI is not used		

Question

1. When the serial communication interface is not used, can the following be utilized as data registers?

   (1) RDR (receive data register)

   (2) TDR (transmit data register)

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
O	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. The answer is as follows:

   (1) RDR is a read-only register, so it cannot be used as a data register.

   (2) TDR can be used as a data register.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 049A
Topic	RDRF bit in SCI		

Question

1. To receive serial data, the receive data register full bit (RDRF) in the serial status register (SSR) must be cleared to 0. What happens if 0 is written in the bit directly, without first reading 1?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
O	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. The RDRF bit retains its 1 value and is not cleared to 0. An overrun error occurs at completion of receiving the next data.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

Similar considerations apply to the transmit data register empty bit (TDRE).

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 050A
Topic	SCI receive error 1		

Question

1. If the receive-error interrupt handler returns to the main program without clearing the overrun flag (ORER), framing error flag (FER), or parity error flag (PER) in the serial status register (SSR) to 0, will a receive error occur again?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
❍	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. After one more instruction is executed in the main program the receive error will occur again, because the error flag itself is the interrupt source.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

This holds for all on-chip supporting modules, excluding only the external interrupts.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 051A
Topic	SCI receive error 2 (clocked synchronous mode)		

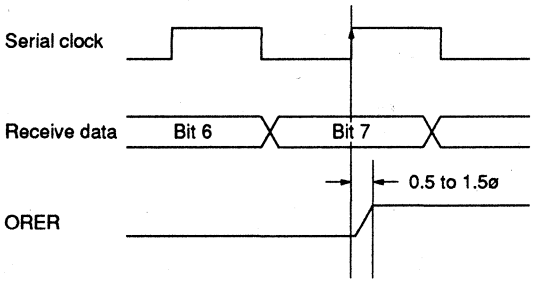
Question

1. When the SCI is used in clocked synchronous mode, at what time is an overrun error detected?

Classification—H8/500	
	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
O	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. The overrun error bit (ORER) is set to 1 after the rise of the serial clock when the most significant data bit (bit 7) is received.



Reception of 8-Bit Data

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 052A
Topic	SCI RxD input example (asynchronous mode)		

Question

1. Suppose the RxD pin is being used as an input port and is now low. Do any precautions have to be taken in order to switch this pin over to its RxD function and receive serial data correctly?

2. Do any precautions have to be taken in order to receive data correctly after detecting the break condition?

Classification—H8/532

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
O	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. Change the RxD input to high before setting the SCI's receive enable bit (RE) to 1.

2. Before reception of the first data, supply high input to the RxD line for at least one frame.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 053A
Topic	SCI transmit start (asynchronous mode)		

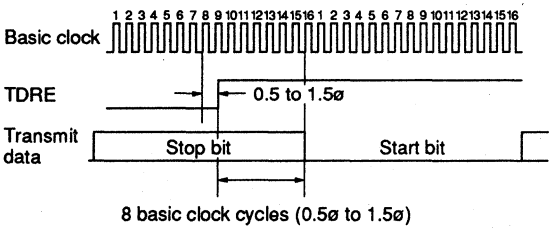
Question

1. In the SCI transmitting sequence, following the transfer of data from TDR to TSR, the transmit data register empty bit (TDRE) in the serial status register (SSR) is set to 1, then the SCI starts transmitting data. How much delay is there from the time when the TDRE bit is set to 1 until output of the start bit?

Classification—H8/500	
	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
O	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. The delay time is eight basic clock cycles (0.5ø to 1.5ø). See the diagram below.



Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

The same timing applies when transmission starts from the setting of the transmit enable bit (TE).

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 054A
Topic	Simultaneous transmit/receive in clocked synchronous mode		

Question

1. During simultaneous transmitting and receiving in clocked synchronous mode, can data be transferred in the state when an overrun error has occurred?

Classification—H8/500	
	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
O	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. Data cannot be transferred.

   In simultaneous transmitting and receiving in clocked synchronous mode, transmitting or receiving cannot proceed independently before the ORER and TDRE bits are both cleared to 0.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 055A
Topic	Clearing the SCI's TDRE bit		

Question

1. When transmitting data, will there be any data transfer problem if we wait until after writing transmit data in the transmit data register (TDR) to read the 1 value of the TDRE bit, then clear this bit to 0?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
O	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	External expansion
	Development tools
	Miscellaneous

Answer

1. No problem will occur.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

If you write in TDR while the TDRE bit is 0, however, you will destroy the previous TDR data.

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 024B
Topic	Start of A/D conversion		

Question

1. Software can select the start of A/D conversion by setting the A/D start bit (ADST) in the A/D control/status register (ADCSR) to 1. What happens if 1 is written in the ADST bit again while A/D conversion is in progress?

2. What happens if A/D conversion starts by detection of the falling edge of the external trigger signal ($\overline{\text{ADTRG}}$), then $\overline{\text{ADTRG}}$ goes high while A/D conversion is in progress?

   (H8/510, H8/520, H8/534, H8/536)

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
O	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. If the ADST bit is set to 1 again during A/D conversion, it will be ignored and A/D conversion will continue.

2. Operation will be normal if the $\overline{\text{ADTRG}}$ signal is low for at least 1.5 cycles. After that, if the $\overline{\text{ADTRG}}$ signal goes high again during A/D conversion, it will be ignored and A/D conversion will continue.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 025B
Topic	Non-use of A/D converter reference voltage lines (AV_{CC}, AV_{SS})		

Question

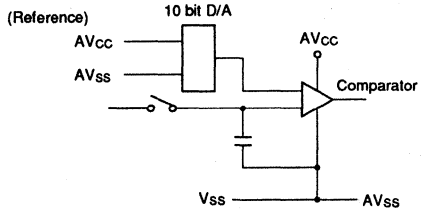
1. When the A/D converter is not used, what should be done with the AV_{CC} and AV_{SS} pins?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
O	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. Even when the A/D converter is not used, AV_{CC} should be connected to V_{CC} and AV_{SS} to V_{SS}.



(1) If AV_{CC} is left open, voltage potentials in the interface to the digital circuits in the A/D converter will be unstable.

(2) AV_{SS} and V_{SS} are shorted inside the chip. Any potential difference between them will cause excessive current drain.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 027B
Topic	Changing A/D conversion mode or channels during conversion		

Question

During A/D conversion, what happens if you:

1. Change the A/D conversion mode?
2. Change the channel selection?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
O	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. Avoid changing the A/D conversion mode during A/D conversion. Conversion accuracy will be degraded.
2. Avoid changing the channel selection during A/D conversion. The same problem will occur as in 1.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

Note: Check the A/D end flag (ADF) in the A/D control/status register (ADCSR), then:
1. Change the A/D conversion mode.
2. Select the channel(s).

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 028B
Topic	Resistor ladder in A/D converter		

Question

1. Are the analog power supplies of the A/D converter connected only to the resistor ladder?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
O	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. The analog power supplies are connected not only to the resistor ladder but also to analog circuits in the comparator etc. They also power the interface to digital circuits in the A/D converter.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 029B
Topic	Rise time of power supplies (AV_{CC}, V_{CC})		

Question

1. Will any problems occur if there is a difference in rise times between the analog power supply (AV_{CC}) and digital power supply (V_{CC})?

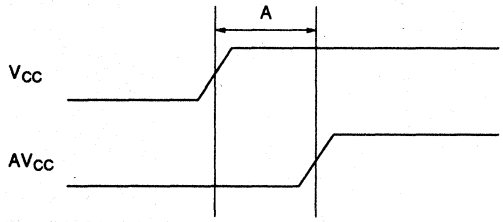
Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
O	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. There is no restriction on the order in which AV_{CC} and V_{CC} are powered up.

   During the interval marked A in the diagram below, voltage potentials in the interface to digital circuits in the A/D converter are unstable, which may cause fluctuations in current drain.



Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 056A
Topic	Allowable impedance of A/D signal sources		

Question

1. Does the allowable signal source impedance remain 10 kΩ even if the A/D conversion time is changed?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
O	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. The low-speed conversion mode should operate even at 20 kΩ, but this is not guaranteed.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/532, H8/534, H8/536	Q&A No.	QA500 - 031B
Topic	DTR of PWM timer		

Question

1. The duty register (DTR) of the PWM timer is set to H'00 for pulses with 0% duty cycle, H'7D for pulses with 50% duty cycle, and H'FA for pulses with 100% duty cycle, but what if a value from H'FB to H'FF is written in DTR?

Classification—H8/532

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
O	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. If a value from H'FB to H'FF is written in DTR, pulses are output with a 100% duty cycle.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/534, H8/536	Q&A No.	QA500 - 057A
Topic	PWM pin assignments		

Question

1. The PWM timer outputs (PW_1 to PW_3) are can be assigned to $P6_1$ to $P6_3$ (multiplexed with $\overline{IRQ_3}$ to $\overline{IRQ_5}$) or $P9_2$ to $P9_4$ (multiplexed with SCK_2, RxD_2, and TxD_2). Can all six pins be used for PWM output?

Classification—H8/534

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
O	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. Yes, they can.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

$P6_1$ to $P6_3$ can be used for both PWM output and IRQ input. $P9_2$ to $P9_4$ can be used for either PWM output or SCI functions, but not both.

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 032B
Topic	Interrupts during DTC operation		

Question

1. During operation of the data transfer controller (DTC), what happens if an interrupt is requested with a priority higher than the interrupt the DTC is serving?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
	PWM
O	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. While the DTC is operating the CPU halts, so no other interrupts can be accepted.

   The DTC therefore completes its interrupt service, after which one instruction is executed; then the pending interrupt-handling sequence begins.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

If the instruction executed after the conclusion of DTC operations is LDC or another instruction that inhibits interrupts, the interrupt-handling sequence will not start until the next instruction after that has been executed (and if that next instruction also inhibits interrupts, another instruction will be executed).

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 033B
Topic	DTC usage		

Question

1. Can DTC register information be located on ROM?

2. After a DTC data transfer, the data transfer count register (DTCR) is decremented by 1, and if the result is 0, the DTC will no longer be activated. If DTC register information is stored on ROM with the DTCR value set to 1, will an interrupt occur after the DTC data transfer?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
	PWM
O	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	External expansion
	Development tools
	Miscellaneous

Answer

1. DTC register information can be located on ROM.

2. An interrupt will be generated. The decision as to whether DTCR = 0 is made when the DTCR value is decremented.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 035B
Topic	Analog input port data register during A/D conversion		

Question

1. During A/D conversion, what happens to the values in the data register (DR) of the input port that is also used for analog input?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
O	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. Pins used for analog input return the value 1 if read during A/D conversion, regardless of the actual input voltage.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 037B
Topic	Port output after reset		

Question

1. To use an input/output port line to output data after a reset, which should be set first: the port's data register (DR) or its data direction register (DDR)?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
O	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. Set these registers in the following order.

   (1) Set the output data in the output port's data register.

   (2) Set the DDR bit of the output line to 1.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

Note: A reset initializes the port data registers to 0.

SECTION 5

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 039B
Topic	$\overline{AS}$ and $\overline{RD}$ signal timing		

Question

1. Are the $\overline{AS}$ and $\overline{RD}$ signals synchronized with the falling edge of the system clock (ø), or with output on the address lines?

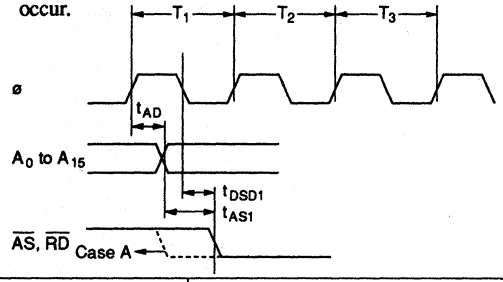
Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
O	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. The $\overline{AS}$ and $\overline{RD}$ signals are synchronized with the falling edge of the system clock in the T_1 state.

   The $\overline{AS}$ and $\overline{RD}$ signals never go low before the falling edge in the T_1 state. Case A in the diagram below cannot occur.



Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/500	Q&A No.	QA500 - 040B
Topic	Unused I/O lines		

Question

1. What should be done with unused I/O port lines?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
O	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. (1) Pull unused input/output port lines up or down through an approximately 10-kΩ resistor.

   (2) Do the same for input-only port lines.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

Connect a separate pull-up or pull-down resistor to each line.

SECTION 5

Technical Question and Answer

Product	H8/520, 532, 534, 536	Q&A No.	QA500 - 041B
Topic	Power dissipation in hardware and software standby modes		

Question

1. Is there any difference in current dissipation between hardware standby and software standby?

Classification—H8/532

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
O	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. Current dissipation satisfies the relationship:

   hardware standby ≤ software standby.

   In hardware standby mode, all lines are placed in the high-impedance state, which reduces current dissipation. In software standby mode I/O ports hold their previous states, so current dissipation varies depending on the state of the port.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/510	Q&A No.	QA500 - 058A
Topic	State of D_0 to D_7 with 8-bit data bus		

Question

1. In 16-bit data bus mode (mode 2 or 4), during access to the area accessed via an eight-bit bus, what are the states of the unused data bus lines (D_0 to D_7) and control signals?

Classification—H8/510

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	Development tools
O	Bus controller
	Miscellaneous

Answer

1. D_0 to D_7 are in the high-impedance state, and $\overline{LWR}$ is always 1.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

Technical Question and Answer

Product	H8/510	Q&A No.	QA500 - 059A
Topic	State of D_0 to D_7 during byte access in 16-bit data bus mode		

Question

1. What are the pin states during access to byte data in 16-bit data bus mode (mode 2 or 4)?

Classification—H8/500

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
O	Bus interface
	External expansion
	Development tools
	Miscellaneous

Answer

1. (1) In write access, the upper data bus (D_{15} to D_8) and lower data bus (D_7 to D_0) both output the same data.

   Control signal states are as follows:

Access to even address	Access to odd address
$\overline{LWR} = 1$	$\overline{LWR} = 0$
$\overline{HWR} = 0$	$\overline{HWR} = 1$

   (2) In read access, the states differ depending on the external circuit configuration.

   Control signal states are as follows:

   $\overline{RD} = 0$

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

1. The minimum RAM standby voltage (VRAM) is specified at 2.0 V. What voltage should be supplied to AV_{CC}?

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Technical Question and Answer

Product	H8/520, 532, 534, 536	Q&A No.	QA500 - 060A
Topic	RAM standby voltage		

Question

Classification—H8/532

	Software
	On-chip ROM
	On-chip RAM
	Clock
	Timers
	Serial I/O
	A/D
	PWM
	DTC
	I/O ports
	Power-down modes
	Elec. characteristics
	Exception handling
	Bus interface
	External expansion
	Development tools
O	Miscellaneous

Answer

1. AV_{CC} should be the same as the RAM standby voltage: 2 V. Setting AV_{CC} to 5 V or VSS will cause excessive current drain.

Related Manuals

Manual Title:

Other Technical Documentation

Document Name:

Related Microcomputer Technical Q&A

Title:

Additional Information

SECTION 5

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H8/520 Device EPROM Security

Tech Notes

Application Engineering

Tom Hampton

EPROM Security

The H8/520 Microcontroller has an EPROM security feature that can be used by the customer. This feature allows the user of the microcontroller to protect parts (or all) of the code programmed into the on-chip EPROM of the device from being read by means other than his own program. This feature cannot be tested by Hitachi and, due to this, is unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

Memory Configuration

The memory matrix of the H8/520 Microcontroller is configured as a dual matrix, one with even addresses and the other with odd addresses. The configuration of each matrix appears as lines of memory 32 bytes wide (32 x 8, 256 bits). This configuration allows an individual memory line to consist of 64 bytes of data (including both even and odd addresses). Each memory line has 1 security bit thus allowing every 64-byte segment to have the option of the security feature. The address of this security bit is the same as the starting address for the memory line.

Security Functions

The security function had two different operations depending upon the mode of operation that the device is placed into; EPROM programming mode or CPU operation mode.

EPROM Programming Mode

In the EPROM programming mode, the ability of the EPROM programmer to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read. If the security bit is a "0" (programmed), then any read operation to the EPROM will result in a "00" being read. This indicates that once the security bit is programmed, the user will be unable to verify the contents of the EPROM.

bit=1	EPROM data can be read (normal)
bit=0	"00" data is always read

H8/520 Device EPROM Security

CPU Operating Mode

In the CPU operating modes, the ability of any device to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read by the CPU. If the security bit is a "0" (programmed), then the read state of the EPROM (from the CPU), depends upon where instruction execution is occurring from.

bit=1 EPROM data can be read by CPU (normal)

bit=0 After RESET, the CPU can read EPROM data until it executes an instruction outside the internal EPROM area (either external memory or internal RAM). Once an instruction is executed outside the internal EPROM memory area, then the EPROM becomes disabled and cannot be accessed any further. This prohibits an external program from being able to "dump" the contents of the on-chip EPROM.

Programming the Security Bit

There exists two EPROM programming modes; normal and security. The normal EPROM programming mode is used to program the code/data area of the on-chip EPROM memory for the H8/520 device. The "security" programming mode is used to program the security bits of the EPROM's memory area. The security function is then implemented by programming a "0" into the address corresponding to the memory line location. Setting the programming mode is done by setting certain I/O port pins to the following states:

	H8/520 Device I/O Port Pin	
Programming Mode	P50/TMCI	P51/FTI1
Normal	1	1
Security	1	0

Again, this feature cannot be tested by Hitachi and thus remains unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H8/534 Device EPROM Security

Tech Notes

Application Engineering

Tom Hampton

EPROM Security

The H8/534 Microcontroller has an EPROM security feature that can be used by the customer. This feature allows the user of the microcontroller to protect parts (or all) of the code programmed into the on-chip EPROM of the device from being read by means other than his own program. This feature cannot be tested by Hitachi and, due to this, is unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

Memory Configuration

The memory matrix of the H8/534 Microcontroller is configured as a dual matrix, one with even addresses and the other with odd addresses. The configuration of each matrix appears as lines of memory 32 bytes wide (32 x 8, 256 bits). This configuration allows an individual memory line to consist of 64 bytes of data (including both even and odd addresses). Each memory line has 1 security bit thus allowing every 64-byte segment to have the option of the security feature. The address of this security bit is the same as the starting address for the memory line.

Security Functions

The security function had two different operations depending upon the mode of operation that the device is placed into; EPROM programming mode or CPU operation mode.

EPROM Programming Mode

In the EPROM programming mode, the ability of the EPROM programmer to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read. If the security bit is a "0" (programmed), then any read operation to the EPROM will result in a "00" being read. This indicates that once the security bit is programmed, the user will be unable to verify the contents of the EPROM.

bit=1	EPROM data can be read (normal)
bit=0	"00" data is always read

H8/534 Device EPROM Security

CPU Operating Mode

In the CPU operating modes, the ability of any device to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read by the CPU. If the security bit is a "0" (programmed), then the read state of the EPROM (from the CPU), depends upon where instruction execution is occurring from.

bit=1 EPROM data can be read by CPU (normal)

bit=0 After RESET, the CPU can read EPROM data until it executes an instruction outside the internal EPROM area (either external memory or internal RAM). Once an instruction is executed outside the internal EPROM memory area, then the EPROM becomes disabled and cannot be accessed any further. This prohibits an external program from being able to "dump" the contents of the on-chip EPROM.

Programming the Security Bit

There exists two EPROM programming modes; normal and security. The normal EPROM programming mode is used to program the code/data area of the on-chip EPROM memory for the H8/534 device. The "security" programming mode is used to program the security bits of the EPROM's memory area. The security function is then implemented by programming a "0" into the address corresponding to the memory line location. Setting the programming mode is done by setting certain I/O port pins to the following states:

	H8/534 Device I/O Port Pin	
Programming Mode	P60/-IRQ2/A16	P61/PW1/-IRQ3/A17
Normal	1	1
Security	1	0

Again, this feature cannot be tested by Hitachi and thus remains unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H8/536 Device EPROM Security

Tech Notes

Application Engineering

Tom Hampton

EPROM Security

The H8/536 Microcontroller has an EPROM security feature that can be used by the customer. This feature allows the user of the microcontroller to protect parts (or all) of the code programmed into the on-chip EPROM of the device from being read by means other than his own program. This feature cannot be tested by Hitachi and, due to this, is unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

Memory Configuration

The memory matrix of the H8/536 Microcontroller is configured as a dual matrix, one with even addresses and the other with odd addresses. The configuration of each matrix appears as lines of memory 32 bytes wide (32 x 8, 256 bits). This configuration allows an individual memory line to consist of 64 bytes of data (including both even and odd addresses). Each memory line has 1 security bit thus allowing every 64-byte segment to have the option of the security feature. The address of this security bit is the same as the starting address for the memory line.

Security Functions

The security function had two different operations depending upon the mode of operation that the device is placed into; EPROM programming mode or CPU operation mode.

EPROM Programming Mode

In the EPROM programming mode, the ability of the EPROM programmer to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read. If the security bit is a "0" (programmed), then any read operation to the EPROM will result in a "00" being read. This indicates that once the security bit is programmed, the user will be unable to verify the contents of the EPROM.

bit=1	EPROM data can be read (normal)
bit=0	"00" data is always read

H8/536 Device EPROM Security

CPU Operating Mode

In the CPU operating modes, the ability of any device to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read by the CPU. If the security bit is a "0" (programmed), then the read state of the EPROM (from the CPU), depends upon where instruction execution is occurring from.

bit=1 EPROM data can be read by CPU (normal)

bit=0 After RESET, the CPU can read EPROM data until it executes an instruction outside the internal EPROM area (either external memory or internal RAM). Once an instruction is executed outside the internal EPROM memory area, then the EPROM becomes disabled and cannot be accessed any further. This prohibits an external program from being able to "dump" the contents of the on-chip EPROM.

Programming the Security Bit

There exists two EPROM programming modes; normal and security. The normal EPROM programming mode is used to program the code/data area of the on-chip EPROM memory for the H8/536 device. The "security" programming mode is used to program the security bits of the EPROM's memory area. The security function is then implemented by programming a "0" into the address corresponding to the memory line location. Setting the programming mode is done by setting certain I/O port pins to the following states:

	H8/536 Device I/O Port Pin	
Programming Mode	P60/-IRQ2/A16	P61/PW1/-IRQ3/A17
Normal	1	1
Security	1	0

Again, this feature cannot be tested by Hitachi and thus remains unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

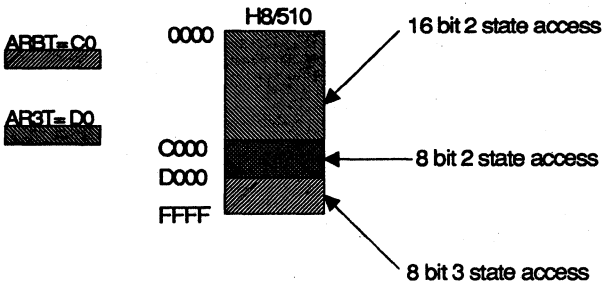
H8/510 Instruction Timing

Tech Notes

Application Engineering

Carol Jacobson

Although the H8/510 uses the same instruction set as the rest of the H8/500 family, the formulas used to calculate instruction fetch and execution times are somewhat different. The H8/510 address range is sectioned by the Three-state Area Top Register (AR3T,) defining two state or three state access field, and Byte Area Top Register (ARBT), defining 16-bit or 8-bit data bus fields. Locations with address values greater than the ARBT register value are accessed via an 8-bit bus. Locations with address values greater than or equal to the AR3T value are accessed using three states, two access plus one additional state for slower peripherials.



Access times for instructions fetched from each area are claculated using slightly different formulas. Therefore, it's important to understand the contents of the AR3T & ARBT before trying to determine execution times. Formulas are given in the H8/500 programming manual section 2.6.4.

SECTION 5

EXAMPLES:

1.

Mode 3 Expanded Maximum 8-bit data bus (ABRT register is ignored)

AR3T=H'90 DP Register=H'A0

address	opcode	operands
00000100	MOV.W	R1,@H'A00020

The instruction is fetched via an 8-bit data bus from a location accessed in two cycles. Word data is moved to a location accessed in three cycles via an 8-bit bus.

From section 2.6.4: Total CPU states (cycles)= (Value in Table 2-8) + 2I + J + K

T= 6 + 2 (2) + 1 + 3 = 14 clocks

2.

Mode 4 Expanded Maximum 16-bit data bus

AR3T=H'A0 ARBT=H'A2

address	opcode	operands
00A00100	MOV.W	R1,@FRT2_OCRA

The instruction is fetched via a 16-bit bus from a location accessed in three cycles. Word data is moved to the on-chip register field which is always accessed in three cycles via an 8-bit bus.

From section 2.6.4: Total CPU states (cycles)= (Value in Table 2-8) + (Value in Table 2-9) + 2I + (J+K)/2

T= 6 + 1 + 2(2) + (1+3)/2 = 13 clocks.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H8/500 Instruction Timing

Tech Notes

Application Engineering

Carol Jacobson

Individual instruction execution times for the H8/532, H8/534, H8/536, H8/520 can be calculated using tables 2-8 (1 thru 6) and table 2-9 of the H8/500 Series Programming Manual. Formulas for the H8/532, H8/534, H8/536, and H8/520 are given in section 2.6.1. H8/510 Instruction timing is discussed in TN-0039.

The main steps used to determine instruction execution timing are:

1. determine the instruction addressing mode, location and operand location.

(note: for indirect addressing modes the operand location is the location of the data)

2. from section 2.6.1 determine the which formula to use

3. apply values from Table 2.8 & 2.9 to the formula

EXAMPLES:

1.

```
Mov.w    #H'DAFE,@FA80        ;addressing mode: @AA:16
```

a. If the instruction and operands reside in on-chip RAM or ROM, the number of CPU clock cycles is the value from the body of Table 2.8 plus the value from Table 2.9.

Table 2.8 # cycles = 9 from Table 2.9 the 'adj. value' = 2 (even address) = 11 CPU clocks total execution time.

b. If the instruction resides in on-chip memory and the operands are from an on-chip module (peripheral) or off-chip, the number of CPU clock cycles is the value from the body of Table 2.8 plus 2 x the I value plus the adj. value from Table 2.9.

Table 2.8 #cycles= 9 + 2(2) plus the 'adj. value' , 2 (even address) = 15 clocks

c. If the instruction resides in off-chip memory and the operands are from on-chip, the number of CPU clock cycles is the value from the body of Table 2.8 plus 2 x the J and K values from Table 2.8.

Table 2.8 #cycles= 9 , K= 3 and J= 3 ; Total execution time = 9 + 2(3 + 3) = 21 clocks

d. If the instruction resides in off-chip memory and the operands are from off-chip memory or on-chip modules the number of CPU clock cycles is the value from the body of Table 2.8 plus 2 x the I, J and K values from Table 2.8.

Table 2.8 #cycles= 9, I= 2, K= 3 and J= 3 ;

Total execution time = 9 + 2(3 + 3 + 2) = 25 clocks

2.

BSR External_ROM ;addressing mode @aa:16

a. If the instruction resides off-chip and the destination is an off-chip 16-bit location and the branch is taken: Table 2.8 # cycles= 7, I= 2, J+K= 5;

Total execution time = 7 + 2(2 + 5) = 21 clocks

b. If the instruction resides on-chip and the destination is an on-chip 16-bit location and the branch is taken: Table 2.8 #cycle= 7, From Table 2.9 'adj. value'= 0 (even address);

Total execution time= 7 + 0 = 7 clocks

Section from Table 2-8

Instruction	I	J \ K	Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@ -Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xxxx:16
			1	1	2	3	1	1	2	3	2	3
MOV.W	2	1	2	5	5	6	5	6	5	6		4
MOV.B #xx:8,<Ea>'	1	2		7	7	8	7	8	7	8		
MOV.W #xx:16,<Ea>	2	3		8	8	9	8	9	8	9		

K values

I values

J values

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H8/532 Microcontroller EPROM Security

Tech Notes

Application Engineering

Tom Hampton

EPROM Security

The H8/532 Microcontroller has an EPROM security feature that can be used by the application programmer. This feature allows the user of the microcontroller to protect parts (or all) of the code programmed into the on-chip EPROM of the H8/532 from being read by means other than his or her own program. This feature cannot be tested by Hitachi and, due to this, is unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

Memory Configuration

The memory matrix of the H8/532 Microcontroller is configured as a dual matrix, one with even addresses and the other with odd addresses. The configuration of each matrix appears as lines of memory 32 bytes wide (32 x 8, 256 bits). This configuration allows an individual memory line to consist of 64 bytes of data (including both even and odd addresses). Each memory line has 1 security bit thus allowing every 64 byte segment to have the option of the security feature. The address of this security bit is the same as the starting address for the memory line.

Security Functions

The security function had two different operations depending upon the mode of operation that the H8/532 device is placed into, EPROM programming mode or CPU operation mode.

EPROM Programming Mode

In the EPROM programming mode, the ability of the EPROM programmer to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read. If the security bit is a "0" (programmed), then any read operation to the EPROM will result in a "00" being read. This indicates that once the security bit is programmed, the user will be unable to verify the contents of the EPROM.

security bit	1	EPROM data can be read (normal)
security bit	0	"00" data is always read

H8/532 Microcontroller EPROM Security

CPU Operating Modes

In the CPU operating modes, the ability of any device to read the EPROM contents is limited by the state of the security bit.

If the security bit is a "1" (unprogrammed state), then the data in the EPROM can always be read by the CPU. If the security bit is a "0" (programmed), then the read state of the EPROM (from the CPU), depends upon where instruction execution is occurring from.

security bit	1	EPROM data can be read by CPU (normal)
security bit	0	After RESET, the CPU can read EPROM data until it executes an instruction outside the internal EPROM area (either external memory or internal RAM). Once an instruction is executed outside the internal EPROM memory area, then the EPROM becomes disabled and cannot be accessed any further. This prohibits an external program from being able to "dump" the contents of the on-chip EPROM.

Programming the Security Bit

There exists two EPROM programming mode; Normal and Security. The normal EPROM programming mode is used to program the code/data area of the on-chip EPROM memory for the H8/532. The "security" programming mode is used to program the security bits of the EPROM's memory area. The security function is then implemented by programming a "0" into the address corresponding to the memory line location. Setting the programming mode is done by setting certain I/O port pins to the following states:

	H8/532 I/O Port Pin	
Programming Mode	P60	P61
Normal	1	1
Security	1	0

Again, this feature cannot be tested by Hitachi and thus remains unguaranteed. It is up to the user to determine whether or not to implement the function of this feature and accept sole responsibility for its outcome.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Section 6

Memory

SECTION 6

HITACHI®

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Word-wide DRAMs

Tech Notes

Application Engineering

Oomer A. Serang

Introduction

Hitachi has introduced an assortment of word-wide 4Meg DRAMs. This Brief will describe the main purpose and application of word wide devices.

Table 1.0 below lists the various x16 offerings.

Part Number	Refresh Rate	Control	Vcc	Icc1 Current Consumption
HM514260-8	512 cycles/ 8ms	2CAS	5.0V +/- 10 %	150mA
HM514270-8	512 cycles/ 8ms	2 WE	5.0V +/- 10%	150mA
HM514170-8	1024 cycles/ 16ms	2 WE	5.0V +/- 10%	120mA
HM51V4160-8	1024 cycles/ 16ms	2 CAS	3.3V +/- 10%	105mA

Table 1.0 Hitachi's 256kx16 parts list.

Applications

The 256kx16 device can be used either to upgrade systems that are based on the older generation 256kx4 DRAMs or to give better performance to new systems that implement word wide busses. The distinction is made when choosing one of the two available refresh rates. The 512cycles /8ms is a direct replacement for older generation 256kx4 DRAMs. This implies that the DRAM controller does not have to be redesigned. While the 1024 cycles/16ms part consumes less current but will require

a new DRAM controller, and hence is targeted for new systems. The current reduction is due to the fact that fewer sense amplifiers are used and hence the overall current consumption is decreased.

Another system design variable is memory bank selection. Some designers prefer to use common /RAS while the /CAS signal serves as a bank select. Hitachi offers the 2CAS signal option precisely for those systems. Hitachi also offers 2WE parts which are best used in graphics applications, since it offers easier control of upper and lower bytes. In particular, the x16 DRAM part is used as a z-buffer while expensive VRAMs would be used as the main graphics buffer.

Lastly, the obvious advantages of designing with 256kx16 devices instead of 256k x4 parts should not be overlooked; less board space is used, memory cost is reduced, and the reliability is higher.

Consequently, an older memory system based on bank selection of 256 k x4 DRAMs and their 8ms refresh requirement can be upgraded with either the HM514260 or HM514270.

Word wide write cycles for the multiple /WE and multiple /CAS type devices can be confusing. To prevent confusion, it is important to understand that address and data is latched internally on a **word-wide** basis at the fall of /RAS and /WE or /CAS regardless of whether the write cycle is a byte write or a word write. This means that the x16 device does NOT permit writing upper and then lower bytes in sequence within the same write cycle. More specifically, early write cycles cannot be mixed with late write cycles in the same cycle. On the other hand, a word can be written with the fall of both upper and lower byte control signals at the same time, and one byte can be written with the fall of either the upper or lower control select signals.

In addition, fast page mode byte read cycles are permitted as long as there is a minimum tCP time separation between the first /UCAS being de-asserted and the second L-CAS becoming asserted as shown in Figure 1.0

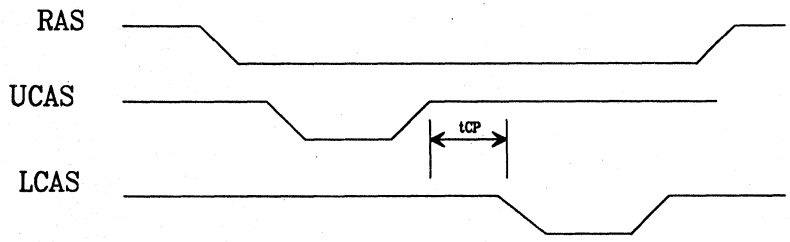


Figure 1.0 Page mode cycle

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Mask History of HN58C256

Tech Notes

Application Engineering

Oomer A. Serang

The original HN58C256, 256k EEPROM was introduced in 1989 as an R0 mask. Since that time the device has gone through 2 mask revisions and this Brief will summmarize the modifications.

The first revision was done in late 1990 to counter potential data destruction due to noise on /CE or /WE while Vcc was powering up or down. Normal program mode dictates that /CE and /WE be low while Vcc and /OE are high. However, while powering on and off unknown levels on /CE or /WE can inadvertently cause erroneous data to be written. Figure 1.0 below, shows the conditions that may lead to an inadvertent program cycle.

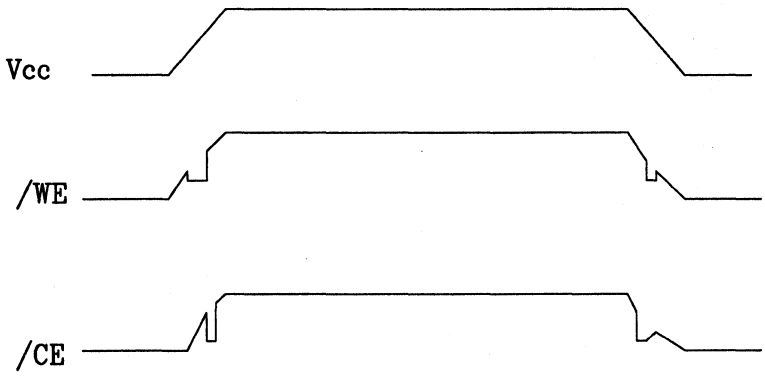


Figure 1.0 Conditions that may cause incorrect write.

Consequently, if a user could guarantee that /WE and /CE are high when Vcc was stabilizing then there could be no possibility of an inadvertent write occuring. Nonetheless, Hitachi decided to improve the chip by decreasing the programming voltage sensitivity level so that any glitches that occurred on /CE or /WE while Vcc was stabilizing were locked out.

From a specification standpoint, the only parameter that changed due to the R1 mask revision was an increase in the Icc1 standby current, from 20μA maximum to 50μA typical and 200μA maximum. Other than the Icc1 change, all other AC and DC parameters remained the same.

Mask History of HN58C256

The second mask revision, called appropriately enough R2, was implemented in the October 1991 timeframe. The R2 revision fixed a rarely occuring problem that caused random address locations to change to FFh. The culprit was found to be a combination of residual voltages coupled with specific rise times on Vcc. The details are best described by looking at Figure 2.0 below.

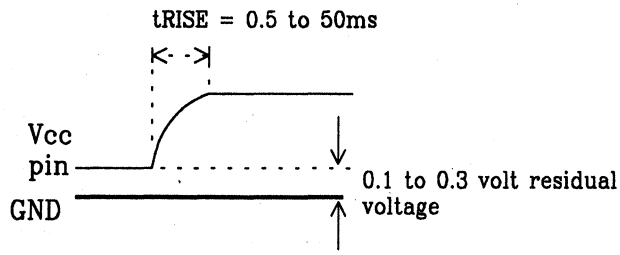


Figure 2.0 Conditions that may cause data to change to FFh.

If a residual voltage in the 0.1 to 0.3 volt range existed on the Vcc pin it would increase the internal capacitance coupling on the /RESET pin. This prevented the reset timer circuits from functioning properly. In fact, the internal reset signal would abort prematurely which briefly activated the write and erase circuits. The presently available R2 mask has countered the aformentioned peculiarity and results in a highly reliable device.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

1M Flash Software Modification

Tech Notes

Application Engineering

Oomer A. Serang

Intel presently offers customers some general 1M Flash code for use in 10Mhz 80186 based systems. This Note will point out the modifications that need to be made to Intel's code so it can support Hitachi's 1M Flash device.

The major differences between Hitachi and Intel are shown in Table 1.0

Parameter	Intel	Hitachi
Write operation duration	10µsec min 25µsec max	25µs min
Erase operation duration	9.5ms min 10.5ms max	9ms min 11ms max
Multiple chip erase	After erase verification of any chip, send reset command (FFH), until all chips have erased.	After erase verification of any chip, send verify command (A0H), until all chips have erased.
Multiple chip programming	After programming completion of any chip, send reset command (FFH) while waiting for all chips to complete programming.	After programming verification of any chip, program and verify chip again but send FFH as program data. Repeat, until all chips have completed programming.

Table 1.0 Major differences between Intel and Hitachi 1M Flash.

1M Flash Software Modification

Table 1.0 indicates that the programming Pulse Width and Erase Time duration for Hitachi's device is different from Intel's. However, if minor code modifications of the original Intel code are done, a Hitachi device can plug into an Intel socket.

For example, in Intel generated code the following instructions are recommended for Programming Pulse Generation in a 10Mhz 80186:

```
%*define (WAIT 10us)
        push cx         ; save old counter register contents
        mov cx, 6       ; put 6 into counter register
        loop $          ; decrement until 0
        pop cx          ; restore old counter contents
```

The code above completes execution in 10us by counting down from 6, after which a control pulse is generated. To generate a 25us pulse width, the number of T states for each instruction is determined and then the appropriate counter value to generate a 25us execution time is calculated. In this case the value required is 17, so

```
        mov cx,6
changes to
        mov cx,17
```

The final code would look like:

```
%*define (WAIT 25us)
        push cx
        mov cx,17
        loop $
        pop cx
```

Lastly, the erase code supplied by Intel generates a 10ms wait and is as follows:

```
%*define(WAIT 10ms)
                push cx
                mov cx, 10
                loop %W10ms
                pop cx
```

Since the 10ms meets Hitachi's erase time specification as well, there is no need to modify any of the above code.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

1M to 4M Flash Memory Upgrade

Tech Notes

Application Engineering

Oomer A. Serang

Introduction

It seems like it was only yesterday when the Hitachi 1M Flash device was introduced and promoted. The differences between our 1M device and a rival's was repeated and re-iterated often. Hence, it is probably very surprising to already begin dicussions on the 4M Flash device. Nonetheless, the 4Meg is here now and the aspects of replacing the 1M device is the topic of this Note.

4M Features

* Automatic chip and BLOCK erase
* Status polling
* Command register based control
* Automatic and Manual Programming

1M to 4M Upgrade

A lot of design effort went into making the 4M Flash as compatible as possible with the 1M Flash device. Consequently, it is no coincidence that the 4M Flash package size is identical to the 1M Flash package size. In addition, the pins between the 1M and 4M devices are similar except for the fact that the 4M Flash has, of course, 2 additional address pins and more importantly no /WE pin. The figure below shows the pin-out difference.

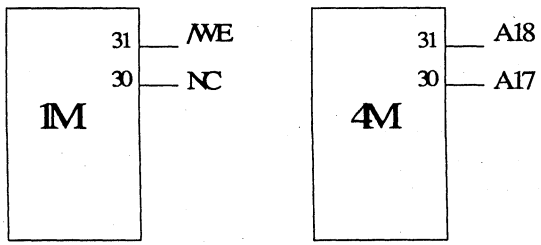


Figure 1.0 Difference in pin configurations between 1M and 4M Flash

1M to 4M Flash Memory Upgrade

Consequently, in the 1M device an address is latched when both the /WE and /CE go low while data is latched when either /WE or /CE goes high. In comparison, the 4M device has address and data latching when /CE goes high.

A lot of additional expense and effort can be avoided if customers design in for the 4M while they are designing in the 1M Flash. The customer should be told that if their 1M system has a trace that serves as either an address line or /WE line, depending upon a jumper position, then the 4M Flash will fit easily into the socket when an upgrade is required. The other address line required for the 4M can be laid out as a trace going to the NC pin of the 1M.

As far as the logic is concerned, the 4M Flash has easier timing in many cases because there is no longer a need to toggle a /WE pin as was the case with the 1M Flash device. In addition, the block select operation is simply controlled by A14-A18 so that incoporating the block select feature consists of only applying a new address and latching it in by the rising edge of /CE. In fact, block accesses can be implemented by "false" CPU write cycles. They are labeled as "false" write cycles because data is not actually written to the device even though the write cycle timing is implemented. An important restriction on the false write cycles is to make sure that the data on the bus is anything EXCEPT FFh when /CE rises. Putting FFh on the data bus will reset the device since FFh is the reset command.

When all the items are taken into account a pre-liminary analysis of the 1M to 4M upgrade yields the following steps:

1) Run an address line trace into the NC of the 1M. When using the 4M the trace will serve as A17.

2) Place a jumper on the board so that the /WE for the 1M can change to A18 for the 4Meg.

3) Modify PAL equations and incorporate false write cycles during block accesses and make sure the data on the bus is not FFh.

As a whole, the above steps indicate that although the 4M upgrade is not trivial, the upgrade is not excessively complicated either.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Low Voltage RAMs

Tech Notes

Application Engineering

Oomer A. Serang

In the past, speed and memory organization were critical parameters in memory system designs. The popularity of notebook computers has now made power consumption of equal if not more importance. A previous note discussed extended refresh cycles as a means to decrease power consumption. This Note will introduce Hitachi's wide variety of low voltage RAM memory products.

Family	Density	Config.	Operating Voltage	Access Time
DRAM	4M	512kx8 256kx16 256kx18	2.7 to 3.6	70/80/100
PSRAM	4M	512kx8	2.7 to 3.6	120/150
SRAM	1M	128kx8	2.7 to 5.5	150

Table 1.0 Hitachi's low voltage RAMs

All of the 4M low voltage DRAMs are manufactured using the same 0.5 µm process that is used to manufacture 16Meg DRAMs. Consequently, the low voltage 4M DRAMs are NOT just high voltage parts that are screened for low voltage operation. In fact, even the low voltage 4M PSRAM is NOT a 5.0 volt PSRAM screened for low voltage operation, but is instead a re-design of the standard part.

As can be imagined, low voltage operation introduces a gamut of potential problems like decreased noise margin and potential increase in soft errors. Noise margin aside, soft errors are most dependent on cycle times and power supply voltages. For a given DRAM cell, decreasing the power supply voltage increases the SER. An effective counter against the SER degradation is to increase the cell capacitance during the chip design stage or to use better dielectrics. Although no reliability data is yet available on the low voltage 4M DRAMs one can assume the SER to be comparable to standard voltage parts.

Extended Refresh DRAMs

Tech Notes

Application Engineering

Oomer A. Serang

Introduction

The explosive growth of notebook computers has not only elevated computing convenience to an unprecedented level but it has also stimulated mainstream suppliers to offer specialized parts for the portable market. For example, 2½" hard disk drives, super-twist color LCDs and miniaturized keyboards are direct consequences of the portable computer revolution. One of the most important objectives of notebook manufacturers is to minimize system power consumption. With reduced system power consumption the end-user can be assured of longer battery life and maximum operating time, thereby making the portable that much more appealing. Consequently, vendors who want to supply the notebook market are aggressively studying and implementing methods to reduce power consumption.

One particularly power hungry aspect of computers is their memory systems. Typical memory systems can consume as much as 30% of the total system power. In particular, the additional overhead of DRAM refresh cycles not only uses precious active bandwidth but also consumes scarce battery current. With this in mind, battery life can be extended if the frequency of memory refresh cycles were decreased. That was the motivating reason to develop and offer extended refresh DRAMs.

Extended refresh DRAMs are DRAMs that have dramatically relaxed refresh requirements. Since cycle times and power consumption are directly related it is possible to reduce power consumption by decreasing refresh frequency. In fact, when compared to standard DRAMs, the extended refresh DRAMs can go as much as 16 times longer without a refresh cycle. Another attraction of extended refresh DRAMs is that Hitachi doesn't consider them to be a low priority product in the memory chain but instead assigns them the same important status as standard 4Meg DRAM products. When low DRAM data retention currents are combined with the high volume manufacturing assurance of Hitachi, then advanced notebook computers, hand-held instruments and other power sensitive applications can become a reality.

Table 1 compares battery life between standard and extended refresh DRAMs. Although no system is going to be in data-retention mode 100% of the time the point to be made is that a SL based system can have a far longer battery life than a system based on standard 4Meg parts.

Part Number	Active Current (80 nsec part)	Data Retention Current	Battery Life (1 pc., 500 mAh)
HM514100/400ASL (1K cycles/256 msec)	90 mA	100 µA (typ.)	208 days (typ.)
HM514100/400A (1K cycles/16 msec)	90 mA	1 mA (typ.)	21 days (typ.)

Table 1: Standard vs. Extended Refresh Battery Life

Trench vs STC

Although standard 4Meg DRAMs are offered by a variety of suppliers as well as a variety of countries the same cannot be said of extended refresh DRAMs. Only a handful of vendors have processes and chip designs superior enough to guarantee extended refresh times.

When a DRAM isn't being accessed or refreshed the amount of time required for the stored charge to dissipate is a function of the dielectric material, cell to cell isolation, parasitic losses and even the cell structure. For example, an analysis of first generation trench cells revealed that they could not meet extended refresh requirements because of leakage between trenches. Since the storage node is directly in the silicon substrate, stress and crystallographic defects were manifested as leakage current. Consequently, the trench cell insulation characteristics was improved upon by depositing an additional SiO_2 layer around each trench and then staggering, rather than lining up in a single file, the cells, to get less cell to cell interaction as well as increased cell density. The improved cell leakage characteristics came at the expense of additional fabrication steps and increased production cost of the trench device.

Even Hitachi's first generation stacked capacitor architecture could not meet the stringent extended refresh specifications. However, Hitachi's second generation "A" mask stacked capacitor architecture yielded minimum leakage specifications primarily because the smaller junction area reduced the rate of leakage current.

Another advantage of extended refresh parts is that they aren't manufactured on special lines or under special circumstances. Hence, qualification of standard parts also immediately qualifies the extended parts. Although extended refresh does not require special manufacturing it does require special screening to meet stringent refresh requirements. The screening process consists of 2 additional steps. The first step is a functional test in which data is written to the DRAM and then after a long pause the device is read to see if the data is still valid. The second step is to measure the V_{CC} current during standby to insure it is vastly smaller than standard DRAMs.

Presently Hitachi offers a number of slow refresh DRAMs. The part numbers and some specifications are shown in Table 2.

Part Number	Active Refresh Rate	Standby Refresh Rate	Standby Current Consumption	Data Retention Current Consumption
HM514100/400 AL	1K/128 msec	1K/128 msec	150 µA (max.)	200 µA (max.)
HM514100/400ASL	1K/16 msec	1K/256 msec	100 µA (max.)	150 µA (max.)

Table 2: Extended 4Meg DRAM Devices

The following calculation shows how the data retention current value is derived for the SL device:

Data retention current = refresh current + standby current

For the SL version

refresh current = (90 mA * 150 nsec * 1024 cycles) / 256 msec
standby current = 100 µA * [256 msec - (150 nsec * 1024 cycles)] / 256 msec
Data retention current = 50 µA + 100 µA
= 150 µA

When adhering to DRAM refresh specifications there are 2 implementation options available to designers. One method consists of stopping every 16 msec and then refreshing the entire DRAM at once and is referred to as a burst refresh. Alternatively, the micro-processor can be interrupted for a refresh cycle every 15.6 µsec and this method is referred to as a distributed refresh. Regardless of which type of refresh is used it is comforting to know that there are no V_{CC} rise or fall time restrictions when entering or exiting data retention mode. Hence, a big advantage of the SL part is not only its ability to operate at a data retention voltage of 4.0 volts, but the V_{CC} can switch from 5.5 Volts to 4.0 Volts without regard to the V_{CC} fall time, or from 4.0 Volts to 5.5 Volts without regard to the rise time.

In practice, one way to control slow refresh DRAMs is to use the DRAM controller chip sets that are made specifically for extended refresh DRAMs. For example, Chips and Technology offers the 82C241 DRAM Controller and the 82C636 Power Control Unit. Effective utilization of the aforementioned parts reduces development cost and speeds the end product into the market place.

Since many customers implement an interrupt driven distributed refresh scheme, another aspect of designing in extended refresh DRAM requires some software modifications. For example, if a certain amount of time has elapsed without a keyboard interrupt then the DRAMs can go into extended refresh mode.

Voltage Degradation (4V operation)

Although Hitachi's SL device can withstand V_{CC} = 4.0 volts during data retention mode a potential drawback is that the Soft Error Rate (SER) will increase. Consequently, customers should be made aware of the relationship between low voltage and SER so they can design their systems accordingly.

Section 7

Support Tools

HITACHI®

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

MRI Toolkit for H8/300

Application Note

Emulator Version XRAY Tutorial

Paul Yiu

INTRODUCTION

This paper will demonstrate some basic functions in XRAY, using ***frank.c*** as an example. ***Frank.c*** is a very basic C program that utilizes loops, counters, and standard I/O routines. The main program, as well as the necessary auxiliary codes and files, are listed at the end of this document.

Note: In the following tutorial, white boxes indicate user inputs, while shaded boxes indicate computer responses.

STARTING XRAY

If you are using XRAY for the first time, be sure to specify the baud rate. At the **c:\xhih83** prompt, type:

xhih83 <filename> -e 9600 <CR>

Note: **Do not add any extensions to the *<filename>* parameter. XRAY will load both the source code and the executable code.**

Once you are in XRAY, it's a good idea to change default baud rate to the baud rate of the ASE. In the command window, type:

option emulator="9600" <CR>

..... sets default baud rate.

startup <CR>

..... saves option to startup.xry, which is called automatically each time XRAY is invoked.

Next time XRAY is called, the baud rate will be set at 9600 by default.

IN XRAY

Let's just run this program a couple of times to see what it does. Type:

go <CR>

This program asks for the user's selection in hex number, then outputs a specific string corresponding to the user's input. Frank Sinatra songs are used as examples here.

First, you will see the message "HITACHI America, Ltd." move from the left edge of the screen to the middle, then the message "Applications Engineering." Finally, there is a message asking for the user's input.

If you look at the source code, you will see that there are only 4 allowed inputs to this program. The other numbers will give an error message and restart the program. Let's choose 7. At program terminal, type:

7 <CR>

The song title ***Fly me to the Moon*** and an airplane are printed on the screen. At this point, the program resets and prints the opening messages again. You can experiment with inputs of 5,12,a,or 7. Also, try inputting an invalid number and see what happens.

The program is very simple; it was written with loops and counters to demonstrate XRAY commands. Now let's take a look at some XRAY debugger commands. At XRAY terminal, type:

<Ctrl> c

```
Stopped due to halt from user
```

In the window at the top right corner, you can see that when we halted the program, ***getchar*** was being executed because the program was waiting for an input from the user.

Some functions are called from the C library; their C source codes aren't always available. To take a look at **getchar** in assembly, press the **F3** key.

Now the screen should show the assembly listing, value of the stack and the registers. Press **F3** again to get back to the original screen.

To see the C source code again, at the XRAY terminal, type:

restart <CR>

```
Note: in startup routine.
Press F9 to go to main.
```

They **F5** key invokes the help screen; it contains brief explantions of each debugger command. You can press **F5** right now and look at some of the explanations.

All the commands you enter are stored in a last-in-first-out fashion. **F7** retrieves your previous commands to save some typing.

DEBUGGING

This program calls a subroutine called ***cpu_init***. Let's take a look at it. First we will set a break point at line 17. Type:

b #17 <CR>

Window #25 will appear at the top of the screen, listing all the break points; we only have one so far. Now type:

go <CR>

```
Break # 1 on instr at loc
0000003E module PT line 98
```

We now see line #98 instead of #17. Line #98 is the beginning of ***cpu_init***; that's why the program stops at line #98.

Another useful function key is the **F9** key. Press **F9** now to allow XRAY to execute one line. Press **F9** two more times to finish ***cpu_init***.

Now line #19 is highlighted, not line #18 because line #18 is only a label, not an instruction.

F9 performs single stepping. The F10 key is similar, but it doesn't take you into each subroutine. Press **F10** now, and line #19 will be executed.

Let's keep track of all the variables as we debug this program. Type:

monitor song,counter,delay <CR>

The values of these variables are listed in the top-left window. Let's set another break point at line #42. Type:

b #42 <CR>

If we start the program again, it should stop executing right after we make our song selection on the program screen. Type:

go <CR>

```
Break # 2 on instr at loc
000000F2 module PT approx line
42 (PC = 000000F6)
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Again we see the title message and the request for input. At the program terminal, type:

```
a      <CR>
```

Nothing happens on the program terminal because XRAY encountered another break point and halted execution. Now, if we look at the value of "song," we see 0x000a. We can also print the value of a variable without monitoring it. Type:

```
p song <CR>
```

```
0x000A
```

Looking at the source code, if we start the program again, we'd expect the ***Stranger in the Night*** message, accompanied by a couple of question marks dancing across the screen. But, one powerful feature of XRAY allows us to "cheat." We can execute C code on the spot. Type:

```
c song=5      <CR>
```

```
Result is: 5  0x05
```

The command "C," followed by an expression, is equivalent to an instruction in the C source code. In the Data window, we can see the value of "song" has been changed to 5. Let's start the program again and see what happens. Type:

```
go     <CR>
```

The "***The way you look tonight***" message came on instead of song number 0x000a. There are more than one break commands in XRAY; let's take a look at them. We'll stop execution and start over. Type:

```
<Ctrl> c
```

```
Stopped due to halt from user
```

```
restart <CR>
```

```
Note: in startup routine.
Press F9 to go to main.
```

```
clear  <CR>
```

The clear command clears all break points. If you wish to clear a specific breakpoint, just type in the breakpoint number after "clear." If you want to see where the breakpoints are, just type:

```
b      <CR>
```

You should get a blank window at this point because we've cleared all the breakpoints.

breakread: If we enter "***br &song***," whenever the program tries to read the value of song, it halts.

breakwrite: Similar to breakread, "bw &song" halts the program whenever it tries to write a value to "song".

breakaccess: The program halts when there is a read or write.

Let's try breakread. Type:

```
br &song      <CR>
```

SECTION 7

```
*** error code = 67
```

Oops, this give an error message. Since we restarted the program, it's not in "main" yet. The variable "*song*" is not yet recognized. Window #4 shows that the variables are not active. Press the spacebar to keep going. Let's get in **main** first, then set the breakpoint. Type:

b #18	<CR>	
g	<CR>	;abbreviation for go

```
Break # 1 on instr at loc
00000042 module PT approx line
#19
```

br &song	<CR>
g	<CR>

```
Break # 2 on instr at loc
0000FF7C (PC = 000000FC)

Break at module PT approx line
42 (PC = 000000FC)
```

At the program terminal, type:

a	<CR>

At line #42, the program tried to print the value of "*song*." The program halted because it tried to read the value of "*song*." Let's keep going.

Type:

g	<CR>

```
Break # 2 on instr at loc
0000FF7C (PC = 0000189C)
Break at module PT approx line
43 (PC = 0000189C)
```

The program halts again at line #43 because the switch statement tried to read "*song*."

We can look at functions that aren't in the window. For example, type:

list cpu_init	<CR>

We can see the listing of subroutine ***cpu_init***. To get back to where we came from, type:

list	<CR>

We are back to line #43 again. Now press F3 to debug at the assembly level. Instead of "list," we say "disassemble" in this mode. Type:

d 0x0030	<CR>

We see the beginning of the program. To get back to the original place, type:

d	<CR>

Press **F3** again to get back into high-level mode. If you have any macros defined, use "show <macro-name>" to see the listing of a macro. Let's define a macro here for practice.

Let's try "blocking out" a song. This will be analogous to locking some channels on cable TV for certain viewers. Say we don't want people to "hear" ***Strangers in the Night***. So, every time someone chooses ***Strangers in the Night***, we "play" ***Fly me to the Moon*** instead. Type the following in the XRAY terminal.

restart <CR>

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

```
Note: in startup routine. Press
 F9 to go to main.
```

clear 2 <CR>

We restarted the program and cleared the breakread breakpoint.

g <CR>

```
Break # 1 on instr at loc
00000042 module PT approx line
19 (PC = 00000046)
```

XRAY halts at line #19. Now, we will define the macro blck.
Type:

```
define int blck()
{
if (song==0x000a)
$c song=0x0007$;
return (1);
}
.
```

After the macro is defined, the command window may be expanded, press F4 to shrink/ expand the active window. Now the macro is defined. Once this macro is invoked, the value of "*song*" is changed to 7 if it's 0x0a. The "$'s" around the c expression indicate that this line is a debugger command. When a macro returns a "1," the program just continues execution after the macro; but, if a macros returns a "0," the program halts after the macro. When do we invoke this macro? It will not be a wise idea to single step and invoke the macro after each step. How about right after the user inputs the selection? Type:

b #42;blck() <CR>

This command tells XRAY to check for value 0x000a after the user inputs his/her selection. Let's see how this breakpoints changes the flow of the program. First we should clear the breakpoint at line #18. Type:

clear 1 <CR>
g <CR>

The program prints the same welcome message on the program terminal, asking for input. Let's try to "hear" ***Stranger in the Night***. At the program terminal, type:

a <CR>

Although we selected ***Stranger in the Night, Fly me to the Moon*** is displayed. One very useful way to utilize macros is to simulate hardware in the simulator version XRAY. For example, in the simulator version, we can set breakpoints and use macros to update status registers, which are normally updated by hardware.

Another useful command is **gostep**; this is especially useful when you are desperate. Say for some reason the stack goes out of bounds, and the program just hangs. We can write a simple macro that returns a "0" when the stack reaches a certain value. Let's say this macro is called ***st_alert()***. Let's assume also we have no idea how and where the stack goes out of bounds. We can enter ***gostep st_alert()***. XRAY will single step through the whole program, calling the ***st_alert*** macro after each step. This is extremely time consuming, but it will locate the problem.

SESSION CONTROL

There is a "save" command that only works in the simulator version; this command saves the register and memory contents into a file. This feature is not implemented in the emulator version because it will take too much time to

actually save all the memory contents.

However, this does not mean we have to do all the debugging in one session. Here is an alternative way:

Before starting to use any XRAY commands, type:

log on=<filename> <CR>

All the commands you enter from this point on will be sent to a viewport. When you are done with the debugging session, type:

log off <CR>

This will save all the commands in the file. Next time XRAY is invoked, type:

include <filename> <CR>

XRAY will then execute all the commands in the log file. This is not as convenient as "save" and 'restore," but it works for both the simulator and emulator versions.

If you wish to record results, as well as the commands, use the "journal" command; it works the same way as the "log" command, the difference being the journal files contain command results.

Note: The information is not automatically saved to a file. It is necessary to type in "log off" or "journal off" when you are done with the session.

SIMULATED I/O

If you are using the simulator version XRAY, you might want to simulate the actual input/output terminal. For example, if the serial port is at address **0xffdb**, we can see the output of the program by typing:

outport [0xffdb],std <CR>

This command tell XRAY to print the value of address 0xffdb to the standard I/O window of XRAY. Instead of "std," you can choose to use "c." This will cause the messages to be printed in the command window.

MEMORY

There are also commands to view or change memory locations. The following commands, followed by memory addresses or range, can perform a variety of functions. Please refer to the ***XRAYH83 H8/300 Debugger*** manual for more details.

Command Name	Summary
COMPARE	Find difference between two memory blocks
COPY	Copy memory block
DUMP	Display memory block in Command window
FILL	Fill memory block with value(s)
SEARCH	Search memory block for patterns of value(s)
SETMEM	Assign values to memory block
SETREG	Change value(s) of various registers

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

We have practiced with starting XRAY, customizing it, monitoring variables and memory locations, setting breakpoints, defining macros, and utilizing macros. These are just the basic commands to get XRAY running. There are more commands for more detailed analysis. Please refer to the ***XRAYH83 H8/300 Debugger*** manual for more commands.

```
/*-----------------------------------------------------------------*/
/*      XRAY DEMO, using H8/325 Flash Board                        */
/*-----------------------------------------------------------------*/
#include "c:\cstuff\ioaddr.c"      /*Address of I/O ports*/
#include "c:\mcch83\stdio.h"
#include <stdlib.h>

void cpu_init();           /*Initialize CPU (I/O ports...)*/
main()
{
unsigned int song;                 /*The chosen song*/
unsigned int counter;              /*generic counter used for loops*/
unsigned int delay;                /*generic delay counter*/
cpu_init();                        /*Initial serial ports*/
beginning:                         /*Program begins here*/
printf ("\r\n");
for (counter=0;counter<20;counter++)
                  {
                          for (delay=0;delay<50000;delay++)
                          {
                          ;
                          }
                  printf (" HITACHI America, Ltd.");
                  printf ("\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b");
                  }
printf ("\r\n");
for (counter=0;counter<20;counter++)
                  {
                          for (delay=0;delay<50000;delay++)
                          {
                          ;
                          }
                  printf (" Applications Engineering");
                  printf ("\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b");
                  }
printf ("\r\n\nThis Program Simulates a Jukebox, with no sound.\r\n");
printf ("\r\nEnter song number in hex : ");
iscanf ("%x", &song);              /*input stored as variable "song" */
printf ("\r\nYou have chosen the %x th song\r\n\n",song);
switch (song)
        {
        case 0x07:        printf ("Fly me to the moon        /-| \r\n");
                          printf ("   0                     /_| | \r\n");
                          printf ("        ___________________| | \r\n");
                          printf ("      /  o o o o o o o o o o  | \r\n");
                          printf ("      |_______ HAL  Air_____| \r\n");
                          printf ("                |     |   \r\n");
                          printf ("                 |    |    \r\n");
                          printf ("                 /____|     \r\n");
                          break;
        case 0x0a:        printf ("Strangers in the Night \r\n");
                          for (counter=0;counter<30;counter++)
                          {
                                  for (delay=0;delay<50000;delay++)
                                  {
                                  ;
```

Listing 1 Frank.c

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

```
                                }
                        printf (" ?? ");
                        printf ("\b\b\b");
                        }
                        printf (":)\r\n");
                        break;
        case 0x12:      for (counter=0;counter<20;counter++)
                        {
                        printf (" New York, New York ");
                        printf ("\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b");
                        }
                        printf ("\r\n");
                        break;
        case 0x05:      printf ("The way you look tonight \r\n");
                        printf ("Some ");
                                for(counter=0;counter<5000;counter++)
                                {
                                ;
                                }
                        printf ("day \r\n");
                        printf ("When I am feeling low \r\n");
                        printf ("When the world seems cold \r\n");
                        printf ("And I feel a glow just thinking of \r\n");
                        printf ("                                  y    y     ooooo
u    u\r\n");
                        printf ("                                   y  y      o   o
u    u\r\n");
                        printf ("                                    yy       o   o
u    u\r\n");
                        printf ("                                    yy       o   o
u    u\r\n");
                        printf ("                                    yy       ooooo
uuuuuu\r\n\n");
                        printf ("And the way you look tonight\r\n");
                        break;
        default: printf ("Song number not in selection!!\n\r");
                        break;
        }
        goto beginning;
}
void cpu_init()
{
     /*Initialize SCI port*/
     *(unsigned char *)sci0_smr=0x00;
     *(unsigned char *)sci0_brr=31;
     *(unsigned char *)sci0_scr=0x30;
}
```

Listing 1 cont'd

Linker command file:

```
start $0030
sect code = $0030
LISTMAP CROSSREF,INTERNALS,PUBLICS
load mainadr
sect mainvec= $0000
sect zerovars=$fb80
order code,const,strings,__INITDATA,heap,zerovars
```

DOS batch file:

```
mcch83 -g -c -o %1.obj %1.c
lnkh83 -c pt.cmd  -o %1.abs -m>%1.map %1.obj iscanf.obj h8325.lib
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

MRI Tools H8/300 Tutorial

Application Note

Software Development from C Source to Executable Files

Paul Yiu

INTRODUCTION

The MRI/Hitachi Toolkit, (consisting of the Compiler, Assembler, Linker, and Librarian) is capable of processing C source codes to create executable code. The final code can then be debugged using XRAY. This paper will serve as a tutorial to demonstrate the process software development, using the Hitachi/MRI Toolkit.

THE SAMPLE ROUTINE

A very basic routine, called ***Sinatra***, (listing #10 at the end of this document) will be used to demonstrate the process. Using the H8/325 as an example, the program prints a menu on the screen, then the user makes his choice. After the choice is made, the program outputs a corresponding message on the screen. The difficulty does not lie in programming, but setting up the standard I/O, registers, and different sections in memory. Fortunately, the set-up only has to be done once. Most of the command files and modules can be used for other programs.

SETTING UP

I/O

The source codes for ***s_write.c*** and ***iscanf.c*** need to be slightly modified to fit the hardware. Basically, we need to assign the correct serial ports for the read/write routines. These source codes are then compiled, assembled and put into a customized library. For further information please refer to Application Note #AE-0028 *H8/325 Standard I/O*.

XRAY

The emulator version of XRAY require some set-up when used for the first time. The default baud rate can be changed, as well as the display mode. Please refer to Tech Note #TN-0021 *Starting up Emulator version XRAY* for details.

Stack

The stack should be set at address 0xFF80 before any instructions begin. Listed below is file stack_ini.c.

```
#pragma asm
    .SECTION stack_ini,TEXT,ALIGN=2
    mov.w #h'ff80,sp
#pragma endasm
```

Listing 1: stack_ini.c

There is a very good reason these lines of code are not put inside ***main***. The compiler, at the start of every routine, puts the value of R6 on the stack. If we put "mov.w #h'ff80,sp" in the first line of ***main***, we would get:

```
_main:
        push  r6
        mov.w sp,r6
        mov.w #h'ff80,sp
```

Listing 2: uninitialized stack

This would mess up the stack. Writing a value on an unknown location could give undetermined results.

There is a routine, called ***s_start.c*** that automatically gets called when the program gets linked. This routine sets up the stack for XRAY. However, this routine is not actually a part of the program. For this program to work without the emulator (i.e. on the chip), the stack needs to be

initialized by the program itself.

Vectors

The first several address spaces are reserved for the vector table, which contains addresses of various interrupt functions. The reset vector is at address 0x0000. It should contain the address of need another file, called ***resetvec.c***, to manipulate the correct address. It is listed below.

```
/*Reset vector. To be put at address 0*/
void (*mainvec[])()=(unsigned int
        *)0x0030;
```

Listing 3: resetvec.c

This routine simply defines a dummy pointer to point to address 0x0030. We will put this pointer at address 0 with the Linker, later.

We don't have to specify a particular address for the pointer to point to. Say we have a function called delay_10 somewhere in the program. If we write another file, called ***delay10.c***. It will look like this:

```
extern void delay_10(void);
void (*vec10[])()={delay_10};
/*function delay_10 can be anywhere*/
```

Listing 4: delay10.c

Here, a dummy pointer points to the function "delay_10." We can assign this pointer to any address; this is useful for interrupt routines.

USING MRI TOOLS

It is a good idea to include the MRI tool directories in the autoexec.bat file. Also, create another directory to store all the programs. Figure 1 shows how the tools work together to create executable codes.

Before we go any further with the program, let's take a brief look at each tool.

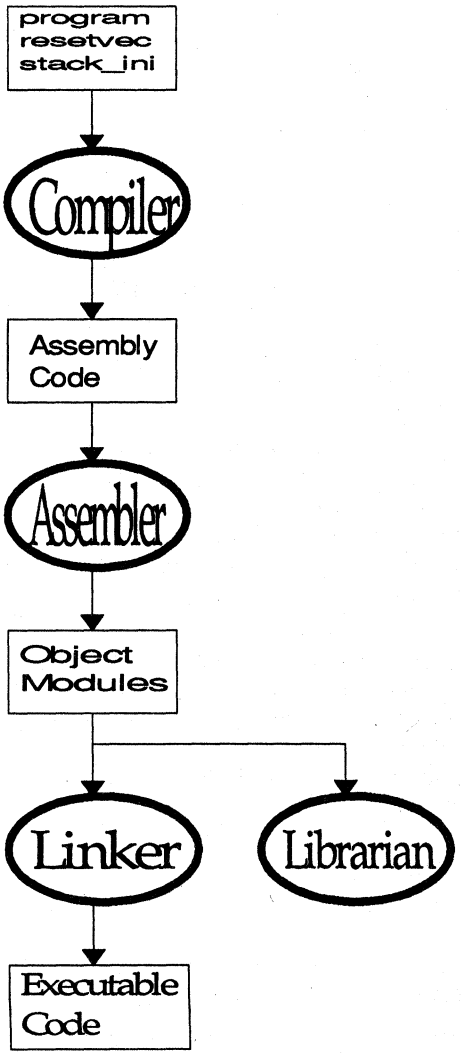


Figure 1 MRI Tools

Compiler

As far as the user is concerned, the compiler simply "turns" C code into Assembly code. The main subject of interest is the long list of command options. All of these command-line options are useful, but only a few are discussed here, for the sake of brevity.

To invoke the compiler, type:

mcch83 [option1 option2 option3.....]
source_filename <CR>

The "mcch83" command invokes more than just the compiler; this command behaves like a DOS batch file. If the user does not tell the software how far to process the program, it will go all the way and create an absolute file. The intermediary files, ***xxxx.src*** and ***xxxx.obj***, will not be available. It is better to specify how far we want the software to process with some command-line options. These options are listed below.

How far:

-S	The .c file gets compiled to create Assembly, a .src file
-c	The .c files gets compiled and assembled to create an object module, a .obj file.
None	The .c file gets processed all the way to create executable code, a .abs file

Table 1: Compiler Options

Debugging information:

-g	Line numbers and trace information are created. This option is extremely important for debugging in XRAY. Use this option almost always.
-Fsm	C source code is mixed into the assembly code as comments.

Table 2: Compiler Options

Output:

-o	This option, followed by a filename, will name the ouput file(s). If this option is not used, the default names (xxxx.src or xxxx.obj) will be used.
-l	This option, followed by a filename, will create a listing file containing the C source code and any errors/warning during compilation.

Table 3: Compiler Options

Naming the sections

-NIname	Thevariable section is renamed to whatever the user entered for "name".
-NCname	The constant section is renamed to whatever the user entered for "name".

Table 4: Compiler Options

The significance of the options will become clear later. Refer to the *Hitachi MCCH83 H8/300 C Compiler* maual for more compiler options.

Assembler

The assembler takes the assembly code and generates an object module. Most of the time the C source code is processed directly to the object module, so the assembler command-line options are not used very often.

To invoke the assembler, type:

asmh83 [option1 option2.....] source-filename <CR>

One very useful option is the "***-l > filename.***" This option generates a listing that contains all the errors and warning during assembly. Refer to the *Hitachi ASMH83 H8/300 Assembler* manual for more assembler options.

Linker

The linker links all the modules together and put in the appropriate places in memory. To invoke the linker, type:

lnkh83 [option1 option2.....] object-filename <CR>

Some important command-line options are listed here.

-c filename	The command file contains the linker commands.
-m> filename	The map file shows the locations of various modules, variable tables, and cross-reference tables.

Table 5: Linker Options

Librarian

The librarian provides a mechanism to create, delete, and edit libraries, which are just collections of object modules. The standard library contains functions like ***printf***, ***getchar***, etc. We can make a custom library for the H8325. Please refer to Application Note #AE-0028 *H8/325 Standard I/O* for more information. To invoke the librarian, type:

libh83 [option] [obj-filename] lib-filename <CR>

or

libh83 < cmd-filename <CR>

Listed below are some librarian command options.

-a	Adds modules to the designated library.
-d	Deletes modules from the designated library.
-e	Extracts modules from the library and generates ".obj" files.
-l	Generates a librarian listing.
-r	Replaces a module in the library with a ".obj" file module.

Table 6: Librarian Options

MAKING EXECUTABLE CODE

1) First we want to compile and assemble

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

the program. Type:

mcch83 -g -c sinatra.c <CR>

We will get a ***sinatra.obj*** file, with debugging information.

2) Next we want to compile and assemble the auxiliary source codes. Type:

mcch83 -g -c -NImainvec resetvec.c
<CR>
mcch83 -g -c stack_ini.c
<CR>

We will get files ***resetvec.obj*** and ***stack_ini.obj***, with debugging information. Since our reset vector is a variable, an "init-vars" section will be created in the ***resetvec*** module (.obj file). Using "***-NImainvec***" renames the *init-vars* section *mainvec*, so it won't be confused with the init-vars section of the main program.

3) Before the modules are linked together, we need to create a command file for the linker, telling the linker how to put modules in memory. This is what the command file, ***sinatra.cmd***, looks like.

```
        start $0030
        LISTMAP
CROSSREF,INTERNALS,PUBLICS
        load resetvec
        load stack_ini
        sect mainvec= $0000
        sect stack_ini=$0030
        sect zerovars=$fb80
        order stack_ini,code,const,strings+
        __INITDATA,heap,zerovars
```

Listing 5: Linker Command File

The first 0x30 bytes of the ROM are reserved for interrupt vectors. To leave the first 0x30 memory addresses alone, we tell the linker to start at address 0x30.

The second line adds a cross-reference table in the map. We will take a look at the map later.

The third and fourth lines simply load the modules and combine them with the program.

The "sect" commands puts sections in absolute addresses. We put the reset vector (0x0030) at address 0x0000, stack_ini code at address 0x0030, and variables at top of RAM. All of the program, with the exception of variables, sit in ROM.

> NOTE: **The format of the executable code will be IEEE. To generate s-record, just add "format s" in the command file. The ASE machine takes s-record format, while XRAY accepts IEEE format.**

The final step is to use the linker. Type:

lnkh83 -c sinatra.cmd -m>sinatra.map -o sinatra.abs sinatra.obj h8325
<CR>

We used "-c sinatra.cmd" to tell the linker to receive commands from the ***sinatra.cmd*** file. We get a map file that lists the locations of various modules by typing "-m>sinatra.map."

The final product, ***sinatra.abs***, is executable code, and it can be used for debugging. To download the program, we'll need to generate S record output. This would be accomplished by adding "***format s***" in the command file. Let's take a look at the ***sinatra.map*** file on page 7 (listing 6) and see where the sections went.

Listing #6 on page 7 shows the sections all went to the right places. We generated a cross-reference table and put it in the map file. Let's take a look at part of the cross-reference table. (listing #7 page 7)

The cross-reference table shows all the functions and variables, as well as the functions that called them.

There is also a module section in the map file; it lists all the modules, either user-defined or from the library, as well their sizes. Part of the module section is shown in listing #8, on page 7.

Some of the steps taken to generate executable code from C source may seem tedious, but they only have to be done once. Only minor modifications need to be done for different programs. For example, we can create a DOS batch file, called mk_code.bat. For the future programs, we only have to type:

mk_code program_name
<CR>

to create executable code.

This is what a ***mk_code.bat*** file might look like:

```
mcch83 -g -c -o %1.obj -l %1.lst %1.c

lnkh83 -c h8325.cmd  -o %1.abs -m>%1.map
%1.obj  h8325.lib
```

Listing 9 ***mk_code.bat***

The file ***h8325.cmd*** will probably look very much like sinatra.cmd. ***H8325.lib*** includes user-defined I/O routines. During execution, the symbol "%1" will be replaced by parameter "program_name." Basically, this batch file performs all the work in one command.

SUMMARY:

1) When compiling the C source code , use the -g and -l command-line options.

2) Create a separate module to set up the stack pointer; this module can be re-used by different programs.

3) Process all C codes with the -c command-line options to create object modules; then, use the linker to link all the modules together.

4) Create a command file for the linker. The command file can place modules in desired order.

5) When debugging with XRAY, use IEEE format output. For debugging with the ASE machine alone, use S record format output.

```
SECTION SUMMARY
---------

SECTION     ATTRIBUTE                    START  END    LENGTH ALIGN

mainvec     NORMAL DATA                  0000   0001   0002   2 (WORD)

stack_ini   NORMAL CODE                  0030   0033   0004   2 (WORD)
code        NORMAL CODE                  0034   19C1   198E   2 (WORD)
const       NORMAL CODE                  19C2   1B6A   01A9   2 (WORD)
strings     NORMAL CODE                  1B6C   1D2B   01C0   2 (WORD)
__INITDATA
            NORMAL DATA                  1D2C   1D2C   0000   2 (WORD)
heap        NORMAL DATA                  1D2C   1D2D   0002   2 (WORD)
zerovars    NORMAL DATA                  FB80   FD0B   018C   2 (WORD)
```

Listing 6: sinatra.map

```
_getNum10       code            0558    iscanf
_getNum16       code            0668    iscanf
_getchar        code            1950    getchar
_hcsr           const           1A46    pt
_ier            const           19EC    pt
_iscanf         code            0186    iscanf
_iscr          const     .............
```

Listing 7: cross-reference table

```
MODULE SUMMARY
--------------

MODULE          SECTION:START    SECTION:END    FILE

pt                const:194E       const:19D7   C:\XHIH83\PROGRAM\pt3.obj
                 strings:1AF8     strings:1BB7
                    code:0030        code:0137
iscanf          zerovars:FB80    zerovars:FB9B  C:\XHIH83\PROGRAM\iscanf.obj
                    code:0138        code:07B9
ctype             const:19D8       const:1AD8   C:\XHIH83\PROGRAM\h8325.lib  L
fakftoa           const:1ADA       const:1AF6   C:\XHIH83\PROGRAM\h8325.lib  L
                    code:07BA        code:07F7
flsbuf              code:07F8        code:090F  C:\XHIH83\PROGRAM\h8325.lib  L
ltostr              code:0910        code:09C5  C:\XHIH83\PROGRAM\h8325.lib  L
malloc          zerovars:FB9C    zerovars:FB9C  C:\XHIH83\PROGRAM\h8325.lib  L
```

Listing 8: modules

```
/*-------------------------------------------------------*/
/*      Program: Sinatra.c                               */
/*-------------------------------------------------------*/
#include "c:\cstuff\ioaddr.c"   /*Address of I/O ports*/
#include "c:\mcch83\stdio.h"
#include <stdlib.h>
void cpu_init();        /*Initialize CPU (I/O ports...)*/

main()
{

unsigned int song;              /*The chosen song*/
cpu_init();
beginning:
printf ("\r\nEnter song number in hex : ");
iscanf ("%x", &song);
printf ("\r\nYou have chosen the %x th song\r\n\n",song);
switch (song)
        {
        case 0x07:      printf ("Fly me to the moon       /-| \r\n");
                        printf ("   0                    /_| | \r\n");
                        printf ("      ___________________| | \r\n");
                        printf ("     /  o o o o o o o o o  | \r\n");
                        printf ("    |_______ HAL  Air_____| \r\n");
                        printf ("              |     |    \r\n   ");
                        printf ("               |    |    \r\n");
                        printf ("               /____|     \r\n");
                        break;
        case 0x0a:      printf ("Strangers in the Night \r\n");
                        break;
        case 0x12:      printf ("New York, New York \r\n");
                        break;
        case 0x05:      printf ("The way you look tonight \r\n");
                        break;
        default: printf ("What? Not on this CD!!\n\r");
                        break;
        }
        goto beginning;
}
void cpu_init()
{
/*-----------*/
/*set for address outputs*/
/*-----------*/
    /*Initialize SCI port*/
    *(unsigned char *)sci0_smr=0x00;
    *(unsigned char *)sci0_brr=31;
    *(unsigned char *)sci0_scr=0x30;
}
```

Listing 10 *Sinatra.c*

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Assembler instructions, after the linker.

```
0030  7907 FF80  mov.w    #H'FF80:16,sp
0032  FF80       mov.b    #H'80:8,spl
0034  6DF6       push.w   r6
0036  0D76       mov.w    sp,r6
0038  7900 0006  mov.w    #6,r0
003A  0006       nop
003C  1907       sub.w    r0,sp
003E  6DF2       push.w   r2
0040  6DF3       push.w   r3
0042  5E00 00FA  jsr      @cpu_init:16
0044  00FA       nop
0046  7900 0001  mov.w    #mainvec+1,r0
0048  0001       nop
004A  6FE0 FFFA  mov.w    r0,@(-6:16,r6)
004C  FFFA       mov.b    #H'FA:8,spl
004E  1900       sub.w    r0,r0
0050  6FE0 FFFC  mov.w    r0,@(-4:16,r6)
0052  FFFC       mov.b    #H'FC:8,spl
0054  1922       sub.w    r2,r2
0056  F801       mov.b    #mainvec+1,r0l
0058  6EA8 FC64  mov.b    r0l,@(-924:16,r2)
005A  FC64       mov.b    #H'64:8,r4l
005C  0B02       adds.w   #1,r2
005E  7900 0011  mov.w    #H'11:16,r0
0060  0011       nop
0062  1D02       cmp.w    r0,r2
0064  4DF0       blt      H'F0  ; =>56
0066  1922       sub.w    r2,r2
0068  6E28 FC64  mov.b    @(-924:16,r2),r0l
006A  FC64       mov.b    #H'64:8,r4l
006C  474C       beq      H'4C  ; =>BA
006E  0D20       mov.w    r2,r0
0070  0900       add.w    r0,r0
0072  0B80       adds.w   #2,r0
0074  0B00       adds.w   #1,r0
0076  6FE0 FFFE  mov.w    r0,@(-2:16,r6)
0078  FFFE       mov.b    #H'FE:8,spl
007A  0D03       mov.w    r0,r3
007C  0923       add.w    r2,r3
007E  400C       bra      H'C  ; =>8C
0080  F800       mov.b    #0,r0l
0082  6EB8 FC64  mov.b    r0l,@(-924:16,r3)
0084  FC64       mov.b    #H'64:8,r4l
0086  6F60 FFFE  mov.w    @(-2:16,r6),r0
0088  FFFE       mov.b    #H'FE:8,spl
```

Listing 11 disassembled code

```
008A  0903        add.w    r0,r3
008C  7900 0011   mov.w    #H'11:16,r0
008E  0011        nop
0090  1D03        cmp.w    r0,r3
0092  4DEC        blt      H'EC  ; =>80
0094  6F60 FFFC   mov.w    @(-4:16,r6),r0
0096  FFFC        mov.b    #H'FC:8,spl
0098  0B00        adds.w   #1,r0
009A  6FE0 FFFC   mov.w    r0,@(-4:16,r6)
009C  FFFC        mov.b    #H'FC:8,spl
009E  6F60 FFFE   mov.w    @(-2:16,r6),r0
00A0  FFFE        mov.b    #H'FE:8,spl
00A2  6DF0        push.w   r0
00A4  6F60 FFFC   mov.w    @(-4:16,r6),r0
00A6  FFFC        mov.b    #H'FC:8,spl
00A8  6DF0        push.w   r0
00AA  7900 1AFA   mov.w    #H'1AFA:16,r0
00AC  1AFA        dec.b    r2l
00AE  6DF0        push.w   r0
00B0  5E00 0BC4   jsr      @printf:16
00B2  0BC4        adds.w   #2,r4
00B4  7904 0006   mov.w    #6,r4
00B6  0006        nop
00B8  0947        add.w    r4,sp
00BA  0B02        adds.w   #1,r2
00BC  7900 0011   mov.w    #H'11:16,r0
00BE  0011        nop
00C0  1D02        cmp.w    r0,r2
00C2  4DA4        blt      H'A4  ; =>68
00C4  6F60 FFFA   mov.w    @(-6:16,r6),r0
00C6  FFFA        mov.b    #H'FA:8,spl
00C8  0B00        adds.w   #1,r0
00CA  6FE0 FFFA   mov.w    r0,@(-6:16,r6)
00CC  FFFA        mov.b    #H'FA:8,spl
00CE  7901 0001   mov.w    #mainvec+1,r1
00D0  0001        nop
00D2  1D10        cmp.w    r1,r0
00D4  4E04        bgt      H'4  ; =>DA
00D6  5A00 004E   jmp      @H'4E:16
00D8  004E        nop
00DA  6F60 FFFC   mov.w    @(-4:16,r6),r0
00DC  FFFC        mov.b    #H'FC:8,spl
00DE  6DF0        push.w   r0
00E0  7900 1B0B   mov.w    #H'1B0B:16,r0
00E2  1B0B        subs.w   #1,r3
00E4  6DF0        push.w   r0
00E6  5E00 0BC4   jsr      @printf:16
00E8  0BC4        adds.w   #2,r4
00EA  0B87        adds.w   #2,sp
```

Listing 11 cont'd

```
00EC  0B87       adds.w   #2,sp
00EE  0180       sleep
00F0  6D73       pop.w    r3
00F2  6D72       pop.w    r2
00F4  0D67       mov.w    r6,sp
00F6  6D76       pop.w    r6
00F8  5470       rts
00FA  6B00 19B6  mov.w    @sci0_smr:16,r0
00FC  19B6       sub.w    r3,r6
00FE  F900       mov.b    #0,r1l
0100  6889       mov.b    r1l,@r0
0102  6B00 19B8  mov.w    @sci0_brr:16,r0
0104  19B8       sub.w    r3,r0
0106  F91F       mov.b    #H'1F:8,r1l
0108  6889       mov.b    r1l,@r0
010A  6B00 19BA  mov.w    @sci0_scr:16,r0
010C  19BA       sub.w    r3,r2
010E  F930       mov.b    #H'30:8,r1l
0110  6889       mov.b    r1l,@r0
0112  5470       rts
```

END of listing 11

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

MRI Toolkit H8/300

Application Note

H8/325 Standard I/O

Paul Yiu

INTRODUCTION

The MRI/Hitachi toolkit for the H8/300 is supplied with standard I/O routines; however, these routines were written to simulate I/O on the PC terminal. This means the user serial I/O interface goes through the debugging terminal, not the serial ports of the CPU. To have a "real" user I/O interface, we need to change some source code, compile it, and put it in the correct library module as replacement routines.

The two most important I/O functions are ***printf*** and ***iscanf***. ***Printf*** outputs variables and strings to the screen, while ***iscanf*** takes user's input or commands.

Code Operation

Figure 1 shows how ***printf*** works. We only need to change ***s_write.c*** to change the ***printf*** routine.

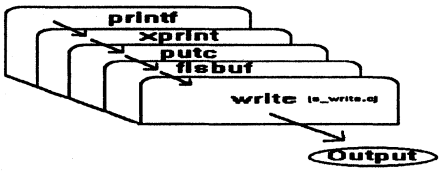


Figure 1: ***printf*** routine

Parts of the original s_write.c are listed here.

```
for (i=nbyte;i!=0;i--)
   _simulated_output=*buffer++;
   return (nbyte);
```

Listing 1: ***s_write.c*** (original)

The output is sent to the variable *_simulated_output*, which is displayed on the XRAY debugging output screen. To make the output go to a serial port, we only need to replace the variable with the serial port address.

The modified ***s_write.c*** is listed below.

```
/*sci0_ssr and sci0_tdr are defined*/

/*as constants in another file*/

extern int const sci0_ssr;

extern int const sci0_tdr;

/*function write*/
int write (int fd, char *buffer,
unsigned nbyte)
{
unsigned int i;

for (i = nbyte; i != 0; i--)
   while ( (*(unsigned char *)sci0_ssr)
< 0x80)
        {
        ;/*wait till ready to transmit*/
        }
  *(unsigned char *)sci0_tdr=*buffer++;
  *(unsigned char *)sci0_ssr =
((*(unsigned char *)sci0_ssr)&0x7f);


        return (nbyte);
        }
```

Listing 2: ***s_write.c***

The two variables, *sci0_tdr* and *sci0_ssr* are defined as constants. To cast the variables into addresses, we put "***(unsigned char *)***" in front of the variables. Another "*" in front accesses the content of a specific address. Please refer to Technote *#TN-0020 Direct Memory Addressing with C Pointers* for more details on pointers and casting.

The code in bold and italic are modified for the H8/325. Bit 7 of **sci0_ssr** (Serial Status Register) , when set to 1, indicates that Transmit

SECTION 7

Data Register (TDR) is empty; this means "ready to transmit." Sci0_tdr is the address of the TDR, where data goes to be output. After the data is sent to the data register, we clear bit 7 of the Serial Status Register because the TDR is now full. The hardware will set bit 7 of Serial Status Register after the data is transmitted to the shift register. Now the CPU is ready to transmit another byte of data.

Input

The standard input routine is ***iscanf***. The source code of ***iscanf*** is available in the package. ***Iscanf*** calls the macro "***get_port***," which in turn calls "***getchar***." ***Getchar*** can be used as a macro or a function. In ***iscanf.c***, ***getchar*** the macro is used, but it is much more convenient to redefine the function, so we won't have to deal with the ***stdio.h*** file. Listed below is part of the supplied ***iscanf.c*** source code:

```
#define GET_PORT() (sf_len++,getchar())
#define UNGET_PORT(c) (sf_len--,ungetc(c,
stdin))

/* Function prototypes */
```

Listing 3: iscanf.c (partial)

To call the function instead of the macro, we just add parenthesis around "***getchar***."
So, the new ***iscanf*** code contains the following:

```
#define GET_PORT() (sf_len++,(getchar)())
#define UNGET_PORT(c) (sf_len--,ungetc(c,
stdin))

/* Function prototypes */
.......
```

Listing 4: iscanf.c (partial)

The original ***getchar*** routine tries to get input from the debugging terminal. We need to modify it to accept characters from the serial port. What we will do is to write a ***getchar.c*** routine, compile and assemble it, then put it in the library, so each time ***getchar*** is called, our modified function will be used. Listed below is the custom-made ***getchar*** routine:

```
extern int const sci0_ssr;
extern int const sci0_rdr;

unsigned char getchar()
{
unsigned char c;
while (((*(unsigned char *)sci0_ssr) |
0xbf) == 0xbf)
        {
        ;/*loop till character a
        vailable*/
        }
c=*(unsigned char *)sci0_rdr;
*(unsigned char *)sci0_ssr=((*(unsigned
char *)sci0_ssr)&0xbf);
printf("%c",c); /*echo input on the
screen*/
return (c);
}
```

Listing 5: ***getchar.c***

Compilation of source codes

Now there are three new C source codes, ***iscanf.c***, ***s_write.c***, and ***getchar.c***. These routines need to be compiled, assembled, and linked into the library for future use. Instead of modifying the original library, we should make a copy of the original and modify the second copy. The library is located in ***c:\xhih83\lib*** directory; it's called ***ch83emc.lib***. First, we'll make a copy of it, called h8325.lib. This library is just a collection of all the modules of standard functions, such as printf, scanf, getchar, putchar, etc. Included in the MRI Toolkit is a librarian utility; we'll use the librarian to modify our H8325.lib file. The librarian utility is a "module manager." The librarian can create, edit, add, or

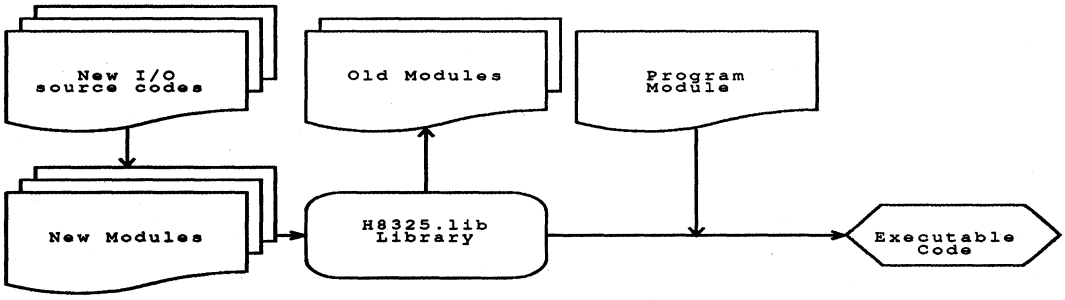


Figure2: Standard I/O Modules

dele te libraries that contain often-used functions. For more information on the librarian, please refer to Application Note AE-0029, *Software Development from C source to S record.*

First we want to take out the old s_write and getchar modules from the library.

c:\> ***libh83 -e s_write h8325.lib***
c:\> ***libh83 -e getchar h8325.lib***

The "-e" flag extracts these modules. Now there are two .obj files called s_write.obj and getchar.obj. These files should be renamed, so they are not confused with our new modules. We'll rename them as s_write.old and getchar.old. Now we want to delete these two modules from the library.

c:\> ***libh83 -d s_write h8325.lib***
c:\> ***libh83 -d getchar h8325.lib***

The "-d" flag deletes modules from the library.

The next step is to compile and assemble our new source codes.

c:\> ***mcch83 -c s_write.c***
c:\> ***mcch83 -c getchar.c***
c:\> ***mcch83 -c iscanf.c***

The "-c" flag tells the compiler to make ***.obj*** files with the source, without creating executable files. Now, there should be **.c**, **.obj**, and **.old** files. The last thing is to put our new modules into the library.

c:\> ***libh83 -a s_write.obj h8325.lib***
c:\> ***libh83 -a getchar.obj h8325.lib***
c:\> ***libh83 -a iscanf.obj h8325.lib***

The "-a" flag adds modules to the library.

The h8325.lib file now contains the updated I/O routines for future use. The h8325.lib file should be linked in when creating executable files. For more information on the linker and its command line options, please refer to Application Note AE-0029, *Software Development from C source to S record.*

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD61830B/LM200

Tutorial

Forth Display Routine Demonstration

Marnie Mar

Introduction

Replacing a debug monitor with a Forth interpreter gives hardware and software developers new options in developing and debugging designs. This Tutorial demonstrates the use of a Forth interpreter through the writing of test and application programs for an LCD display system.

Forth interpreters are now available for use with Hitachi's H8/532 and H8/330 microcontroller evaluation boards. The interpreters are available in object code form to be programmed into EPROM.

This Tutorial assumes some knowledge of Forth, and/or access to a Forth reference document. While the Forth examples shown here apply to an H8/532-based LCD display system, the ideas used apply to other microcontroller-based system developments.

A detailed description of this LCD Display system is available in a Tutorial on the HD61830B and LM200. This Tutorial is document number AE150, available from your local Field Application Engineer. Please refer to this document for system details.

Forth Simplifies Debug Monitor Functions

Using a Forth interpreter speeds up microprocessor-based hardware checkout by allowing users to exercise the circuitry without generating native object code. Hardware designers can test circuitry by using Forth commands for interactively writing to and reading from memory locations and hardware registers in the memory or I/O map.

Memory interfaces can be tested by writing to memory locations and reading back to see that data was written. Peripheral device interfaces are tested similarly, and peripheral functions are tested by initializing control registers and checking for the expected operation.

When using a debug monitor, the user is required to perform many steps just to get a test program into memory. One method is to hand enter machine code into memory to be executed using monitor commands. Another method relies on the monitor to provide a line assembler, which requires the user to type assembly commands line by line. The third method requires an editor and a cross-assembler running on a PC, which are used to generate an object code file that must then be downloaded to target memory.

The Forth interpreter eliminates these steps by allowing the user to enter high-level commands which control the system. These commands are executed (interpreted) with each carriage return.

Obtaining Forth

The object code files for the Forth interpreters (and the demonstration files used here) are available on the HAL Application Engineering Bulletin Board system. Please contact your local FAE for information on accessing this bulletin board and obtaining these files.

Installing Forth

The Forth object code should be programmed into an HN27C256 or HN27512 EPROM by downloading the S-type or binary format object code into a device programmer. If an HN27512 device is used for the H8/532 board, program the Forth object code starting at device address 8000h. Otherwise, program the object code starting at address 0h.

Once programmed, the Forth EPROM replaces the debug monitor EPROM on the H8/330 Evaluation Board (US338EVB01H) or the H8/532 Evaluation Board (US538EVB21H). Both boards use the shipping switch configurations with the following exception: on the H8/330 Evaluation Board, jumper W2 should be set for the EPROM size used.

Starting Forth

After installing the Forth EPROM and confirming jumper placement, connect a terminal or a PC running terminal emulation software to the evaluation board. Configure the terminal for 9600 baud, no parity, one stop bit. Apply power to the board, and a Forth sign-on message followed by the "ok" prompt should appear.

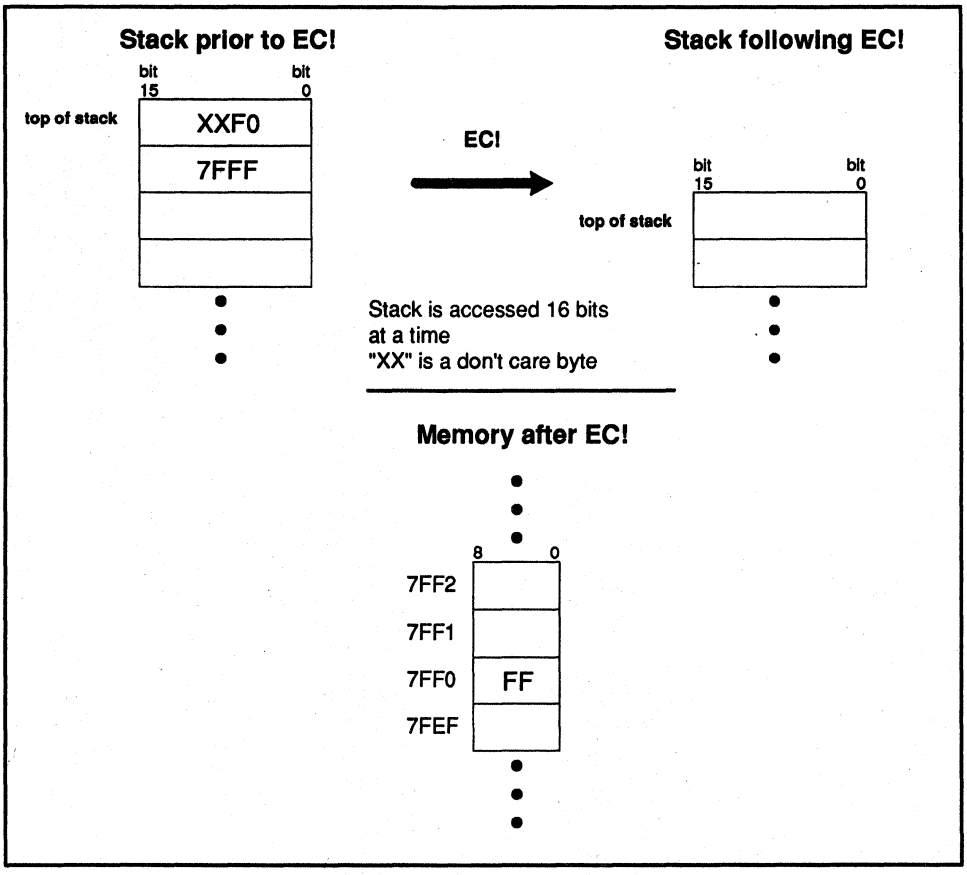


Figure 1 - EC! Stack and Memory Operation

Using Forth

Following the "ok" prompt, type VLIST (note that all Forth words are defined in capital letters, and text entries are case sensitive). This lists all words defined in the Forth dictionary. Some of these words are standard Forth commands and should be described in your Forth reference document. Other words are recognizable as constant names defined for on-chip peripheral register locations.

At this point, Forth commands can be entered, and values can be placed on the stack. How are values placed on the stack? When a user types in information following the "ok" prompt, Forth first determines if the typed value is the name of an operation currently defined in the dictionary (either part of the kernel or defined by the user). If so, the defined operation is carried out. If not, Forth assumes this is a value to be placed on the stack.

To view the contents of the stack, use ".S". This displays stack contents without changing the stack. "." displays the value on the top of the stack and also removes the value fromt he stack.

Testing an interface

The first task is to determine if the H8/532 to HD61830B interface is performing properly. To check this interface, values can be written to registers in the device, then read back

for verification. This is accomplished in H8 Forth using the operators EC! (to write to an address) and EC@ (to read from an address).

EC! and EC@ were defined for H8 Forth based on the standard Forth operators C! and C@. The H8 microcontroller instruction set includes instructions that synchronize data transfers to and from peripheral addresses using E clock timing. C! and C@ are based on reads and writes using standard H8 bus timing, while EC! and EC@ are based on reads and writes using the E clock timing.

Forth is a stack-based language that takes input parameters for an operation off the stack, and places operation results back on the stack. For EC!, Forth pops the word value on the top of the stack and accepts it as a memory address. The next byte on the stack is popped as a value, which is written to the popped address. This operation stores no results to the stack. See Figure 1 for a diagram of this operation.

The EC@ operation reads a value from an address. The address is popped from the top of the stack. A byte is read from this address, and is pushed onto the stack to be used by Forth operations to be executed in the future.

Documenting Forth

Listings of Forth operations are commented using a description of what is popped from the stack and what is pushed to the stack. For instance, in the case of EC!, the listing would show:

```
EC!   ( address value – )
```

EC! is the operation performed. The opening paren signals the Forth interpreter that a comment follows. It must be followed by a space. EC! pops first an address, then a value. The "–" separates what is popped from what is pushed. In this example, no data is pushed back on the stack, so none is shown. The closing paren terminates the comment.

Similarly, EC@ would be commented as follows:

```
EC@   ( address – value )
```

showing that EC@ pops an address from the stack, and places the value read from the address on the stack following the operation.

Testing a peripheral interface

The following sequence can be used to test the interface between the microcontroller and an E clock peripheral with a read/write register at address 8000h:

```
HEX   ( all numerical values are hex )
FF 8000 EC! ( write FFh to  8000h )
8000 EC@       ( read  location 8000h, place value
                on stack )
      ( remove top stack value and display )
```

The word HEX causes all further numerical inputs to be recognized as hex values. These commands are typed in response to the "ok" prompt. When the value read from the register is displayed, the user can see if the it is FFh as expected. Other write values can be tested in the same way.

Testing the HD61830B interface

Testing the interface between the H8/532 and the HD61830B is not as simple as performing a write and a read, since this device does not have directly addressed registers. To access a register, the user must first write the address of the register to the device's instruction register, then either read or write data in a data holding register.

Labels can be used in place of the numerical register addresses and register numbers. The registers and register numbers can be defined for the HD61830B as follows:

```
7FF1 CONSTANT INSTREG
7FF0 CONSTANT WRITREG
7FF1 CONSTANT BFLAG
0 CONSTANT MODEREG
1 CONSTANT CHARPITCH
2 CONSTANT NUMCHAR
3 CONSTANT NUMTIMES
4 CONSTANT CURPOS
8 CONSTANT DSTARTLO
9 CONSTANT DSTARTHI
A CONSTANT CURSLO
B CONSTANT CURSHI
C CONSTANT WRTDSP
D CONSTANT RDDISP
E CONSTANT CLRBIT
F CONSTANT SETBIT
```

Each line would be entered followed by a carriage return, and the interpreter would respond with "ok" as the constant is accepted.

Once the register addresses have been defined as constants, writing of data can be simplified using a Forth *colon definition* called WRITVAL:

```
: WRITVAL
  INSTREG EC!   ( write to INSTREG )
  WRITREG EC!   ( write to WRITREG )
  BEGIN
    BFLAG EC@   ( read BFLAG  )
    80 AND      ( AND with 80h )
    80 <>   ( loop until not equal )
  UNTIL   ( CONDITION - )
;
```

A colon definition defines a new operation, which becomes part of the Forth dictionary. Whenever WRITVAL is typed followed by a carriage return, this operation is carried out. A colon definition is terminated by a semicolon. When such a definition is entered to a Forth interpreter, the "ok" prompt is not displayed until the entire definition has been entered, and the semicolon terminator has been received. During definition (between the ":" and the ";") no execution takes place. The defined word must be entered as a command in order for the defined function to take place.

As shown in the comment on the first line of WRITVAL, this operation takes a register value and a register number off the stack, and places no data back on the stack. In other words, when the WRITVAL operation is requested, the user must make sure that these two values are on the stack, or an error will occur.

To write a value the value 12h into the mode register, use the following sequence (output from interpreter is underlined):

place 12h on the stack:

```
ok  12
```

place the register number on the stack:

```
ok  MODEREG
```

execute WRITVAL:

```
ok  WRITVAL
```

The function is complete when the "ok" prompt is returned.

The HD63810B should not be accessed again until the requested operation (write to the mode register) has completed. The device sets the most significant bit of the BFLAG register when it is busy, and clears this bit when the operation has completed. The second half of the WRITVAL colon definition takes care of this.

The BEGIN—UNTIL sequence is a looping construct. When the UNTIL is reached, Forth pops the value at the top of the stack. A value of zero is a false condition, which causes a loop back to BEGIN. A non-zero value is taken to be a true condition, which terminates the loop.

The busy flag is checked by first reading the BFLAG register, ANDing the value with 80h, then comparing the value with 80h. If the value is equal, a busy condition exists, and the loop should continue. Otherwise, the loop ends.

Writing and Reading Display RAM

Since the HD61830B lacks read/write registers, a write then read back test cannot be performed. An alternative method of testing is to cause the HD61830B to write to display RAM, then read back the written value. To write to the display RAM, the HD61830B must be initialized as to operating mode, display RAM start location, and cursor start location. The cursor moves forward with each display RAM access, so to read the same location that was written, the cursor must be backed-up. The operation is as follows:

initialize mode
initialize display RAM start address
initialize cursor location
write to display memory
move cursor back
read from display memory
compare read and written values

A sequence of Forth commands to perform this operation is shown in Appendix A.

Writing display demonstrations

Once the interface has been tested, Forth can be used to write display programs for demonstration. Rather than enter each program line by line to the interpreter, it is possible to combine Forth operations and colon definitions into a text file. These files can then be downloaded from a PC to the user system using the file transfer capabilities of a terminal emulator program. The Forth interpreter accepts these inputs as if they were inputs from the keyboard.

The interpreter interprets Forth operations line by line. Therefore, there may be a delay between the time a line of your file is accepted, and the time the interpreter is ready for the next line. If the next line is sent by the PC too soon, the Forth interpreter will miss characters.

To minimize the chance of missed characters, a feature of the terminal emulation package is used to cause a pause between

download of a carriage return and the following line. This pause can be adjusted to suit the file being downloaded.

The four demonstration files perform the following tasks:

- initialize the LCD controller in character mode, and display the full character set available on the HD61830B device

- initialize the LCD controller in graphics mode, and display a checkerboard pattern

- initialize the LCD controller in graphics mode, and tile the display with different tile patterns

- initialize the LCD controller in graphics mode, prompt the user for an eight byte tile pattern (8 dot x 8 dot tile), and tile the display with this pattern.

The Forth files required to run these programs are listed in Appendix B.

Running the demonstration programs

Running the demonstration programs requires an LCD Software Development Station (available from Hitachi America, Ltd.), a Forth EPROM for the H8/532 evaluation board, a PC, the demonstration Forth files and a terminal emulation software package. PROCOMM Plus, available from Datastorm Technologies, Inc. was used during development.

The terminal emulation software should be set up to provide a delay following each line downloaded. PROCOMM Plus can be setup to perform this line pacing by accessing the line pacing parameter in the ASCII Transfer Options Setup screen. The demonstrations shown here were transferred with 30 second pauses between lines. This parameter should be lengthened if necessary so that no missed characters occur between the time that a Forth command is sent and the time that the interpreter is ready to receive another command.

When the "ok" prompt is on the screen, initiate the download using the file transfer features of the terminal emulation package. Lines of the file will be displayed, followed by "ok". The Forth interpreter executes the commands as if they were typed from the keyboard by the user. Watch the LCD display for the program results.

When the download is completed, remember that the colon definitions of the file have been loaded to the dictionary and can be entered at the keyboard and used for further demonstration. For instance, following the download of the pattern tiling file, the pattern names defined can be entered to cause these patterns to display on the LCD.

Since each file redefines the CONSTANT values, the H8/532 board of the Software Station should be reset prior to downloading the another demonstration file.

Summary

A Forth interpreter replacing the debug monitor on an H8/532 or H8/330 evaluation board can be used to test hardware interfaces and generate hardware test programs. These programs can be stored in ASCII file format for downloading for future execution. Once downloaded, colon definitions defined in the file can be used as Forth commands to the interpreter.

The hardware-friendly interface of Forth minimizes the time and code generation required to test system circuitry. Writing application software in any language is greatly simplified when the hardware has been well tested with Forth routines.

For more information on any of the hardware or software tools described in tutorial, please contact the Field Application Engineer in your local Hitachi Field Sales office.

Appendix A - RAM write and read through HD61830B

```
( Forth commands to test writing then reading of
(  Display RAM using the HD61830B in character mode )

( values to be entered in hexadecimal )
HEX

( define HD61830B register address constants )
7FF1 CONSTANT INSTREG
7FF0 CONSTANT WRITREG
7FF0 CONSTANT RDREG

( define HD63180B register number constants )
0 CONSTANT MODEREG
4 CONSTANT CURPOS
8 CONSTANT DSTARTLO
9 CONSTANT DSTARTHI
A CONSTANT CURSLO
B CONSTANT CURSHI
C CONSTANT WRTDISP
D CONSTANT RDDISP


MODEREG INSTREG EC!      ( write reg # to INSTREG )
1C WRITREG EC!           ( write mode )

DSTARTLO INSTREG EC!
00 WRITREG EC!           ( load Disp RAM start addr lo)

DSTARTHI INSTREG EC!
00 WRITREG EC!           ( load disp RAM start addr hi)

CURSLO INSTREG EC!
00 WRITREG EC!           ( initialize cursor)

CURSHI INSTREG EC!
00 WRITREG EC!           ( initialize cursor)

WRTDISP INSTREG EC!
AA WRITREG EC!           ( write AA to display RAM )

CURSLO INSTREG EC!
00 WRITREG EC!           ( move cursor back - write to RAM automatically moves it forward)

CURSHI INSTREG EC!
00 WRITREG EC!           ( move cursor back )

RDDISP INSTREG EC1       ( read RAM and display )
RDREG EC@
                         ( value displayed should equal value written )
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Appendix B - Forth Demonstration Files

Character Mode Demonstration

```
( Demonstration Program for H8/532 and HD61830B system )
(        Demonstrates character display mode )

( values to be entered in hexadecimal )
HEX

( define HD61830B register address constants )
7FF1 CONSTANT INSTREG
7FF0 CONSTANT WRITREG
7FF1 CONSTANT BFLAG

( define HD63180B register number constants )
0 CONSTANT MODEREG
1 CONSTANT CHARPITCH
2 CONSTANT NUMCHAR
3 CONSTANT NUMTIMES
4 CONSTANT CURPOS
8 CONSTANT DSTARTLO
9 CONSTANT DSTARTHI
A CONSTANT CURSLO
B CONSTANT CURSHI
C CONSTANT WRTDISP
D CONSTANT RDDISP
E CONSTANT CLRBIT
F CONSTANT SETBIT

( WRITVAL writes a word to an HD61830B register, then waits
( for Busy condition to clear)
: WRITVAL  ( REGVAL REGNUM — )
INSTREG EC!  ( REGNUM — )
WRITREG EC!  ( REGVAL — )
BEGIN
BFLAG EC@  ( — FLAG )
80 AND     ( FLAG — BIT7 TEST)
80 <>  ( BIT7 TEST — CONDITION )
UNTIL  ( CONDITION — )
;
( CLEARSCREEN moves the cursor to the starting position and
( write 20h to screen to display spaces )
: CLEARSCREEN
00 CURSLO WRITVAL
00 CURSHI WRITVAL
1FF 0 DO
20 WRTDISP WRITVAL
LOOP
;
```

(continued on next page)

SECTION 7

Appendix B - Forth Demonstration Files - continued

Character Mode Demonstration

```
( HD61830B Initialization commands)
1C MODEREG WRITVAL   (  char mode, display off)
95 CHARPITCH WRITVAL (  10 x 6 block char )
27 NUMCHAR WRITVAL
1F NUMTIMES WRITVAL
08 CURPOS WRITVAL  (  display cursor at line 9)
00 DSTARTLO WRITVAL
00 DSTARTHI WRITVAL

CLEARSCREEN
3C MODEREG WRITVAL  (  turn on display )
( DISPSCREEN loops from 20h to FFh, displaying the
( character corresponding to the loop variable )
: DISPSCREEN
  00 CURSLO WRITVAL
  00 CURSHI WRITVAL
  FF 20 DO
   I WRTDISP WRITVAL
  LOOP
;
( DISPLAY CHARACTER SET )
DISPSCREEN
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Appendix B - Forth Demonstration Files - continued

Checkerboard Demonstration

```
(    Forth commands to display a checkerboard pattern on )
(  LCD panel using HD61830B graphics mode )

( all following numeric entries in hexadecimal )
HEX

(  define HD61830B register address constants )
7FF1 CONSTANT INSTREG
7FF0 CONSTANT WRITREG
7FF1 CONSTANT BFLAG

( define HD63180B register number constants )
0 CONSTANT MODEREG
1 CONSTANT CHARPITCH
2 CONSTANT NUMCHAR
3 CONSTANT NUMTIMES
4 CONSTANT CURPOS
8 CONSTANT DSTARTLO
9 CONSTANT DSTARTHI
A CONSTANT CURSLO
B CONSTANT CURSHI
C CONSTANT WRTDISP
D CONSTANT RDDISP
E CONSTANT CLRBIT
F CONSTANT SETBIT

( WRITVAL writes a word to an HD61830B register, then waits
( for Busy condition to clear)
: WRITVAL  ( REGVAL REGNUM — )
  INSTREG EC!   ( REGNUM — )
  WRITREG EC!   ( REGVAL — )
  BEGIN
   BFLAG EC@   ( — FLAG )
   80 AND      ( FLAG — BIT7 TEST)
   80 <>      ( BIT7 TEST — CONDITION )
  UNTIL   ( CONDITION — )
;

( CLEARSCREEN moves the cursor to the starting position and
( write 20h to screen to display spaces )
: CLEARSCREEN
  00 CURSLO WRITVAL
  00 CURSHI WRITVAL
  A00 0 DO
   00 WRTDISP WRITVAL
  LOOP
;
```

(continued on next page)

SECTION 7

Appendix B - Forth Demonstration Files - continued

Checkerboard Demonstration

```
( HD61830B Initialization commands)
12 MODEREG WRITVAL    ( graphics mode, display off)
07 CHARPITCH WRITVAL ( 8 bits per byte storage)
1D NUMCHAR WRITVAL
1F NUMTIMES WRITVAL
00 DSTARTLO WRITVAL
00 DSTARTHI WRITVAL

CLEARSCREEN

(  move cursor back to start position )
00 CURSLO WRITVAL
00 CURSHI WRITVAL

32 MODEREG WRITVAL    ( TURN ON DISPLAY )

( BLACKROW displays a row of checkerboard pattern starting
( with black square)
: BLACKROW
    0F 00 DO
      FF WRTDISP WRITVAL
      00 WRTDISP WRITVAL
    LOOP
;
( WHITEROW displays a row of checkerboard pattern starting
( with white square)
: WHITEROW
    0F 00 DO
       00 WRTDISP WRITVAL
       FF WRTDISP WRITVAL
    LOOP
;
( CHECKBD uses WHITEROW and BLACKROW to display the
( checkerboard pattern)
: CHECKBD
04 00 DO
 08 00 DO
  WHITEROW
 LOOP
 08 00 DO
  BLACKROW
 LOOP
LOOP
;
( DRAW CHECKERBOARD PATTERN )
CHECKBD
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Appendix B - Forth Demonstration Files - continued

Tiled Pattern Demonstration

```
( Tiles the LCD display with an 8 bit x 8 bit pattern.
(  LOADWHEEL, LOADDIAMOND, LOADTHATCH, LOADWEAVE, and
(   LOADK fill the array with a preset pattern  )

( all values following accepted as hexadecimal )
HEX

( register address definitions for HD61830B )
7FF1 CONSTANT INSTREG
7FF0 CONSTANT WRITREG
7FF1 CONSTANT BFLAG

( register number definitions for HD61830B )
0 CONSTANT MODEREG
1 CONSTANT CHARPITCH
2 CONSTANT NUMCHAR
3 CONSTANT NUMTIMES
4 CONSTANT CURPOS
8 CONSTANT DSTARTLO
9 CONSTANT DSTARTHI
A CONSTANT CURSLO
B CONSTANT CURSHI
C CONSTANT WRTDISP
D CONSTANT RDDISP
E CONSTANT CLRBIT
F CONSTANT SETBIT

( writes a byte value to a register )
: WRITVAL   ( REGVAL REGNUM — )
  INSTREG EC!   ( REGNUM — )
  WRITREG EC!   ( REGVAL — )
  BEGIN
   BFLAG EC@   ( — FLAG )
   80 AND      ( FLAG — BIT7 TEST)
   80 <>      ( BIT7 TEST — CONDITION )
  UNTIL   ( CONDITION — )
;
( Clears the screen by writing 00h to each display byte )
: CLEARSCREEN
  00 CURSLO WRITVAL
  00 CURSHI WRITVAL
  A00 0 DO
   00 WRTDISP WRITVAL
  LOOP
;
```

(continued on next page)

Appendix B - Forth Demonstration Files - continued

Tiled Pattern Demonstration

```
( Moves cursor position to upper left corner )
: CURINIT
00 CURSLO WRITVAL
00 CURSHI WRITVAL
;

( Initialization for character write mode )
12 MODEREG WRITVAL     ( 1C TO mode register)
07 CHARPITCH WRITVAL ( 95 to char pitch register)
1D NUMCHAR WRITVAL     ( 27 TO number of chars reg)
1F NUMTIMES WRITVAL    ( 1F to number of times reg)
00 DSTARTLO WRITVAL
00 DSTARTHI WRITVAL

CLEARSCREEN

( Return cursor to start position )
CURINIT

32 MODEREG WRITVAL    ( TURN ON DISPLAY )

( Array definitions for tiling routine )
0 VARIABLE BYTEARRAY 8 ALLOT


0 VARIABLE BYTESTORE 2 ALLOT


( Tiles one row with the 8 bytes in BYTEARRAY )
: TILEROW
 08 0 DO
  1E 0 DO
   BYTEARRAY J + C@
   WRTDISP WRITVAL
  LOOP
 LOOP
;
```

(continued on next page)

Appendix B - Forth Demonstration Files - continued

Tiled Pattern Demonstration

```
( Tiles display using TILEROW )
: TILE
  10 0 DO
    TILEROW
  LOOP
;

: LOADWHEEL
  14 BYTEARRAY 0 + C!
  0C BYTEARRAY 1 + C!
  C8 BYTEARRAY 2 + C!
  79 BYTEARRAY 3 + C!
  9E BYTEARRAY 4 + C!
  13 BYTEARRAY 5 + C!
  30 BYTEARRAY 6 + C!
  28 BYTEARRAY 7 + C!
  CURINIT
  TILE
;

: LOADDIAMOND
  20 BYTEARRAY 0 + C!
  50 BYTEARRAY 1 + C!
  88 BYTEARRAY 2 + C!
  50 BYTEARRAY 3 + C!
  20 BYTEARRAY 4 + C!
  00 BYTEARRAY 5 + C!
  00 BYTEARRAY 6 + C!
  00 BYTEARRAY 7 + C!
  CURINIT
  TILE
;

: LOADTHATCH
  88 BYTEARRAY 0 + C!
  54 BYTEARRAY 1 + C!
  22 BYTEARRAY 2 + C!
  45 BYTEARRAY 3 + C!
  88 BYTEARRAY 4 + C!
  15 BYTEARRAY 5 + C!
  22 BYTEARRAY 6 + C!
  51 BYTEARRAY 7 + C!
  CURINIT
  TILE
;

: LOADWEAVE
  F8 BYTEARRAY 0 + C!
  74 BYTEARRAY 1 + C!
  22 BYTEARRAY 2 + C!
  47 BYTEARRAY 3 + C!
  8F BYTEARRAY 4 + C!
  17 BYTEARRAY 5 + C!
  22 BYTEARRAY 6 + C!
  71 BYTEARRAY 7 + C!
  CURINIT
  TILE
;

: LOADK
  11 BYTEARRAY 0 + C!
  09 BYTEARRAY 1 + C!
  05 BYTEARRAY 2 + C!
  03 BYTEARRAY 3 + C!
  05 BYTEARRAY 4 + C!
  09 BYTEARRAY 5 + C!
  11 BYTEARRAY 6 + C!
  00 BYTEARRAY 7 + C!
  CURINIT
  TILE
;
( type LOADWHEEL, LOADWEAVE, LOADTHATCH,
LOADDIAMOND, LOADK
(  to see these patterns drawn )
```

SECTION 7

Appendix B - Forth Demonstration Files - continued

Interactive Tiling Demonstration

```
( Tiles the LCD display with an 8 bit x 8 bit pattern.
(  ASK requests 8 bytes for the pattern and stores
(   these into an array
(  TILE tiles the display with these patterns

( all values following accepted as hexadecimal )
HEX

( register address definitions for HD61830B )
7FF1 CONSTANT INSTREG
7FF0 CONSTANT WRITREG
7FF1 CONSTANT BFLAG

( register number definitions for HD61830B )
0 CONSTANT MODEREG
1 CONSTANT CHARPITCH
2 CONSTANT NUMCHAR
3 CONSTANT NUMTIMES
4 CONSTANT CURPOS
8 CONSTANT DSTARTLO
9 CONSTANT DSTARTHI
A CONSTANT CURSLO
B CONSTANT CURSHI
C CONSTANT WRTDISP
D CONSTANT RDDISP
E CONSTANT CLRBIT
F CONSTANT SETBIT

( writes a byte value to a register )
: WRITVAL  ( REGVAL REGNUM - )
  INSTREG EC!  ( REGNUM - )
  WRITREG EC!  ( REGVAL - )
  BEGIN
   BFLAG EC@  ( - FLAG )
   80 AND     ( FLAG - BIT7 TEST)
   80 <>      ( BIT7 TEST - CONDITION )
  UNTIL  ( CONDITION - )
;

( Clears the screen by writing 00h to each display byte )
: CLEARSCREEN
  00 CURSLO WRITVAL
  00 CURSHI WRITVAL
  A00 0 DO
   00 WRTDISP WRITVAL
  LOOP
;
```

(continued on next page)

Appendix B - Forth Demonstration Files - continued

Interactive Tiling Demonstration

```
( Moves cursor position to upper left corner )
: CURINIT
00 CURSLO WRITVAL
00 CURSHI WRITVAL
;

( Initialization for character write mode )
12 MODEREG WRITVAL    ( 1C TO mode register)
07 CHARPITCH WRITVAL ( 95 to char pitch register)
1D NUMCHAR WRITVAL    ( 27 TO number of chars reg)
1F NUMTIMES WRITVAL   ( 1F to number of times reg)
00 DSTARTLO WRITVAL
00 DSTARTHI WRITVAL

CLEARSCREEN



( Return cursor to start position )
CURINIT

32 MODEREG WRITVAL    ( TURN ON DISPLAY )

( Array definitions for tiling routine )
0 VARIABLE BYTEARRAY 8 ALLOT


0 VARIABLE BYTESTORE 2 ALLOT

( Gets a byte of data from keyboard and converts to hex )
: GETBYTE
 KEY          ( -- Key_value1 )
 DUP          ( -- Key_value1 )
 BYTESTORE 0 + C!        ( place value1 in array[0] )
 EMIT         ( Key_value1 -- )
 KEY          ( -- Key_value2 )
 DUP          ( -- Key_value2 )
 BYTESTORE 1 + C!        ( place value2 in array[1] )
 EMIT         ( Key_value2 -- )
;

( Converts two ASCII bytes into a hex byte )
: CONVERTER
BYTESTORE 0 + C@   ( -- Key_value1 )
30 -               ( Key_value1 -- hex1 )
DUP                ( -- hex1)
9 >                ( hex1 -- condition)
```

(continued on next page)

SECTION 7

Appendix B - Forth Demonstration Files - continued

Interactive Tiling Demonstration

```
IF 7 -        ( condition hex -- newhex)
ELSE
THEN
10 *          ( Hex -- HexMSN): generate Most Significant Nibble in HEX
BYTESTORE 1 + C@    (-- Value2)
30 -          ( Value2 -- hexLSN )
DUP           ( -- hexLSN )
9 >           ( hexLSN -- condition )
IF 7 -        ( condition -- hexLSN )
ELSE
THEN
+             ( hexLSN hexMSN -- hex_equivalent )
;

( Prompts user for 8 hex bytes to be tiled)
: ASK
 CR
 8 0 DO
 ." ENTER BYTE "
 I .
 GETBYTE
 CR
 CONVERTER
 BYTEARRAY I + C!
 LOOP
;

( Tiles one row with the 8 bytes in BYTEARRAY )
: TILEROW
 08 0 DO
  1E 0 DO
   BYTEARRAY J + C@
   WRTDISP WRITVAL
  LOOP
 LOOP
;

( Tiles display using TILEROW )
: TILE
  10 0 DO
    TILEROW
  LOOP
;

( type ASK, enter bytes, then type CURINIT to initialize
(   cursor, then TILE to display pattern

(   this sequence can be performed repeatedly  )
```

Hitachi Emulators

Applications Guide

Emulator to PC Interface Guide

Marnie Mar

Introduction

Designers using Hitachi microcontrollers can simplify system implementation by using a PC for all development steps from code generation and hardware testing to system integration and debugging.

Using a PC and cross software tools available from Hitachi and third party vendors, users can compile, assemble, and link code destined to run on their target processor. The object file resulting from this process is ready to be tested for proper operation using software simulators or hardware emulators.

When using a software simulator that also runs on the PC, this object file is simply read by the simulator program, and debugging begins. However, when the debugging environment is a separate piece of hardware for emulation, the transfer of object information is less trivial.

This guide describes how to interface Hitachi emulators to a PC, so that object code can be transferred and debugging sessions can be carried out using the PC's keyboard and display as the user interface. Four types of emulators will be discussed, giving details on interface cable specifications, software required and where to obtain it, emulator features, and tips on using the emulator.

The following Hitachi emulators will be covered:

- H Series Adaptive System Evaluator (ASE)
- 64180 Family Adaptive System Evaluator (ASE)
- 63xx Family Emulators
- 400 Series Device Emulators

This guide assumes the user has access to the following:

- A PC with unused COM1: or COM2: serial port. If the PC is not an IBM PC/XT or AT, compare the serial port pin-out of your machine with those shown in diagrams in this document to ensure that the recommended cables will provide the proper connections.
- Cross software for the processor to be emulated which will generate an object format acceptable to the emulator (acceptable formats are shown for each emulator type).
- Emulator and User's Manual for the processor to be emulated.
- An RS-232C breakout box or equipment to build the recommended cables as shown in this guide (in some cases, the cables shipped with the emulators will not operate correctly when connected to the PC).
- Hitachi America, Ltd.'s Application Engineering Bulletin Board System for downloading files. For more information contact your local Field Application Engineer.
- A terminal emulation and file transfer program, such as Procomm, Crosstalk, PC-Talk, or many others running on the PC (required for some 64180 interface configurations).

The following abbreviations are used throughout this guide:

ASE - Adaptive System Evaluator: a hardware development tool which emulates device operation
HAL - Hitachi America, Ltd.
MRI - Microtec Research Inc.
HINT - H-Series Interface Software
BBS - HAL Application Engineering Bulletin Board System

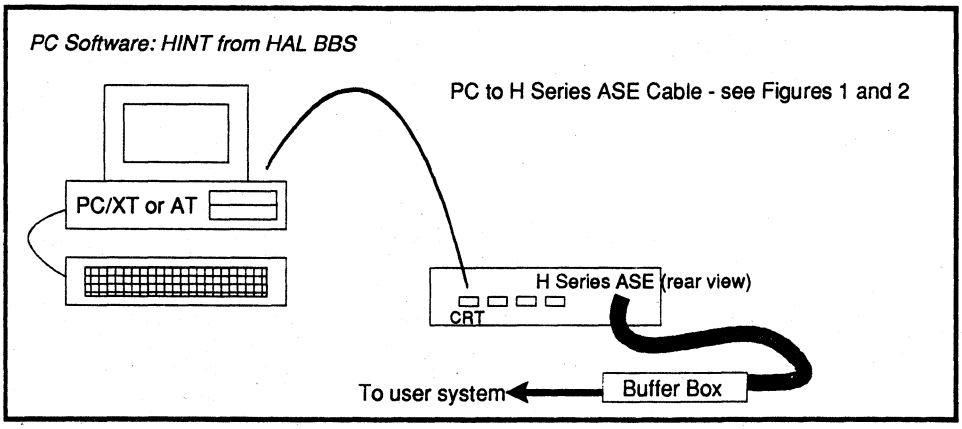
For more information on Hitachi products, please contact your local Field Sales Office.

For information on Microtec Research, Inc. tools, call 1-800-950-5554.

H Series ASE Interface

Hitachi provides emulation capability for users of H Series devices in the form of the ASE, or Adaptive System Evaluator. An ASE consists of a main station common to all H Series devices, a buffer box specific to the device being emulated, and a target probe determined by the package type being designed in.

System Configuration:



ASE Part Number:

HS640AST01H

Interface Software:

HINT from HAL BBS. Download file named HINT22A.EXE from area O: Special Function Programs. This is a self-extracting file that will unarchive itself when executed.

Interface Cables:

See Figure 1 or 2, depending on whether PC/XT or AT computer is used. The HINT program assumes communications through COM1: of the PC.

ASE Communications:

Execute the H Series Interface program HINT22. EXE. The user communicates with the ASE debug monitor through the CRT port using XON/XOFF flow control supported by both the HINT program and the ASE operating system. The ASE ignores any input on the CTS pin (pin 4). The RTS output is always high, so this signal can be input to the PC CTS signal to ensure that the PC always detects a Clear To Send condition.

HINT assumes a communications speed of 9600 bits per second, with 8 bit data, 1 stop bit, no parity. The ASE CRT port is configured for this speed at shipping. Refer to the User's Manual for more information on configuring this port.

Object Code File Transfers:

Object code information can be uploaded and downloaded through either the ASE HOST or CRT port. In order to use the CRT port (and remove the need for an additional terminal or PC), the ASE must be booted in the proper mode. To do this, turn the ASE on *without* the floppy disk latched into the drive. This causes the ASE to prompt for an operation. Select the I command, to use an ASE interface.

If the ASE is started with the floppy disk in the drive, the ASE system will automatically load from the floppy, and the ASE will assume that uploads and downloads will take place through the HOST port. This would result in a configuration similar to the two-display configuration shown for the 64180 ASE, and is not receommended.

The ASE performs object file transfers using software handshaking. This additional handshaking is provided by the HINT program, and is described in the *ASE User's Manual*. The handshaking method is specific to these emulators, and is not supported by common terminal emulator/file transfer software packages.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

H Series ASE Interface (continued)

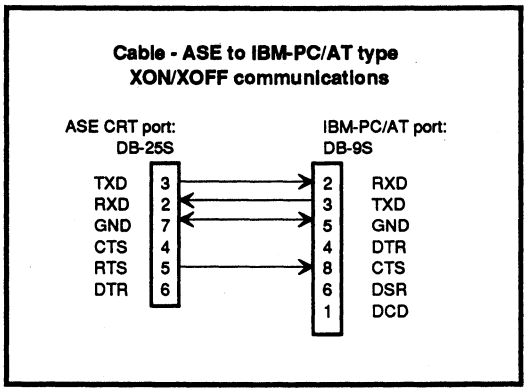


Figure 1

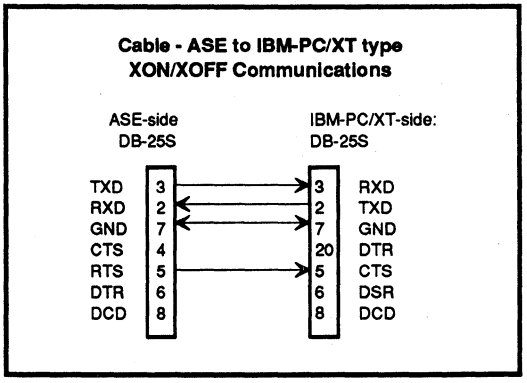


Figure 2

Object File Formats:

S records
Intel Hex records
ASCII symbol and S-record files generated by HAL cross-software tools

Symbol Capability:

The ASE provides the capability of referring to addresses by associated symbol name. Symbol names are assigned to addresses at link time, and are based on the symbol names defined at assembly time. These are user-defined in the assembler source, or compiler-defined when the compiler generates the assembly listing.

For HAL cross software tools, symbolic information is loaded to the ASE from a symbol and S-record output file using a special feature of the HINT interface program.

Notes on use of the H Series ASE:

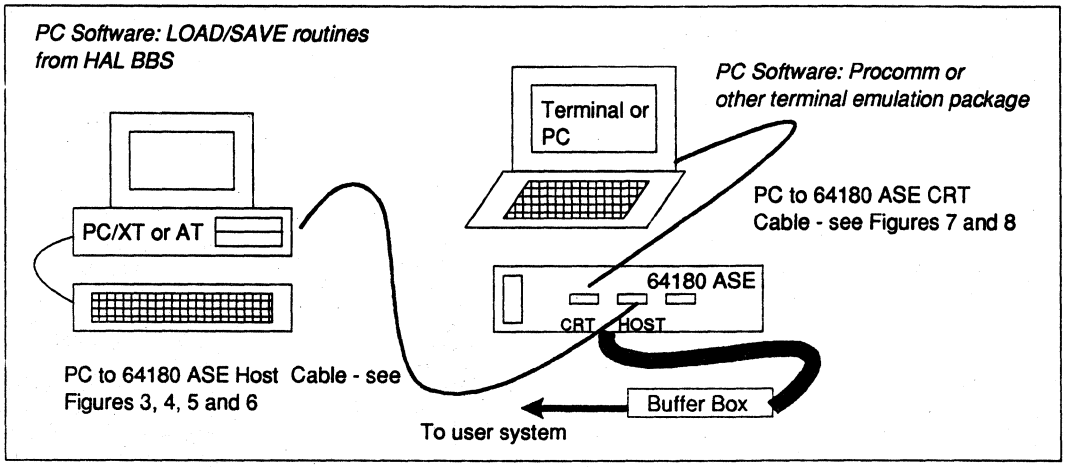
A buffer box and end user cable defined by the device begin emulated must be purchased separately. Some devices are supported by buffer boxes which allow additional memory to be added to expand the user memory space for larger code size applications. These add-on memory boards fit into the buffer box.

The ASE must be used with a buffer box attached. The ASE main station operating system software is contained on a disk that comes with each buffer box, and is specific to that buffer box. No software disks are shipped with the ASE alone.

64180 Family ASE Interface

Emulation of the 64180 family of devices is performed using an ASE, or Adaptive System Evaluator. An ASE consists of a main station common to all 64180 devices, a buffer box specific to the device being emulated, and a target probe determined by the package type being designed in.

Dual Display System Configuration:



ASE Part Number:

HS180AST01H (also H180AS01)

Interface Software

LOAD and SAVE available by downloading file ASECOMM.EXE from Area O: Special Function Programs of the BBS
Terminal Emulator/File Transfer software package

- Dual Display System:
LOAD and SAVE programs to run on PC connected to HOST port of ASE (COM1: port of PC must be used) -OR-
Terminal emulator and File Transfer software to run on PC connected to HOST port of ASE

If a PC is connected to the CRT port, terminal emulation software must be used to communicate with the ASE debug monitor through the CRT port.

In the dual display configuration, one display (connected to HOST port) must be associated with a PC and is used to initiate file transfers for upload and download. The other display can be a terminal or PC running terminal emulation software and is used to communicate with the ASE debug monitor through the CRT port.

- Single Display System:
PROCOMM/other terminal emulator to run on a PC connected to both HOST and CRT port of ASE using Y connector.

Interface Cables:

Select the cable configuration from Figures 3 through 9, depending on the type of system you plan to use. Configure the cables required for your system as shown. The cables to use depend on the type of PC being interfaced, which software package is used, and whether XON/XOFF or RTS/CTS data flow control is used.

Note that the Y connector requires -12V level for proper operation. This can be provided by external power supply, or from the user system. See Figure 9 for this cable configuration.

Interface Communications - CRT Port:

The 64180 ASE CRT port requires that the CTS and DTR inputs to the ASE be active before transmission of data can

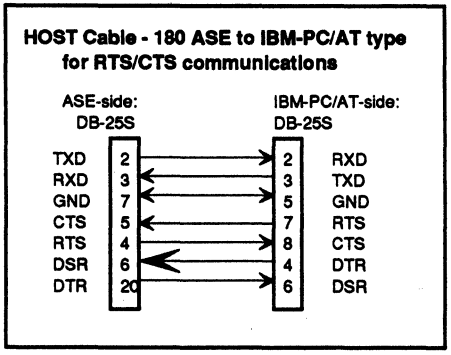


Figure 1

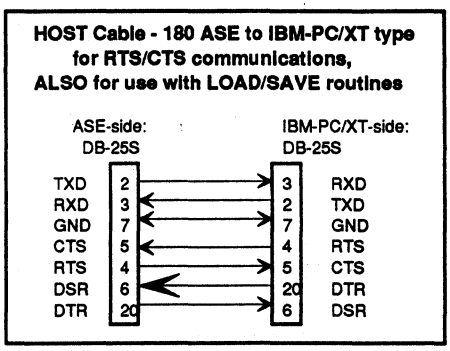


Figure 4

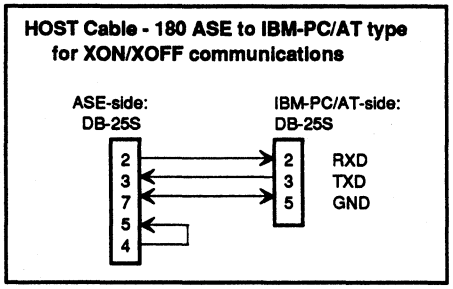


Figure 5

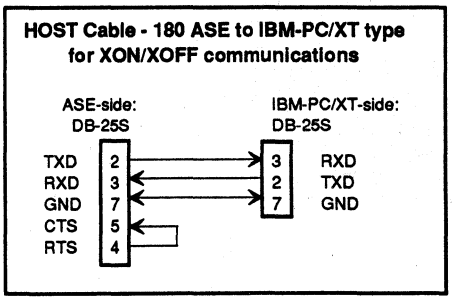


Figure 6

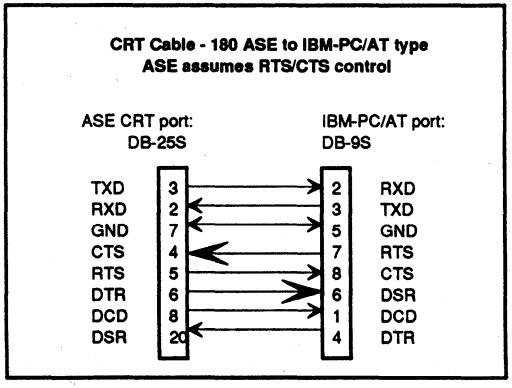


Figure 7

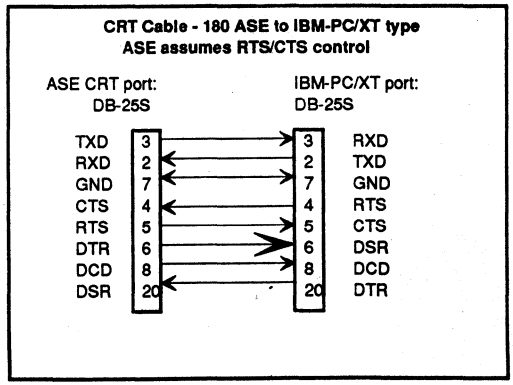
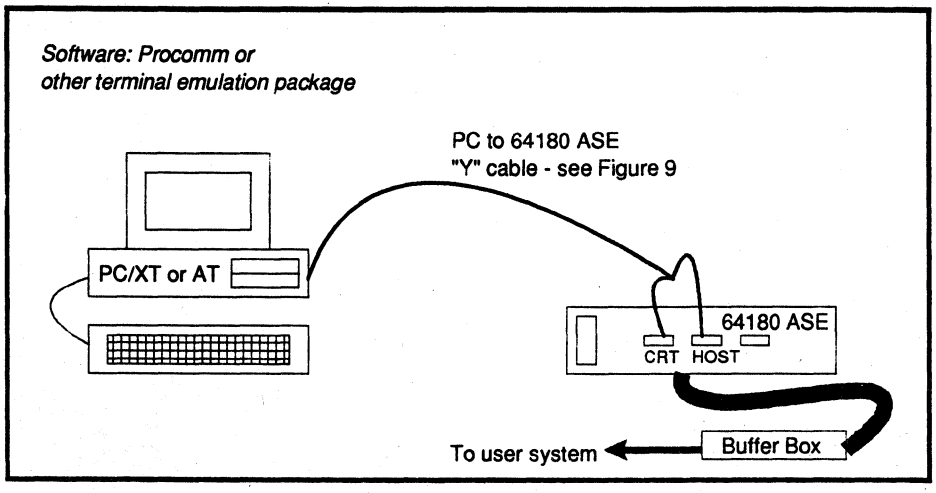


Figure 8

64180 Family ASE Interface (continued)

Single Display System Configuration:



occur. The CTS signal controls the flow of data transmitted from the ASE to the CRT. The RTS line output from the ASE CRT port is always high, since flow control from the CRT device is not critical (keyboard input).

Operating speed of the CRT port can be selected by setting the switches of the ASE control board as shown in the *ASE User's Manual*. These switches are set to an invalid code at shipping. The data format is 8 data bits, 1 start bit, 1 stop bit, no parity. After the speed has been selected on the ASE, set up the terminal or terminal emulator software accordingly.

Interface Communications - HOST port:

The 64180 HOST port can be software configured for RTS/CTS (hardware) or XON/XOFF (software) data flow control. Thi selection is made by executing the ASE's *HOST* command. The LOAD/SAVE programs assume RTS/CTS control is used. When PROCOMM or another terminal emulation package is used, either XON/XOFF or RTS/CTS control can be selected, with the software package and the ASE configured accordingly.

Operating speed of the HOST port is defined to be 9600 bits per second if the LOAD/SAVE programs are used, but can be user selected if a terminal emulation package is used for file transfer to the HOST port. The *HOST* command of the ASE is used to specify the parameters of HOST port commmunications, including baud rate and XON/XOFF or RTS/CTS handshaking.

Object File Transfers:

In the Dual Display configuration, either the LOAD/SAVE routines from the BBS or a terminal emulation/file transfer program can be used to transfer files to and from the ASE's HOST port. If the LOAD/SAVE routines are used, see the User's Guide information archieved with these files on the BBS.

If terminal emulation software is used, the procedure is similar to that for the Single Display configuration. The *LOAD, VERIFY* or *SAVE* command should be issued to the ASE. After this is done, execute the steps necessary to cause an ASCII file transfer as required by the software package you are using. Once the ASE has received a *LOAD* or *VERIFY* command, it will wait for a file to be received through the HOST port. If the ASE receives a *SAVE* command, it will begin sending data to the HOST port.

Object File Formats:

S records
Hitachi S6 symbol records can be included in S record files
Intel Hex records

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

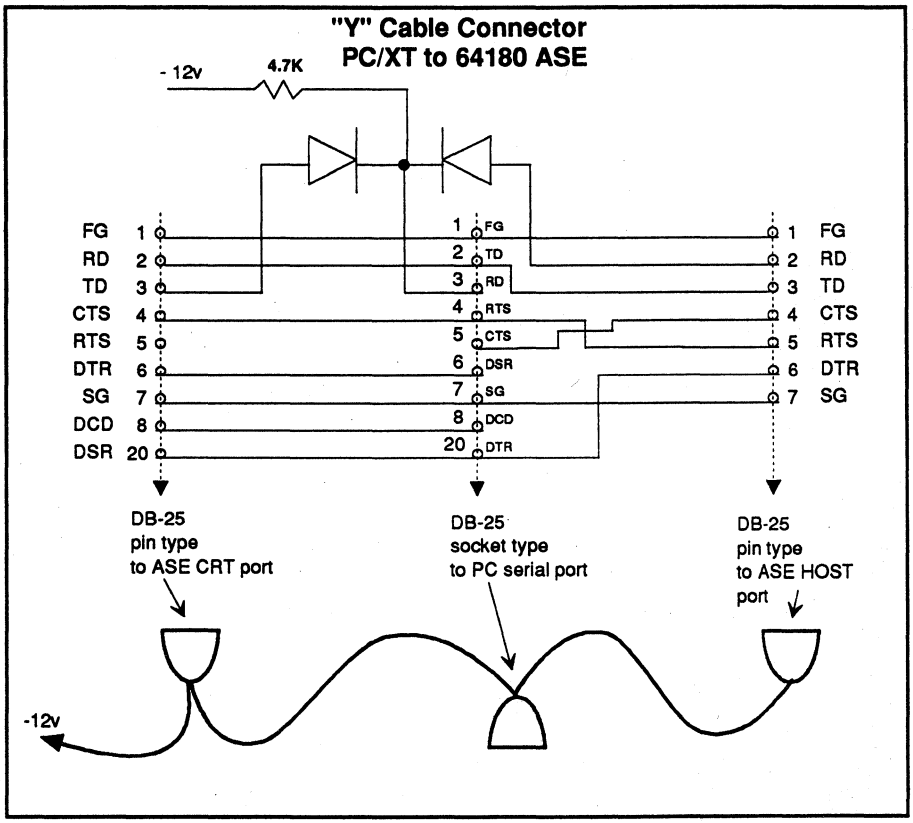


Figure 9

Symbol Capability:

Symbol names can be assigned to physical address values. MRI Cross Software Tools generate symbol information records (S6 records) which can be downloaded to enter symbol information into memory.

Notes on Emulator Use:

A buffer box and end user cable defined by the device being emulated must be purchased separately from the ASE. The ASE comes with 8Kbyte of emulation RAM for user development. This RAM space can be increased by purchasing additional (up to two) 256Kbyte memory boards, which are placed in the ASE main station.

Hitachi sells an ASE package which consists of the ASE station (HS180AST01H) and a buffer box supporting the 64180R device (HS180ABX02H) as part number HS180ASE02H. The ASE and this buffer box can also be purchased separately. Buffer boxes supporting all other 64180 devices are sold separately. For devices with package type options, end-user target cables are sold separately to emulate other package types. .

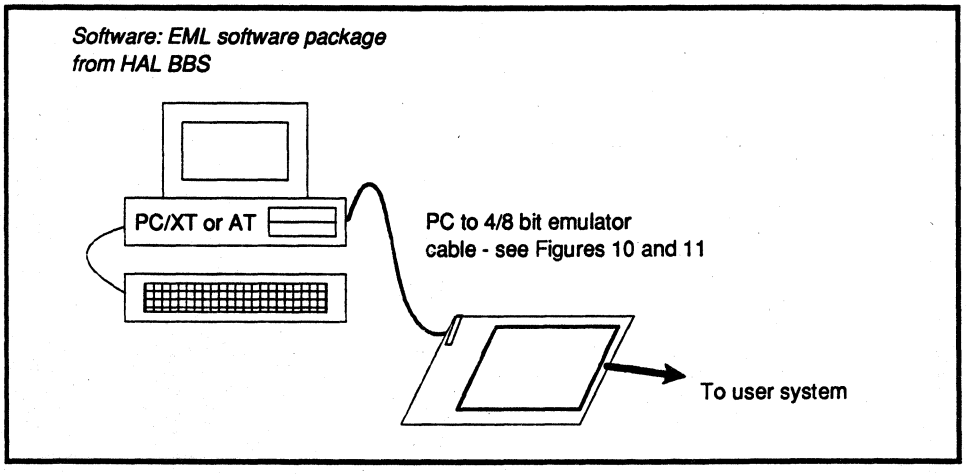
The ASE cannot be used without a buffer box, since at power up, a check for a buffer box is made. If none is found, operation will not continue. ASE system software on floppy disk is included with each buffer box. No system software is shipped with the ASE alone.

Older versions of the ASE are marked H180AS01. These units will work with all 64180 family buffer boxes, and have the same functionality as the H180AST01H.

63xx Family Emulator Interface

Hitachi manufactures and sells emulators which support a specific device or set of devices in the HD63xx micro families. These emulators have similar capabilities, and each has a serial interface which allows connection to a PC for download and upload of code, and for communication with the emulator monitor for debugging sessions.

System Configuration:



Emulator part numbers:

Includes H31MIX2,3,4, H35MIX3,5

Interface Software:

EML interface software, available on the HAL BBS. Download file EML.EXE from area O: Special Function Programs. This file is a self-extracting archive file which will unarchive itself when executed.

Interface Cable:

See Figures 11 and 12. Configure a cable for your system as shown to connect the emulator serial port to either COM1: or COM2: serial port of the PC.

Interface Communications:

The interface program comes in two versions which allow the user to communicate through either COM1: or COM2: serial port of the PC. Execute either EML1.EXE or EML2.EXE to communicate through COM1: and COM2:, respectively.

The EML programs operate at 9600 BPS,8 data bits, 1 start and 1 stop bit, so for proper operation, the emulator must be configured for operation at this speed. Set SW3 of the emulator as shown in Figure 10 to allow communications between the emulator and a PC running EML.

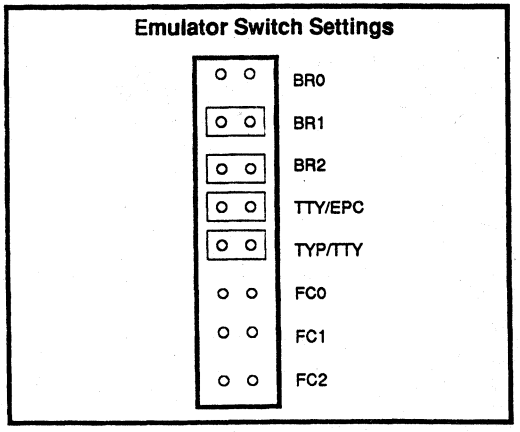


Figure 10

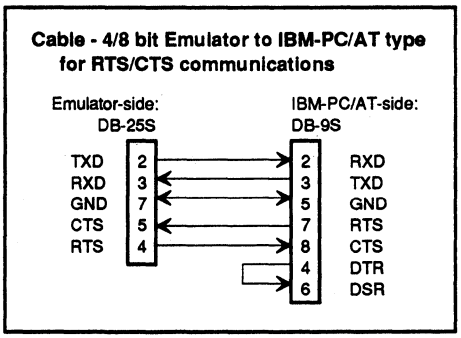


Figure 11

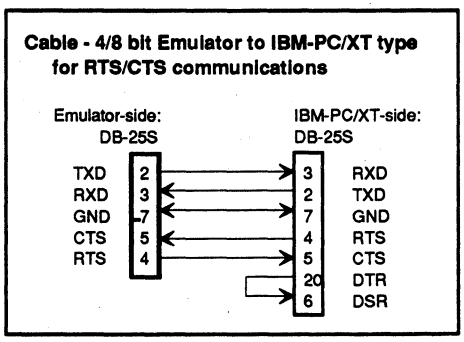


Figure 12

Object Code File Transfers:

File transfers require a special software handshaking protocol unique to these Hitachi emulators, which is supported by EML. While the various terminal emulation packages available on the market can be used to communicate with the emulator debug monitor, these packages do not support this upload/download handshaking protocol.

The following commands are used to transfer object files (<CR> represents pressing the Carriage Return or Enter key on the PC):

Loading object file:
L <filename> <CR>
Verifying object file:
V <filename> <CR>
Saving (punching) new object file:
P <filename> <CR>

The E (End) command followed by <CR> at the prompt will terminate the interface program and return control to DOS.

Object file format for download:

Motorola S records (S0, S1, S9)
Intel Hex records

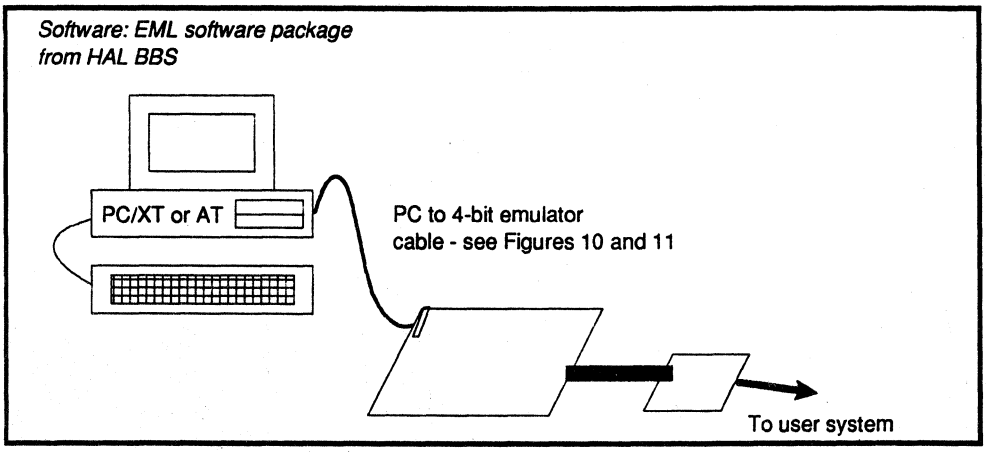
Notes on EML Use:

A User's Manual for the EML program is not included in the BBS distribution file, however, the important information from this manual (hardware interface, upload/download commands) is listed here.

4-bit Microcontroller Emulators

System Configuration:

Hitachi emulators for devices in the 400-Series are similar to the 63xx emulators in that they use the same serial interface for connection to a PC. The EML programs metioned in the 63xx section is also used for communication with these emulators.



Emulator part numbers:

HS400EUA01H /HS400EUA02H emulator station
H400CMIX2 emulator

Interface Software:

EML interface software available on the HAL BBS. Download file EML.EXE from Area O: Special Function Programs. This is an archive file which will unarchive automatically when it is executed.

Interface Details and Object File Transfers:

See information on EML program use in the 63xx Family Emulators section. For the HS400EUA01H, configure the Emulator Operation Selection Switch settings as shown at right. For the H400CMIX2, configure as shown in Figure 10.

Object File Format for Download:

S record files
Hitachi S6 record files (symbol information only) for the HS400EUA01H/02H
Intel Hex record files

Symbolic Capabilities:

Symbol names can be assigned to address locations manually using the monitor SYM command. Symbol files can also be downloaded to the emulator using the *LOAD* command.

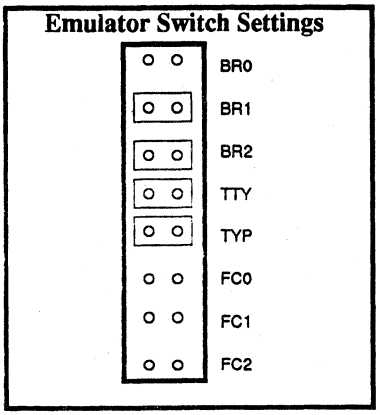


Figure 13

Notes on the 400 Series Emulators:

The HS400EUA01H/02H connects to a series of target probes that each allow emulation of a different device in this family. Each target probe connects to a User Cable which connects the target probe to the user system These pieces must be purchased separately by part number, make sure to get the correct target probe and user cable as required for your application. The HS400EUA01H cannot operate without a target probe attached.

H Series Support Tools

July, 1990

Application Note

Symbolic Debugging with the H Series ASE and MRI Tools

Marnie Mar

Objective

Designers using the Hitachi Adaptive System Evaluator (ASE) emulator and Microtec Research, Inc. (MRI) software development tools can use symbols associated with assembly-level source code to aid in debugging. This note discusses how to use the MRI tools to generate an object code file containing symbol information, how to modify this file so that the information is acceptable to the ASE, and how to load the modified file information to the ASE from a floppy disk.

Procedure overview

The MRI linker will generate an S-type linked object file that contains symbol information. This file can be edited using a word processor into two separate files, one containing only S-record information and the other containing symbol information in a form acceptable to the ASE. Once these two files are available, standard ASE commands can be used to read these files from the disk.

file cc.bat - uses an option file to specify compiler directives. Resulting S record object file containing symbols will be placed by default in file **sieve.abs**.

```
mcch83 -dcoption.cmp sieve.c
```

file coption.cmp - compiler options list, includes specification of options to the assembler, and a command file containing linker options

```
-Fsm
-Vi
-l>test.lst
-Wa,-fde,-l>sieve.lst
-esform.cmd
```

file sform.cmd - linker option comand file

```
format s
listmap publics,internals
listabs publics,internals
debug_symbols
extern mri_start
load ch83isc.lib
load ch83isf.lib
end
```

Figure 1 - Compiling C code to S-type object

file sieveasm.bat - batch file to assemble and link an application written in assembler. Linker commands are read from a command file. Resulting linked object and symbol information is placed in **sieve.abs**.

```
asmh83 -fde -l>sieve.lst sieve.src
lnkh83 -csform.cmd -osieve.abs sieve
```

file sform.cmd - linker command file to generate S-type object file.

```
format s
listmap publics,internals
listabs publics,internals
debug_symbols
extern mri_start
load ch83isc.lib
load ch83isf.lib
end
```

Figure 2 - Assembling source code to S-type object

Generating the S-type linked object file

Figure 1 and Figure 2 show examples of generating S-record object files with symbol information, for C language sources and assembler sources, respectively. These batch and command files assume that the proper "path" statements have been set up so that the current directory has access to the Compiler, Assembler, and Linker executable files, as well as the source and object involved. These figures show examples only, and the flexibility of the MRI tools allows users to arrive at similar results using various combinations of command files, batch files, command line options, and assembler source file directives.

A useful option to the compiler is the "-Fsm" option, which causes the C source code lines to be intermixed as comments into the resulting assembler source file. The resulting assembler listing file will assist in locating code when debugging using the ASE.

Assembler options are specified to the compiler driver using the "-Wa..." compiler option. These options can also be included in the assembler command line. Options to note include the "-fde" option, which is required to cause symbol information to be placed in the object file generated by the assembler, and the "-l > *filename*" option which causes a listing to be generated and redirected to the file *filename*.

Linker commands can be entered either on the command line, or in a command file as shown in the examples. When linking, the "debug_symbols" entry in the command file is required to cause the internal symbols to be output to the resulting S-type linked object file. These internal symbols correspond to local labels used in assembler source files.

The "listmap publics,internals" command in conjunction with the "debug_symbols" command ensures that both global and local symbols are output to the object file, making them available for use in debugging.

Editing the MRI S-record linked object file

Excerpts from an linked object file generated using the command sequences discussed above are shown in Figure 3. In order to use this file information with the ASE, a word processor must be used to divide this symbol and S-record information into two separate files. One file contains only the S-record information, starting with a record beginning with the characters "SO". The S record information is acceptable by the ASE without modification.

The other file, which contains the symbol information, must be edited to allow it to be read by the ASE. The ASE uses two commands that will be used to load this symbol information into ASE memory. The Command_Chain, or cc command is used to read and execute valid ASE commands from a file in the floppy drive. The Symbol, or Sy command allows the user to define symbols by inputting the symbol name followed by the symbol address, in the form:

: sy !newsymbol=h'3000<cr>

where newsymbol is the name of the symbol to be defined. The ":" is the prompt output by the ASE. All symbols must be preceeded by an exclamation point (!) when they are defined, and the h' prefix indicates to the ASE that the number following refers to a hexadecimal value. The lines starting with "$$" in the MRI generated file are informational only, and should be delected.

```
$$ sieve
  __environ $11AC __com_line $11B0 _flags $13B0
__iob $13C4
$$ sieve
  _main $0080 L7001 $0128 L5 $009E L14 $00DA
  S0 $004C L13 $00CC L1 $0096 L11 $0108
  L9 $00B2 S1 $0040
$$ fakftoa
  __fltused $115C __ftoa $0146 __dldd $016E
$$ flsbuf
  __flsbuf $0170
$$ imul
  __aimul $0298 __imul $0280
$$ land
  __axor $02E8 __not $0302 __lognot $031E __aand
$02B4
  __neg $030C __aor $02CE
$$

  .
  .
  .

S00600004844521B
S11400400A2564207072696D65730A00207072696D86
S10E005165202564203D2025640A0082
S11400806DF60D767900000619076DF26DF3790000AE
  .
  .
  .
S10411680082
S114112A6DF60D766DF26DF36F6300080D3240146F2F
S114113B6000060D010B016FE1000668086A881168EE
S113114C1B020D2246E80D306D736D726D76547072
S9031036B6
```

Figure 3 - Excerpts from MRI's linked S-record file

The editted version of the symbol information is shown in Figure 4. Each symbol definition must be placed on a separate line of the file, and the "!" and the "=" sign must be added to each line. The "$" that is generated by the MRI tools must be replaced by the "h'" address prefix recognized by the ASE. Global replace features of word processors can

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

```
sy !environ = h'11AC
sy !com_line = h'11B0
sy !_flags = h'13B0
sy !iob = h'13C4

sy !_main = h'0080
sy !L7001 = h'0128
sy !L5 = h'009E
sy !L14 = h'00DA

sy !S0 = h'004C
sy !L13 = h'00CC
sy !L1 = h'0096
sy !L11 = h'0108
sy !L9 = h'00B2
sy !S1 = h'0040
sy !fltused = h'115C
sy !ftoa = h'0146
sy !dldd = h'016E
sy !flsbuf = h'0170
sy !aimul = h'0298
sy !imul = h'0280
sy !axor = h'02E8
sy !not = h'0302
sy !lognot = h'031E
sy !aand = h'02B4
sy !neg = h'030C
sy !aor = h'02CE
   .
   .
   .
```

Figure 4 - Edited symbol information

be exploited to aid in converting the MRI symbol information into a file readable by the ASE.

The names of the symbols in this file are used only to assist the developer with debugging, so it is possible to modify these names to simplify their use. For instance, leading underscores ("_") generated by the MRI tools to indicate global symbols can be eliminated, long symbol names can be shortened, and names can be made more descriptive.

Loading object code to the ASE

Once the S record file and the new symbol file have been generated, they should be copied to a disk formatted by the ASE using the 1.2MByte drive on a PC. To load the S-record file to the ASE, use the Floppy_Load or FL command. This command offers the option of specifying an offset to the load addresses contained in the object module file, however, if this offset is used, the symbol table you generated will not match up with the code downloaded. The object code and associated symbols can be assigned to a specific start address at link time by using the -B or BASE linker option. This command is executed as follows:

: FL *filename* <cr>

where *filename* is the name of the S-record file on the disk in the ASE floppy drive.

Loading symbol information to the ASE

The newly generated symbol file is loaded using the Command_Chain command:

: CC *filename* <cr>

where *filename* is the name of the modified symbol file placed on the disk in the ASE floppy drive.

Conclusion

It is possible to load both symbol and object code information into ASE memory using ASE commands. These commands rely on the availability of object code and symbol information files that can be generated using MRI tools and any word processor.

Loading code and symbol information in this matter allows the user the ability to perform symbolic debug of code, and eliminates the need for downloading using a serial link from the development computer.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

HD64180S Development Board

Update to #U17

User's Manual Update

Interfacing to the ASE Emulator

The NPU Development Board (part number US180EVB01H) can be used as an in-circuit target for the Hitachi ASE Emulator and NPU Buffer Box. In order to do this, however, a modification to the board must be made.

This modification adjusts the timing of the signals that access the DRAM module on the NPU Board. Although the access signals are changed, no additional wait states must be added to access this memory. This modification will not affect the operation of the board with an NPU device installed.

The modification consists of adding a 74HCT32 IC, cutting a trace, and adding jumper wires. The changes are shown in Figure 1.

Page 1 of the circuit diagram (on page 56 of the manual) is affected by these changes. New connections to U8 (DS1005N1500 delay line), shown on the bottom right of this page of the circuit diagram, are shown in Figure 2.

This modification results in delaying the leading edge of the CAS signal that is input to the DRAM module. The OR gate ensures that the trailing edge of the CAS meets requirements.

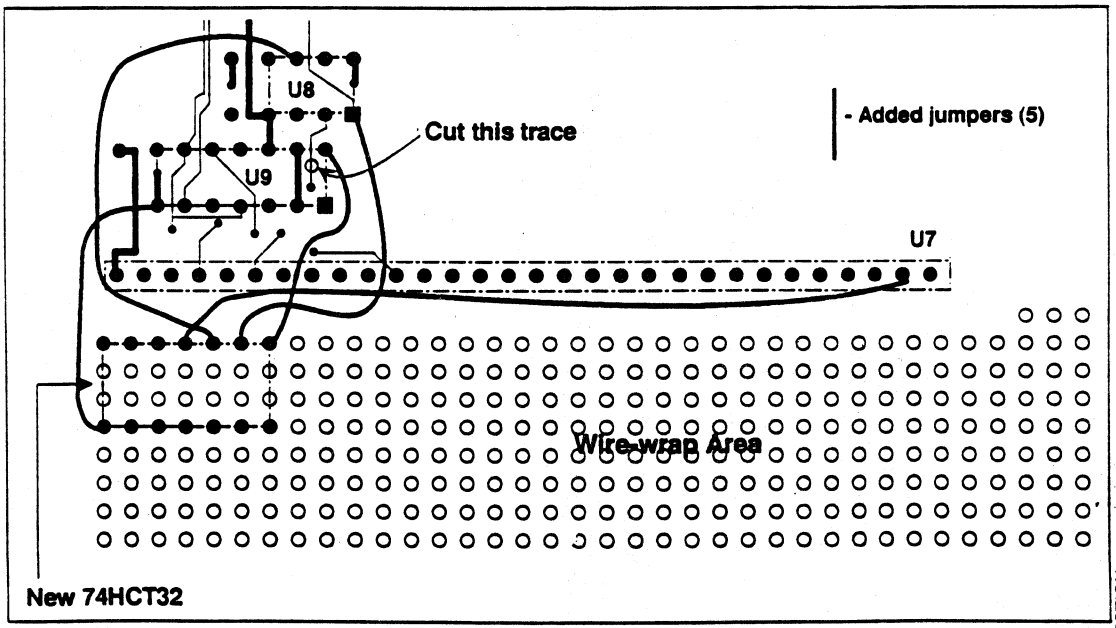


Figure 1 - Board modifications for interface to ASE (non-component side of board)

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

Linker-Generated Disassembled Code

Tech Notes

Application Engineering

Paul Yiu

Introduction

An assembler generates relocatable modules from assembly source codes; these relocatable modules are then put into designated addresses by the linker. Though the assembler can produce a listing, the listed addresses are relative. Using the "-m" command line option can produce a map file that lists the location of individual modules; however, it is often desirable to have access to disassembled code that has already been put into their correct addresses.

If the user only wants to look at the disassembled code after linker, he/she only needs to invoke XRAY, where the source code is listed. We can use the "macro" feature of XRAY to obtain a hardcopy of the disassembled code.

First, in XRAY, type:

fopen 60,*filename*.out

This command creates a window #60 and dumps its data in a text file named *filename*.out. The lower windows are reserved for default XRAY use. For example, we type in XRAY commands in window #10. Windows 50 to 256 are user-defined.

Before we write the macro, be sure your linked file is loaded. When invoking XRAY, type:

xhih83 filename

Note: Don't include the filename extension. Please refer to technote #TN-0021 for details on initial baud rate and display if you are using the emulator version XRAY.

Next we will define a macro in XRAY. This macro takes two arguments that signify the beginning and ending addresses of your listing; the macro then prints the disassembled code to window #60, which in turn dumps its data into the text file we defined. The macro is listed on the next page.

Linker-Generated Disassembled Code

```
: define void outcode(beg,end)
: int beg;
: int end;
: {
: int counter;
: for (counter=beg;counter<=end;counter=counter+2)
:       {
:       $fprintf 60,"%m\r\n",counter$;
:       }
: }
: .
```

Let's go through the macro definition line by line.

: define void outcode(beg,end)

This informs XRAY we are starting to define a macro named outcode,with two arguments,***beg*** and ***end***.

: int beg;

"Beg" is the starting address of our disassembled code.

: int end;

We can scroll thorugh the screen and find the end of our code.

: {

The macro body starts here.

: int counter;

"Counter" is a variable in the macro.

: for (counter=beg;counter<=end;counter=counter+2)

This starts a loop, very similar to a "for" loop in C.

: {

Start of loop.

: $fprintf 60,"%m\r\n",counter$;

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

```
            The "$'s" indicate that fprintf is an XRAY
            command.We need to put dollar signs around an
            XRAY commands.

:     }
            End of loop.

:}
            End of macro body.

:.
            The period tells XRAY your definition is done.
```

Basically, this macro takes two arguments, the beginning and ending addresses of the code we wish to print. Lines of code are printed to the designated window until the counter reaches the end of the block.

The next command line in XRAY prints assembly code in the first 100 hex address space.

> macro outcode(0,0x100)

This may seem a bit tedious, to define a whole macro just to take a look at the disassembled code, but it only has to be done once. The macro can be saved in an include file, which can be called up each time XRAY is invoked.

Starting up Emulator Version XRAY

Tech Notes

Application Engineering

Paul Yiu

Introduction

The Hitachi/Microtec XRAY for Emulators is designed to allow the ASE machine to communicate with an IBM-PC or compatible. However, XRAY's default baud rate is 19.2K while the default baud rate for ASE is 9200. As a result, the first-time user may get the error message "Problem communicating with the CPU." This is because XRAY and the ASE are communicating at different speeds. This problem is very simple to correct. Simply add "-e 9600" at the end of the command line when calling XRAY; this will set the XRAY to communicate at 9600 baud.

C:\XHIH83> ***xhih83 <filename> -e 9600*** enter

In XRAY

Once you are in XRAY, you can use the OPTION command to change some default settings.

option emulator="9600" ..... sets default baud rate

startup ..... saves option to startup.xry, which is called automatically each time XRAY is called.

Using a monochrome LCD display

Because monochrome LCD displays cannot fully take advantage of all of XRAY's colors and highlights, it may be difficult for the user to see highlighted material or error messages. Here is how to fix the problem.

Once the user is inside XRAY, there is an option color command that can change the display.

option color=none ..... changes color to white/blue

option highlight=inverse ..... when highlight=bright, it's not visible in the no color mode.

startup ..... saves options to startup.xry

Refer to the XRAY manual, Debugger Commands, for more options. Be sure to enter ***STARTUP*** to save the options for future use.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

ASMH83 Assembler/Linker

Tech Notes

Application Engineering

Carol Jacobson

Using the ASE emulator, system designers can download and execute software routines from either target memory or emulator on-chip ROM space. Prior to download, source text files containing address information, assembly instructions and labels must be converted to a hex code format which can be interpreted by the CPU and system. The three formats accepted by the ASE, SYSROF, Intellec HEX and S-record (Motorola) are detailed in the ASE 8/3xx Series Users Manual. Hitachi's ASMH83 Assembler translates H8/300 assembly files to S-record hex files containing lines of hex code with each line preceeded by the address assigned in the source file. Labels are converted to hex address locations and user comments are removed.

Creating Hex Files

Converting source to hex files requires three steps:.

1. Generating the assembly source file containing H8/300 code, assembler directives and labels
2. Assembling the source file to produce an object file (*.obj)
3. Linking the object file to produce an absolute hex file in S format (*.abs).

The Source File

The source text file can be created using any basic editor but must have the following format:

```
Assembler Directives      (See ASMH83 H8/300 Manual)
label:      H8/300 opcode operand,operand      ;comments
    .end
```

The Assembler

To start assembly, from a DOS environment, enter the directory containing the ASMH83.EXE file and type the command line:

ASMH83 -l >*.lis [source DOS path] (*.lis = name assigned to listing file)

Assembly usualy takes about 10 sec for 400-500 lines. At the end of assembly you will be notified of errors and warnings and two new files will have been created in the current directory: *.obj and *.lis. If you received error or warning messages, using an editor, review the listing file (*.lis) for error information, correct the errors in the source file and re-assemble. All errors must be removed to

produce a valid hex file.

CAUTION: If you have more than a few hundred lines of code and assembly seems to complete in 1 or 2 seconds without error, verify that a new .obj file was produced. If the .obj file was not produced, the assembler may not have located the source file or understood the command line as entered. Verify the source location and command format and re-enter the command.

The Linker

For most purposes the linker performs two critical tasks, it links several sections of code together to produce one program file and it outputs the program file in an executable form, in this case S-record. For each program a short command file containing at least the linker format and load instructions must be created. This is a batch file which always takes the extension .cmd. For example a command file, PRG.cmd, for linking file PRG.obj, may contain only the lines:

format s	(output= S records)
load pgm.obj	(load file pgm.obj)
load xxx.obj	(load any other files to be linked with prg.obj)
base 3000	(base offset = h'3000, ie code starts at h'3030)

To invoke the linker, from the directory containing the file LNKH83.EXE enter the command line:

LNKH83 -c [command file DOS path]

When the linker has finished you will receive a message notifying you of any errors and a file, *.abs will have been created in the current directory (*.abs is given the same file name as the .cmd file). The file, *.abs, is the S-record form of the linked files and can be downloaded to the ASE.

The summary given here is by no means complete, but it should be sufficient to get you started. Take time to look through the ASMH83 H8/300 Assembler manual. There are several options not covered in this TechNote which may greatly simplify and enhance your code.

Direct Memory Addressing with C Pointers

Tech Notes

Application Engineering

Paul Yiu

Direct Addressing

In Embedded Programming, the C source code often needs to access memory addresses directly to drive the hardware peripherals, such as I/O ports, timers, registers, etc.

The best way to demonstrate direct addressing is by example. If we have a timer, called tmr0, at addressing 0xffc8, and we want to change its value. This is the simplest way:

1 Define tmr0 as a constant:

In C source code:

unsigned int const tmr0 = 0xFFC8;

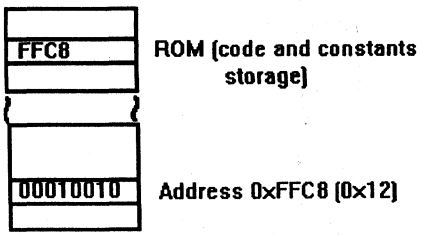
a) It's best to define addresses as unsigned numbers, so the compiler won't mistake 0xffc8 as a negative number.
b) The compiler puts our variable, tmr0, in ROM because it's declared as a constant, thus freeing up more RAM.
c) In C, hex numbers are preceded by 0x(zero x).
d) Pointers are extremely tricky, and they take up extra memory space, so we define tmr0 as just an integer. We can cast this number to be a pointer later.

2 Cast this integer as a pointer to a memory address:

In C source code:

****(unsigned char *)tmr0 = 0x12;***

a) ***(unsigned char *)*** casts tmr0 to be a pointer to an unsigned character. Now, (unsigned char *)tmr0 refers to address hex FFC8. Adding a '*' in front of it makes the expression content of address 0xFFC8.



SECTION 7

b) If we want to put an integer in address FFC8 and FFC9, we can just change the casting to (unsigned int *), then if we say ***(unsigned int *)tmr0=0x1234;***, we will have 0x12 in FFC8, 0x34 in FFC9.

c) The benefit of defining address as integers instead of pointers is we don't need to define this pointer as a character or an integer until we want to put numbers in these addresses. These integers, casted as pointers, are much more flexible.

Pointers to Strings

In C, the simplest way to define a string is to define a pointer that points to that string.

In C source code:

```
unsigned char *sinatra="Fly me to the moon.";
```

Compiled:

```
.EXPORT _sinatra
_sinatra   .DATA.W     S0
.SECTION        strings,TEXT,ALIGN=2
S0 .SDATA       "Fly me to the moon"<0>
```

a) The compiler will put this string in the strings section. The pointer, "sinatra" refers to the address of the first character, "F."

b) *sinatra refers to the character "F."

Say we have defined another pointer, called ptr.

In C source code:

```
unsigned char *ptr;
ptr=sinatra;
```

Compiled:

```
.IMPORT _ptr
.comm _ptr,H'2
mov.w     @_sinatra,r0
mov.w     r0,@_ptr
```

a) Here, we first define a pointer, called ptr, not pointed to anything yet. Secondly, we tell the compiler to let "ptr" point to whatever "sinatra" is pointing to. *ptr now is character "*F*." Since the string is stored in memory in order, if we increment ptr, we get the next character.

In C source code:

```
ptr++;
```

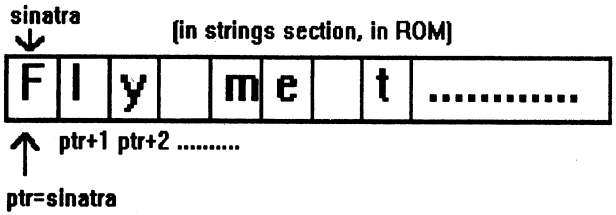
Compiled:

```
adds      #1,r0
mov.w     r0,@_ptr
```

a) *ptr now is character " *l*."

b) ptr corresponds to the address of the letter "*l*."

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300


sinatra
(in strings section, in ROM)
F l y m e t .....
ptr+1 ptr+2 .....
ptr=sinatra

Hitachi America, Ltd.

SEMICONDUCTOR & I.C. DIVISION
San Francisco Center
2000 Sierra Point Parkway
Brisbane, CA 94005–1819
Telephone: 415–589–8300
Fax: 415–583–4207

REGIONAL OFFICES

NORTHEAST REGION
Hitachi America, Ltd.
77 South Bedford Street
Burlington, MA 01803
Telephone: 617-229-2150
Fax: 617-229-6554

NORTH CENTRAL REGION
Hitachi America, Ltd.
500 Park Boulevard, Suite 415
Itasca, IL 60143
Telephone: 708-773-4864
Fax: 708-773-9006

NORTHWEST REGION
Hitachi America, Ltd.
1900 McCarthy Boulevard, Suite 310
Milpitas, CA 95035
Telephone: 408-954-8100
Fax: 408-954-0499

SOUTHEAST REGION
Hitachi America, Ltd.
5511 Capital Center Drive, Suite 204
Raleigh, NC 27606
Telephone: 919-233-0800
Fax: 919-233-0508

SOUTH CENTRAL REGION
Hitachi America, Ltd.
Two Lincoln Centre, Suite 865
5420 LBJ Freeway
Dallas, TX 75240
Telephone: 214-991-4510
Fax: 214-991-6151

SOUTHWEST REGION
Hitachi America, Ltd.
2030 Main Street, Suite 450
Irvine, CA 92714
Telephone: 714-553-8500
Fax: 714-553-8561

PACIFIC MOUNTAIN REGION
Hitachi America, Ltd.
4600 S. Ulster Street, Suite 700
Denver, CO 80237
Telephone: 303-740-6644
Fax: 303-740-6609

AUTOMOTIVE REGION
Hitachi America, Ltd.
330 Town Center Drive, Suite 311
Dearborn, MI 48126
Telephone: 313-271-4410
Fax: 313-271-5707

MID-ATLANTIC REGION
Hitachi America, Ltd.
325 Columbia Turnpike, Suite 203
Florham Park, NJ 07932
Telephone: 201-514-2100
Fax: 201-514-2020

DISTRICT OFFICES

CANADA
Hitachi (Canadian) Ltd.
320 March Road, Suite 602
Kanata, Ontario, Canada K2K 2E3
Telephone: 613-591-1990
Fax: 613-591-1994

FLORIDA
Hitachi America, Ltd.
4901 N.W. 17th Way, Suite 302
Fort Lauderdale, FL 33309
Telephone: 305-491-6154
Fax: 305-771-7217

MINNESOTA
Hitachi America, Ltd.
3800 W. 80th Street, Suite 1050
Bloomington, MN 55431
Telephone: 612-896-3444
Fax: 612-896-3443

IBM REGION
Hitachi America, Ltd.
21 Old Main Street, Suite 104
Fishkill, NY 12524
Telephone: 914-897-3000
Fax: 914-897-3007

TEXAS
Hitachi America, Ltd.
10777 Westheimer, Suite 1040
Houston, TX 77042
Telephone: 713-974-0534
Fax: 713-974-0587

Hitachi America, Ltd.
9600 Great Hills Trail, Ste. 150W
Austin, TX 77042
Telephone: 512-345-9983
Fax: 512-343-2759

MANUFACTURING FACILITY
Hitachi Semiconductor (America) Inc.
6321 East Campus Circle Drive
Irving, TX 75063-2712

ENGINEERING FACILITY
Hitachi Micro Systems, Inc.
179 East Tasman Drive
San Jose, CA 95134

Technical product or pricing questions can be answered by your nearest Hitachi office.

You may order product literature either by calling your nearest Hitachi office or by calling 1-800-285-1601.

HITACHI®

HITACHI®

Our Standards Set Standards

Hitachi America, Ltd.
Semiconductor & I.C. Division
San Francisco Center
2000 Sierra Point Parkway, Brisbane, CA 94005-1819
1-415-589-8300

892/5M/GI/LP/RD
Order Number: M00T021
Printed on recycled paper.