Intel 186 EB/EC Evaluation Board User's Manual

80C186EC/80C188EC
80L186EC/80L188EC
and
80C186EB/80C188EB
80L186EB/80L188EB

March 1997

Order Number: 272986-001

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

CONTENTS

CHAPTER 1
ABOUT THIS MANUAL
1.1 CONTENT OVERVIEW 1-1
1.2 NOTATION CONVENTIONS 1-2
1.3 RELATED DOCUMENTS 1-3
1.4 ELECTRONIC SUPPORT SYSTEMS 1-4
1.4.1 FaxBack Service 1-4
1.4.2 World Wide Web 1-4
1.5 TECHNICAL SUPPORT 1-5

CHAPTER 2
GETTING STARTED
2.1 SYSTEM REQUIREMENTS 2-3
2.2 WHAT'S IN YOUR KIT 2-3
2.3 VIEWING THE BOARD SCHEMATICS 2-4
2.4 SETTING UP THE EVALUATION BOARD AND THE HOST PC 2-4

CHAPTER 3
HARDWARE OVERVIEW
3.1 JUMPER SUMMARY 3-1
3.2 MICROPROCESSOR 3-2
3.2.1 Packaging 3-2
3.3 MEMORY CONFIGURATION 3-3
3.3.1 Flash (Program Memory) 3-5
3.3.1.1 Configuring the Board for Flash Downloading 3-5
3.3.2 SRAM (Static Memory) 3-7
3.4 PROGRAMMABLE LOGIC 3-7
3.5 POWER SUPPLY 3-8
3.6 SERIAL INTERFACE 3-9
3.7 EXPANSION INTERFACE 3-12
3.8 LCD INTERFACE 3-15
3.8.1 LCD Interface Demo 3-15

CHAPTER 4
INTRODUCTION TO THE SOFTWARE
4.1 SOFTWARE FEATURES 4-1
4.2 RESTRICTIONS 4-2
4.3 EMBEDDED CONTROLLER MONITOR (ECM) 4-2
4.4 USER INTERFACE 4-3
4.4.1 Numeric Input 4-3
4.4.2 Controlling Lengthy Commands 4-3
4.4.3 Aborting from iECM-86 4-3
4.5 INITIATING AND TERMINATING iECM-86 4-3
4.5.1 ECM86 4-3
4.5.2 -COM2, -COM1 4-4
4.5.3 -DIAG 4-4
4.5.4 -POLL, -SIGNAL 4-5
4.5.5 RESET SYSTEM, RES SYSTEM, RESET, RES 4-5
4.5.6 DOS 4-5
4.5.7 QUIT 4-5
4.6 RELATED INFORMATION 4-6
4.6.1 Reserved Functions 4-6
4.6.2 Reserved Memory 4-6
4.6.3 Reserved I/O 4-6

CHAPTER 5
iECM-86 COMMANDS
5.1 ENTERING COMMANDS 5-1
5.2 FILE OPERATIONS 5-2
5.2.1 Loading and Saving Object Code 5-2
5.2.2 Other File Operations 5-3
5.3 PROGRAM CONTROL 5-5
5.3.1 Resetting the Target 5-5
5.3.2 Breakpoints 5-5
5.3.3 Program Execution 5-7
5.3.4 Program Stepping 5-8
5.4 DISPLAYING AND MODIFYING PROGRAM VARIABLES 5-10
5.4.1 Supported Data Types 5-10
5.4.2 BYTE Commands 5-11
5.4.3 WORD Commands 5-12
5.4.4 DWORD Commands 5-13
5.4.5 STACK Commands 5-14
5.4.6 STRING Commands 5-15
5.4.7 PORT Commands 5-15
5.4.8 WPORT Commands 5-16
5.4.9 Processor Variables 5-17

CHAPTER 6
iRISM-186 COMMANDS
6.1 IRISM VARIABLES 6-1
6.1.1 Other Variables 6-1
6.2 RISM STRUCTURE 6-2
6.3 RECEIVING DATA FROM THE HOST 6-2
6.4 SENDING DATA TO THE HOST 6-2
6.5 RISM COMMANDS 6-2
6.5.1 SET_DATA_FLAG (Code 00H) 6-3
6.5.2 TRANSMIT (Code 02H) 6-3
6.5.3 READ_BYTE (Code 04H) 6-3
6.5.4 READ_WORD (Code 05H) 6-3
6.5.5 READ_DOUBLE (Code 06H) 6-3
6.5.6 WRITE_BYTE (Code 07H) 6-3
6.5.7 WRITE_WORD (Code 08H) 6-3
6.5.8 WRITE_DOUBLE (Code 09H) 6-4
6.5.9 LOAD_ADDRESS (Code 0AH) 6-4
6.5.10 READ_PC (Code 10H) 6-4
6.5.11 WRITE_PC (Code 11H) 6-4
6.5.12 START_USER (Code 12H) 6-4
6.5.13 STOP_USER (code 13H) 6-4
6.5.14 TRAP_ISR 6-5
6.5.15 REPORT_STATUS (Code 14H) 6-5
6.5.16 MONITOR_ESCAPE (Code 15H) 6-5
6.5.17 READ_BPORT (Code 16H) 6-5
6.5.18 WRITE_BPORT (Code 17H) 6-5
6.5.19 READ_WPORT (Code 18H) 6-5
6.5.20 WRITE_WPORT (Code 19H) 6-6
6.5.21 STEP (Code 1AH) 6-6
6.5.22 READ_REG (Code 1BH) 6-6
6.5.23 WRITE_REG (Code 1CH) 6-6
6.5.24 Start Up Commands (/ or \) 6-7

APPENDIX A
PARTS LIST

FIGURES

2-1 Intel 186 EB Evaluation Board Layout 2-1
2-2 Intel 186 EC Evaluation Board Layout 2-2
3-1 Physical Memory Map 3-4
3-2 Jumper Assembly for Flash Downloading 3-6
3-3 E1 Jumper 3-8
3-4 J2 Power Connector 3-8
3-5 25-Pin to 9-Pin Adaptor 3-11
3-6 186 EC Peripheral Expansion Connector JP2 (40 pin) 3-12
3-7 186 EB Peripheral Expansion Connector JP2 (24 pin) 3-13

TABLES

1-1 Customer Support Telephone Numbers 1-5
3-1 80x186EB/EC Evaluation Board Jumper Settings 3-1
3-2 Logical Memory Map 3-3
3-3 P1 Host Serial Connector 3-9
3-4 P2 Serial Channel 0 3-10
5-1 Supported Data Types 5-10
6-1 iRISM Variables 6-1
6-2 iRISM Registers 6-6
A-1 80186 EB Board Manual Parts List A-1
A-2 80186 EC Board Manual Parts List A-4

1

About This Manual

CHAPTER 1
ABOUT THIS MANUAL

This manual describes how to set up and use the Intel 186 EB/EC Evaluation Board. The board is used to evaluate hardware and software performance and provide an “emulation-like” feel when executing and debugging user-written code. This board operates at either 3.3 volts or 5.0 volts. It supports the following processors:

- 80C186EB/80C188EB
- 80L186EB/80L188EB
- 80C186EC/80C188EC
- 80L186EC/80L188EC.

The 3.3 V, 16 MHz 80L186EB or 80L186EC processor is installed on the evaluation board. This manual covers both processors.

1.1 CONTENT OVERVIEW

Chapter 1, About This Manual — This chapter contains an overview of this manual.

Chapter 2, Getting Started — This chapter describes the Intel 186 EC/EB Evaluation Board, and provides setup instructions.

Chapter 3, Hardware Overview — This chapter describes the evaluation board hardware, such as connectors, jumpers, memory configuration, and power supply.

Chapter 4, Introduction to the Software — This chapter provides an overview of the software used on the evaluation board and the host computer.

Chapter 5, iECM-86 Commands — This chapter describes the iECM-86 software, which runs on the host computer.

Chapter 6, iRISM-186 Commands — This chapter describes the iRISM-186 software, which runs on the evaluation board.

Appendix A, Parts List — This chapter contains a part list for both the EB and EC versions of the evaluation board.

1.2 NOTATION CONVENTIONS

The following notation conventions are used in this manual.

#	Pound symbol (#) appended to a signal name indicates that the signal is active low.
italics	Italics identify variables and indicate new terms.
bold sans-serif	In text, identifies commands (instructions).
`typewriter font`	This font is used for code examples. All characters are equal width; this is useful for maintaining accurate character spacing.
UPPERCASE	In text, signal names are shown in uppercase. When several signals share a common name, each signal is represented by the signal name followed by a number; the group is represented by the signal name followed by a variable (*n*). In code examples, signal names are shown in the case required by the software development tool in use.
Designations for hexadecimal and binary numbers	Hexadecimal numbers are represented by a string of hex digits followed by the letter *H*. A zero prefix is added to numbers that begin with *A* through *F*. (*FF* is shown as *0FFH*.) For binary numbers, the letter B may be appended for clarity.

Units of Measure

mA	milliamps, milliamperes
A	amps, amperes
Kbit, Kbyte	kilobits, kilobytes
KΩ	kilo-ohms
Mbit, Mbyte	megabits, megabytes
KHz, MHz	kilohertz, megahertz
ms	milliseconds
µs	microseconds
ns	nanoseconds
µF	microfarads
W	watts
V	volts

NOTE:
Units listed are frequently used; other units and symbols are used as necessary.

1.3 RELATED DOCUMENTS

You can order Intel product literature from the following Intel literature centers.

1-800-548-4725	U.S. and Canada
708-296-9333	U.S. (from overseas)
44(0)1793-431155	Europe (U.K.)
44(0)1793-421333	Germany
44(0)1793-421777	France
81(0)120-47-88-32	Japan (fax only)

The following documents may be useful for designing applications using this evaluation board.

Document Name	Intel Order #
80C186EB/80C188EB Microprocessor User's Manual	270830
80C186EC/80C188EC Microprocessor User's Manual	272047
80C186EB/80C188EB and 80L186EB /80L188EB datasheet	272433
80C186EC/80C188EC and 80L186EC/80L188EC datasheet	272434
Flash Memory databook	210830
Application Notes	
AP484: Interfacing a Floppy Disk Drive to an 80C186EX Family Processor	272339
AP730: Interfacing the 82C59A-2 to Intel186 Family Processors	272822
AP731: Understanding the Interrupt Control Unit of the 80C186EC/80C188EC	272823
ApBuilder and Hypertext	
80C186EC/80C188EC Hypertext Manual & Datasheet	272298
ApBuilder Interactive Programming Tool Software Package	272216

1.4 ELECTRONIC SUPPORT SYSTEMS

Intel's FaxBack* service provides up-to-date technical information. Intel also offers a variety of information on the World Wide Web. These systems are available 24 hours a day, 7 days a week, providing technical information whenever you need it.

1.4.1 FaxBack Service

FaxBack is an on-demand publishing system that sends documents to your fax machine. You can get product announcements, change notifications, product literature, device characteristics, design recommendations, and quality and reliability information.

1-800-525-3019 (US or Canada)

+44-1793-496646 (Europe)

+65-256-5350 (Singapore)

+852-2-844-4448 (Hong Kong)

+886-2-514-0815 (Taiwan)

+822-767-2594 (Korea)

+61-2-975-3922 (Australia)

1-503-264-6835 or 1-916-356-3105 (Worldwide)

1.4.2 World Wide Web

Intel offers a variety of information through the World Wide Web (http://www.intel.com/).

1.5 TECHNICAL SUPPORT

Table 1-1. Customer Support Telephone Numbers

Customer Support (US and Canada)	800-628-8686
Australia National Sydney	008-257-307 61-2-975-3300 61-3-810-2141
Belgium, Netherlands, and Luxembourg	010-4071-111
Canada	Contact local distributor
Finland	358-0-544-644
France	33-1-30-57-72-22
Germany	Hardware: 49-89-903-8529 Software: 49-89-903-2025
Israel	972-3-548-3232
Italy	39-02-89200950
Japan	0120-1-80387
Sweden	46-8-7340100

2

Getting Started

CHAPTER 2
GETTING STARTED

This chapter describes the Intel 186 EC/EB Evaluation Board kit, and provides setup instructions. Figure 2-1 shows the 80x186 EB Evaluation Board layout, and Figure 2-2 shows the EC board layout. Refer to these figures when you are following the instructions in this chapter for setting up your evaluation board.

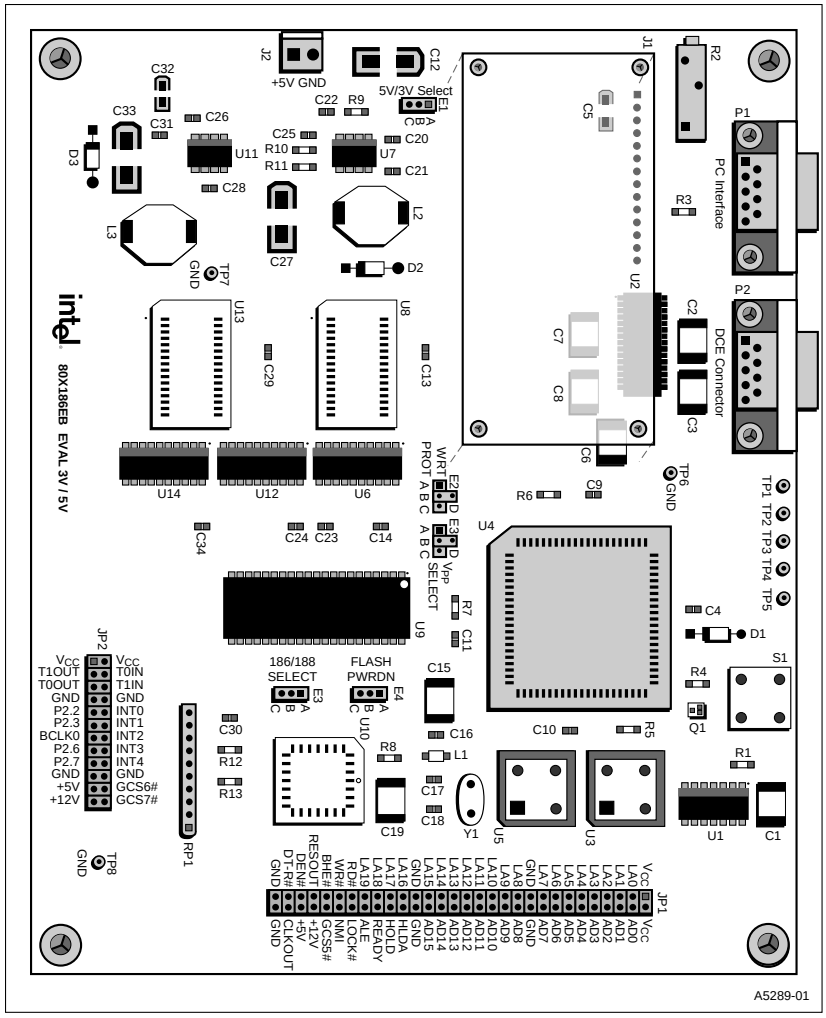


Figure 2-1. Intel 186 EB Evaluation Board Layout

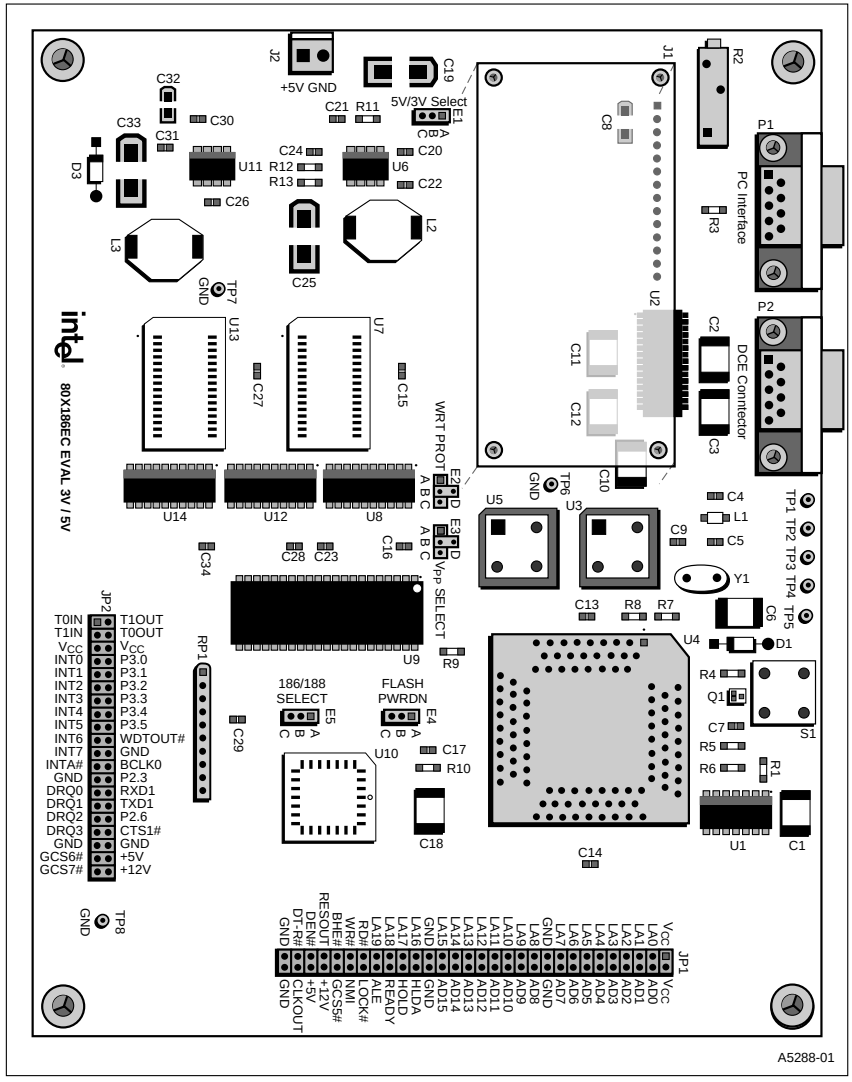


Figure 2-2. Intel 186 EC Evaluation Board Layout

2.1 SYSTEM REQUIREMENTS

- IBM* PC AT, XT or BIOS-compatible computer host system (interfaces via COM1 or COM2 at 9600 baud).
- 5 V power supply (the connector housing and contact pins are included in the kit).

2.2 WHAT'S IN YOUR KIT

Evaluation Board	Your kit includes a board with either a 3.3 volt, 16 MHz 80L186EB or 80L186EC microprocessor installed. Separately packaged components included with the board are 5 VDC versions of the microprocessor and SRAM for conversion to a 5 VDC evaluation platform.
Monitor Program	The Embedded Controller Monitor (ECM) program supports basic software and hardware evaluation and basic debug facilities (LOAD, GO, STEP, etc.) on the evaluation board. The ECM consists of two programs: RISM-186 executes in the evaluation board and ECM-86 executes in an IBM PC or BIOS-compatible computer, called the host PC. These two programs communicate through an asynchronous serial channel using a binary protocol defined specifically for this application. The source code for the monitor software is provided on a diskette included in your kit; this allows you to update the software for various operating conditions in your target application.
Contents on Disk	In addition to Flash downloading software, a diskette provided in the kit contains schematics, a pld file for the programmable logic device used on the board, and a sample assembly file for working the with LCD display. Compiler software is not included in the kit.
Software Development Kit	The kit provides a software development kit, which includes a software debugger, locator, and sample code.
Flash Loading Utility	Users can download application programs to the on-board Flash memory for execution. The Flash loading utility is contained on a diskette, and a separate manual, the *CQI Flash Loader User's Manual*, provides instructions for using this utility.
Serial Cable	A serial cable is provided to connect the evaluation board to the host PC.

2.3 VIEWING THE BOARD SCHEMATICS

The schematics provided on the diskette are in the Adobe* Acrobat .pdf format. You can view and print the schematics using the Acrobat Reader. The Reader is available at no charge from the Intel World Wide Web site (http://www.intel.com/) or from the Adobe site (http://www.adobe.com/).

2.4 SETTING UP THE EVALUATION BOARD AND THE HOST PC

This section tells you how to set up the board for use with a host PC. This section assumes you won't be using some of the advanced features of the board when you first power it up. For additional options, such as selecting 80188 evaluation mode, refer to Chapter 3, "Hardware Overview."

1. Make sure you are in a static-free environment before removing any components from their anti-static packaging. The evaluation board is susceptible to electro-static discharge damage; such damage may cause product failure or unpredictable operation.

2. Inspect the contents of your kit. Make sure that all items are included. Check for damage that may have occurred during shipment. Contact your sales representative if any items are missing or damaged.

CAUTION: Many of the connectors on the evaluation board provide power through non-standard pins. Connecting the wrong cable or reversing the cable can damage the evaluation board and may damage the device being connected. Use extreme caution when preparing to connect cables to this product.

3. Connect the power supply. The Intel 186 EC/EB Evaluation Board operates from a 5 VDC ± 10% power supply plugged into the J2 power connector (see Figures 2-1 and 2-2). This 5 volt signal is stepped down to 3.3 volts on the board. The connector housing and contact pins provided in your kit match the power supply to the J2 connector.

   To select 5 V, place a jumper on pins B and C of jumper E1. To select 3 V, place a jumper on pins A and B of jumper E1. See Figures 2-1 and 2-2 for jumper locations.

   All devices on the board operate at both 3.3 volts and 5.0 volts (except the LCD display, which is hardwired to 5 volts). This option allows comparison of current consumption when running code at either voltage. Separately packaged 5 V versions of the 80C186 processor and SRAM must be installed on the board for 5 V operation.

4. Apply power to the host PC and the evaluation board.

   When power is applied to the board, the message “i186 Ex 3V/5V EV” should appear across the LCD display. This message indicates board initialization is complete. If the message does not appear, press the reset button (S1).

   Connect one end of the standard 9-pin AT-type serial connector to header P1 on the evaluation board. Connect the other end to the COM1 port of the host computer. (You can use COM2 if you need to, but you’ll have to specify COM2 when you run the Monitor Software.) The PC and board communicate at 9600 baud.

   After connection to the PC, the processor may appear to be held in the reset state. The reason this occurs is that one of the host signals is used to reset the board. This signal may be active prior to invoking the ECM86 host software on the PC. The PC and board communicate at 9600 baud.

5. Insert the ECM-86 floppy disk provided with your kit in the floppy drive on the host PC. You can run the ECM86 program directly from the diskette or copy the contents of the diskette to your hard drive.

6. At the DOS prompt, change to the floppy disk drive (or to the directory to which you copied the files in the previous step) and enter this command:

   ECM86

   After a moment, the PC should display the ECM86 monitor screen.

Complete information on using the monitor software is located in Chapters 4 and 5.

3

Hardware Overview

CHAPTER 3
HARDWARE OVERVIEW

The evaluation board comes with a 16 MHz 80L186 EB or EC processor, 512 Kbytes of Flash (containing the iRISM-186 monitor and a Flash loader utility in the boot block), and 256 Kbytes of SRAM. The expansion connector (JP1) supports up to 1 Mbyte of external memory and 64 Kbytes of external I/O. Refer to Figures 2-1 and 2-2 for the exact locations of connectors, jumpers and headers listed in this chapter.

The board utilizes the high peripheral integration of the 186 product family. The programmable chip-selects support on-board memory, expansion memory, and the LCD interface. The timer/counter unit controls timing for LCD display accesses. The serial control unit communicates with the host PC through the iECM-86 software and the Flash loader host software. Finally, the I/O port unit controls on-board power management functions (enable/disable serial drivers and +12 volts).

Other on-chip peripherals are made available for hardware expansion via the JP1, JP2, and P2 connectors. The following sections describe in detail the specific devices used on the board.

3.1 JUMPER SUMMARY

Table 3-1. 80x186EB/EC Evaluation Board Jumper Settings

Jumper	Name	Description	Options
E1	5 V/3 V Select	Selects voltage (5 V or 3.3 V) that will be present on V_{CC} power plane.	A-B = 3.3 V† B-C = 5 V
E2	LA19/WRT PROT	Selects options for Flash WP# pin. Includes option to make LA19 available to Flash pin 2 for upgrading to 8-MBIT component (PA28F800BV).	A-B = Write protect boot block† B-C = Unlock boot block B-D = Add LA19 for 8- MBIT Flash
E3	V_{PP} Select	Selects 5 V or 12 V programming voltage, as well as GND to remove all program and erase capabilities.	A-B = Total WRT protect† B-C = 12 V program voltage B-D = 5 V program voltage
E4	Flash Powerdown Select	Selects options for Flash RP# pin. For normal operation, SW-RES# is selected. To unlock boot block (regardless of WP#), 12 V is selected.	A-B = Normal† B-C = Program boot block override
E5	186/188 Select	Jumper for appropriate processor type.	A-B = 188 processor installed B-C = 186 processor installed†

† Default setting

3.2 MICROPROCESSOR

The core of the evaluation board is the 80x186 microprocessor. This processor operates at 3.3 volts up to 16 MHz in this board. Alternatively, the board can be configured to run at 5 volts up to 33 MHz. To vary the CPU clock speed, an appropriate frequency value oscillator must be installed at location U3 on the EC board and at location U5 on the EB board. The oscillator operates at twice the frequency of the installed processor.

The 80x186 processor offers the following features:

- 16-bit data bus
- 1 Mbyte address space
- 2 on-chip UARTs
- 10 programmable chip-selects
- Interrupt control unit
- 3 programmable timer/counters
- Power management unit
- 32-bit watchdog timer (EC only)
- 4 DMA channels (EC only)

The 8-bit bus version of the processor (80C188/80L188) may also be used in this board. To configure the board to operate with an 8-bit bus, jumper E5 must be in the A–B position. To configure the board to operate with a 16-bit bus, jumper E5 must be in the B–C position. Many of the processor's on-chip peripherals can be accessed using the two expansion connectors on the board (JP1 and JP2).

NOTE

Because host communications use the on-chip serial ports, changing the operating frequency of the board requires the processor serial ports to be reconfigured. The RISM monitor source code is provided on a floppy diskette in your kit and is commented to indicate current register values.

3.2.1 Packaging

The 80x186 EC is packaged in a 100 lead PQFP and socket and the 80x186 EB is packaged in an 84 lead PLCC package and socket. Adaptors are available from Applied Microsystems Corp.* and Emulation Technologies, Inc.* to allow for the connection of in-circuit emulators.

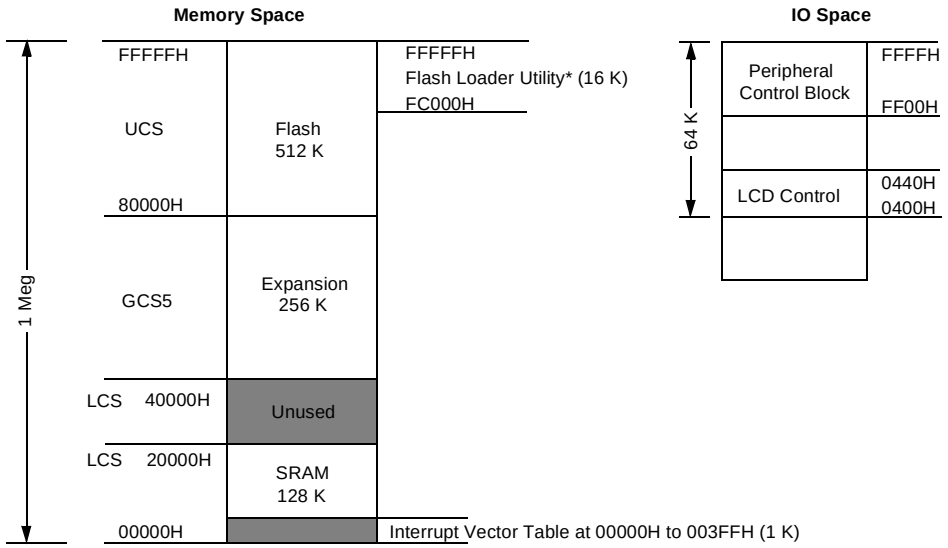
3.3 MEMORY CONFIGURATION

The memory on the evaluation board can be divided into three types: Flash, SRAM, and expansion. Flash memory contains the Flash loader utility, located in the boot block boundary, and the RISM monitor program, beginning at F800:0000. Users can execute their test code from boot-up using the Flash loader utility. Refer to the *CQI Flash Loader Reference Manual* for instructions on programming the Flash memory. SRAM memory is used for the processor interrupt vector table, stack allocation, and RISM data variables, and as a possible destination for user-written code downloaded on the host interface. Expansion memory can be accessed through the expansion interface, if required.

Table 3-2 shows the logical memory map and Figure 3-1 shows the physical memory map of the evaluation board.

Table 3-2. Logical Memory Map

Memory Area	Start (H)	Stop (H)	Size
SRAM	0000:0000	2000:0000	128 Kbytes
Flash	8000:0000	F000:FFFF	512 Kbytes
Flash Boot Block	FC00:0000	F000:FFFF	16 Kbytes
Expansion	4000:0000	8000:0000	256 Kbytes
LCD (I/O)	0000:0400	0000:0440	64 bytes



UCS — Upper Chip Select
Start: 80000H Stop: FFFFF (Flash 512 K)

- 3 Wait States
- Active for Memory Bus Cycles
- Bus Ready Not Required

LCS — Lower Chip Select
Start: 0000H Stop: 40000H (SRAM 128 K)

- 2 Wait States
- Stop Address Required
- Active for Memory Bus Cycles
- Bus Ready Ignored

GCS5 — Expansion Memory
Start: 40000H Stop: 80000H (Expansion 256 K)

- Zero Wait States
- Ignore Stop Address
- Active for Memory Bus Cycles
- Bus Ready Required

GCS7 — Expansion I/O
Start: 400H Stop: 500H

- Zero Wait States
- Ignore Stop Address
- Active for I/O Bus Cycles
- Bus Ready Required

GCS2 — LCD Display I/O
Start: 400H Stop: 440H

- 8 Wait States
- Active for I/O Bus Cycles
- Bus Ready Ignored

* As shipped, RISM is located at F800:0000 and pointed to by the Flash loader utility during boot-up.

Figure 3-1. Physical Memory Map

3.3.1 Flash (Program Memory)

Flash memory, as configured in the RISM monitor, is mapped to the upper 512 Kbytes of the 1 Mbyte 80x186 processor address space. The board includes a single 4 Mbit, 32-pin PSOP Flash device at location U9 with 110 ns access time at 3.3 V and 60 ns access time at 5 V. This memory runs with one wait state at 5 volts/20 MHz and 3.3 volts/16 MHz.

The device data bus can be configured to be either 8 or 16 bits wide (corresponding to the 80x188and the 80x186 processor, respectively). Jumper E5 determines the Flash bus width. When E5 is in the A–B position, the bus is 8 bits wide; when E5 is in the B–C position, the bus is 16 bits wide. The configurable bus width allows access to all 512 Kbytes of Flash memory.

If a user application requires nonvolatile memory for storage, Flash can be erased and written by jumpering E3 for either 5 V or 12 V programming voltage (V_{PP}) and using the proper programming algorithm. The SmartVoltage* Flash device can be programmed using either voltage. The Flash loader utility is located in the Flash boot block (upper 16 Kbytes, FC000h to FFFFFh). Writes to this region are prohibited, regardless of the voltage on V_{PP}, unless the RP# input is at +12 volts or jumper E2 is set to unlock the boot block. Jumper E4 controls the voltage on RP#. When E4 is in the B–C position, the +12 volt supply is connected to RP#. When E4 is in the A–B position, RP# is connected to the board reset signal.

CAUTION: To access boot block memory, E4 must be in the B–C position and Port Pin 1.1 must be programmed to a logic 0 (enabling +12 volts). Accessing the boot block is **not** recommended, as the Flash loader utility code could be corrupted.

3.3.1.1 Setting Up the Board for Flash Downloading

You can use the Flash utility host program, FLASHLDR.EXE, provided in the kit to download your application program to the Flash memory. Upon reset or power-up, the Flash loader reads port pin to determine whether to execute a loaded program, such as RISM, or download new software to Flash memory.

To set up the board for Flash downloading:

1. Power-off the evaluation board and disconnect the serial cable from the PC.
2. Port pin P2.6 on the secondary header (JP2) controls which programs execute at start-up. Connect P2.6 to the +5 volt pin with jumper wire and 10 kΩ resistor. Figure 3-2 illustrates this connection.

CAUTION: A 10 kΩ resistor is required when jumpering from the P2.6 pin to the 5 volt pin; if this configuration is not used, the processor's port control hardware could be damaged.

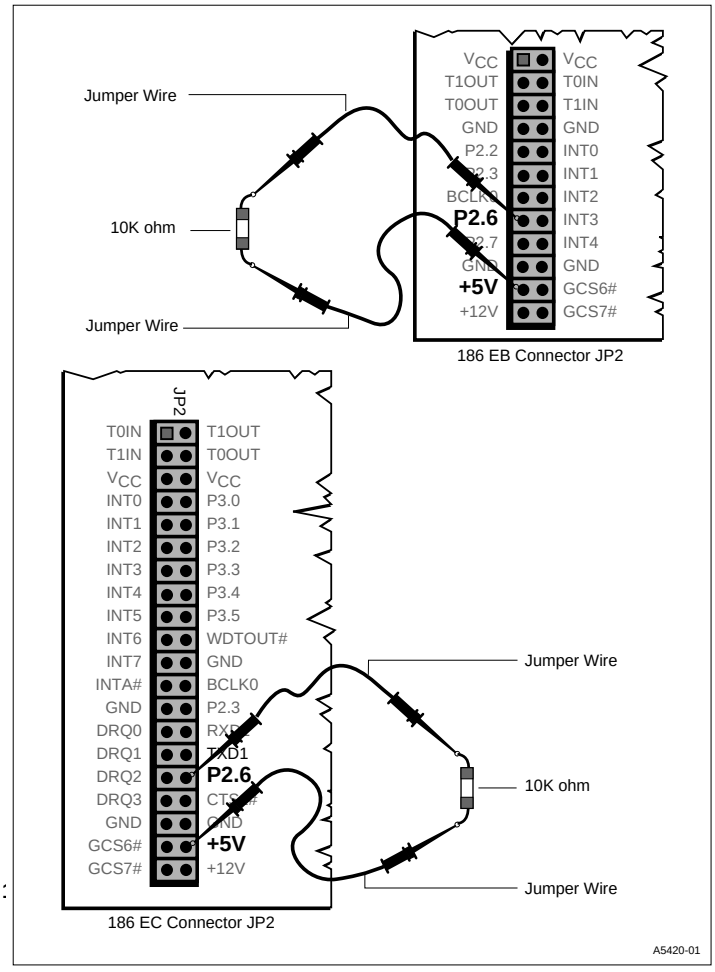


Figure 3-2. Jumper Assembly for Flash Downloading

3. Reconnect the serial cable and power-up the board.

   You should notice that the text CQFLASH LOADER now displays on the LED, signaling that the board is ready for Flash downloading.

When the jumper assembly is installed, the Flash target program waits for commands from the PC host, allowing you to use the provided Flash loader utility program to download programs to the Flash.

If this text does not display on the LED, indicating a problem with the jumper assembly, the board boots as if no Flash loader assembly is installed; that is, the Flash target program immediately starts the loaded user application program (for example, the iRISM monitor software).

You can find complete instructions for using the Flash utility program in the *CQI Flash Loader User Manual* included in your kit.

3.3.2 SRAM (Static Memory)

SRAM occupies the lower 128 Kbytes of memory starting at location 00000H. This memory is used by the processor for interrupt vectors and stack allocation, by the RISM for program variables, and by the user for downloaded code. The board includes two 1-Mbit, 32-pin SRAMs with 17 ns access time at 3.3 volts. SRAMs are socketed to allow installation of 5 V SRAMs (17 ns access time).

To allow insertion of both the 80x186 processor and the 80x188 processor, the memory is configured such that only 128 Kbytes of the SRAM is accessible, even though 256 Kbytes of SRAM are installed on the board.

3.4 PROGRAMMABLE LOGIC

All glue logic required by the evaluation board is implemented on a GAL 22LV10C-15. The PLD file located on the floppy diskette in your kit includes logic equations for this device. The logic implemented includes the following:

- Inverting the Port Pin signal controlling V_{PP} (so V_{PP} is disabled at reset)
- Controlling the 8-bit/16-bit configuration for the Flash device
- Decoding the Enable signal for the LCD display

3.5 POWER SUPPLY

The power supply connects to J2 on the board schematic. Pin 1 must connect to +5 volts and pin 2 must connect to ground. The supply is then regulated to 3.3 volts by the on-board circuitry. The V_{CC} for the board is controlled by jumper E1. When E1 is in the A–B position, V_{CC} = 3.3 volts; when E1 is in the B–C position, V_{CC} = 5.0 volts. V_{CC} is converted to +12 volts for optional Flash programming voltage.

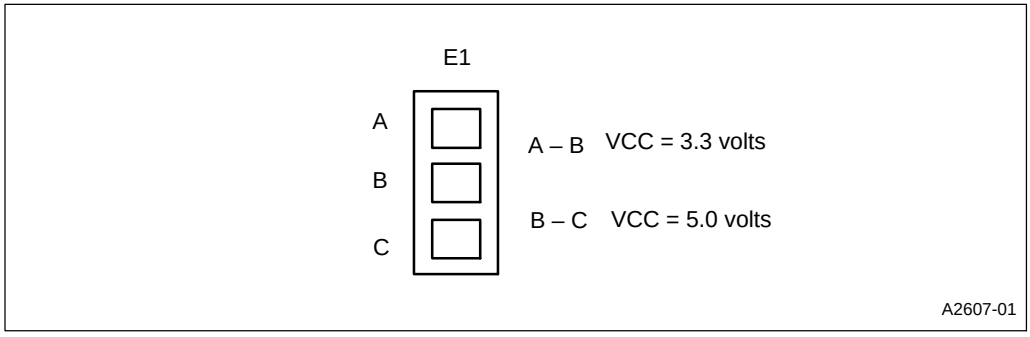


Figure 3-3. E1 Jumper

The LCD display controller V_{CC} pin connects directly to the 5 volt supply, not the V_{CC} plane, allowing 5 volt operation only.

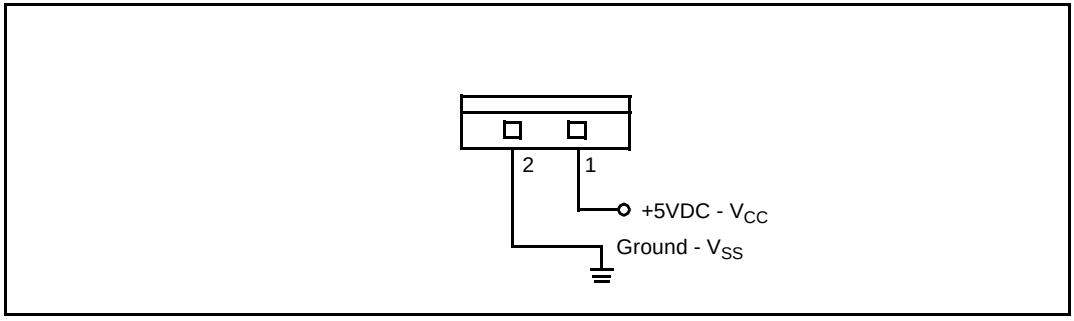


Figure 3-4. J2 Power Connector

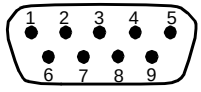
The Maxim* MAX750 component at U6 (EC) or U7 (EB) is a current-mode DC-DC converter. This device takes the 5 volt supply and steps it down to 3.3 volts. This voltage output is always supplied to provide V_{CC} for the processor, memory, and logic when selected at E1.

The Maxim MAX734 located at U11 is also a current-mode DC-DC converter. This device steps up the V_{CC} voltage to +12.0 volts. This voltage output is supplied to provide a V_{PP} option for Flash memory programming. The SHDN# input (pin 1) connects to a port pin (P1.1) on the processor through an inverter. At reset, SHDN# is driven low to disable the +12 volt signal. The output remains disabled until Port Pin 1.1 is programmed to a logic 0. When SHDN# is low, the output (pin 8) is V_{CC} minus a diode drop. The evaluation board uses SmartVoltage Flash. To prevent unintentional writes to Flash, set jumpers E2 and E3 as indicated in Table 3-1.

3.6 SERIAL INTERFACE

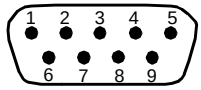
Connector P1 connects to your PC's serial port. P1 interfaces pin-to-pin with a standard nine-pin RS-232 serial connector. Verify that the cable being used provides all signals required.

Table 3-3. P1 Host Serial Connector

P1 Connector	Pin Nos.	Host RS-232 Signal Name	Connection on Evaluation Board
	1 (CF)	DCD Data Carrier Detect	DTR P1-pin 4
	2 (BB)	RxD Receive Data	TxD of MAX561
	3 (BA)	TxD Transmit Data	RxD of MAX561
	4 (CD)	DTR Data Terminal Ready	INIT
	5 (AB)	SG Signal Ground	Digital Ground
	6 (CC)	DSR Data Set Ready	DTR P1-pin4
	7 (CA)	RTS Request To Send	CTS P1-pin8
	8 (CB)	CTS Clear To Send	RTS P1-pin7
	9 (CE)	RI Ring Indicator	Run Indicator

Connector P2 is an additional serial port for user applications. Receive, Transmit, and Clear-to-Send are connected. Other connector pins are routed to test points on the board.

Table 3-4. P2 Serial Channel 0

P2 Connector	Pin Nos.	Host RS-232 Signal Name	Connection on Evaluation Board
	1 (CF)	DCD Data Carrier Detect	Test Point 1
	2 (BB)	RxD Receive Data	RxD
	3 (BA)	TxD Transmit Data	TxD
	4 (CD)	DTR Data Terminal Ready	Test Point 4
	5 (AB)	SG Signal Ground	Digital Ground
	6 (AB)	DSR Data Set Ready	Test Point 2
	7 (CD)	RTS Request To Send	Test Point 3
	8 (BA)	CTS Clear To Send	CTS
	9 (BB)	RI Ring Indicator	Test Point 5

The two serial connectors are connected to the Maxim MAX561, an EIA/TIA-562 Driver/Receiver. This device operates from a 3.3 volt V_{CC} (or 5 volts, optionally). The EIA/TIA-562 standard is a low voltage serial communications protocol. This protocol operates at ±3.7 volts. The 3.3 volt signals from the board are charge-pumped to ±6.6 volt levels internally, conforming to this standard. Signals from the serial connectors, P1 and P2, are translated to a 3.3 volt level. Output from this device is recognized by EIA/TIA-232-D receivers, and inputs can handle EIA/TIA-232-D levels without damaging the device. The MAX561 SHDN pin (pin 25) connects to port pin 1.0 on the 80x186 processor. When this pin is programmed to a logic 1, the Maxim device will go into shutdown mode, reducing current consumption to leakage. During initialization, port pin 1.0 is programmed to a logic 0 to enable communication with the host PC.

Serial communications on the evaluation board are controlled by the 80x186 processor on-chip serial ports. Serial Port 0 on the microprocessor handles PC communications via connector P1. Serial Port 1 is available for user applications via connector P2. The 80x186 processor supports synchronous serial communications as well as various modes of asynchronous communications. The time base for the host interface is a 6.0 MHz oscillator connected to BCLK0, the external serial clock input on the 80x186 processor. This allows the user to change the processor operating frequency without altering the baud rate.

NOTE

The BCLK0 input must be less than half the processor operating frequency (which is half the clock input frequency). Operating the processor below 12.288 MHz requires reprogramming the serial control unit on the 80x186 processor. The source code for the RISM monitor is provided on a floppy diskette included in your kit for this purpose.

Figure 3-5 on page 3-11 illustrates the adaptor cable needed if your PC has a 25-pin serial port connector.

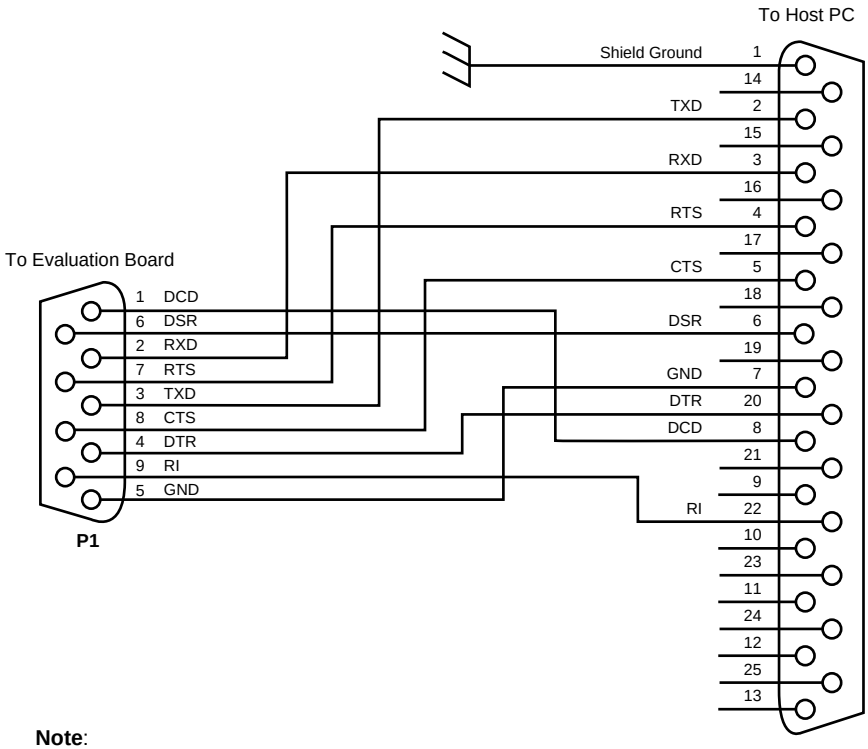


Figure 3-5. 25-Pin to 9-Pin Adaptor

3.7 EXPANSION INTERFACE

There are two expansion connectors on the evaluation board. Refer to the schematics included on a floppy diskette in your kit for representation of the connector pinouts. The 60-pin JP1 connector (Figure 3-7) provides latched address pins and the address/data bus signals. This connector also provides access to all bus-control signals, programmable chip-selects, +3.3 volts, +5 volts, and +12 volts. The JP2 connector provides access to on-chip peripherals of the 80x186 processor. This connector allows access to interrupt inputs, timer inputs and outputs, port pins, CLKOUT, RESOUT, +3.3 volts, +5 volts, and +12 volts. The JP2 connector contains 40 pins for the EC processors (see Figure 3-6) and 24 pins for the EB processors (see Figure 3-7).

NOTE

3.3 volts is available on the connector only when jumper E1 selects V_{CC} = 3.3 volts; otherwise, these pins are 5 volts. +12 volts is available on the connector only when Port Pin 1.1 is programmed to a logic 0; otherwise, these pins are V_{CC} minus a diode drop.

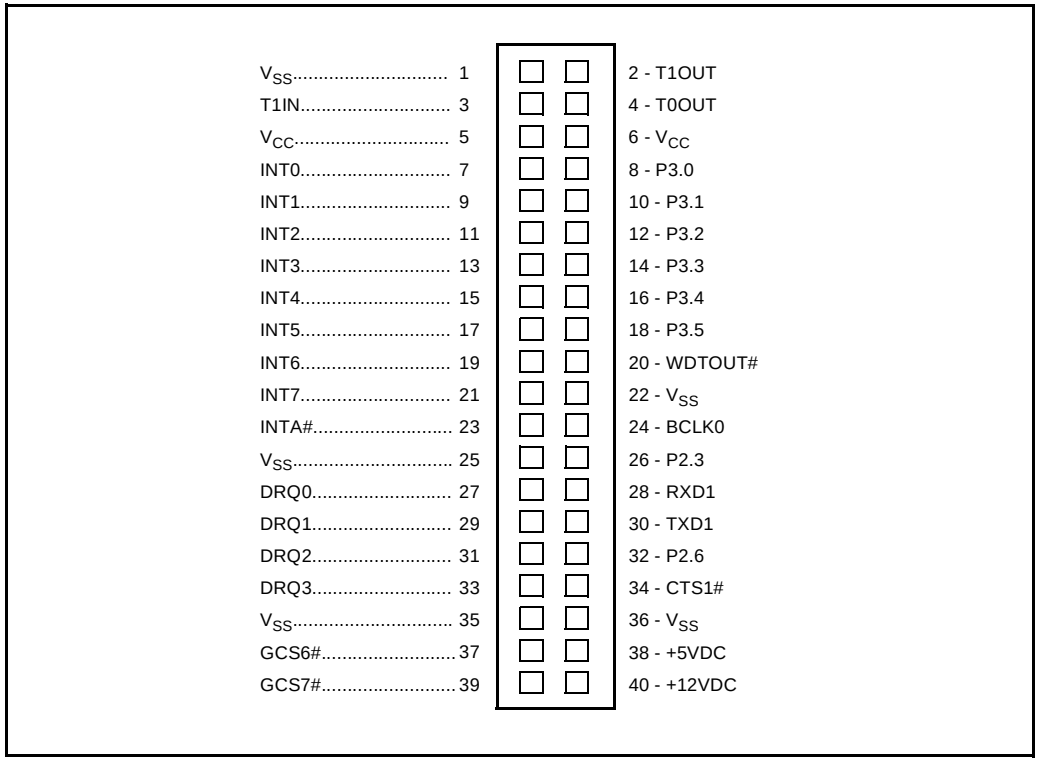


Figure 3-6. 186 EC Peripheral Expansion Connector JP2 (40 pin)

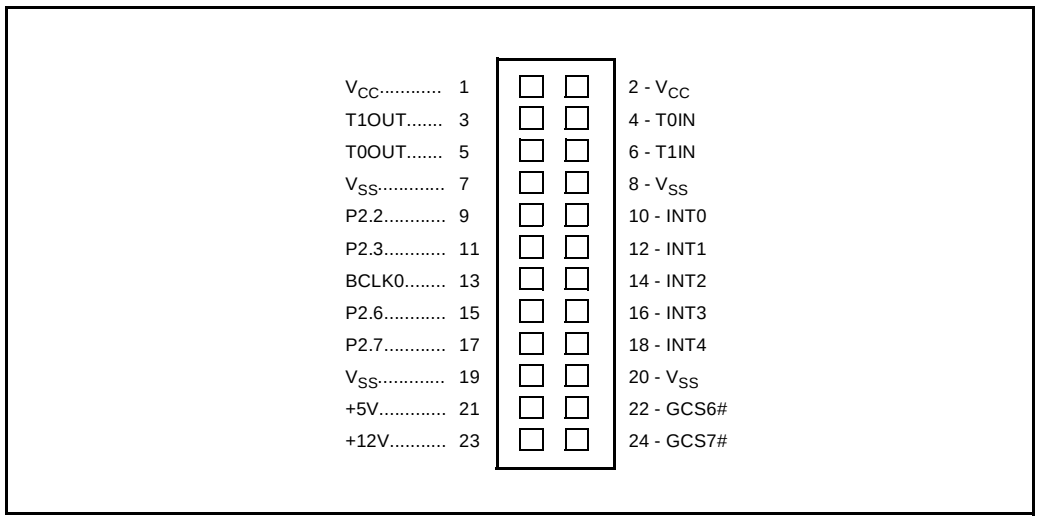


Figure 3-7. 186 EB Peripheral Expansion Connector JP2 (24 pin)

JP1 Memory - I/O Expansion Connector
2x30 Pin Molex* 39-51-6004 or Equivalent

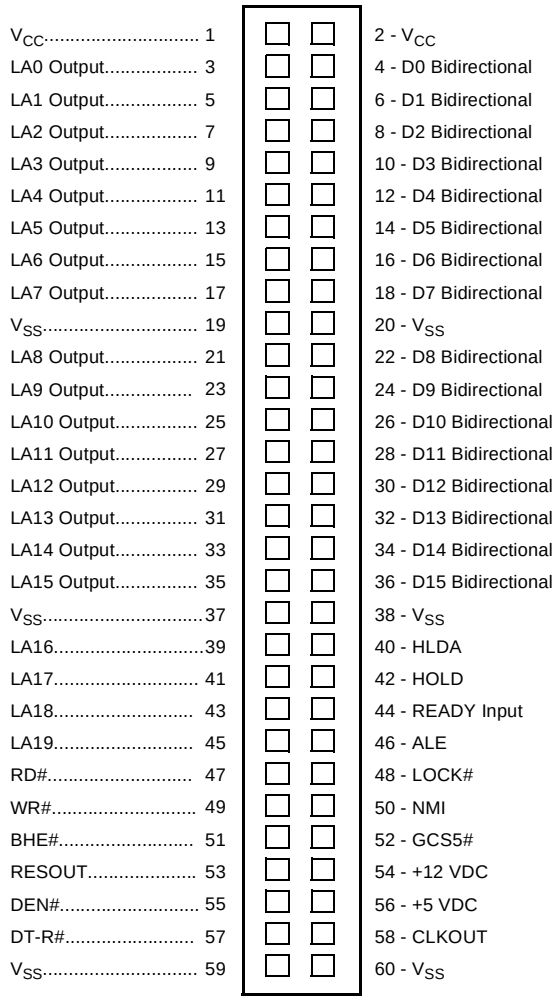


Figure 3-8. CPU Bus Expansion (EB and EC)

3.8 LCD INTERFACE

The evaluation board includes a 16-character by 1-line LCD display. The display has an 8-bit interface and is designed to operate at up to 20 MHz. The display includes a Hitachi* 44780 LCD display controller that takes care of functions such as character interpretation and display refresh.

The display is write-only. This is because the display controller operates at 5 volts V_{CC}. A 5-volt part driving a 3.3-volt bus can damage parts operating at 3.3 volts V_{CC}. This means that the BUSY pin of the processor cannot be monitored to determine when the processor is ready for the next command, so a delay loop must be used to allow the display to finish commands.

Signals from the 80x186 processor can be connected directly to the LCD controller inputs, regardless of V_{CC}, because 3.3 volt and 5 volt outputs are compatible with 5 volt TTL level inputs.

The LCD display is mapped in I/O space at 400H to 440H. All command and data writes to the display are to this address. Port pin 1.4 is used to control which LCD register is accessed. P1.4 = 0 accesses the command register; P1.4 = 1 accesses the data register.

3.8.1 LCD Interface Demo

The diskette provided in your kit includes a file, LCD_DEMO.ASM, that contains source code you can assemble and load onto the board (using iECM). You can execute the program for a demonstration of the basic principals of operating the LCD display module. This program prints a static message to the display. The source code is commented to serve as a tutorial and can be adapted as needed for other applications and messages. Note that although the LCD module is capable of displaying standard ASCII (characters 32 through 125) or custom characters, this demo uses only ASCII characters.

For more information regarding the operation of the display controller, please refer to the Hitachi *LCD Controller/Driver LSI Data Book*.

4

Introduction to the Software

CHAPTER 4
INTRODUCTION TO THE SOFTWARE

The Intel 186 EC/EB Evaluation Board uses an Embedded Controller Monitor (ECM) written for the 80x186 family of 16-bit microprocessors. This monitor supports basic debug facilities (LOAD, GO, STEP, etc.) in the user's target system. The ECM is broken into two independent programs. One of these (iRISM-186) executes in the evaluation board and the other (iECM-86) executes in an IBM PC or BIOS-compatible computer. These two programs communicate via an asynchronous serial channel using a binary protocol defined specifically for this application.

The partitioning of the ECM into two separate programs supports a number of goals:

- The system is easy to adapt to a new target because the code that runs in the target is very simple and small.
- The feature set of the user interface is not limited by the resources of the target, since the user interface is implemented in the host PC.
- Concurrent operation of the ECM and the target system is easily achieved. This allows you to interrogate and (carefully) modify the state of the target system while it is running.

This chapter describes the user interface provided by the iECM-86, the interface between this PC-resident software and the target-resident software, and the structure of the software in the target. The board uses the internal 80x186 EB/EC serial port for host communications.

The iECM-86 software was created by Intel to support users of the 80x186 architecture and is placed in the public domain with no restrictions or warranties of any kind.

4.1 SOFTWARE FEATURES

The iECM-86 software has the following features:

- Sixteen software execution breakpoints
- Concurrent interrogation of target memory and registers
- Supports BYTE, CHARACTER, WORD, STRING, DOUBLE WORD, and REAL variable types
- Supports LOAD, SAVE, LIST, LOG, and command INCLUDE files

4.2 RESTRICTIONS

Two words of the user stack are reserved for use by the iRISM-186 software. Other memory and/or registers in the target memory are used by the iRISM-186 software. The exact amount and location of this memory is implementation-dependent.

An asynchronous serial port capable of operation at 9600 baud must be available in the target system. The RISM described in this document uses the 80x186 EB/EC internal serial port.

The TRAP instruction is reserved.

Breakpoints and program stepping will not operate if the user's code is in Flash or other nonchangeable memory.

4.3 EMBEDDED CONTROLLER MONITOR (ECM)

An ECM (Embedded Controller Monitor) is installed in your target system to provide basic debug capability. Capabilities include loading object files into system RAM, examining and modifying variables, executing code, and stepping through code. A personal computer acts as the host for program translation and emulates a video display during user interaction with the ECM. The ECM developed for the 80x186 family makes the assumption that the user interface is a personal computer; no provision is made for interface to a CRT terminal. By making this assumption, it is possible to reduce the size and complexity of the code that must be installed in the target system. The term coined for this target-resident code is Reduced Instruction Set Monitor (RISM).

The RISM consists of about 2200 bytes of 80x186 code that provides primitive operations. Software running in the host uses the RISM commands to provide a complete user interface to the target system. The advantage of this approach is that the ECM can be readily adapted to different target systems and requires only a small part of the available target memory space. The disadvantage is that the user interface must be provided by a personal computer.

RISM is structured as a short section of initialization code and an interrupt service routine (ISR). The ISR processes interrupts from the host system. The RISM ISR consists of a short prologue and a case-jump to one of 20 to 25 command executors. These executors are simple and short; the flow though the entire ISR (including the prologue) is 15-20 instructions. The serial communication occurs at 9600 baud, which limits the frequency of these interrupts to 1 KHz. In the worst case, the board will be slowed by the execution of a fairly short RISM ISR every millisecond while executing user code. It is possible to operate the board so that no real time is lost to the iECM-86 unless the user is actively interrogating the target. See "Initiating and Terminating iECM-86" on page 4-3 and the description of the RISM REPORT_STATUS code (Code 14H) on page 6-5 for details.

4.4 USER INTERFACE

The user interface to the iECM-86 supports commands to initiate and configure the ECM-86, perform I/O operations involving DOS files, execute user programs, and interrogate variables in the target system. Interrogation can be done in a number of formats and in most cases can be done concurrently with user code execution.

4.4.1 Numeric Input

The command parser used by the iECM-86 software requires that numeric inputs always start with the digits 0-9. Hexadecimal numbers that start with A–F must be preceded by a zero. For example, enter “0AA55” instead of “AA55.” This requirement is similar to that of ASM86.

4.4.2 Controlling Lengthy Commands

Most of the commands supported by iECM-86 appear to complete without delay. Some commands (for example, displaying or filling a large area of memory) take an appreciable length of time to complete. In general, these commands can be aborted by pressing Enter. Those commands that display a large amount of information can be paused by pressing the spacebar. After you have checked the data on the screen, you can press the spacebar again to resume the output.

4.4.3 Aborting from iECM-86

Press Ctrl+C to close any open files and return to DOS.

4.5 INITIATING AND TERMINATING iECM-86

This section describes the commands for invoking iECM-86 from DOS and exiting back to DOS.

4.5.1 ECM86

This command, entered at the DOS prompt, loads the iECM-86 software and executes it. Several options are available with this command. Option strings always start with a hyphen (-) and can be entered in upper or lower case. The operation of these options is described below. Any or all of these options can be entered in any order. If the options are contradictory, the actual option accepted is the last one entered.

4.5.2 -COM2, -COM1

These options tell the iECM-86 software which serial communication port is to be used. If neither option is entered, COM1 is used as a default. If iECM-86 detects valid CTS (Clear to Send) and DSR (Data Set Ready) signals from the appropriate COM port, it signs on and displays a command prompt. When the target is stopped, the command prompt is an asterisk (*). When the target is already running, the prompt is a greater-than sign (>).

4.5.3 -DIAG

If CTS and DSR are not present, iECM-86 displays a warning message. You can choose to proceed or exit. It is possible, but not likely, that iECM-86 will operate properly even after the warning. It is more likely that there is a problem with the serial port or the cabling that prevents proper operation.

If the problem is not obvious, such as a disconnected cable or no power to the target hardware, use the -DIAG invocation option to help isolate the problem. The -DIAG option puts the iECM-86 system in a special mode that allows many tests to be used to find interfacing problems or target bugs.

The diagnostic mode is intended to support debugging of boards that use iECM-86 software. It also provides a simple routine to check the communications interface between the host and the target.

In the board, a serial port loop-back mode allows debugging the host/board interface. Upon reset, the board is in the echo mode. Until it receives an ASCII slash (/) or backslash (\), it increments every character it receives from the host and sends the incremented value back to the host. The LCD displays the word "**DIAGNOSTICS**" when the board is in echo mode. If a backslash is received by the RISM, the board leaves echo mode and starts normal operation. When a slash is received, the board stops echoing incremented received data and starts responding to RISM commands with the diagnostic flag set.

NOTE

The target hardware has to be reset before using the -DIAG option. When executing diagnostic routines from Flash, certain commands such as breakpoints and stepping will not work because they need to modify the code to work properly.

When the host software is invoked in the diagnostic mode, it prompts you to enter characters on the keyboard. These characters are sent to the target, and the response from the target is displayed on screen. This is a simple confidence check on the serial communication channel. You are told to enter a slash or backslash to terminate this mode and proceed in either the diagnostic mode or the normal user's mode. If the user interface is invoked without the -DIAG option, the software immediately transmits a reverse-slash, which should put the target in the normal mode.

4.5.4 -POLL, -SIGNAL

These two options control how the host software detects whether or not the user's code is running. If poll mode is selected, the host periodically polls the target with a REPORT_STATUS command. This takes no additional hardware, but it forces the target to spend instruction cycles responding to the poll. The signal mode avoids this overhead, but it requires that the target set the Ring Indicator modem line before it issues a REPORT_STATUS command. If neither option is selected, the signal mode is selected as a default. On the board, the P1.3 pin of 80x186 processor is used to generate this running signal. Therefore, the signal mode is recommended. (The REPORT_STATUS command is described on page 6-5.)

4.5.5 RESET SYSTEM, RES SYSTEM, RESET, RES

This command and its abbreviations reset the entire target hardware system. This command operates by dropping the DTR modem control line. This comes into the target as DSR. After dropping DTR, the iECM-86 software waits about 1 second to allow the target to complete its initialization routines. The iECM-86 warns of this time delay and then ignores input from the host PC until it expires. Unless special precautions are taken in the design of a target system, any data in RAM (including downloaded object code) may be corrupted by the reset. On the board, the RAM contents should not be affected by a reset.

4.5.6 DOS

This command enables you to temporarily leave iECM-86 and return to DOS. Once you have suspended iECM-86, you may perform other functions in DOS, including using other software programs such as ASM86, as long as there is sufficient memory to do so.

To re-enter iECM-86, type **exit** at the DOS prompt. iECM-86 returns with all conditions that were in effect at the time it was suspended.

4.5.7 QUIT

This command closes any files that iECM-86 has opened and exits to DOS. Note that this command can be used even if the target is running. iECM-86 sets the selected COM port to 9600 baud, 8 bits, no parity and one stop bit. The port is left in this state by iECM-86 when control is returned to DOS.

4.6 RELATED INFORMATION

All unreserved functions of the processor are available to you, except the Non-Maskable Interrupt (NMI), the Breakpoint instruction (INT 3), the Trap Flag (TF), 16 Kbytes of address space, and 128 bytes of I/O space.

4.6.1 Reserved Functions

The Trap Flag and its vector in memory locations 4H–7H are reserved for use by the SSTEP command and BREAKPOINTS.

The NMI pin and its vector in memory locations 8H–0BH are reserved for use by the host interface.

The INT 3 instruction and its vector in memory locations 0CH–0FH are reserved for use by the SSTEP command and BREAKPOINTS.

4.6.2 Reserved Memory

On-board Flash memory, as shipped, is 32 Kbytes from address 0H to 7FFFH.

Addresses 0H–3FFH are the interrupt vectors for the processor.

You must not alter the interrupt vectors from 4H–0FH.

Memory locations 400H–415H are reserved for use by the RISM monitor code. You must ensure that no locations in this partition are used by code that is to operate with the RISM. The easiest way of doing this is to generate an ASM-86 module that declares a DATA SEGMENT at 400H that is 22 bytes long. This module can then be linked into the final program to prevent the linker from assigning these registers to another module.

Fourteen words of user stack space must be reserved for use by the iRISM-186 software while the board is processing a host interrupt. The CS:SP register pair is initialized by RISM to 0000H:0800H, providing a total stack size of 501 words before RISM data variables are overwritten. If this is insufficient for your application, your code should alter the SP to a large enough value. Normally, you should write your code to begin at address 800H and download it to Flash memory using iECM-86. You should use any space left beneath your code as data memory.

4.6.3 Reserved I/O

The I/O space from 400H–47FH is reserved for use by the host interface.

5

iECM-86 Commands

CHAPTER 5
iECM-86 COMMANDS

This chapter defines the iECM-86 software commands.

5.1 ENTERING COMMANDS

The syntax for iECM commands is shown below:

COMMAND *metasymbol*

iECM-86 command definitions use one or more of the following metasymbols:

addr
address iECM-86 is able to interpret the microprocessor's address space as either a flat 20-bit array or through segmentation. A location anywhere within the 1 Mbyte memory range may be specified by its complete physical address, such as 0F1AC9H.

segment*:*offset Memory may also be accessed by segments. Valid segment references are the following (where segment and offset are valid integers):

CS:*offset*
DS:*offset*
ES:*offset*
SS:*offset*

When using CS, DS, ES or SS, the full address is calculated using the actual value of the appropriate target processor segment register.

In addition to the above registers, iECM-86 maintains four user-definable registers that may be used for segment variables:

CB:*offset*
DB:*offset*
EB:*offset*
SB:*offset*

This facility is useful when reading from assembler listings, which are typically offset from 0000. These base registers are used, for example, as a base pointer to a block of memory for debug purposes. CB could be loaded with the base address of a code module, then breakpoints could be set using offsets from that base. Using these internal iECM registers has no effect on the values of the target processor's registers.

bp_number	Sixteen breakpoints are available to the user. This number selects which breakpoint to access.
code_addr	The code address may be specified by either *segment: offset*, *CS:offset*, or *CB:offset*.
count	This denotes the number of times a command executes.
filename	This is the location (path) and name of the file you want to reference (e.g., \progdir\program.obj).
value	Data to be entered in the current base notation.

5.2 FILE OPERATIONS

iECM-86 uses files in the host system to load and save object code, to enter predefined strings of commands, to keep a log of commands that are entered by the user, and to keep a record of an entire debug session that includes both the characters entered by the user and the responses generated by iECM-86 on the host screen. The commands that operate with files are described in the following sections.

5.2.1 Loading and Saving Object Code

iECM-86 accepts object files that are generated by Intel's development tools. iECM-86 will not accept files that contain unresolved externals or files that contain re-locatable records. These files must be passed through LINK86 and/or LOC86 to resolve the externals and/or absolutely locate the re-locatable segments. iECM-86 will also not accept HEX format files. The iECM-86 commands that operate on object files are the following:

LOAD *filename*

SAVE *addr* TO *addr* IN *filename*

The metasymbol *filename* means that a valid MS-DOS file name must be entered in that position of the command string.

LOAD *filename*	This command loads the content records of the object file *filename* into the target memory.
SAVE *addr* TO *addr* IN *filename*	This command saves a region of memory as an object file that can be reloaded into the target memory at some later time.

5.2.2 Other File Operations

In addition to object files, the iECM-86 makes use of include files, log files, and list files. Include files contain commands to be executed by iECM-86. They must contain the exact sequence of ASCII characters that you would enter from the keyboard to execute the command. Include files can be tedious to create with a text editor, so iECM-86 can generate log files that store characters entered by the user. These log files may be used later as include files to recreate command sequences. List files keep a running record of commands entered by the user and the responses generated by iECM-86. Comments can be included in list and log files to make them easier to understand. A comment starts with a semicolon (;) and ends with a carriage return or ESC. The semicolon is considered part of the comment, but the carriage return or ESC is not. The command parser ignores comments but puts them in the list and log files.

The list and log file commands allow for default file names and allow either overwriting existing data in the file or appending data at the end of the file. This allows you to gather list and log data in the default files, which avoids creating and managing a large number of separate files. Log and list files are stamped with the date and time whenever they are opened to facilitate using this capability, then going back to sort out the data from several debug sessions with a text editor.

The following commands are used in include, log, and list operations.

INCLUDE *filename*

PAUSE

LIST

LIST *filename*

LOG

LOG *filename*

LISTOFF

LISTON

LOGOFF

LOGON

Three of these commands require you to supply a valid file name; the rest use the appropriate file name that has already been entered.

INCLUDE *filename* This command attempts to open *filename* as a read-only file. If the file can be opened, the command parser takes commands from that file until the end of the file is reached. The INCLUDE file is then closed. Only one INCLUDE file is opened at a time.

PAUSE This command is documented in this section because it is intended to be used as part of INCLUDE files. It is not really a file-oriented command itself. When this command is entered, the iECM-86 stops

parsing commands until a space character is entered from the keyboard (the space character can't come from an INCLUDE file). This allows the user to pause in the middle of an INCLUDE file operation to see what is occurring and then acknowledge the pause condition by pressing the space bar.

LIST This command behaves like the LIST *filename* command described below, except that it uses the last file name that was entered as part of a LIST *filename* command. If no such command has been entered, the default filename LIST.ECM is used.

LIST *filename* This command attempts to open *filename* as a writable file. If *filename* already exists, then iECM-86 asks if the file is to be overwritten or if the new data should be appended to the existing file. It then opens the file and stamps it with the current date and time from the system clock. Subsequent commands entered by the user and the responses generated by iECM-86 are recorded in the file.

LOG This command behaves like the LOG *filename* command described below, except that it uses the last file name that was entered as part of a LOG *filename* command. If no such command has been entered, the default file name LOG.ECM is used.

LOG *filename* This command attempts to open *filename* as a writable file. If *filename* already exists, iECM-86 asks if the file is to be overwritten or if the new data should be appended to the file. It then opens the file and stamps it with the current date and time. Subsequent commands entered by the user are recorded in the file. Note that this file may contain nonprintable characters (e.g., ESC).

LISTOFF
LISTON The LISTOFF command closes a list file that has been specified by the LIST command. This stops new list information from being recorded. The LISTON command reopens the list file in the append mode so that recording can start again. LISTON also stamps the list file with the current date and time from the system clock.

LOGOFF
LOGON The LOGOFF command closes a log file that has been specified by the LOG command. This stops new log information from being recorded. The LOGON command reopens the log file in the append mode so that recording can start again. LOGON also stamps the list file with the current date and time from the system clock.

5.3 PROGRAM CONTROL

Commands that control program execution allow you to reset the processor, set execution breakpoints, start execution, stop execution, step, and super step. The commands are grouped by their major functions for the sake of discussion.

5.3.1 Resetting the Target

The processor can be reset by executing the following iECM-86 command:

RESET CHIP

RES CHIP

This command physically resets the processor by setting the RISM_DATA register to 0XXXX0001 and issuing a MONITOR_ESC RISM command, which causes the target to perform a JMP FFFF:0000H instruction.

RESET SYSTEM

RES SYSTEM

RES

This command resets the entire iECM-86 system, including the target. It operates by bringing the DCD line of the serial port low. This, with appropriate circuitry in the target system, resets the target processor. During this process, the iECM-86 software must wait about one second to allow the main board to complete its initialization routines. The iECM-86 warns of this time delay and then ignores the user until it expires. Any user code in the Flash must be reloaded after this command.

5.3.2 Breakpoints

iECM-86 provides sixteen program execution breakpoints. If a given breakpoint is inactive, it is set to zero; if it is active, it is set to the address of the first byte of an instruction. Breakpoints set to addresses that are not the first byte of an instruction cause unpredictable errors in the execution of the user's code. When execution is started, iECM-86 saves the user code byte at any active breakpoint and substitutes an INT3 instruction for that byte. Executing an INT3 instruction causes the iECM-86 to restore the user code bytes where the INT instructions were substituted and then decrement the user's program counter so that it points to the original instruction. The user's program appears to stop execution immediately before executing the instruction with a breakpoint set on it. All INT instructions are removed from the user's code and the original code is restored.

NOTE

Most monitor programs similar to iECM-86 display a message on the console when a break occurs (e.g., "Program break at 1234H"). This is not done in iECM-86 because the system supports concurrent interrogation of the target on which the user's code is running; it is possible that the break will occur while you are in the middle of displaying or modifying the state of the target. Any special break message would have to interrupt the execution of the command. Because of this, the iECM-86 does not output a special break message. You have two ways to find out that a break occurred:

- The prompt changes from a greater-than sign (>) to an asterisk (*).
- The status of the processor shown in the "control panel" at the top of the console screen changes from "running" to "stopped."

Commands which set the breakpoint array are:

BR

BR [*bp_number*]

BR [*bp_number*] = *code_addr*

The square brackets in the latter two commands are part of the command syntax and must be entered by the user; the angle brackets are part of the "meta" language used to describe the syntax. Breakpoints can be displayed while your code is running, but they cannot be modified.

NOTE

BR[0] and BR[1] can also be set by the GO command by using the TILL clause; all breakpoints are cleared by the GO command if the FOREVER clause is used.

BR — This command displays all of the active breakpoints (i.e., those not set to zero). You are also informed if no breakpoints are active.

BR [*bp_number*] — This command displays the setting of the selected breakpoint and waits for input from you. If you enter a carriage return, the command terminates. If you enter an ESC, the next sequential breakpoint is displayed. If you enter a numeric value, the selected breakpoint is loaded with the value and the iECM-86 again waits for input. At this point, you can enter either a carriage return or an ESC. As before, the ESC causes the iECM-86 to display the next breakpoint and the carriage return terminates the command. This command wraps around from the last breakpoint (15t) to the first breakpoint (0).

BR [*bp_number*] = *code_addr* — This command sets the specific breakpoint specified by *bp_number* to the value *code_addr*.

5.3.3 Program Execution

These commands start and stop execution of user code. The commands provided are:

GO

GO FOREVER

GO FROM *code_addr*

GO FROM *code_addr* FOREVER

GO FROM *code_addr* TILL *code_addr*

GO FROM *code_addr* TILL *code_addr* OR *code_addr*

GO TILL *code_addr*

GO TILL *code_addr* OR *code_addr*

HALT

If a GO with breakpoint command is entered, the user code bytes at the breakpoints are saved and INT3s are substituted. When a breakpoint is reached, the user's software stops **before** the instruction that caused the breakpoint and the iECM-86 software restores the original user code. Note that this differs from the operation of most ICE modules, which stop just **after** the instruction executes. A problem associated with stopping before the break instruction executes is that subsequent GO commands may run into the breakpoint before any user code is executed. The iECM-86 avoids this problem by skipping the setting of any breakpoints set on the instruction that the current PC points to. If this happens to remove the last breakpoint set, you are warned, but the GO still executes with no breakpoints enabled. If this happens, you can use the HALT command to stop the program.

None of the GO commands can be executed while the user's code is already running; the HALT command cannot be executed if the user's code is not running. The GO commands that set breakpoints use BR[0] and possibly BR[1]. Any break value already in one of these breakpoints is overwritten and destroyed by these GO commands. If possible, the user should reserve the first two breakpoints for use by the GO commands, and set the remaining breakpoints (if required) explicitly with the BR commands.

GO — This command starts execution of the user's code using the current value of user's program counter (PC) and the current breakpoint array.

GO FOREVER — This command clears the breakpoint array and starts execution at the current value of the user's PC.

GO FROM *code_addr* — This command loads the user's PC with *code_addr* and starts execution of the user's code using the current breakpoint array.

GO FROM *code_addr* FOREVER

This command loads the user's PC with *code_addr*, clears the breakpoint array, and starts execution of the user's code.

GO FROM *code_addr* TILL *code_addr*

This command loads the user's PC with the *code_addr* that follows the FROM keyword, sets the first breakpoint (BR[0]) to the *code_addr* that follows the TILL keyword, and starts execution of the user's code.

GO FROM *code_addr* TILL *code_addr* OR *code_addr*

This command acts like the previous command, except that it also sets the second breakpoint (BR[1]) to the *code_addr* that follows the OR keyword.

GO TILL *code_addr* This command sets the first breakpoint (BR[0]) to *code_addr* and starts the execution of user code using the current setting of the user's PC and the breakpoint array.

GO TILL *code_addr* OR *code_addr*

This command acts like the previous command, except that it also sets the second breakpoint (BR[1]) to the *code_addr* that follows the OR keyword.

HALT This command stops execution of user code by forcing the processor to execute a jump to self instruction in a reserved location.

5.3.4 Program Stepping

These commands allow stepping through programs one instruction at a time. Between instructions, the iECM-86 commands can be used to check the state of the variables changed by the instruction, to ensure that the program is operating properly. Stepping through code allows a more detailed look at what is going on in the program. The price paid for this detail is that stepping does not occur in real time; this makes it difficult, or perhaps impossible, to use on code that is tied to real-time events.

Stepping while interrupts are enabled would be confusing, since interrupt service routines would be stepped through as well as sequential code. iECM-86 avoids this problem by artificially locking out interrupts while stepping, ignoring the state of the interrupt enable (IF) or interrupt mask.

Super-stepping is similar to stepping, except that the super-step command treats an interrupt service routine or a subroutine call (and the body of the subroutine that is called) as one indivisible instruction. This allows the user to ignore the details of subroutines and interrupt service routines while evaluating code. This may allow limited stepping through code while operating in a concurrent environment, but the system will not operate in real time. A better approach is to use the GO command to execute to a specified breakpoint and then step through the code being tested, looking for proper operation.

iECM-86 implements the step operation by using the trap flag (TF). To step over a given instruction, iECM-86 sets the trap flag to put the processor into single-step mode. In this mode, the CPU automatically generates an internal interrupt after each instruction, allowing a program to be inspected as it executes. After the processor receives this trap interrupt, it restores all of the user flags overwritten by the iECM flags.

Super-stepping is also accomplished by setting the trap flag, except for CALL instructions, which are treated as a special case. During a STEP, the iECM-86 sets the trap flag; during a super-step an INT3 is placed at the instruction following the CALL. Interrupts are suppressed during STEP operations by saving the user's IF bit, clearing it before the STEP occurs, and then restoring it. During a GO or SSTEP command, all instructions are executed by the target.

The iECM-86 commands that implement step operations are the following:

STEP

STEP *count*

STEP FROM *code_addr*

STEP FROM *code_addr count*

SSTEP

SSTEP *count*

SSTEP FROM *code_addr*

SSTEP FROM *code_addr count*

Aside from the style of the actual step operation, the SSTEP and STEP commands behave the same. They are called single-stepping commands are described as follows.

{STEP | SSTEP} This command single-steps one time.

{STEP | SSTEP} *count* This command single-steps *count* times.

{STEP | SSTEP} FROM *code_addr*
This command loads the user's program counter (PC) with *code_addr* and then single-steps one time.

{STEP | SSTEP} FROM *code_addr count*

This command loads the user's program counter (PC) with *code_addr* and then single-steps *count* times.

5.4 DISPLAYING AND MODIFYING PROGRAM VARIABLES

iECM-86 provides commands to display and modify program variables in several formats. In addition to simple variables such as bytes and words, more complicated variables such as reals and character strings are supported. iECM-86 commands allow variables to be displayed or initialized either individually or as regions of memory that contain variables of the given type.

5.4.1 Supported Data Types

Table 5-1. Supported Data Types

Data Type	Description
BYTE	A BYTE is an eight-bit variable. No alignment rules are enforced for BYTE variables.
CHAR	A CHAR is a special case of a BYTE. CHAR variables are displayed as ASCII characters.
WORD	A WORD is a 16-bit variable. The address of a WORD is the address of its least significant byte.
DWORD	A DWORD is a 32-bit variable. The address of a DWORD is the address of its least significant byte.
STACK	A STACK variable is a 16-bit variable that resides in the system stack. The address of a stack variable (*stack_addr*) is taken to be relative to the current stack pointer
STRING	A STRING is a sequence of ASCII characters that are terminated by the NUL character. The ASCII character NUL has the binary value of zero.
PORT	A PORT is an 8-bit I/O port. No alignment rules are enforced for PORTs.
WPORT	A WPORT is a 16-bit I/O port. The address of a WPORT is the address of its least significant byte. A WPORT must start at an even address.

In addition to supporting access to variables of the above types, iECM-86 also provides commands to access the microprocessor registers and other special program variables such as PC (program counter), and SP (stack pointer). These commands include AX, AH, AL, BX, BH, BL, CX, CH, and CL.

5.4.2 BYTE Commands

There are four forms for the BYTE commands:

BYTE *byte_address*

BYTE *byte_address* = *byte_value*

BYTE *byte_address* TO *byte_address*

BYTE *byte_address* TO *byte_address* = *byte_value*

All of these commands can be used whether or not the user's program is running.

BYTE *byte_address* This form is used to examine and then possibly change one or more sequential BYTE variables. When this command is invoked, iECM-86 displays the *byte_address* in hexadecimal notation and the value of the BYTE in the default base, then waits for an input from you. You can respond with a carriage return character, an ESC character, or a numeric value. A carriage return terminates the command. An ESC results in the display of the next sequential BYTE variable. If a numeric value is entered, the BYTE variable is set to this value and the iECM-86 again waits for input. At this point, you can respond only with an ESC or carriage return. As before, the ESC displays the next sequential BYTE and the carriage return terminates the command.

BYTE *byte_address* = *byte_value*

This form is used to set an individual BYTE variable without first checking its current value. When invoked, this command sets the BYTE variable at *byte_address* to *byte_value.*

BYTE *byte_address* TO *byte_address*

This form is used to display a region of memory as a sequence of BYTE variables. When this command is invoked, iECM-86 starts by displaying the current default base and then a series of lines showing the contents of the selected memory region. The next line starts with a hexadecimal display of the address of the next BYTE variable to be displayed, followed by the display of up to 16 bytes of memory as BYTE variables in the default base. A new line start whenever 16 bytes of memory have been displayed on the line. The command terminates when all of the BYTE variables in the selected range have been displayed. During lengthy displays, you can stop the output to the console by pressing the space bar. You can resume the display by pressing the space bar a second time. You terminate the command by entering a carriage return.

BYTE *byte_address* TO *byte_address* = *byte_value*

This form is used to initialize a region of memory to the given *byte_value*. Note that this command takes a little over a millisecond (at 9600 baud) for each BYTE loaded. You can terminate this command by entering a carriage return, but terminating the command leaves only part of the memory region initialized.

5.4.3 WORD Commands

There are four basic forms for the WORD commands:

WORD *word_address*

WORD *word_address* = *word_value*

WORD *word_address* TO *word_address*

WORD *word_address* TO *word_address* = *word_value*

All of these commands can be used whether or not the user's program is running.

WORD *word_address* This form is used to examine and then possibly change one or more sequential WORD variables. When this command is invoked, iECM-86 displays the *word_address* in hexadecimal notation and the value of the WORD in the default base, then waits for an input from you. You can respond with a carriage return, an ESC, or a numeric value. A carriage return terminates the command. An ESC results in the display of the next sequential WORD variable. If a numeric value is entered, the WORD variable is set to this value and the iECM-86 again waits for input. At this point you can respond only with an ESC or carriage return. As before, the ESC displays the next sequential WORD and the carriage return terminates the command.

WORD *word_address* = *word_value*

This form is used to set an individual WORD variable without first checking its current value. When invoked, this command sets the WORD variable at *word_address* to *word_value*.

WORD *word_address* TO *word_address*

This form is used to display a region of memory as a sequence of WORD variables. When this command is invoked, iECM-86 starts by displaying the current default base and then a series of lines showing the contents of the selected memory region. The next line starts with a hexadecimal display of the address of the next WORD variable to be displayed, followed by the display of up to 16 bytes of memory as WORD variables in the default base. A new line starts whenever 16 bytes of memory have been displayed on the line. The

command terminates when all of the WORD variables in the selected range have been displayed. During lengthy displays, you can stop the output to the console by pressing the space bar. You can resume the display by pressing the space bar a second time. You terminate the command by entering a carriage return.

WORD *word_address* TO *word_address* = *word_value*

This form is used to initialize a region of memory to the given *word_value*. Note that this command takes a little over a millisecond (at 9600 baud) for each WORD loaded. You can terminate this command by entering a carriage return, but terminating the command leaves only part of the memory region initialized.

5.4.4 DWORD Commands

There are four basic forms for the DWORD commands:

DWORD *dword_address*

DWORD *dword_address* = *dword_value*

DWORD *dword_address* TO *dword_address*

DWORD *dword_address* TO *dword_address* = *dword_value*

All of these commands can be used whether or not the user's program is running.

DWORD *dword_address*

This form is used to examine and then possibly change one or more sequential DWORD variables. When this command is invoked, iECM-86 displays the *dword_address* in hexadecimal notation and the value of the DWORD in the default base, then waits for an input from you. You can respond with a carriage return, an ESC, or a numeric value. A carriage return terminates the command. An ESC results in the display of the next sequential DWORD variable. If a numeric value is entered, the DWORD variable is set to this value and the iECM-86 again waits for input. At this point you can respond only with an ESC or carriage return. As before, the ESC displays the next sequential DWORD and the carriage return terminates the command.

DWORD *dword_address* = *dword_value*

This form is used to set an individual DWORD variable without first checking its current value. When invoked, this command sets the DWORD variable at *dword_address* to *dword_value*.

DWORD *dword_address* TO *dword_address*

This form is used to display a region of memory as a sequence of

DWORD variables. When this command is invoked, iECM-86 starts by displaying the current default base and then a series of lines showing the contents of the selected memory region. The next line starts with a hexadecimal display of the address of the next DWORD variable to be displayed followed by the display of up to 16 bytes of memory as DWORD variables in the default base. A new line starts whenever 16 bytes of memory have been displayed on the line. The command terminates when all of the DWORD variables in the selected range have been displayed. During lengthy displays, you can stop the output to the console by pressing the space bar. You can resume the display by pressing the space bar a second time. You terminate the command by entering a carriage return.

DWORD *dword_address* TO *dword_address = dword_value*

This form is used to initialize a region of memory to the given *dword_value*. Note that this command takes a little over a millisecond (at 9600 baud) for each DWORD loaded. You can terminate this command by entering a carriage return, but terminating the command leaves only part of the memory region initialized.

5.4.5 STACK Commands

There are two basic forms for the STACK command:

STACK *stack_address*

STACK *stack_address* TO *stack_address*

Both of these commands can be used whether or not the user's program is running.

STACK *stack_address*

This command is useful for accessing a 16-bit variable that is known to be a fixed offset in the system stack. When this command is invoked, iECM-86 executes a WORD *word_address* command in which the *word_address* is formed by adding *stack_address* to the current value of the system stack pointer.

STACK *stack_address* TO *stack_address*

This command is useful for accessing a sequence of 16-bit variables that are known to start at a fixed offset in the system stack. When this command is invoked, iECM-86 executes a WORD *word_address* TO *word_address* command in which both *word_address* fields are formed by adding the corresponding *stack_address* to the current value of the system stack pointer. During lengthy displays, you can stop the output to the console by pressing the space bar. You can

resume the display by pressing the space bar a second time. You terminate the command by entering a carriage return.

5.4.6 STRING Commands

There is only one form of the STRING command:

STRING *byte_address*

The line starts with a hexadecimal display of *byte_address* followed by the NUL-terminated ASCII string starting at that address. For long strings, only the first 60 characters are displayed. When trailing characters are stripped, decimal points (.) are substituted for the first three characters stripped.

5.4.7 PORT Commands

There are four forms for the PORT command:

PORT *port_address*

PORT *port_address* = *byte_value*

PORT *port_address* TO *port_address*

PORT *port_address* TO *port_address* = *byte_value*

All of these commands can be used whether or not the user's program is running.

PORT *port_address* This form is used to examine and then possibly change one or more sequential PORT variables. When this command is invoked, iECM-86 displays the *port_address* in hexadecimal notation and the value of the PORT in the default base, then waits for an input from you. You can respond with a carriage return, an ESC, or a numeric value. A carriage return terminates the command. An ESC results in the display of the next sequential PORT variable. If a numeric value is entered, the PORT variable is set to this value and the iECM-86 again waits for input. At this point, you can respond only with an ESC or carriage return. As before, the ESC displays the next sequential PORT and the carriage return terminates the command.

PORT *port_address* = *byte_value*

This form is used to set an individual PORT variable without first checking its current value. When invoked, this command sets the PORT variable at *port_address* to *byte_value.*

PORT *port_address* TO *port_address*

This form is used to display a series of PORT variables. When this command is invoked, iECM-86 starts by displaying the current

default base and then a series of lines showing the contents of the selected ports. The next line starts with a hexadecimal display of the address of the next PORT variable to be displayed, followed by the display of up to 16 PORT variables in the default base. A new line starts whenever 16 ports have been displayed on the line. The command terminates when all of the PORT variables in the selected range have been displayed. During lengthy displays, you can stop the output to the console by pressing the space bar. You can resume the display by pressing the space bar a second time. You terminate the command by entering a carriage return.

PORT *port_address* TO *port_address* = *byte_value*

This form is used to initialize a set of ports to *byte_value.* Note that this command takes a little over a millisecond (at 9600 baud) for each PORT loaded. You can terminate this command by entering a carriage return, but terminating the command leaves only part of the memory region initialized.

5.4.8 WPORT Commands

There are four basic forms for the WPORT commands:

WPORT *wport_address*

WPORT *wport_address* = *word_value*

WPORT *wport_address* TO *wport_address*

WPORT *wport_address* TO *wport_address* = *word_value*

All of these commands can be used whether or not the user's program is running.

WPORT *wport_address*

This form is used to examine and possibly change one or more sequential WPORT variables. When this command is invoked, iECM-86 displays the *wport_address* in hexadecimal notation and the value of the WPORT in the default base, then waits for an input from you. You can respond with a carriage return, an ESC, or a numeric value. A carriage return terminates the command. An ESC results in the display of the next sequential WPORT variable. If a numeric value is entered, the WPORT variable is set to this value and the iECM-86 again waits for input. At this point, you can respond only with an ESC or carriage return. As before, the ESC displays the next sequential WPORT and the carriage return terminates the command.

WPORT *wport_address* = *word_value*

This form is used to set an individual WPORT variable without first checking its current value. When invoked, this command sets the WPORT variable at *wport_address* to *word_value.*

WPORT *wport_address* TO *wport_address*

This form is used to display a series of WPORT variables. When this command is invoked, iECM-86 starts by displaying the current default base and then a series of lines showing the contents of the selected ports. The next line starts with a hexadecimal display of the address of the next WPORT variable to be displayed, followed by the display of up to 16 WPORT variables in the default base. A new line starts whenever 16 bytes of memory have been displayed on the line. The command terminates when all of the WPORT variables in the selected range have been displayed. During lengthy displays, you can stop the output to the console by pressing the space bar. You can resume the display by pressing the space bar a second time. You terminate the command by entering a carriage return.

WPORT *wport_address* TO *wport_address* = word_value

This form is used to initialize a set of ports to *word_value*. Note that this command takes a little over a millisecond (at 9600 baud) for each WPORT loaded. You can terminate this command by entering a carriage return, but terminating the command leaves only part of the memory region initialized.

5.4.9 Processor Variables

Several commands are provided to access variables that are associated with the processor rather than with the program:

AX
AH
AL
BX
BH
BL
CX
CH
CL
DX
DH
DL

CS
DS
SP
BP
DI
ES
FLAGS
IP
PC {=<*code_address*>}
REGS
SI
SP = <*word_address*>
SS

The processor variables can be modified only while the target is stopped. They can be read at any time. These commands allow the display and loading of the internal target processor registers. Display is in the default base. Addresses are displayed in the last format used (i.e., if PC was loaded with the PC=segment:offset command, addresses will be displayed in that format).

The REGS command displays the register contents of the microprocessor in the current base. The 80C186EC registers may be individually displayed or changed by referring to them by name. The registers with suffixes of 'H or 'L are byte-wide; all others are word-wide.

The PC and SP commands are special cases because they modify both segment and offset (i.e., CS:IP, SS:SP).

NOTE

The examination of the SP will be confusing if you don't understand the following paragraph.

The iECM-86 software uses twenty-eight words in the user's stack to store all internal CPU registers during a host interface interrupt. When the user displays the SP (or uses the STACK command), the value shown for SP is adjusted by the correct number of bytes to compensate for this overhead so that it becomes invisible to the user (the user must still allow for the extra stack space used).

6

iRISM-186 Commands

CHAPTER 6
iRISM-186 COMMANDS

This chapter describes the elements of iRISM-186 monitor code. This information is common to all implementations.

6.1 iRISM VARIABLES

The following table lists the RISM variables and provides a description of each.

Table 6-1. iRISM Variables

Variable	Description
RISM_DATA	A 32-bit register that acts as the primary data interface between software running in the host and the RISM running in the target.
RISM_ADDR	A 32-bit register that contains the address to be used for reading and writing target memory. The base address is contained in the most significant word and the offset is in the least significant word.
RISM_STATUS	An 8-bit register used to store RISM status and state information. This register contains the following boolean flags:
BOOLEAN FLAGS	
DATA_FLAG	Indicates that the next character received by the RISM should be treated as a data byte, even if its value corresponds to an implemented command.
RUN_FLAG	Indicates that the target is running user code. It can modify the operation of some RISM commands.
TRAP_FLAG	Indicates that the target was running user code, but that a software trap suspended its execution. The TRAP_FLAG is cleared whenever RISM starts execution of user code.
DIAG_FLAG	An optional flag that indicates that the target is operating in a diagnostic mode. Details of this flag are implementation-dependent.
USER_CS / USER_IP	Used to save the user's program counter while the user's code is not executing.
USER_FLAGS	Saves the user's program status word while the user's code is not executing.

6.1.1 Other Variables

Specific implementations of RISMs will require other variables for temporary storage.

6.2 RISM STRUCTURE

The RISM resides in the target system and provides the interface between the target system and the user interface, which resides in the host system. The RISM is compact and simple. This serves two purposes:

1. The RISM can reside in a user's system with minimal impact on available memory.
2. The RISM is easy to port into the target's environment.

The internal state structure of the RISM was kept as simple as possible. There are only three internal flags that can change the way that the RISM deals with a character sent by the host.

6.3 RECEIVING DATA FROM THE HOST

When the RISM receives a character from the host, its first task is to determine whether the character represents a command or data. When the character is less than 32 (decimal), it is assumed to be a command. When the character is more than 32 (decimal), it is assumed to be data. When the host needs to send a data byte that has a value less than 32, it first must issue a SET_DATA_FLAG command. When the DATA_FLAG is set, the next character received by the RISM is interpreted as data (even if it is less than 32), and the DATA_FLAG is cleared. Once the RISM determines that the received character is a data byte, it processes it by shifting the 32-bit RISM_DATA register left eight places and then placing the data byte in the lower byte of the RISM_DATA register. The data shifted out of the upper byte of the RISM_DATA register is discarded.

6.4 SENDING DATA TO THE HOST

When the host expects data to be returned from the RISM, it sends a TRANSMIT command byte and waits for a response. The RISM transmits the lower byte of the 32-bit RISM_DATA register and right shifts the RISM_DATA register by eight bits. As part of this command, the RISM increments its RISM_ADDR register. The RISM transmits data only in response to a TRANSMIT command, never on its own initiative or even in response to other commands from the host.

6.5 RISM COMMANDS

This section details the operation of each of the commands sent to the RISM.

6.5.1 SET_DATA_FLAG (Code 00H)

This command sets the DATA_FLAG. This forces the next character received by the RISM to be treated as data, even if its value corresponds to a RISM command. The code that overrides the normal selection of command or data also clears the DATA_FLAG so that it applies only to the first character received after the SET_DATA_FLAG command.

6.5.2 TRANSMIT (Code 02H)

This command transmits the lower eight bits of the RISM_DATA register to the host, right shifts the data register eight places, and increments the RISM_ADDR register. Sequential TRANSMIT commands are used to read the RISM_DATA register; the RISM_ADDR register indicates the address that corresponds to the least-significant byte in the RISM_DATA register.

6.5.3 READ_BYTE (Code 04H)

This command reads the byte of memory pointed to by the RISM_ADDR register and places the result in the least-significant byte of the RISM_DATA register.

6.5.4 READ_WORD (Code 05H)

This command reads the word of memory pointed to by the RISM_ADDR register and places the result in the least-significant word of the RISM_DATA register.

6.5.5 READ_DOUBLE (Code 06H)

This command reads the double-word of memory pointed to by the address register and places the result in the RISM_DATA register.

6.5.6 WRITE_BYTE (Code 07H)

This command stores the least-significant byte of the RISM_DATA register in the byte of memory pointed to by the RISM_ADDR register and increments the RISM_ADDR register (by one) to point to the next memory byte.

6.5.7 WRITE_WORD (Code 08H)

This command stores the least-significant word of the RISM_DATA register in the word of memory pointed to by the RISM_ADDR register and increments the RISM_ADDR register (by two) to point at the next memory word.

6.5.8 WRITE_DOUBLE (Code 09H)

This command stores the RISM_DATA register in the double-word of memory pointed to by the RISM_ADDR register and increments the RISM_ADDR register (by four) to point at the next memory double-word.

6.5.9 LOAD_ADDRESS (Code 0AH)

This command loads the RISM_ADDR register with the least-significant word in the RISM_DATA register.

6.5.10 READ_PC (Code 10H)

This command loads the RISM_DATA register with the CS (Code Segment) and IP (Instruction Pointer) associated with the user's code. Most RISM implementations have to check RUN_FLAG to determine how to access the user's PC.

6.5.11 WRITE_PC (Code 11H)

This command loads the CS (Code Segment) and the IP (Instruction Pointer) associated with the user's code from the RISM_DATA register. The host software will invoke this command only while user code is not running.

6.5.12 START_USER (Code 12H)

This command starts execution of user code, clears the TRAP_FLAG, and sets the RUN_FLAG. The action of this command relies on its being executed as part of an ISR (Interrupt Service Routine). At the start of the ISR, the current CS:IP and FLAGS are pushed into the stack. If the user code is not running, the CS:IP and FLAGS that are pushed into the stack are associated with an idle loop that the RISM runs while it waits for an interrupt. The START_USER command deletes the CS:IP and FLAGS from the stack and replaces them with USER_CS, USER_IP and USER_FLAGS. When control returns from the ISR, the user's code (rather than the idle loop) executes. The host software will not issue a GO command if the user code is already running.

6.5.13 STOP_USER (code 13H)

This command stops the execution of user code and clears the RUN_FLAG. The action of the HALT command mirrors that of the GO command. In the case of the HALT command, the user's CS:IP and FLAGS are pushed into the stack upon entry to the ISR. The STOP_USER command saves this user information in USER_CS, USER_IP, and USER_FLAGS and replaces it with CS:IP and FLAGS values associated with the idle loop. When control returns from the ISR, the idle loop (rather than the user's code) executes. The host software will not issue a HALT command unless the user code is running.

6.5.14 TRAP_ISR

This is a pseudo-command. It cannot be issued directly by the host software, but is executed when an INT3 is executed. The INT3 instruction is used by iECM-86 for implementing software breakpoints and for single-stepping. A separate entry point into the STOP_USER command is provided for the INT3 vector. Code at this entry point sets the TRAP_FLAG and then drops into the code that implements the STOP_USER command.

6.5.15 REPORT_STATUS (Code 14H)

This command loads the least-significant word of the RISM_DATA register with status information. Valid status values are 0, 1, and 2:

0—indicates that user code is stopped (RUN_FLAG and TRAP_FLAG are both FALSE)

1—indicates that user code is running (RUN_FLAG is TRUE)

2—indicates that user code executed a TRAP instruction (TRAP_FLAG is TRUE)

The host software periodically polls the target system to check on its status, and this polling can rob execution time from the user's program. This loss of target processor cycles can be avoided by setting the Ring Indicator modem status line signal whenever the RUN_FLAG is set. The host software assumes that the target is running user code whenever it detects the ring indicator and issues REPORT_STATUS commands only if the ring indicator is off.

6.5.16 MONITOR_ESCAPE (Code 15H)

This command provides for the addition of RISM commands for special purposes; it uses the RISM_DATA register to extend the command set of the RISM. The basic RISM requires only one of these "extended" commands; if the lower 16-bits of the RISM_DATA register is one (RISM_DATA = 0XXXX0001H), the target processor should execute either a RST (ReSeT) instruction or a software initialization routine.

6.5.17 READ_BPORT (Code 16H)

This command reads the 8-bit input port pointed to by the RISM_ADDR register and places the result in the least-significant byte of the RISM_DATA register.

6.5.18 WRITE_BPORT (Code 17H)

This command stores the least-significant byte of the RISM_DATA register in the 8-bit output port pointed to by the RISM_ADDR register.

6.5.19 READ_WPORT (Code 18H)

This command reads the 16-bit input port pointed to by the RISM_ADDR register and places the result in the least-significant word of the RISM_DATA register.

6.5.20 WRITE_WPORT (Code 19H)

This command stores the least-significant word of the RISM_DATA register in the 16-bit output port pointed to by the RISM_ADDR register.

6.5.21 STEP (Code 1AH)

This command sets the target processor's TRAP_FLAG and the RUN_FLAG and steps one instruction. After setting these flags, the action of this command is similar to the START_USER command followed by a TRAP.

6.5.22 READ_REG (Code 1BH)

This command reads the word value of a CPU register pointed to by the most-significant word of the RISM_DATA register and places the result in the least-significant word of the RISM_DATA register. Registers are accessed as shown in Table 6-2:

Table 6-2. iRISM Registers

MSW of RISM_DATA	Register
0000H	SS
0001H	ES
0002H	DS
0003H	DI
0004H	SI
0005H	BP
0006H	SP
0007H	BX
0008H	DX
0009H	CX
000AH	AX
000BH	IP
000CH	CS
000DH	FLAGS

6.5.23 WRITE_REG (Code 1CH)

This command stores the least-significant word of the RISM_DATA register in the CPU register pointed to by the most-significant word of the RISM_DATA register. Registers are accessed as shown in Table 6-2.

6.5.24 Start Up Commands (/ or \)

Upon reset, the board is in the echo mode. Until it receives an ASCII slash (/) or reverse-slash (\), it increments every character it receives from the host and sends the incremented value back to the host. It also displays the binary code of the character received on the LEDs. If a reverse-slash is received by the RISM, the board leaves the echo mode and starts normal operation. If a slash is received, it stops echoing incremented received data and starts responding to RISM commands with the diagnostic flag set. (See the option “-DIAG” on page 4-4 for additional information on the diagnostic mode.)

A

Parts List

APPENDIX A
PARTS LIST

Table A-1 provides the board location, manufacturer, and description of each part on the 80186 EB Evaluation Board. Table A-2 provides the same information for the 186 EC Evaluation Board.

Table A-1. 80186 EB Board Manual Parts List (Sheet 1 of 3)

LOCATION	MANUFACTURER PART NUMBER	DESCRIPTION	FOOTPRINT	COMMENTS
C31	Kemet # C0805C102K5RAC	CAP, .001µF	CC0805	SMT Chip Cap
C22,C26	Kemet # C0805C103K5RAC	CAP, .01µF	CC0805	SMT Chip Cap
C5,C32	Kemet # T491C106K010AS	CAP, 10µF	6032	SMT Tant.
C4,C9,C10,	Kemet # C0805C104K5RAC	CAP, .1µF	CC0805	SMT Chip Cap
C11,C13,C14,				
C20,C21,C23,				
C24,C28,C29,				
C30,C34				
C1,C2,C3,	Kemet # C2220C105J5RAC	CAP, 1µF	CC2220	SMT Chip Cap
C6,C7,C8,				
C15,C19				
C33	Kemet # T491D476K016AS	CAP, 47µF	7343	SMT Chip Cap
C12,C27	Kemet # T495X107K010AS	CAP, 100µF	7343H	SMT Tant. ESR<.25 Ω
C25	Kemet # C0805C331K5RAC	CAP, 330pF	CC0805	SMT Chip Cap
C17,C18	Kemet # C0805C220J5GAC	CAP, 22pF	CC0805	SMT Chip Cap
C16			CC0805	Not installed, but place footprint
D1	Philips #1N4148	DIODE, 1N4148	DO-35	Axial lead Diode
D2,D3	Motorola # 1N5817	DIODE, 1N5817	DO-41	Axial lead Schottky Diode
E1,E4,E5	3M #23036111TG	3 PIN HEADER	JUMP3	3 pin header for jumper
E2,E3	3M #23066121TG	4 PIN HEADER	JUMP4	4 pin header for jumper, cut 2 x 6
J2	Methode #3100-8-102-01	2 PIN PWR CONN	CN2PMLX	2 pin power connector

Table A-1. 80186 EB Board Manual Parts List (Sheet 2 of 3)

LOCATION	MANUFACTURER PART NUMBER	DESCRIPTION	FOOTPRINT	COMMENTS
J1	AMP #544282-3	14 PIN SIP SKT	SIP14	14 pin SIP socket terminal strip
JP1	AMP # 4-87227-0	2 X 30 HEADER	HDR2X30	Cut to size (3-103186-0)
JP2	AMP # 1-103186-2	2 X 12 HEADER	HDR2X12	
L3	Coilcraft #D03316P-103	10 µH INDUCTOR		SMT inductor, 10µH
L2	Coilcraft #D03316P-104	100µH INDUCTOR		SMT inductor, 100µH
L1			3216CHIP	Not installed, but place footprint
P1,P2	AMP # 748875-3	DB9 RECEPTACLE	DB9FM1	9 pin, sub-D, R/A, female
Q1	Motorola #MMBT2907ALT1	PNP TRANSISTOR	SOT23	SMT PNP transistor,2907A
R1	Dale # CRCW1206	RES, 100k Ω	CR1206	SMT resistor, 100k
R4,R5,R6,	Dale # CRCW1206	RES, 10k Ω	CR1206	SMT resistor, 10k
R7,R8,R12,				
R13				
R9	Dale # CRCW1206	RES, 220k Ω	CR1206	SMT resistor, 220k
R11	Dale # TNPW-1206-1692-B-T-2	RES,16.9k,.1%	CR1206	SMT resistor,16.9k, .1% tolerance
R10	Dale # TNPW-1206-1002-B-T-2	RES,10k,.1%	CR1206	SMT resistor,16.9k, .1% tolerance
R3	Dale # CRCW1206	RES, 1.5k Ω	CR1206	SMT resistor, 1.5k
R2	Bourns 3006P,50k, potentiometer	POT,50k	PT3006P	Thru-hole potentiometer,50k
RP1	Bourns # 4610X-101	RPACK, 10K Ω	SIP10	Thru-hole,10 pin, 10k RPACK
S1	ITT Cannon # KSAOM211	Mom. switch	RESET	Thru-hole, 4 pin (Include square blk button #)
TP1 - TP8	Mill-Max # 3132-0-00-15-00-00-08-0	Testpoint turret	EPOINT	
U11	Maxim # MAX734CSA	12V Supply device	S08	Pending SMT availability
XU8, XU13	Berg McKenzie # SOJ32P-4.0	SMT 32 pin socket	SOJ32/400	SRAM Sockets
U8,U13	NEC # D431008LLE-A17	3.3V,1Mb,SRAM		To be installed (socketed)
U8,U13	NEC # D431008LE-17	5V,1Mb,SRAM		Bagged, to be included in kit package
U8,U13	Hitachi #HM62W8127HLJP-35	3.3V,1Mb,SRAM		Possible secondary source for 3V & 5V SRAM

Table A-1. 80186 EB Board Manual Parts List (Sheet 3 of 3)

LOCATION	MANUFACTURER PART NUMBER	DESCRIPTION	FOOTPRINT	COMMENTS
XU9	Meritec # 980021-44-01	SMT 44 pin socket	SOP44	SMT 44 pin socket,w/o alignment pins
U9	INTEL #PA28F400BV-T60	4Mb,boot blk,flash		Socketed
U6,U12,U14	Motorola # MC74AC573DW	74AC573	SO20W	SMT Octal latch
U7	Maxim # MAX750CSA	Step-down regulator	SO8	Pending SMT availability
XU10	AMP # 822039-3	28 Pin PLCC socket	SOCKET28	SMT 28 pin PLCC socket
U10	Lattice #GAL22LV10C-15LJ	Low voltage GAL		Socketed
U1	Motorola # MC74AC14D	74AC14	SO14	SMT Hex Schmitt Trg. inverter
U2	Maxim # MAX561CWI	562 Ser. xceiver	SO28W	SMT 3.3V 562 Interface device
XU5, XU3	Thru-hole DIP socket	4 pin socket for osc	XTAL8	4 pin socket in 8 pin DIP size for oscillators
U5	CTS 32MHZ,half size can	32MHZ Oscillator		Socketed, Digi-Key # CTX174-ND
U3	CTS 6MHZ,half size can	6MHZ Oscillator		Socketed, Digi-Key # CTX159-ND
XU4	Samtec #PLCC-084-T-N	84 ld PLCC socket	SOCKET84	SMT 84 pin socket,w/o alignment pins
U4	INTEL # N80L186EB-16	3.3V 80X186EB		Socketed microprocessor
Y1			XTALV	Not installed, but place footprint
Jumper shunts				Configureable options
J1	Sharp # LM16155	LCD Display		Socketed at J3, mount holes will require stndoff
Standoff H/W	EF Johnson #J234-ND (D/K)	Round Spacer, .375		2/Bd.Require (2) 2-56 nylon screw,nut,washer
	H538-ND,H612-ND, #2 washer (D/K)	Nylon #2 H/W		Nylon#2-56,3/4 screw,nut,washer as per above

Table A-2. 80186 EC Board Manual Parts List (Sheet 1 of 3)

LOCATION	MANUFACTURER PART NUMBER	DESCRIPTION	FOOTPRINT	COMMENTS
C31	Kemet # C0805C102K5RAC	CAP, .001µF	CC0805	SMT Chip Cap
C21,C30	Kemet # C0805C103K5RAC	CAP, .01µF	CC0805	SMT Chip Cap
C8,C32	Kemet # T491C106K010AS	CAP, 10µF	6032	SMT Tant.
C7,C13,C14,	Kemet # C0805C104K5RAC	CAP, .1µF	CC0805	SMT Chip Cap
C15,C16,C17,				
C20,C22,C23,				
C26,C27,C28,				
C29,C34				
C1,C2,C3,	Kemet # C2220C105J5RAC	CAP, 1µF	CC2220	SMT Chip Cap
C6,C10,C11,				
C12,C18				
C33	Kemet # T491D476K016AS	CAP, 47µF	7343	SMT Chip Cap
C19,C25	Kemet # T495X107K010AS	CAP, 100µF	7343H	SMT Tant. ESR<.25 Ω
C24	Kemet # C0805C331K5RAC	CAP, 330pF	CC0805	SMT Chip Cap
C5,C9	Kemet # C0805C220J5GAC	CAP, 22pF	CC0805	SMT Chip Cap
C4			CC0805	Not installed, but place footprint
D1	Philips #1N4148	DIODE, 1N4148	DO-35	Axial lead Diode
D2,D3	Motorola # 1N5817	DIODE, 1N5817	DO-41	Axial lead Schottky Diode
E1,E4,E5	3M #23036111TG	3 PIN HEADER	JUMP3	3 pin header for jumper
E2,E3	3M #23066121TG	4 PIN HEADER	JUMP4	4 pin header for jumper, cut 2 x 6
J2	Methode #3100-8-102-01	2 PIN PWR CONN	CN2PMLX	2 pin power connector
J1	AMP #544282-3	14 PIN SIP SKT	SIP14	14 pin SIP socket terminal strip
JP1	AMP # 4-87227-0	2 X 30 HEADER	HDR2X30	Cut to size
JP2	AMP # 2-87227-0	2 X 20 HEADER	HDR2X20	
L3	Coilcraft #DO3316P-103	10 µH INDUCTOR		SMT inductor, 10µH
L2	Coilcraft #DO3316P-104	100µH INDUCTOR		SMT inductor, 100µH

Table A-2. 80186 EC Board Manual Parts List (Sheet 2 of 3)

LOCATION	MANUFACTURER PART NUMBER	DESCRIPTION	FOOTPRINT	COMMENTS
L1			3216CHIP	Not installed, but place footprint
P1,P2	AMP # 748875-3	DB9 RECEPTACLE	DB9FM1	9 pin, sub-D, R/A, female
Q1	Motorola # MMBT2907ALT1	PNP TRANSISTOR	SOT23	SMT PNP transistor,2907A
R1	Dale # CRCW1206	RES, 100k Ω	CR1206	SMT resistor, 100k
R4,R5,R6,	Dale # CRCW1206	RES, 10k Ω	CR1206	SMT resistor, 10k
R7,R8,R9,				
R10				
R11	Dale # CRCW1206	RES, 220k Ω	CR1206	SMT resistor, 220k
R13	Dale # TNPW-1206-1692-B-T-2	RES,16.9k,.1%	CR1206	SMT resistor,16.9k, .1% tolerance
R12	Dale # TNPW-1206-1002-B-T-2	RES,10k,.1%	CR1206	SMT resistor,16.9k, .1% tolerance
R3	Dale # CRCW1206	RES, 1.5k Ω	CR1206	SMT resistor, 1.5k
R2	Bourns 3006P,50k, potentiometer	POT,50k	PT3006P	Thru-hole potentiometer,50k
RP1	Bourns # 4610X-101	RPACK, 10K Ω	SIP10	Thru-hole,10 pin, 10k RPACK
S1	ITT Cannon # KSAOM211	Mom. switch	RESET	Thru-hole, 4 pin (Include square blk button #)
TP1 - TP8	Mill-Max # 3132-0-00-15-00-00-08-0	Testpoint turret	EPOINT	
U11	Maxim # MAX734CSA	12V Supply device	S08	Pending SMT availability
XU7, XU13	Berg McKenzie # SOJ32P-4.0	SMT 32 pin socket	SOJ32/400	SRAM Sockets
U7,U13	NEC # D431008LLE-A17	3.3V,1Mb,SRAM		To be installed (socketed)
U7,U13	NEC # D431008LE-17	5V,1Mb,SRAM		Bagged, to be included in kit package
U7,U13	Hitachi # HM62W8127HLJP-35	3.3V,1Mb,SRAM		Possible secondary source for 3V & 5V SRAM
XU9	Meritec # 980021-44-01	SMT 44 pin socket	SOP44	SMT 44 pin socket,w/o alignment pins
U9	INTEL # PA28F400BV-T60	4Mb,boot blk,flash		Socketed
U8,U12,U14	Motorola # MC74AC573DW	74AC573	SO20W	SMT Octal latch
U6	Maxim # MAX750CSA	Step-down regulator	SO8	Pending SMT availability

Table A-2. 80186 EC Board Manual Parts List (Sheet 3 of 3)

LOCATION	MANUFACTURER PART NUMBER	DESCRIPTION	FOOTPRINT	COMMENTS
XU10	AMP # 822039-3	28 Pin PLCC socket	SOCKET28	SMT 28 pin PLCC socket
U10	Lattice # GAL22LV10C-15LJ	Low voltage GAL		Socketed
U1	Motorola # MC74AC14D	74AC14	SO14	SMT Hex Schmitt Trg. inverter
U2	Maxim # MAX561CWI	562 Ser. xceiver	SO28W	SMT 3.3V 562 Interface device
XU3, XU5	Mill-Max #S190-00-000-00-22000	4 pin socket for osc	XTAL8	4 pin socket in 8 pin DIP size for oscillators
U3	CTS 32MHZ,half size can	32MHZ Oscillator		Socketed, Digi-Key # CTX174-ND
U5	CTS 6MHZ,half size can	6MHZ Oscillator		Socketed, Digi-Key # CTX159-ND
XU4	3M # 2-0100-07243-000-018-007	100 ld PQFP socket		Thru-hole socket
U4	INTEL # KU80L186EC-16	3.3V 80X186EC		Socketed microprocessor
Y1			XTALV	Not installed, but place footprint
Jumper shunts				Configureable options
J1	Sharp # LM16155	LCD Display		Socketed at J3, mount holes will require stndoff
Standoff H/W	EF Johnson #J234-ND	Round Spacer, .375		2/Bd.Require (2) 2-56 nylon screw,nut,washer
	H538-ND,H612-ND,#2 washer	Nylon #2 H/W		Nylon#2-56,3/4 screw,nut,washer,as per above

80C186EB/EC features, 3-2
80C188EB/EC, configuring board jumpers, 3-2
8-bit bus, configuring the board for, 3-2

A

adaptor
    25-pin to 9-pin, 3-11
    for in-circuit emulation, 3-2

B

BCLK0 input, 3-10
breakpoints, 5-5
bus expansion, 3-14

C

connectors
    P1, 3-9
    P2, 3-10
customer service, 1-4

D

data types, supported by iECM-86, 5-10
display controller, 3-15

E

E1 jumper, 3-8
EIA/TIA-562 protocol, 3-10
Embedded Controller Monitor (ECM), 4-1, 4-2
evaluation board
    layout of EB, 2-1
    layout of EC, 2-2
    setting up, 2-4–2-5
expansion connectors, 3-12
Expansion memory, 3-3

F

FaxBack service, 1-4
Flash loader utility, 3-5
Flash memory, 3-3
    bus width configuration, 3-5
    downloading to, 3-5–3-6
    mapping, 3-5
    on-board, 3-1

H

hardware overview, 3-1
Hitachi 44780 LCD display controller, 3-15

I

I/O port unit, 3-1
I/O space, reserved, 4-6
iEC-86
    program variables, 5-10
iECM-86, 4-1
    breakpoints, 5-5
    features, 4-1
    program stepping, 5-8
    supported data types, 5-10
iECM-86 commands, 4-3
    BR, 5-6
    BYTE, 5-11
    DWORD, 5-13
    GO, 5-7
    GO FOREVER, 5-7
    GO FROM, 5-7
    GO TILL, 5-8
    HALT, 5-7, 5-8
    INCLUDE, 5-3
    LIST, 5-4
    LISTOFF/ON, 5-4
    LOAD, 5-2
    LOG, 5-4
    LOGOFF/LOGON, 5-4
    PAUSE, 5-3
    PORT, 5-15
    RESET CHIP, 5-5
    RESET SYSTEM, 5-5
    SAVE, 5-2
    STACK, 5-14
    STEP | SSTEP, 5-9
    STRING, 5-15
    WORD, 5-12
    WPORT, 5-16
include files, 5-3
iRISM-186, 4-1
    registers, 6-6
    restrictions, 4-2

J

JP1 expansion connector, 3-12
jumpers
    configuring for an 8-bit bus, 3-2
    E1, 3-8
    J2 (power connector), 3-8
    summary, 3-1

L

LCD interface, 3-15
list files, 5-3
log files, 5-3

M

Maxim MAX561, 3-10
Maxim MAX750, 3-8
memory configuration, 3-3
memory map, 3-3
memory, reserved, 4-6

N

non-maskable interrupt, 4-6
notational conventions, 1-2

P

P1 connector, 3-9
P2 connector, 3-10
peripheral expansion connector, 3-12, 3-13
power supply, 3-8
processor, selecting type using jumper, 3-1
program control, 5-5
program stepping, in iECM-86 programs, 5-8
programmable chip-selects, 3-1

R

reserved I/O space, 4-6
reserved memory, 4-6
RISM
    receiving data from host, 6-2
    sending data to the host, 6-2
    structure, 6-2
    variables, 6-1
RISM commands
    LOAD_ADDRESS, 6-4
    MONITOR_ESCAPE, 6-5
    READ_BPORT, 6-5
    READ_BYTE, 6-3
    READ_DOUBLE, 6-3
    READ_PC, 6-4
    READ_REG, 6-6
    READ_WORD, 6-3
    READ_WPORT, 6-5
    REPORT_STATUS, 6-5
    SET_DATA_FLAG, 6-3
    start-up commands, 6-7
    START_USER, 6-4
    STEP, 6-6
    STOP_USER, 6-4
    TRANSMIT, 6-3
    TRAP_ISR, 6-5
    WRITE_BPORT, 6-5
    WRITE_BYTE, 6-3
    WRITE_DOUBLE, 6-4
    WRITE_PC, 6-4
    WRITE_REG, 6-6
    WRITE_WORD, 6-3
    WRITE_WPORT, 6-6
RISM monitor, 3-3

S

segment variable registers, 5-1
serial control unit, 3-1
serial port connector (P1), 3-9
serial ports
    on-chip, 3-10
    reconfiguring for different operating frequency, 3-2
SRAM memory, 3-3
SRAM, mapping, 3-7
super-stepping, 5-9

T

technical support, 1-5
timer/counter unit, 3-1
Trap Flag, 4-6
    used in step operation, 5-9

V

voltage, selecting using jumper, 3-1

W

World Wide Web, 1-4