USER'S GUIDE FOR MICE-II

8086/8088(MIN)
8086/8088(MAX)
80186/80188
80286

3rd Edition
October 1986

2nd Printing
April 22, 1987

Printed in Taiwan, ROC
April 1987

PREFACE

Microtek International Inc. proudly presents the award winning Micro-In-Circuit-Emulator, MICE-II. You have made an excellent purchase and will soon benefit from the many advance features that Microtek designs into all of its products.

With the detailed instructions provided in this manual, you can rapidly set up and operate your new MICE-II. The information provided in this user's guide is believed accurate and complete, but no responsibility is assumed for any error or ommision.

Microtek is dedicated in providing leadership in the microcomputer-based industry. We continue to design new emulators with enhanced features, low cost and always geared toward the microprocessor designer's need. Thank you for choosing Microtek; and may we suggest that you also investigate the many other fine products that Microtek has to offer.

The sale of any Microtek product is subject to all Microtek Terms and Conditions of Sale and Sales Policies, copies of which are available upon request.

Microtek supplies a variety of hi-tech equipment to the microprocessor-based industry. For further information on other Microtek products and/or any other questions regarding this manual, please contact Microtek at the following location:

MICROTEK® INTERNATIONAL INC.

No. 6 Industry E. Road 3
Science-Based Industrial Park
Hsinchu, Taiwan 30077, R.O.C.
TEL: (035) 772155
TLX: 32169 MICROTEK
FAX: (035) 772598

TABLE OF CONTENTS

APPENDICES

CHAPTER 1
GENERAL INTRODUCTION

1.1 Introduction to MICE

Microtek's versatile Micro-In-Circuit-Emulator (MICE) is a low-cost development tool that emulates most industry-standard microprocessors. It has set new standards for universal, high-performance emulation, at an exceptionally low cost-per-function.

Traditionally, microprocessor development was done on dedicated systems. The computer system had its complement of peripheral devices and sufficient mass storage to accomodate user programs, etc. A dedicated emulator was attached to the system, which allowed the designer to assure that both hardware and software were functioning properly.

Recently, a variety of general purpose computers have been designed for use in microprocessor development. Through the use of cross assemblers, code can be generated for the target system. However, these emulator products are still dedicated to that particular computer.

With MICE, a third more practical approach to microprocessor development is available; one that uses fewer resources, performs most of the same functions, yet costs only a fraction of its predecessors. The MICE module is controlled via an RS-232C compatible interface. All software necessary to operate the MICE module is contained in EPROMs within the module. MICE can be operated using only a display terminal or in conjunction with a computer system. Different processors can be emulated by merely changing the personality card and associated EPROMs.

Some key features of MICE are:

* Operation at speeds up to the maximum rated frequency of the specified microprocessor.

* Target processor retains its entire memory and I/O space.

* Enabling and disabling of hardware control signals to the processor with console commands.

* Resident assembler and two-pass disassembler which assigns labels to subroutine and branch addresses.

* Built-in memory diagnostics and block memory transfer for target processor memory.

* Downloading and uploading of target program between MICE and host computer system.

* Help command that lists all commands along with the proper syntax.

1.2 Introduction to MICE-II

MICE-II is an enhanced member of the MICE family. All basic features of the original MICE are retained; however, MICE-II has substantially upgraded attributes. These include:

* High Performance Emulation Memory (HUEM) containing 128K bytes of static RAM (with 2 independent 64K byte segment settings).

* Emulation memory expandable in 128K byte blocks by adding optional memory boards.

* 8K byte block enable/disable capability.

* User-qualified trigger to specify start or end of tracing for source code and machine statuses. Up to 2048 cycles and 40 channels of signals may be recorded for program debug.

* Supports up to 8 hardware trace points for program debug.

* Two realtime breakpoints.

* Execution breakpoint command that stops tracing only if the trigger address is actually executed and not just prefetched. (All four breakpoints are for instructions in RAM, except for the 80286 where one breakpoint is for instructions in ROM/RAM and the other three are for instructions in RAM.)

* Supports disassembly for both Instruction Step and List Trace commands.

New BPP board option (see Appendix I) has the following capabilities:

* Sophisticated breakpoint logic, with up to 120 new breakpoint constructs.

* Flexible trigger constructs that can define single events, multiple activities or external hardware signals.

* Two data breakpoints and two external hardware breakpoints.

* Breakpoint interval timer that displays the interval from initial trigger until emulation stops.

MICE-II is a low-cost powerful emulation instrument consisting of 3 boards:

Control Emulation Processor (CEP) - also called personality board
Realtime Trace (RTT)
High Performance Universal Emulation Memory (HUEM) - also called memory board

Remember that the CEP board can be interchanged to form any other MICE-II. If more emulation memory is needed, additional HUEM boards may be added. And where even more powerful emulation breakpoint or triggering capability is required, the BPP board is also available.

At half the cost of competitive units, MICE-II provides features not available with other emulators. Some of the advantages of MICE-II over other emulators are:

universality A wide variety of microprocessors can be emulated using this single development tool, thus avoiding both the inflexibility of dedicated systems and the heavy capital investment required for general purpose systems.

versatility Emulation of a different processor only requires changing the personality board. Multiple design projects can be carried out concurrently, and the CEP board can be reused for different projects.

flexibility With MICE-II providing its own resident assembler and disassembler, it can be minimally configured with just a display terminal and power supply. After the target program has been downloaded from the host computer, testing can be done off-line, thereby freeing the computer.

1.3 MICE-II Operating Configurations

MICE-II can be used in several cofigurations. The simplest configuration requires only an RS-232C compatible terminal as the display console and command entry device. A computer system can also be configured as the controlling device.

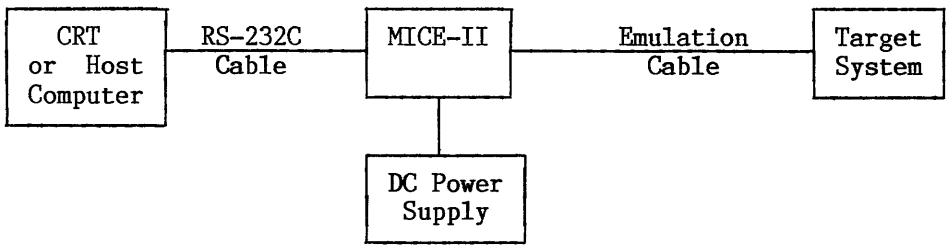


Figure 1-1
Configuring MICE-II with an RS-232C Compatible Terminal or Host Computer

When a computer system is interfaced with MICE-II, a driver program must be resident in the system. Driver programs and symbolic translator drivers are available for various systems. (Refer to Appendix E for a detailed listing.)

Microtek has access to sources of various driver programs. For updated information contact your local representative.

1.3.1 Microtek Personal Development System (MPDS)

MPDS is a fully integrated development system. It provides advanced support for all MICE-II emulators. Cross-assemblers, linkers and advanced software are just a few of its major features. Symbolic debug, logic state analysis, software performance analysis, emulation and interface utilities make up a rich software base.

A wide variety of development and debugging tools guarantee maximum performance. The extended assembler and linker supports a powerful set of macro commands and generation of symbol table files. MPDS has a large internal memory, dual disk drives and a high resolution CRT. MPDS supports comprehensive development and debugging with these powerful functions:

* **XASM and Linker**
* **Emulation**
* **MICE Utilities**
* **Symbolic Debug**
* **Software Performance Analysis**
* **Logic State Analysis**

The MPDS-MS/DOS software package is fully compatible with the IBM-PC/XT/AT. All the power and flexibility required for target system hardware debug, signal analysis (including waveform display), and software analysis makes MPDS a truly advanced micro-development solution.

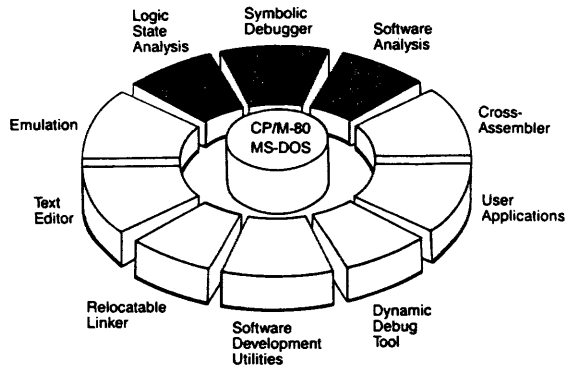


Figure 1-2
Overview of MPDS Software Environment

1.4 MICE-II Applications

Because of its unique design, MICE-II offers new applications which include:

1) Inexpensive evaluation of new microprocessors without purchasing a special evaluation board or expensive development system.

2) Several designers can share the use of a single development system, eliminating the difficult problem of allocating a single resource and the need for multiple work-stations. Large programs can be edited, assembled or compiled, and then downloaded to the target system. Since MICE-II has its own assembler and disassembler, the programs can then be tested using only a display terminal.

3) MICE-II's compact size, light weight, and rugged construction makes it an ideal field service instrument. Easily transported and set up in remote locations, MICE-II can reduce downtime, provide on the spot diagnosis and resolution of field problems, avoiding customer inconvenience and expensive service delays. With its RS-232C interface, MICE-II can be quickly interfaced with any compatible display terminal. Diagnostic programs can be generated using the resident assembler.

4) Personal computers can be upgraded to development systems at a fraction of the typical costs. Driver programs for various computers have already been written, allowing programs assembled or compiled to be downloaded.

1.5 MICE-II Definitions

MICE is a patented trade name for Microtek emulators and stands for Micro-In-Circuit-Emulator.

ICE cable is an In-Circuit-Emulator cable that joins MICE-II to the target.

MICE-II module is a conventional MICE emulator module consisting of three interconnected printed circuit boards. These boards are:

1) Control Emulation Processor board.

2) Realtime Trace board.

3) High Performance Emulation Memory Board.

4) Breakpoint Processor (BPP) Board, which may be optionally included depending on the user's specifications.

CHAPTER 2
MICE-II INSTALLATION PROCEDURES

MICE-II comes from the factory preset and completely assembled. The MICE-II module consists of three printed circuit boards: Control Emulation Processor board (CEP), Realtime Trace board (RTT) and High Performance Universal Emulation Memory board (HUEM). ICE cables are also included which must be connected between MICE-II and the target system.

2.1 Setting Up MICE-II

Detailed instructions for setting up the MICE-II module are as follows:

2.1.1 Opening the MICE-II Case

MICE-II is contained in a two piece high impact plastic case. To open the case, hold the lower portion of the vertical end (rear) firmly with both hands and with your thumbs slide the top half forward about one half inch. This will separate the top from the bottom half.

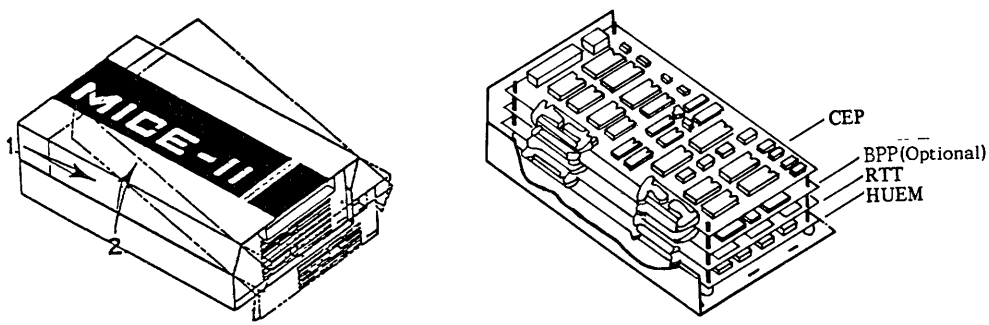


Figure 2-1
Opening the MICE-II Case

2.1.2 Replacing the Personality Board (CEP) or Emulation CPU

1) To install a personality board in place of the current card, the cover of the MICE-II case must first be removed. With power off, carefully remove the top personality card (CEP) from the MICE-II module by disconnecting the two ribbon cable connectors which join the CEP board to the RTT board below, and then removing the six locking screws and six brass spacers.

   Install the new personality board into the module by refastening the brass spacers and locking screws to hold the replacement card permanently in position. Then reconnect the two ribbon cable connectors from the RTT board below to the CEP board; the arrows indicating pin 1 must be aligned.

2) Only one CPU is supplied with MICE-II per the customer's specification. To emulate the other processor, replace the CPU as indicated below:

CPU	Location	MICE-II
8086 or 8088	U29	8086/88(MIN)
8086 or 8088	U39	8086/88(MAX)
80186 or 80188	U43	80186/188
80286	U44	80286

2.1.3 Internal Adjustments

Access to all adjustment switches is possible with the MICE-II cover removed. The DIP switch on the personality board is readily accessed from the top; and the three DIP switches for memory board selection are located in the opening at the end of the MICE-II case, which results when the cover is removed. Preset instructions are explained in the following sections of this chapter.

2.1.4 Memory Expansion (Adding HUEM Boards)

Instructions for adding memory boards are detailed in Appendix F.

2.1.5 Connecting the ICE Cable

1) 8086/88(MIN/MAX)

The ICE (In-Circuit-Emulator) cable assembly consists of one flat-wire cable with a 40-pin connector at one end and an IC header at the other end. To install the ICE cable after setup has been completed, remove the cover to MICE-II, and attach the 40-pin connector at position J5 on the CEP board. Then thread the IC header end through the rectangular opening in the MICE-II case and slide the cover shut (section 2.1.6). Note that the ICE cable for MICE-II 8086/88(MIN/MAX) cannot be interchanged with other MICE-II models.

2) 80186/188

The ICE (In-Circuit-Emulator) cable assembly consists of four flat-wire cables with two 38-pin connectors at one end and an IC header at the other end. To install the ICE cable after setup has been completed, remove the cover of MICE-II and unfasten the upper retainer next to J6 on the CEP board (Figure 2-2). Attach the two 38-pin connectors at positions J5 and J6; then refasten the retainer (Figure 2-3). Thread the IC header end through the rectangular opening in the MICE-II case and slide the cover shut (section 2.1.6). The alligator clip on the ICE cable (Figure 2-3) should be attached to a ground point on the target board, as close to the IC header as possible. Note that the ICE cable for MICE-II 80186/188 cannot be interchanged with other MICE-II models.

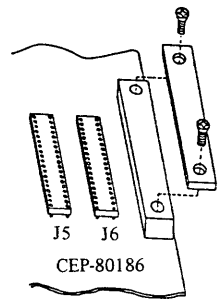


Figure 2-2

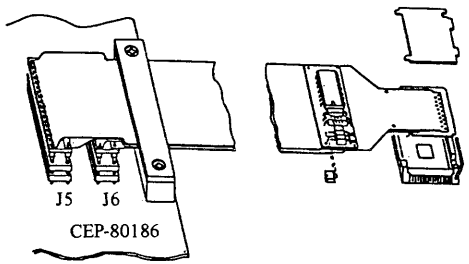


Figure 2-3

Note: When purchasing a target processor header assembly, a metal cover may be provided. If this cover is used on the chip carrier with MICE-II 80186/188, MICE-II will not function properly and may incur damage. A plastic cover is provided with MICE-II 80186/188; only this cover is authorized for use with the header assembly.

3) 80286

The ICE cable assembly is installed in the same manner as described for the 80186/188 above. However, there is no clock adaptor mounted on the 80286 ICE cable since the 80286 CPU will not accept input from a crystal oscillator for the clock source. Also note that two different connector options are supported for target interface, depending on the user's requirements. Cable assemblies with connectors for the standard chip carrier type socket and for the pin grid array type socket are indicated in figures 2-4 and 2-5 respectively.

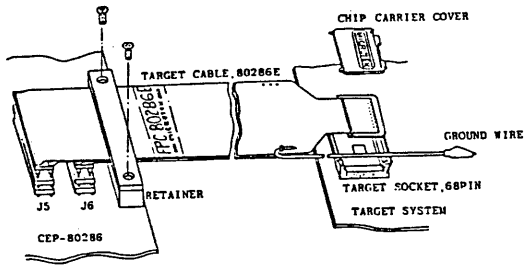


Figure 2-4

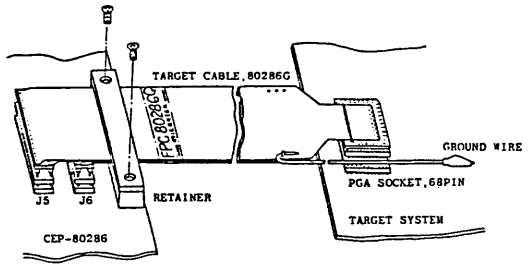


Figure 2-5

2.1.6 Closing the MICE-II Case

To close the cover, set it over the base leaving a 1/2 inch gap. Holding the base firmly in the front, use your thumbs to push the cover the remaining distance. This will force the locking tabs into position and firmly attach the top to the base. However, remember not to close the case during operation for the 8086/88(MIN/MAX), or when four boards installed, otherwise an intermittent problem with overheating may result.

2.2 MICE-II Specifications

Mechanical specifications for MICE-II are:

Width	20.0 cm	(7.87 in)
Height	8.0 cm	(3.15 in)
Length	33.0 cm	(13.00 in)
Weight	2.0 kg	(4.40 lb)

The minimum and maximum operation and storage limits for temperature and humidity are:

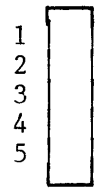
Operating temperature: 0^{o}-50^{o}C (32^{o}-122^{o}F)
Storage Temperature: -10^{o}-65^{o}C (14^{o}-149^{o}F)
Relative Humidity: 20 -80%

MICE-II requires three external power supply voltages, each of which must be regulated to within 5 percent of nominal.

Power Consumption	Ripple
+ 5VDC at 4.0A (maximum)	50m Vp-p
+12VDC at 0.1A (maximum)	100m Vp-p
-12VDC at 0.1A (maximum)	100m Vp-p

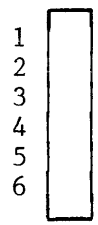
Note: Power supply loading is based on conventional MICE-II modules. Since MICE-II supports emulation memory expansion, additional power supply requirements may be necessary depending on the number of memory boards added and/or cooling fan is installed. Each additional memory board requires 0.8 amp at +5 VDC and each cooling fan requires 0.35 amp at +12 VDC. The switching power supply should also be grounded.

Using the supplied power cable, the mating female connector is attached to the pins at the rear of MICE-II with the locking tab pointing up. The power connector pinout is as follows:



Pin	Description
1	+5VDC
2	Ground
3	+12VDC
4	not connected
5	-12VDC

To improve the contact of power cable, the connector pinout of the latest and future MICE-II models are changed to the following manner:



Pin	Description
1	+5VDC
2	+5VDC
3	GND
4	+12V
5	GND
6	-12V

Warning: Be sure that the correct voltages are connected to the proper pins, otherwise MICE-II may suffer severe damage to its circuitry. When turning off the power to MICE-II, wait a few seconds to allow the capacitors to fully discharge before turning the power on again.

2.3 Communicating with MICE-II

Whether MICE-II is connected to a terminal or to a computer system, the connection between the controlling device and MICE-II is across a programmable RS-232C compatible interface.

2.4 Control Emulation Processor Board (CEP) Setup

Other MICE-II emulators can be formed simply by replacing the personality (CEP) board. If a change of personality boards is required, or modification to the factory preset conditions is necessary, the adjustment options are described as follows:

On the CEP board locate the six position DIP switch [8086/88(MIN)-U39, 8086/88(MAX)-U54, 80186/188-U17 and 80286-U18]. This switch sets the following communication modes which are only examined by the control processor during power-up or the software reset "r" (section 2.8).

Switch Selection	Description
S1-S2-S3	Baud Rate
S4	7/8 Data Bits
S5	Disable/Enable Parity
S6	Odd/Even Parity

The number of stop bits is permanently set at two; and communication is full duplex. Each data frame consists of 1 start bit, 2 stop bits, 7/8 data bits and 1 parity bit if parity is enabled.

The transmission rate can be specified from 150~19200 baud by setting S1-S2-S3 of the DIP switch as follows:

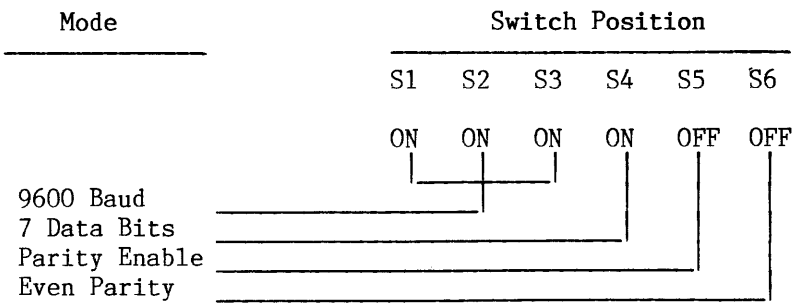
Baud Rate	Switch Section S1	S2	S3
150	ON	OFF	OFF
300	OFF	ON	OFF
600	ON	ON	OFF
1200	OFF	OFF	ON
2400	ON	OFF	ON
4800	OFF	ON	ON
9600	ON	ON	ON
19200	OFF	OFF	OFF *

* This setting is only effective for the following (or later) firmware versions: 80286 - V3.0, all others - V3.2. For any previous versions, this switch setting indicates 110 baud.

The number of data bits and parity can be specified by setting S4, S5 and S6 of the DIP switch as follows:

Mode	Switch Section S4	S5	S6
8 Data Bits	OFF		
7 Data Bits	ON		
Parity Enable		OFF	
Parity Disable		ON	
Even Parity			OFF
Odd Parity			ON

MICE-II personality boards are shipped from the factory preset to 9600 baud, 7 data bits, and parity enabled at even. Switch position for this default selection is as follows:



Default setting

When parity is disabled, the odd/even parity switch section can be set to either position since it is ignored.

2.5 High Performance Emulation Memory Board (HUEM) Setup

There are four DIP switches located at the front of the HUEM board (bottom board in the MICE-II module); U14, U28, U36 and U43. The individual keys on the switches are designated S1 through S8 or S10.

There are two separate 64K memory banks in the HUEM. U14 and U28 set bank 1 (memory is at U1-4 and U15-18). U36 and U43 set bank 2 (memory is at U5-8 and U19-22).

The DIP switches are set up per the following instructions:

2.5.1 U14 and U36: Emulation Memory Enable/Disable

The user can select any memory block of 8K bytes to disable or enable depending on the amount of emulation memory needed. Set individual keys ON for enable and OFF for disable.

```
   S1       S2       S3       S4       S5       S6       S7       S8
Lowest 8K ..................................................... Highest 8K
```

Position S9 and S10 are used as follows:

Mode	S9	S10
Write Enable	ON	
Write Protect	OFF	
Memory Enable		OFF
Memory Disable		ON

2.5.2 U28 and U43: Memory Segment Select

Memory segments may be selected within 16M bytes by setting U28 and U43 as follows:

Segment	Start Address	S1	S2	S3	S4	S5	S6	S7	S8
0K	0H	ON	ON	ON	ON	ON	ON	ON	ON
64K	10000H	OFF	ON	ON	ON	ON	ON	ON	ON
128K	20000H	ON	OFF	ON	ON	ON	ON	ON	ON
"	"				"				
"	"				"				
16320K	FF0000H				A11 OFF				

Note: If bank 1 and bank 2 have the same start address (i.e. U28 and U43 have the same key setting), HUEM will select bank 1 only and disregard the U34 key setting for bank 2.

2.5.3 Factory Preset

HUEM is preset at the factory with the following switch positions (0-128K bytes):

U14,U36	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
	ON	ON	ON	ON	ON	ON	ON	ON	ON	OFF

In this configuration all blocks (64K byte x 2) of emulation memory are enabled and write enabled.

U28	S1	S2	S3	S4	S5	S6	S7	S8
	ON	ON	ON	ON	ON	ON	ON	ON

All U28 keys are in the ON position. In this configuration, the bank 1 memory segment start address is 0000H.

U43	S1	S2	S3	S4	S5	S6	S7	S8
	OFF	ON	ON	ON	ON	ON	ON	ON

Only S1 is OFF. In this configuration, the bank 2 memory segment start address is 10000H.

Setting U14, U28, U36 and U43 completes setup of the HUEM board.

2.6 RS-232C Cable Connection

After selecting the proper data rate and transmission characteristics, next determine whether the controlling device has a data terminal equipment (DTE) interface or a data communication equipment (DCE) interface, with or without handshaking. Display terminals are usually equipped with DTE interface; computer systems usually have both.

There are several methods for determining the type of interface on the controlling device. The first method is by simple trial and error. If this fails, procedures two and three listed below can be used to determine the interface type. (It must be known beforehand whether or not handshaking is selected as a software option for the controlling device, as it cannot be easily detected by examining hardware signals.) Subsequent instructions detail how to accomplish interface header rewiring if required.

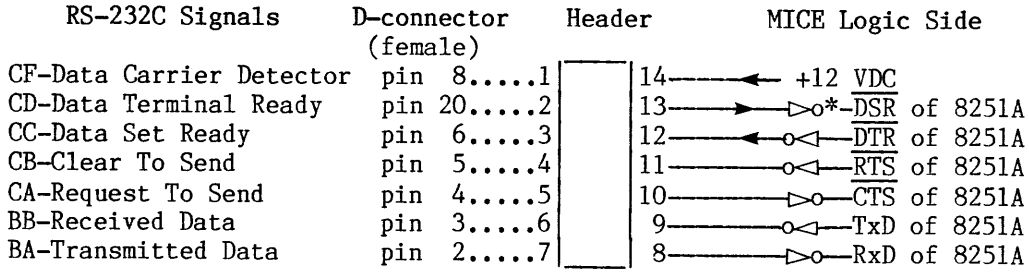
1) Try connecting the two devices together. If the response is correct, the interfaces match and no further adjustment is required. If the interconnect does not work, a mismatch exists and indicates that a connection change is necessary.

2) Data are transmitted on pin 2 and received on pin 3 of the D-connector for DTE devices; the reverse is true for DCE devices.

3) When not transmitting, the voltage (with respect to pin 7 [ground] of the D-connector) is -12 VDC on pin 2 for DTE devices and -12 VDC on pin 3 for DCE devices.

The header designated as U13 on the CEP board is used to configure the interface between MICE-II and the controlling device. MICE-II are shipped from the factory with a straight-wired header for DTE with handshaking.

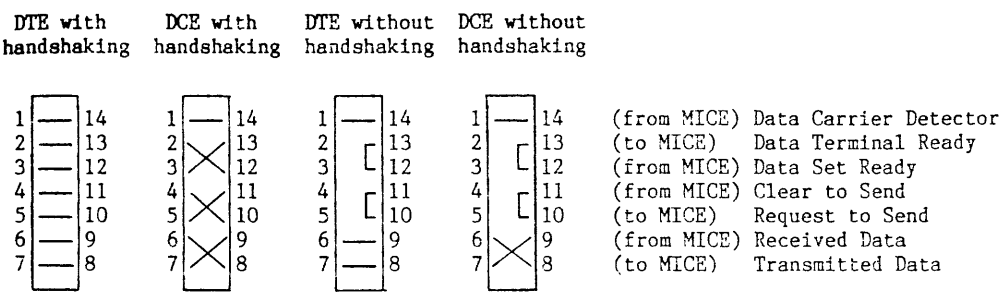
Pins 1-7 on the interface header are connected to the female D-connector (J2) while pins 8-14 are connected to the MICE-II, RS-232C interface logic. Pin definitions on the header are as follows:



*RS-232C interface gate for 1488 or 1489 depending on signal Input or Output.

The wiring configuration for all three header types is shown below. Note that if the controlling device has a DCE interface, incoming signals are now outgoing, and vice versa, on the same pins. By removing the cover to the header, the interface type used by MICE can be determined according the following wiring configuration.

DTE with handshaking	DCE with handshaking	DTE without handshaking	DCE without handshaking



When wired for DTE, connect MICE-II using a straight-wire cable with male D-connectors; for other header types, the cable must be rewired accordingly. If a display terminal is connected after making the proper connections, MICE-II should respond when power is applied. (A display terminal should always be used to check new MICE-II units to ensure that they are properly functioning.)

Finallly, MICE-II requires that both Request to Send and Data Terminal Ready inputs, pins 10 and 13 respectively on the logic side of the header, be at +12 VDC before it transmits any data. If the controlling device does not supply the necessary voltage, it can be obtained by reconnecting pins 10 and 13 directly to pin 14 (Received Line Signal Detector) of header U13 which is always at +12 VDC.

To accommodate computer systems which are sending commands and data too fast for MICE-II, the Data Set Ready output, normally at +12 VDC, is pulled low by MICE-II to -12 VDC. The signal is restored to +12 VDC when MICE-II is again ready. Also, to accommodate display terminals with slow carriage-returns or line-feeds, six null characters are always transmitted after a carriage-return is issued.

2.7 Applying Power to MICE-II

Connect your RS-232C cable from the controlling device to the female D-connector at the rear of MICE-II. (Remember that when powering on the system, first apply power to the target and then to MICE-II; and use the opposite sequence when powering off.) With the controlling device ready, connect the power cable with the locking tab pointing up, and apply power. Within a few seconds, the following start-up message should display:

```
***MICE-II-type V#.#***
>
```

type identifies target processor being emulated by the personality card.

#.# indicates version number of controlling program on personality card.

> is the prompt character indicating that MICE-II is ready for a command.

In response to the command prompt character ">" enter a question mark "?", immediately followed by a carriage-return. If the data bits and parity are correctly matched, the MICE-II command summary is listed; otherwise, the error message "WHAT?" is printed. Reset the switch section where necessary and remember to wait a few seconds before power is turned on again.

Notes 1) If target VCC is not provided when MICE-II is powered up, the following message will display:

8086/88(MIN) - NO TARGET VCC; NMI, HOLD, INTR DISABLED!
8086/88(MAX) - NO TARGET VCC; NMI, INTR, RQ/GT0, RQ/GT1 DISABLED!
80186/188 - NO TARGET VCC; NMI, HOLD, INTR, DRQ DISABLED!
80286 - DEBUG MODE; NMI AND PEREQ DISABLED!
NO TARGET VCC; HOLD, INTR AND RESET DISABLED!

2) The CPU is automatically reset after power-up, the software reset command "r" (section 2.8) or the MICE Reset command X (section 5.7). If the target system has any peripheral devices or slave processors which must be synchronized with a CPU reset, then the reset must be performed on the target side. A target reset signal will also reset the emulation CPU.
However, note that this does not apply for the 80186/188 CPU which supports a RESET output signal, and therefore eliminates the need to reset on the target side for peripheral devices or slave processors that are connected to pin-57.

3) If the processor used is an 8086 (80186) and the clock is not ready, an incorrect title may be displayed - MICE-II 8088 (80188).

4) After powering up the 80286, the PC is automatically set to FFFFFOH.

2.7.1 No Response

If there is no response from MICE-II, check the following items:

1) Check the RS-232C cable connection at both ends.
2) Check the power supply connections and voltages.

3) Check that the header has the proper interface.

4) Check that both Request to Send and Data Terminal Ready, pins 10 and 13 on the header are at +12 VDC.

5) Check that the controlling device has the proper voltage level requirements on its RS-232C inputs for transmission and reception.

6) Check that the baud rates of the controlling device and MICE-II are the same, resetting if necessary. If a message does appear but is garbled, any combination of the baud rate, data length or parity could be incorrectly set.

7) Check the RS-232C cable for incorrectly wired or loose pins.

8) If a computer system is the controlling device, check that the driver program is running and the RS-232C cable is connected to the correct port and that the port is working.

Note: When changing the communication configuration switch, note that it is only read during a power-up or reset. After turning the power off, wait a few seconds before the power is turned on again to allow the capacitors to fully discharge.

If the problem still cannot be found, contact your local Microtek representative for further assistance.

2.7.2 Failure Device

During the delay prior to the start-up message the RAMs and EPROMs on the CEP board are tested. If any component failures exist, they are listed as detected in the following format: "U## - FAILURE", where ## is the component number on the personality board.

2.8 Control Processor Software Reset Command

The character "r" is the command to reset MICE-II. Remember that except for this command all alphabetic characters must be entered in upper-case; no other lower-case characters are recognized.

2.9 MICE Operation Modes (80286 only)

1) Debug and Trace Mode

The MICE-II 80286 uses two separate modes of operation to emulate the Intel 80286 microprocessor. Debug Mode supports all MICE commands for iAPX 86 Real Address Mode (with the NMI signal disabled by MICE); while Trace Mode supports MICE trace commands (B/BS/BT/C/CW/D/E/F/G/H/L/X) for both Real Address Mode and Protected Virtual Address Mode (where the NMI signal is controlled by the user).

2) Applying Power

After executing either a hardware power-up (cold start) or software reset command "r" (warm start), MICE-II 80286 is automatically set to Debug Mode, the clock source defaults to the on-board 6MHz clock, the PEREQ input signal for coprocessor communication is disabled, and emulation memory access uses the on-board Ready Signal (with zero wait cycles). The Non-maskable Interrupt Request (NMI) signal is always disabled while in Debug Mode; however, the status of other control signals will depend on whether or not power is applied to the target.

80286 Power Reset Status (for cold or warm start)
a) Debug Mode enabled. b) Clock source is 6MHz internal clock. c) NMI always disabled. d) PEREQ is disabled.* e) Target Ready Signal disabled (wait state) when accessing emulation memory. f) Status of following signals depends on whether or not power is applied to the target: I - Maskable Interrupt Request H - Bus Hold Request R - System Reset

* The PEREQ signal is automatically disabled after a power reset. If an 80287 coprocessor is used in the target, then key in the "EP" command (Chapter 6) to enable debug for 80287 numeric instructions (section 4.4).

3) Mode Selection

The Enable Control Signal command is used to change operation modes. (Note that MICE does not have to be connected to a power-applied system to change modes.) The following commands are used to select MICE operation modes.

ED - Enable Debug Mode
ET - Enable Trace Mode

Changing from Debug to Trace mode does not affect the system; but changing from Trace to Debug mode while the CPU is located at an NMI service routine will generate a "TARGET CAN'T STEP" message if an A/BP/I/J/M/O/R/S/T/U/Z or download command is keyed in. When executing the "ED" command, MICE will display the following message:

"WARNING: CHANGING TO DEBUG MODE; MAKE SURE NOT IN NMI ROUTINE!"

If your program is within the NMI service routine at this time, then perform the following steps to avoid a system error.

a) Key in the "ET" command and return to Trace mode. (Remember that the NMI signal is not enabled when entering Trace mode; however, subsequent status will depend on input of D/E commands.)

b) Key in a C/G/F/B command to let the CPU advance beyond the NMI routine and then execute any Debug mode commands.

CHAPTER 3
MICE-II COMMAND LANGUAGE

All MICE-II products have a common set of commands identified by a single or double character regardless of the processor being emulated. No additional time or effort is required to learn a new command language when the target processor changes.

These commands are described in the following chapters along with a variety of available options, though not all options are applicable for the different types of processors. For the specific processor being emulated, consult the Help "?" command. The commands described in the following chapters are grouped as follows:

MICE-II Utility Commands
- ? Help Command
- ! Attention Command
- K Clock Selection
- !M Coprocessor Mode Selection
- Mode Real/ Protected Virtual Addrress Mode Selection (for 80286 only)

Memory, Port and Register Commands
- M Memory Display/Examine/Modify/Fill/Search Command
- T Memory Checksum/Test/Transfer/Compare Command
- A Line Assembly Command
- Z Disassembly Command
- I Port Input Command
- O Port Output Command
- R Register Display/Modify Command
- J Jump/Branch Command
- X Reset/Initialization Command

Control Signal Commands
- D Disable/Display Control Signal Command
- E Enable/Display Control Signal Command

Emulation and Trace Control Commands
- G Go/Execution Command
- H Halt/Breakpoint Set Command
- F Forward Trace Command
- B Backward Trace Command
- BP Execution Breakpoint Command
- L List Trace Buffer Command

Stepped Emulation Commands
- C Single Cycle, Step Command
- S Instruction Step Command

Utility Commands Involving a System
- : Download Command (Intel Format)
- / Download Command (Tektronix Format)
- U Upload Command

3.1 Command Syntax

MICE-II indicates that it is ready to accept a command line by printing a greater-than character ">" on a new line. A command may then be entered and must be terminated by a carriage-return <CR>. The general syntax of MICE-II commands is:

command [parameters] <CR>

where: **command** is the command representation.

parameters are one or more variable data supplied with the command. Parameters are alphanumeric; when a numeric parameter is called for, it must be entered in hexadecimal.

Where a space is shown in the syntax, either a space or comma can be used. A <CR> must be used to terminate a command input line. In most cases line feed <LF> or <CR> have the same effect, except where otherwise noted. Note that brackets [ ] and braces { } are used for describing command syntax only, and are not used in command input.

3.2 Notations and Conventions

A set of conventions is used to describe the structure of commands. The notations and rules are as follows:

1) An upper-case entry must be input.

2) A lower-case entry in the description of a command is the class-name for a parameter. A particular value for this class must be entered. A class-name never appears in an actual operable command. For example, the lower-case entry "**start-address**" means that MICE-II will only accept a hexadecimal value as an address in the target processor's memory space.

3) A required entry is shown without any enclosures; whereas an optional entry is denoted by enclosing it in brackets. For example, the command description "? [B]" indicates that the command "?" is required, and the

brackets around the entry "B" means that it is optional in this command.

Where brackets are within another set of brackets, the entry enclosed by the inner brackets may only be entered if the items outside those inner brackets are first entered. For example, in the command description - "I port[count[time]]", the command "I" is required; and the brackets mean that the selection of "count" and "time" is optional in the command. However, a "count" must first be entered if a "time" value is to be specified.

4) Where an entry must be selected from a choice of two or more, the choices for the required entry are enclosed in braces and separated by vertical bars. For example - "{S|M|a3[V]}" indicates that either "S", "M" or "a3" must be entered.

5) Where a choice exists for an optional entry, the choices are enclosed in brackets and separated by vertical bars. For example, "[CS|DS|SS|ES]" indicates that either "CS", "DS", "SS" or "ES" may be entered.

6) "Addx" is a hex address with optional byte pair of "XX" or an optional nibble of "X" [only for the fifth digit (e.g. X2345)]; where X indicates that the digits are don't care. For example - "12XX" means all addresses between 1200H and 12FFH.

7) Commands A/C/S/I/R all use carriage-return <CR> or line-feed <LF> to display the next line.

8) In the Memory Modify (M) command, use a <CR> to advance to the next location and a <LF> to return to the previous location.

9) Entries underlined in command examples indicate user input.

3.3 Editing Characters

Each character entered on the keyboard is stored in a line editing buffer until <CR> is entered. If more than 80 characters are entered without inputting a <CR>, an error message is printed, and the command is ignored.

The line editing buffer can be edited or entirely deleted by using special non-printable editing characters. Control characters are entered by holding down the control key <CTRL> while the character is typed. Control character input is expressed with the control command enclosed in < > brackets.

BACKSPACE deletes the preceding character from the line buffer and from the display. Repeated usage is allowed. <CTRL-H> performs the same function. When a hardcopy terminal is used instead of a display screen, RUBOUT should be used.

RUBOUT deletes the preceding character from the line buffer and echoes the deleted character on the display, preceded by a backslash character. Repeated usage is allowed. On some terminals, this key may be labeled as DELETE or DEL.

ESCAPE ignores the current contents of the line buffer and prompts ">" for a new command on the next line. This key is also used to terminate commands in process and to return to the prompt state. ESCAPE is also expressed as <ESC>. On some terminals, this key may be labeled ESC. <CTRL-Y> performs the same function.

<CTRL-R> causes a <CR> or <LF>, followed by a redisplaying of the current undeleted contents in the line buffer. This is useful to see a clean copy of the command line after RUBOUT has been used.

<CTRL-X> ignores the current contents of the line buffer and shifts the cursor to the first position of the next line, awaiting input of new data.

3.4 Control Characters and Delimiters

The following control characters have special meaning for all MICE-II firmware:

<CTRL-J>: is the same as <LF>.
<CTRL-M>: is the same as <CR>.
<CTRL-S>: stops data transmission from MICE-II.
<CTRL-Q>: continues data transmission from MICE-II.
<CTRL-Y>: is the same as <ESC>.

CHAPTER 4
MICE-II UTILITY COMMANDS

These commands allow the user to query MICE-II for a summary of the commands and syntax that are available for the target under emulation. In addition, the user can query to display the processor type being used.

Notes 1) If any code other than OFFH *1 is placed in firmware at the location indicated in table 4-1, it will cause the handshaking code at this location to be sent to the host computer (or terminal) when MICE is waiting for input. This extra code can be used to improve interface efficiency with the host.

2) The handshaking code 003H *2 is sent to the host computer when MICE is waiting for input. Any code other than 003H may be placed in firmware (table 4-1) to suit the user's specific requirements.

3) When using the MPDS software package MICEA86, version V2.2 or earlier, the handshaking code used in MICE firmware (table 4-1) must be OFFH. For MPDS version 2.3 or later, there is no limitation on the handshaking code used in MICE.

MICE-II	CEP Location	Address	Handshaking Code	
			OFFH *1	003H *2 (ETX)
8086/88(MIN/MAX)	U4	1FFDH	V1.0-V3.1	V3.2 & later
80186/188	U1	3FFDH	V1.0-V3.1	V3.2 & later
80286	U1	7FFDH		V3.1 & later

Table 4-1 Handshaking Codes

4.1 Help Command - ?

?

? is the command keyword for Help.

The command summary for the target processor currently being emulated is displayed on the terminal. All commands are common regardless of target processor type, but command parameters may differ for the different processors under emulation.

Example: Display the command summary for MICE-II 8086/88(MAX).

```
>?
ASSEMBLY              A [[CS|DS|SS|ES|seg:]addr]
BACKWARD TRACE        B [R]addx[ c[ q]]
CYCLE STEP            C [W|c]
DISABLE               D [N|I|H|C]
ENABLE                E [N|I|H|C|
FORWARD TRACE         F [R]addx[ c[ q]]
EXECUTION             G [[seg:]addr]
BREAKPOINT            H [0[1[2]|1 [addx[ c[ q]]]|2 [addr]]
INPUT                 I [W]port[ c[ time]]
JUMP                  J [seg:]addr
LIST TRACE            L [step[ a1[ a2[ q...]]]|S[step]|Z[step]|N]
MEMORY                M [[W][CS|DS|SS|ES|seg:]a1[ a2[ d1[ ...d8]][ S]]
OUTPUT                O [W]port d1[ ...d8]
REGISTER              R [AX|BX|CX|DX|SP|BP|SI|DI|DS|ES|SS|CS|IP|FS]
INSTRUCTION STEP      S [S|R][c]|[Z]|[a1 a2]
TRANSFER/TEST         T [CS|DS|SS|ES|seg:]a1 a2 {S|M|[CS|DS|SS|ES|seg:]a3[ V]}
UPLOAD                U [CS|DS|SS|ES|seg:]a1 a2 [T|I]
RESET                 X [seg[:addr]]
DISASSEMBLY           Z [[CS|DS|SS|ES|seg:]a1 [a2]]
DOWNLOAD              : (INT),/ (TEK)
HELP                  ? [B]
ATTENTION             !
>
```

The above summary does not fully follow the notations and conventions previously described to avoid a lengthy display.

MICE-II 8086/88(MAX) is used as the basis for all examples in this manual. Command syntax varies for different MICE-II emulators; for the proper syntax consult the Help (?) command.

4.2 Attention Command - !

!

! is the command keyword for Attention.

The target processor type currently being emulated by the personality card in MICE-II is displayed on the terminal. Six characters are used for identification.

Example: Display the processor type currently being emulated [8086(MAX)].

```
>!
MX8086
>
```

The messages for 16-bit Intel series processor types are:

MICE-II	PROCESSOR TYPE
8086/8088(MIN)	MN8086 or MN8088
8086/8088(MAX)	MX8086 or MX8088
80186/80188	M80186 or M80188
80286	M80286

4.3 Clock Selection - K (80286 only)

K[I|T]

K is the clock selection. If no other parameters are specified, inputting "K<CR>" will display the current clock selection for the emulation proccessor.

I is internal clock.

T selects the external target clock.

After a cold/warm start, MICE defaults to the internal clock. Changing the current clock selection will reset the emulation processor. Note that if the current clock source is selected, MICE displays:

"SAME CLOCK SLECTION, SOURCE NOT CHANGED!"

Example:

```
>K                          ;display clock selection.
INTERNAL CLOCK              ;internal clock 10MHz.
>
```

4.4 Coprocessor Mode Selection - ! (80286 only)

!M[87|287]

!M is the command for coprocessor mode selection. If no other parameters are specified, inputting "!M<CR>" will display the current coprocessor mode setting. Default is for 80287.

87 selects 8087 mode. When a "WAIT" (09BH) and 8087 instruction sequence occurs, LS/LZ/SR/SS/SZ commands (Chapter 7) and the Z command (section 5.4) display these two instructions as one instruction.

287 selects 80287 mode, where the "WAIT" (09BH) code is always treated as a separate instruction.

The 8086 assembler automatically inserts a "WAIT" instruction before each numeric instruction. However, the 80286 assembler does not generate "WAIT" instructions before numeric instructions, because the 80286 CPU automatically tests the BUSY line from the 80287 to ensure that it has completed the previous instruction before executing the next numeric instruction. Therefore, the correct disassembly format (87 or 287) must be selected to permit proper display of programs generated by either an 8086 or 80286 assembler. Also note that the "A" command does not insert "WAIT" instructions prior to numeric instructions in either 8087 or 80287 mode.

Example: Disassemble a brief program containing both 8087 and 80287 instructions using 8087 coprocessor mode.

```
>!M87
>Z0 23
LOC          OBJ              LINE LABEL      SOURCE CODE
0100:0000    9BDF6020         0001            FBLD    [BX][SI]20
0100:0004    9B               0002            WAIT
0100:0005    8B05             0003            MOV     AX,[DI]
0100:0007    9B2EDFB73412     0004            FBSTP   CS:[BX]1234
0100:000D    9B               0005            WAIT
0100:000E    268A0C           0006            MOV     CL,ES:[SI]
0100:0011    9B               0007            WAIT
0100:0012    2E9BDA5090       0008            FICOM   DWORD [BX][SI]90
0100:0017    9B               0009            WAIT
0100:0018    2E9B             0010            WAIT
0100:001A    01C8             0011            ADD     AX,CX
0100:001C    2E9B             0012            WAIT
0100:001E    2E8B05           0013            MOV     AX,CS:[DI]
0100:0021    2E9B             0014            WAIT
0100:0023    2EDF160004       0015            FIST    WORD CS:0400
     DISASSEMBLY COMPLETED
>
```

Example: Now disassemble in the same program using 80287 coprocessor mode.

```
>!M287
>Z0 23
LOC          OBJ              LINE LABEL      SOURCE CODE
0100:0000    9B               0001            WAIT
0100:0001    DF6020           0002            FBLD    [BX][SI]200
0100:0004    9B               0003            WAIT
0100:0005    8B05             0004            MOV     AX,[DI]
0100:0007    9B               0005            WAIT
0100:0008    2EDFB73412       0006            FBSTP   CS:[BX]1234
0100:000D    9B               0007            WAIT
0100:000E    268A0C           0008            MOV     CL,ES:[SI]
0100:0011    9B               0009            WAIT
0100:0012    2E9B             0010            WAIT
0100:0014    DA5090           0011            FICOM   DWORD [BX][SI]90
0100:0017    9B               0012            WAIT
0100:0018    2E9B             0013            WAIT
0100:001A    01C8             0014            ADD     AX,CX
0100:001C    2E9B             0015            WAIT
0100:001E    2E8B05           0016            MOV     AX,CS:[DI]
0100:0021    2E9B             0017            WAIT
0100:0023    2EDF160004       0018            FIST    WORD CS:0400
     DISASSEMBLY COMPLETED
>
```

Example: Assemble (section 5.3) the same program first in 8087 mode and then in 80287 mode. Note that there is no difference in assembly between 8087 and 80287 modes.

```
>!M87
>A
LOC          OBJ              LINE LABEL      SOURCE CODE
0100:0000    D9C1             0001            FLD
0100:0002    D9C1             0002            FLD ST(1)
0100:0004    DD31             0003            FSAVE [BX][DI]
0100:0006    DD7180           0004            FNSAVE [BX][DI]80
0100:0009    DBE0             0005            FENI
0100:000B    DBE0             0006            FNENI
>!M287
>A
LOC          OBJ              LINE LABEL      SOURCE CODE
0100:0000    D9C1             0001            FLD
0100:0002    D9C1             0002            FLD ST(1)
0100:0004    DD31             0003            FSAVE [BX][DI]
0100:0006    DD7180           0004            FNSAVE [BX][DI]80
0100:0009    DBE0             0005            FENI
0100:000B    DBE0             0006            FNENI
>
```

CHAPTER 5
MEMORY, PORT AND REGISTER COMMANDS

These commands give access to the contents or current values stored in designated memory locations, I/O ports and registers. The maximum amount of addressable memory is 1M byte per memory type [CS|DS|ES|SS]; and the total amount of addressable I/O is 64K bytes.

Logical addresses* (e.g. 6000:1234) must be used for M/T/A/Z/J/X commands (described in this chapter). For M/T/A/Z/J commands, if only one address is input (e.g. 1234), it indicates the offset (instruction pointer) and default for memory type is CS. For the X command, if only one address is input, it indicates the segment value and default for the offset is 0H.

MICE-II always checks that memory type and memory/port addresses are valid before an operation is performed. References to an invalid type or invalid address result in an error message being printed and the command ended. MICE-II, however, does not verify that there is memory or anything connected to the port when completing an operation. In these cases, data written is lost, and data read is random.

Note: When performing READ/WRITE functions, if external circuitry does not respond with the READY signal for 8086/88(MIN/MAX) and 80286 or the SRDY/ARDY signal for 80186/188, "TARGET IS NOT READY!!" will appear on the console and the contents of the registers will be retained.

* (Segment Value x 10H) + (Instruction Pointer) = Program Counter. Also note that this is the only method used by MICE-II to calculate logical addresses for both Real Address Mode and Protection Mode for the 80286.

5.1 Memory Display/Examine/Modify/Fill/Search Command - M

M[[W][type]start-address[end-address[data-1...data-8]][S]]

M	is the command keyword for Memory Display/Examine/Modify/Fill/Search. If no other parameters are specified, inputting "M<CR>" will display the next 256 bytes starting at the current PC.
W	specifies a memory operation with memory content organized in word format.
type	is the type of memory to be used, and is indicated by alphabetic characters CS/DS/ES/SS or a 4-digit hexadecimal segment value followed by a colon (:); default is for Code Segment.
start-address	is a hexadecimal address of the target processor indicating a memory location where the operation is to begin.
end-address	is a hexadecimal address of the target processor indicating the last memory location of the specified range.
data-1...8	are hexadecimal or ASCII data. Data in ASCII must be enclosed within two apostrophes (e.g. 'AR'). Where a "block fill" or "block search" operation is specified, the block may be up to 8 bytes in hex or 8 characters in ASCII. Combined use of hex and ASCII is permitted. If the address range (from start to end-address) is smaller than the block size (data1 - data8), then block-fill/search is still executed but excess trailing data is ignored. (Note that the apostrophe ['], lower case "r" and MICE editing/control characters are not allowed for ASCII input.)
S	defines a "search" option which looks for specified data.

5.1.1 Memory Display

M[[W][type]start-address[end-address]]

Input a start and end-address to display the content of a memory range. Memory contents are displayed in hexadecimal and ASCII. (Note that data without an ASCII equivalent is indicated by a period.) The end-address must be greater than or equal to the start-address or an error message is printed and the command ends. The display can be terminated by entering an <ESC>. Type specification can be used to indicate the memory type to be displayed; CS is always selected when the type specification is omitted.

Example: Display program memory contents 600:209H to 600:217H. (Note that the code segment may also be set with the X/R/J commands.)

```
>M600:209 217
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0600:0200                                 69 00 C4 11 65 F8 7E          i...e.~
0600:0210  BE 00 40 BF 00 30 B9 10                            ..@..0..
>
```

5.1.2 Memory Examine/Modify

M[W][type]start-address

If only the start-address is specified, the content of that memory location is first displayed. MICE-II then waits for input. To advance to the next memory address, a <CR> should be entered; the next memory location's content is then displayed, and MICE-II again waits. To go back to the previous memory address a <LF> should be entered; the previous memory location's content is then displayed, and MICE-II again waits. Entering a <CR> or <LF> repeats the entire sequence. If an <ESC> is entered, the command ends.

To change the memory content displayed, enter a new value and a <CR> or <LF>. (Note that the apostrophe ['], lower case "r" and MICE editing/control characters are not allowed for ASCII input.) Where other than default memory type is required, the type of memory to be changed must be specified. After writing data to memory, verification of memory starts automatically (I/O is also verified if the target has memory mapped I/O). If any data does not match, the following message displays: "MEMORY VERIFICATION FAILURE!".

Example: Examine and modify the memory contents from address 600:210H, then display the results. Program memory type "CS" is assumed.

```
>M600:210
 0600:0210 41 BE<CR>
 0600:0211 FF 00<CR>
 0600:0212 00 '@'<CR>
 0600:0213 BF <LF>
 0600:0212 40 <ESC>
>M600:210 217
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0600:0210  BE 00 40 BF 00 30 B9 10                          ..@..0..
>
```

Example: Examine and modify the memory contents from address location 0600:0020H with word option, then display the results.

```
>MW 600:20
 0600:0020 FFBF 00BE<CR>
 0600:0022 2001 BF40<CR>
 0600:0024 3000 <ESC>
>M0600:20 26
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0600:0020  BE 00 40 BF 00 30 B9                             ..@..0.
>
```

5.1.3 Memory Fill

M[W][type]start-address end-address data-1[...data-8]

Input a start and end-address for the memory range to be filled. A specified data value or data block may be written into the defined range. Where other than default memory type is required, the type of memory to be used must be specified.

Example: Write the value 28H into the memory range 0024H to 0033H and then display the memory contents for the range 0020H to 003FH.

```
>M600:24 33 28
>M600:20 3F
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0600:0020  BF FF 01 20 28 28 28 28 28 28 28 28 28 28 28 28  ... ((((((((((((
0600:0030  28 28 28 28 FF FF 00 00 FF FF 00 00 FF FB 00 00  ((((............
>
```

5.1.4 Memory Search

M[W][type]start-address end-address data-1[...data-8] S

Input a start and end-address for the memory range to be searched, a specified data value or data block to be searched for within the defined range, and "S". Where other than default memory type is to be searched, the type of memory must be specified.

Example: Search for string "BA 10" in the range 210H to 21FH, which is not found; then search for string "'0' B9" in the same address range.

```
>M600:0210 21F
           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0600:00210  BE 00 40 BF 00 30 B9 10 00 F3 A5 EB F3 FF 00 00  ..@.0...........
>M600:210 21F BA 10 S
 MEMORY SEARCH FAILURE !
>M600:200 21F '0' B9 S
 MEMORY MATCH AT ADDRESS 0600:0215
>
```

5.2 Memory Checksum/Test/Transfer/Compare Command - T

```
T[type]start-address end-address {S|M|address-3[ V]}
```

T	is the command keyword for Memory Checksum/Test/Transfer/ Compare.
type	is the type of memory to be used, and is indicated by alphabetic characters CS/DS/ES/SS or a 4-digit hexadecimal segment value followed by a colon (:); default is for Code Segment. Specifying more than one memory type is permitted.
start-address	is a hexadecimal address of the target processor indicating a memory location where the operation is to begin.
end-address	is a hexadecimal address of the target processor indicating the last memory location of the specified range.
S	displays the checksum of the contents in the specified range.
M	performs a memory test for the specified range.
address-3	is a hexadecimal address of the target processor specifying either a destination address where data is to be transferred, or the start of a second memory range used in "block compare" (section 5.2.4). Note that if a segment value is not specified for this address, the default is the segment value for the start-address.
V	specifies "block compare".

Define the start and end-address for a memory range in the target processor. The end-address must be greater than or equal to the start-address or an error message is printed and the command ends. Code Segment memory is always selected when the type specification is omitted.

5.2.1 Memory Checksum

T[type]start-address end-address S

An "S" following the range specification causes the checksum to be displayed. The checksum is calculated by taking the hexadecimal sum of the contents for the indicated range, with carry added back, modulo 256.

Example: First display the program memory contents for the range 0105H to 010AH, and then calculate the checksum for the same range.

```
>M0600:105 10A
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0600:0100                 30 B9 10 00 F3 00                     0.....
>T600:105 10A S
THE CHECKSUM IS: ED
>
```

5.2.2 Memory Test

T[type]start-address end-address M

An "M" following the range specification causes a memory test to be performed on the indicated range. The upper-byte and lower-byte of the address whose memory is to be tested are exclusive-or'ed (XOR) and written into memory for the entire range. The data are verified and then their complements are written into the entire range and thoroughly checked again. If the comparison fails in either pass, the test is stopped at the failed address and that address is displayed. The original contents in memory are destroyed in this test.

Example: Test the memory range 0100H through 010AH, and then display the memory contents for the range 0100H to 010AH.

```
>T0600:100 10A M
 RAM OK!
>M0600:100 10A
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0600:0100  FE FF FC FD FA FB F8 F9 F6 F7 F4                  ..........
>
```

Example: Test the memory range 3FF0H through 400FH, where memory is enabled only through 3FFFH.

```
>T600:3FF0 400F M
ADDRESS (4000) RAM ERROR!
>M600:3FF0 400F
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0600:3FF0 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F  0123456789:;<=>?
0600:4000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
>
```

5.2.3 Memory Transfer

T[type]start-address end-address address-3

An address-3 (dest-address) specification indicates that the memory content in the defined range is to be transferred into memory beginning at address-3. Data is transferred, one location at a time beginning at the start address of the destination range. If the memory ranges defined in the transfer command overlap, the transfer will begin at the end address of the destination range to prevent overwrite.

Example: First display the memory content for the range 0100H to 011FH, then transfer the memory content in the range 0100H through 010AH to memory beginning at 0112H. Finally, display the results of the transfer.

```
>M600:100 11F
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0600:0100  BE 00 40 BF 00 30 B9 10 00 F3 A5 EB F3 D4 90 D4  ..@..0..........
0600:0110  4E 45 57 20 44 41 54 41 4E 45 57 20 44 41 54 41  NEW DATANEW DATA
>T600:100 10A 112
>M600:100 11F
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0600:0100  BE 00 40 BF 00 30 B9 10 00 F3 A5 EB F3 D4 90 D4  ..@..0..........
0600:0110  4E 45 BE 00 40 BF 00 30 B9 10 00 F3 A5 41 54 41  NE..@..0.....ATA
>
```

5.2.4 Memory Compare

T[type]start-address end-address address-3 V

A "V" following the address-3 specification executes block compare for the indicated memory ranges. The block from start-address to end-address is compared with the block beginning at address-3. If comparison is successful, "COMPARISON OK!" will display; otherwise the failure addresses along with the incorrect data are displayed.

Example: First display the memory contents from 0100H to 011FH, and compare the data block from 0100H to 010AH with the address range beginning at 0112H.

```
>MCS:100 11F
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0600:0100  BE 00 40 BF 00 30 B9 10 00 F3 A5 EB F3 D4 90 D4  ..@..0..........
0600:0110  4E 45 BE 00 40 BF 00 30 B9 10 00 F3 A5 41 54 41  NE..@..0.....ATA
>T100 10A 112 V
  COMPARISON OK!
>
```

Example: Then compare the data block from 0100H to 010AH with the address range beginning at 0110H.

```
>TCS:100 10A 110 V
 (0600:0100) = BE  ;(block 1 failure address) = data 1
 (0600:0110) = 4E  ;(block 2 failure address) = data 2
 >
```

5.3 Line Assembly Command-A

A[[type] start-address]

A is the command keyword for Assembly. If no other parameters are specified, inputting "A<CR>" will execute assembly beginning at the current PC.

type is the type of memory to be used, and is indicated by alphabetic characters CS/DS/ES/SS or a 4-digit hexadecimal segment value followed by a colon (:); default is for Code Segment.

start-address is a hexadecimal address in the target processor's program memory where MICE-II begins storing the entered assembly language program.

Rather than enter programs or program changes in machine code using the previously described memory (M) command, the MICE-II resident assembler accepts and converts mnemonic inputs into machine code. The converted code is then stored in the target processor's program memory starting at the indicated start-address. The assembler does not recognize symbolic labels or constants other than hexadecimal values. The instruction mnemonics accepted are those adopted by the original manufacturer for the processor being emulated. In addition to these mnemonic codes (listed in the appendices), the following three instructions are also supported during assembly.

DB - Define Byte	(1-6 bytes)
DW - Define Word	(2 bytes)
DS - Define Storage	(OH-FFFFH)

DB and DW accept both hex and ASCII codes, while DS is restricted to hex data. (Note that the apostrophe ['], lower case "r" and MICE editing/control characters are not allowed for ASCII input.) If more than 6 bytes are keyed in for DB, excess data is truncated on the right and MICE-II displays: "WARNING: ONLY 6 BYTES ARE VALID!". If more than 2 bytes are keyed in for DW or DS, the input data is ignored and "ERROR CODE, TRY AGAIN!" is displayed.

After inputting the assembly command, MICE-II displays the following column headings:

```
>AO
LOC          OBJ        LINE          LABEL          SOURCE CODE
0600:0000                0001                         *
```

where the decimal value of 0001 under the column "LINE" indicates the line number being entered and the "*" indicates the new cursor position. MICE-II then waits for an assembly language line input from the user. The source code column consists of an opcode and operand, and must be terminated with either a <CR> or <LF>. MICE-II assembles the line and stores the machine code beginning at the indicated start-address. The code for the program memory location to be stored is printed under the column "OBJ". The next line number is printed and MICE-II again waits. If an <ESC> is entered instead, the command ends.

MICE-II performs data verification after executing a memory WRITE. If any data does not match, the following message displays: "MEMORY WRITE FAILURE!". If an invalid program instruction is entered, MICE-II displays: "ERROR CODE, TRY AGAIN!".

Example: Enter a simple program starting at location 0210H.

```
>A210
LOC          OBJ        LINE          LABEL          SOURCE CODE
0600:0210    BE0040     0001                         MOV     SI,#4000<CR>
0600:0213    BF0030     0002                         MOV     DI,#3000<CR>
0600:0216    B91000     0003                         MOV     CX,#0010<CR>
0600:0219    F3         0004                         REP<CR>
0600:021A    A5         0005                         MOVS    WORD<CR>
0600:021B    EBF3       0006                         JMP     0210<CR>
0600:021C               0007                         <ESC>
>
```

Example: An error occurs when an incorrect mnemonic is entered. In this case, non-hexadecimal data cannot be used in the data field.

```
>AO
LOC          OBJ        LINE          LABEL          SOURCE CODE
0600:0000                0001                         MOV SI,#400G<CR>
ERROR CODE, TRY AGAIN!
0600:0000    B0040      0001                         MOV SI,#4000<CR>
0600:0003               0002                         <ESC>
>
```

When making changes in an existing program, it is advisable to check the program around the modified area using the Disassembly command (section 5.4) before and after the change. Rechecking the modified area is important to assure that new program changes do not affect the surrounding program.

Notes: 1) Opcodes ADD, ADC, CMP, SBB or SUB used with a WORD address and immediate data in the operand (e.g. CMP WORD 2000,#data) are treated as follows when keying in the immediate data:

```
   0H to   7FH - opcode is 83H
  80H to FFFFH - opcode is 81H
 -80H to   -1H - opcode is 83H (data as 2's complement)
less than -80H - is illegal for immediate data
```

and vice versa for the Disassembly command.

2) 8086 mode consists of 8086 and 8087 instruction sets.

3) Code conversion for A/Z commands into 8089 instructions is specified by entering a ";" before <CR> (i.e. A/Z[type]start-address;).

4) If the target uses an 8087 or 8089 coprocessor, the ESC code coprocessor program and memory for DMA must reside in target memory and not in MICE-II emulation memory. Therefore, the trace cannot record DMA cycles.

5) 80286 mode consists of 80286 and 80287 instruction sets. (Refer to section 4.4.)

5.4 Disassembly Command - Z

Z[[type] start-address[end-address]]

Z — is the command keyword for Disassembly. If no other parameters are specified, inputting "Z<CR>" will execute disassembly for the next 256 bytes starting at the current PC.

type — is the type of memory to be used, and is indicated by alphabetic characters CS/DS/ES/SS or a 4-digit hexadecimal segment value followed by a colon (:); default is for Code Segment.

start-address — is a hexadecimal address in target program memory where the display of disassembled memory content begins.

end-address — is the hexadecimal address in the target processor's program memory indicating the last memory location of the range to be disassembled and displayed.

If only the start-address is specified, the content for that memory location is disassembled and displayed; more data are then read from subsequent locations to complete the instruction if necessary. An end-address specification causes the memory contents in the defined memory range to be disassembled and displayed. The end-address must be greater than or equal to the start-address or an error message is printed (INPUT ADDRESS ERROR) and the command ends. If an illegal machine code is encountered, disassembly terminates at the illegal code's address.

MICE-II uses a two-pass disassembler with all branch and subroutine call addresses first identified and converted to labels; a total of 900 labels can be stored for disassembly. Depending on the range to be disassembled, there may be a pause before any data is displayed.

Example: Disassemble the previously entered program. Note that the byte following the end-address is also read in order to complete the instruction for this example. Note that labels include a prefix (B for branch or S for subroutine) and address.

```
>Z210 21C
LOC         OBJ       LINE         LABEL          SOURCE CODE
0600:0210   BE0040    0001         B0210          MOV     SI,#4000
0600:0213   BF0030    0002                        MOV     DI,#3000
0600:0216   B91000    0003                        MOV     CX,#0010
0600:0219   F3        0004                        REP
0600:021A   A5        0005                        MOVS    WORD
0600:021B   EBF3      0006                        JMP     0210
                           DISASSEMBLY COMPLETED
>
```

Example: If an illegal machine code is encountered, disassembly terminates at the illegal code's address.

```
>Z300:210,21C
LOC         OBJ       LINE         LABEL          SOURCE CODE
0300:0210   BE0040          0001   B0210          MOV     SI,#4000
0300:0213   FFFF            0002                      ERROR CODE
>
```

5.5 Port Input Command - I

I[W] port[count[duration]]

I is the command keyword for Input.

W is an option to read 16-bit data in word format from two 8-bit ports simultaneously (e.g. IW0 reads port 1 as the high byte and port 0 as the low byte).

port is the hexadecimal address of the target processor's input port that is to be read and displayed.

count is a hexadecimal value from 00H to FFH specifying the number of times the input port is to be read, with 00H indicating 256 times.

duration is a hexadecimal value from 00H to FFH specifying the interval in milliseconds between each read, with 00H indicating 256 milliseconds.

The input port command has two modes of operation. If neither the count nor the duration is specified, a range of port contents can be read and displayed beginning with the indicated port address.

MICE-II first reads and displays the contents of the port specified and waits for input from the user. To advance to the next port, enter either a <CR> or <LF>; the next port's content is then read and displayed, and MICE-II again waits. Entering a <CR> or <LF> repeats the entire sequence. The command is terminated by inputting an <ESC>.

Example: Read and display the contents of ports 4, 5 and 6.

```
>I4
 04 DC<CR>
 05 87<LF>
 06 FF<ESC>
>
```

Specifying a count with or without a duration interval, MICE-II first reads from the specified input port the number of times indicated and then displays the contents.

Duration defines the interval between each read in milliseconds, permitting compensation for data inputs which are time critical. If no interval is specified, then data is read at one millisecond intervals.

Example: Read and display the next 64 (40H) values for port D0H at 10 (0AH) millisecond intervals.

```
>ID0 40 A
PORT 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
00D0 36 22 AE 9B 2B 18 C7 77 26 D6 86 35 E5 95 44 F4  6"..+..w&..5..D.
     A3 53 03 B2 62 12 C1 71 20 D0 80 2F DF 8E 3E EE  .S..b..q ../..>.
     9D RD FD AC 5C 0B BB 6B 1A CA 7A 29 D9 88 38 E8  .M..\..k..z)..8.
     97 47 F6 A6 56 05 B5 65 14 C4 73 23 D3 82 32 E0  .G..V..e..s#..2.
>
```

Example: Read and display the next 32 (20H) word values for port D0H at 16 (10H) millisecond intervals.

```
>IW D0 20 10
PORT       00    02    04    06    08    0A    0C    0E     ASCII-CODE
00D0     FFFD  FF90  FF20  FF60  FF44  FF30  FF68  FF04  .......`.D.0.h..
         FF91  FF76  FFB5  FF40  FFD9  FF18  FFFD  FF20  ...v...@........
         FF20  FF94  FF43  FF64  FF67  FF30  FF91  FF00  .....C.d.g.0....
         FFB4  FF00  FFD8  FF50  FFFC  FF50  FF19  FF28  .......P...P....
>
```

Note that the interval can be up to 256 milliseconds (approximately a quarter of a second). If the number of times the port is to be read is also 256, the elapse time before any data is displayed is approximately 65 seconds. Therefore, to end the command before display begins, enter an <ESC>.

5.6 Port Output Command - O

O[W] port data-1[...data-8]

O is the command keyword for Output.

W is an option to write 16-bit data in word format to two 8-bit ports simultaneously (e.g. OWO writes the high byte to port 1 and the low byte to port 0).

port is a hexadecimal address of the target processor's output port.

data-1...8 are hexadecimal values to be written into the specified output port of the target processor in sequential order. Note that if word option is selected, though, only 4 values can be specified.

If only one value is specified, it is written to the indicated output port. If more than one value is specified, each value is written to the indicated port at one millisecond intervals with the first value going out immediately.

Example: Write the value 48H to port 01H.

```
>O 1 48
>
```

Example: Write values 1, 2 and 3 to port 05H. Value 1 is written immediately followed by value 2 after one millisecond and value 3 after another millisecond.

```
>O 5 1 2 3
>
```

Example: Write values to port 1234 in word format. These values are written to the output port at one millisecond intervals.

```
>OW 1234 41ED 0356 3C00 0032
>
```

5.7 Reset/Initialization Command - X

X [segment[:address]]

X is the command keyword for Reset/Initialization.

segment is the segment value of a hexadecimal address in the target processor's program memory where emulation is to begin.

address is the offset (instruction pointer) of a hexadecimal address in the target processor's program memory where emulation is to begin (the default is for 0).

A signal pulse is generated on the appropriate pin of the target processor under emulation to reset it. The processor's program counter, internal registers and flags are all altered. A start address can be defined which sets the target processor's program counter to the specified program memory address. Only the target processor is reset, any other circuits in the target system that are also connected to the same pin are not affected by the reset.

In addition, control and interrupt signals to the target processor that are selectively controllable by Disable and Enable commands are activated if MICE-II is connected to a power-applied target system. If no target system is connected to MICE-II, those signals, except for Software Control [signal C for 8086/88(MIN/MAX)], will be disabled.

Example: Reset a target processor, showing changes in the program counter (PC) and flags before and after. After reset, the code segment (CS) becomes "segment" and instruction pointer (IP) becomes "address".

```
>R
 AX= 1111 BX= 2222 CX= FFFF DX= FFFF SP= FFFF BP= FFFF SI= 0000 DI= 0000
 DS= 0000 SS= 0000 ES= 0000 CS= FFFF IP= 0000 FS= FF02 PC= FFFF0
>X12:34
>R
 AX= 1111 BX= 2222 CX= FFFF DX= FFFF SP= FFFF BP= FFFF SI= 0000 DI= 0000
 DS= 0000 SS= 0000 ES= 0000 CS= 0012 IP= 0034 FS= F002 PC= 00154
>
```

Example: If neither the "segment" nor "address" is entered, the code segment (CS) becomes FFFFH and the instruction pointer (IP) becomes 0000H.

```
>X
>R
 AX= 1111 BX= 2222 CX= FFFF DX= FFFF SP= FFFF BP= FFFF SI= 0000 DI= 0000
 DS= 0000 SS= 0000 ES= 0000 CS= FFFF IP= 0000 FS= F002 PC= FFFF0
>
```

Example: If "address" is not entered, the instruction pointer becomes 0000H.

```
>X10
>R
 AX= 1111 BX= 2222 CX= FFFF DX= FFFF SP= FFFF BP= FFFF SI= 0000 DI= 0000
 DS= 0000 SS= 0000 ES= 0000 CS= 0010 IP= 0000 FS= F002 PC= 00100
>
```

1) When executing the X command "without" address specification, the PC is set to FFFFFOH.

   When executing the X command "with" address specification, the sixth digit of the PC is forced to zero (i.e. OxxxxxH). Therefore, special care should be taken when working with a target decoding circuit where the sixth digit of the PC must be interpreted.

2) If the Stack Pointer (SP) is less than 6H when executing the R command, MICE will automatically increase the SP to 6H and display the following message: "SP CAN'T BE LESS THAN 6H; SP HAS BEEN INCREASED TO 6H!!".

3) The X command affects signal status as indicated below.

MICE Reset Command (X) Status
a) Current selection for MICE Operation Mode (section 2.9) and the control signal "W" (Chapter 6) is not changed.
b) Clock source is not changed.
c) PEREQ - if MICE-II is attached to a power-applied target, then signal status will not be affected; otherwise this signal is always disabled.
d) NMI - when in Debug Mode, this signal is always disabled. when in Trace Mode, this signal's status depends on whether or not power is applied to the target.
e) Status of the following signals depends on whether or not power is applied to the target: I - Maskable Interrupt Request H - Bus Hold Request R - System Reset

5.8 Register Display/Modify Command - R

R[register]

R is the command keyword for Register.

register are alphabetic characters indicating the target register whose content is to be displayed or modified. The register set which can be displayed and changed by the R command is as follows:

Mnemonic	Register	Contents
AX	Accumulator	16 bits
BX	Base Register	16 bits
CX	Count Register	16 bits
DX	Data Register	16 bits
SP	Stack Pointer	16 bits
BP	Base Pointer	16 bits
SI	Source Index	16 bits
DI	Destination Index	16 bits
DS	Data Segment	16 bits
ES	Extra Segment	16 bits
SS	Stack Segment	16 bits
CS	Code Segment	16 bits
IP	Instruction Pointer	16 bits
FS	Status Flag	16 bits

In addition to the registers indicated above, the MICE-II 80286 can also display/modify the following registers.

Mnemonic	Register	Contents
MS	Machine Status word	16 bits
GDT	Global Descriptor Table Base	24 bits
	Global Descriptor Table Limit	16 bits
IDT	Interrupt Descriptor Table Base	24 bits
	Interrupt Descriptor Table Limit	16 bits

5.8.1 Display Registers

R

All register contents are displayed if no specific register is defined. Note that the displayed value of the program counter (PC) always indicates the address for the next instruction's opcode.

Example: Display register contents for the 8086 microprocessor.

```
>R
AX= 1234 BX= 5678 CX= FFFF DX= FFFF SP= FFFF BP= FFFF SI= 0000 DI= 0000
DS= 0000 SS= 0000 ES= 0000 CS= 0100 IP= 0034 FS= F002 PC= 01034
>
```

Example: Display register contents for the 80286 microprocessor.

```
>R
AX=1390 BX=1291 CX=9078 DX=5634 SP=12DF BP=9078 SI=5634  FS  -NPLODITSZ-A-P-C
DI=1291 DS=0000 SS=0000 ES=0000 CS=F000 IP=FFF0 MS=FFF0 0002 0000000000000010
GDT.BASE=FFFFF0 GDT.LIMIT=FFFF  IDT.BASE=000000 IDT.LIMIT=FFFF  PC=0FFFF0
>
```

5.8.2 Display Internal Control Block

RI[address] or RIM address

I is the command to read an internal control block. (Only applicable for 80186/188.)

M specifies an internal control block in memory space; default is for I/O space.

address is the base address for the internal control block (the 8 Most Significant Bits for I/O space and the 12 Most Significant Bits for memory space); default is for FFH in I/O space. The address range of the Internal Control Block = (base address x 100H) to ([base address x 100H] + FFH).

The internal control block is displayed in I/O space with the RI command, and in memory space with the RIM command. To change the address range for the internal control block, use the MW command when located in memory space and the OW command when located in I/O space; otherwise an error message (INTERNAL CONTROL BLOCK ADDRESS MISMATCH!) will display. This error message will also display if the location specified by the RI[M] command is not the current base address.

Example: Display the internal control block at the default I/O address range of FF00H to FFFFH.

```
>RI
RELOCATION       REG.  (FFFE)= 20FF
                               0000 0002 0004 0006 0008 000A 000C 000E
INTERRUPT  CNTL REG.   (FF20)  0000 80FF 0000 0000 00FD 0007 0000 0000
                       (FF30)  0000 000F 000F 000F 000F 000F 000F 000F
TIMER  0&1 CNTL REG.   (FF50)  0000 FFFF FFFF 203F 0000 FFFF FFFF 203F
TIMER   2  CNTL REG.   (FF60)  0000 FFFF FFFF 0000
MCS & PCS  CNTL REG.   (FFA0)  FFFB 36B8 7DF8 63F8 B4B8
DMA DESCRIPTION CH0    (FFC0)  BE7B 0000 7BCB 0000 B20D 0000
DMA DESCRIPTION CH1    (FFD0)  AA23 0000 47A3 0000 B19A 0000
>
```

Example: Set and display the internal control block at memory address range 18000H to 180FFH.

```
>OW FFFE 3180    ;First modify the relocation register to memory address 3180.
>RIM180          ;Then display internal control block.
RELOCATION       REG. (180FE)= 3180
                               0000 0002 0004 0006 0008 000A 000C 000E
INTERRUPT  CNTL REG. (18020)   0000 80FF 0000 0000 00FD 0007 0000 0000
                     (18030)   0000 000F 000F 000F 000F 000F 000F 000F
TIMER  0&1 CNTL REG. (18050)   0000 FFFF FFFF 203F 0000 FFFF FFFF 203F
TIMER   2  CNTL REG. (18060)   0000 FFFF FFFF 0000
MCS & PCS CNTL REG.  (180A0)   FFFB 36B8 7DF8 63F8 B4B8
DMA DESCRIPTION CH0  (180C0)   BE7B 0000 7BCB 0000 B20D 0000
DMA DESCRIPTION CH1  (180D0)   AA23 0000 47A3 0000 B19A 0000
>
```

Example: Attempt to display the internal control block in I/O space without first changing the relocation register to I/O space with the MW command.

```
>RI
INTERNAL CONTROL BLOCK ADDRESS MISMATCH!
>MW 1000:80FE 80FF 20FF<CR>
>
```

In the preceding example, memory fill was used instead of memory modify to change the relocation register because it does not verify memory after execution, and thereby avoids generating an unnecessary memory verification failure message. The memory modify command can also be used to change the relocation register, but because the internal control block is transferred to I/O space FF00H-FFFFH, the associated data cannot be correctly verified.

Example: Change the relocation register to I/O space using memory modify.

```
>MW 1000:80FE
 1000:80FE 3180 20FF<CR>
MEMORY VERIFICATION FAILURE!
 1000:8100 FFFF <ESC>
>
```

Note: The DHLT (DMA Halt) bit of Interrupt Status Register for the internal control block is supposed to be set by the CPU NMI sequence. If it is set by the user's program, an A/I/J/M/O/R/S/T/Z command will reset it. This bit is always read as "Ø" in the RI command and "1" in the Memory/Input command; these values are invalid because the actual data is not available.

5.8.3 Modify Registers

R register

If a register is specified, the content of that register is first displayed; MICE-II then waits for input. To advance to the next register enter either a <CR> or <LF>; the next register's content is then displayed and MICE-II again waits. Entering a <CR> or <LF> repeats the entire sequence unless the current register being displayed is "FS" (Status Flag). If an <ESC> is entered, the command ends. To change the content of the displayed register, enter a new value in hex or ASCII (enclosed with two apostrophes, e.g. 'AR') and <CR> or <LF>.

Example: Examine the specified register and change its content.

```
>RAX
 AX 1234 1111<CR>
 BX 5678 2222<CR>
 CX FFFF<ESC>
>R
 AX= 1111 BX= 2222 CX= FFFF DX= FFFF SP= FFFF BP= FFFF SI= 0000 DI= 0000
 DS= 0000 SS= 0000 ES= 0000 CS= 0100 IP= 0034 FS= F002 PC= 01034
>
```

For the 80286, remember that the limit of the IDT cannot be set to less than 0FH; otherwise the Register command would not work and the emulation processor would hang up. If you attempt to modify the limit of the IDT to less than 0FH, "IDT.LIMIT CAN'T BE LESS THAN 0FH!!" will be displayed.

Example: Modify the base and limit for both the GDT and IDT registers of an 80286 target.

```
>RGDT
 GDT.BASE FFFFF0 FFFFFF<CR>
    .LIMIT  FFFF 1234<CR>
 IDT.BASE 000000 111111<CR>
    .LIMIT  FFFF 1234<CR>
 AX FFFF <ESC>
>
```

Modifying the CS or IP register changes the program counter. The program counter is calculated as (Code Segment)*10H + (Instruction Pointer). The Jump command can also be used to change the program counter (PC), as described in the following command description.

Example: Change the program counter by modifying CS and IP registers.

```
>RCS
 CS 0100 10<CR>
 IP 0034 22<CR>
 FS F002<ESC>
>R
 AX= 1111 BX= 2222 CX= FFFF DX= FFFF SP= FFFF BP= FFFF SI= 0000 DI= 0000
 DS= 0000 SS= 0000 ES= 0000 CS= 0010 IP= 0022 FS= F002 PC= 00122
>
```

5.9 Jump/Branch Command - J

```
J [segment:]address
```

J is the command keyword for Jump/Branch.

segment is the segment value of a hexadecimal address in the target processor's program memory where emulation is to begin. (The default for memory type is the current CS.)

address is the offset (instruction pointer) of a hexadecimal address in the target processor's program memory where the program counter will be set.

Changing the target processor's program counter causes subsequent emulation to continue from that address in program memory space.

Example: Change the PC from the current address to 200H.

```
>J100
>R
 AX= 1111 BX= 2222 CX= FFFF DX= FFFF SP= FFFF BP= FFFF SI= 0000 DI= 0000
 DS= 0000 SS= 0000 ES= 0000 CS= 0010 IP= 0100 FS= F002 PC= 00200
>
```

Example: Change the PC by modifying the code segment and instruction pointer from the current address to 210H.

```
>J20:10
>R
 AX= 1111 BX= 2222 CX= FFFF DX= FFFF SP= FFFF BP= FFFF SI= 0000 DI= 0000
 DS= 0000 SS= 0000 ES= 0000 CS= 0020 IP= 0010 FS= F002 PC= 00210
>
```

CHAPTER 6
CONTROL SIGNAL COMMANDS

These commands enable or disable some of the control signals going to the processor. The available signal is either enabled or disabled at the input pin on the target processor. Only sense input to the processor is affected; circuits which generate the control signal in the target system are unaffected. The control characters and signal designation for hardware control signals (including pin numbers) and software control signals that are affected by the Disable and/or Enable commands are listed below:

Signal Designation	8086/88(MIN)	8086/88(MAX)	80186/188	80286
C - Software Control	*1	*1		
D - Direct Memory Access Request			DRQ0 - 18 DRQ1 - 19	
D - Debug Mode Selection				*2
H - Bus Hold Request	HOLD - 31 HLDA - 30	$\overline{RQ0}/\overline{GT0}$ - 31 $\overline{RQ1}/\overline{GT1}$ - 30	HOLD - 50 HLDA - 51	HOLD - 64 HLDA - 65
I - Interrupt Request	INTR - 18	INTR - 18	INT0 - 45 INT1 - 44 INT2/$\overline{INTA0}$ - 42 INT3/$\overline{INTA1}$ - 41	INTR - 57
N - Non-maskable Interrupt Request	NMI - 17	NMI - 17	NMI - 46	NMI - 59
R - System Reset		RESET - 21		RESET - 29
P - Processor Extension Request				PEREQ - 61
T - Trace Mode Selection				*2
W - Use Target Ready Signal (Wait State) when Accessing Emulation Memory				*3

*1 Software control; once enabled, emulation memory will not distinguish any memory type. When disabled, only the code segment can be mapped (loaded) to emulation memory. After reset, default for all segments is in emulation memory.

*2 These are software control functions; no pin signals are affected. Only use the Enable for mode selection (section 2.9).

*3 If memory selection is at HUEM, the on-board ready signal is selected by disabling the emulation control signal "W", and the target ready signal is selected by enabling control signal "W". However, if target memory is selected, then the status of control signal "W" will be disregarded.

Notes for 80286

1) Remember that MICE automatically enters Debug Mode after a cold/warm start, where NMI is always disabled, and control signals "W" and "PEREQ" are initially disabled (i.e. subsequent status depends on input of D/E commands).

2) If the status of the NMI signal input from the target is (TTL) high when operating in Trace mode (section 2.9), inputting "DN" and then "EN" commands will generate an NMI to the emulation CPU.

6.1 Disable/Display Control Signal Command - D

D[control-signal]

D — is the command keyword for Disable. If no signals are specified, keying in "D" will display the control signals that are disabled.

control-signal — is a single alphabetic character indicating a control signal to the target processor.

The control signal specified is deactivated at the pin of the target processor. When resetting the processor using the X command (section 5.7) these signals are activated if MICE-II is connected to a power-applied target system. Note that control signal C, for 8086/88(MIN/MAX) emulation memory type, is always activated after resetting the processor.

Example: Deactivate the interrupt request (INTR) control signal, thereby not processing any interrupts which may occur. Note that this command does not affect the interrupt mask list in the processor status register.

```
>DI
>
```

Example: Display the disabled control signals.

```
>D
DISABLE-INTR
>
```

6.2 Enable/Display Control Signal Command - E

E[control-signal]

E — is the command keyword for Enable. If no signals are specified, keying in "E" will display the control signals that are enabled.

control-signal — is a single alphabetic character indicating a control signal to the target processor.

The control signal specified is activated at the pin of the target processor. When resetting the processor with the X command (section 5.7), these signals, except for C [for 8086/88(MIN/MAX) emulation memory type], are not activated unless MICE-II is connected to a power-applied target system. Note that control signal C, is always activated after resetting the processor.

Example: Activate the interrupt request control signal.

```
>EI
>
```

Example: Display the enabled control signals.

```
>E
ENABLE-INTR
>
```

CHAPTER 7
EMULATION AND TRACE CONTROL COMMANDS

These commands perform free-running emulation or tracing for the target processor in real time. Program operation may be recorded with user defined trigger addresses to specify the start or end of tracing. Up to 2048 machine cycles can be recorded. In Forward and Backward Trace commands, the trigger address is set for BUS activity and not the actual program counter. If the trigger address must be specified after a JMP or conditional Jump/Branch instruction, set it four or six bytes beyond the actual program counter for Forward and Backward Trace. Note that the Execution Breakpoint does not have this limitation.

Absolute addresses (e.g. 1234) are used for the L command, logical addresses (e.g. 6000:1234) for G/BP commands, and absolute or logical addresses may be used for H/F/B commands. All referenced commands are described in this chapter.

Status information displayed by these commands use the following column headings: "ADDRESS DATA STATUS SEGMENT IE SPARE (7/8 BITS) DMA FLUSH"

where:
- ADDRESS is the hexadecimal value on the address bus.
- DATA is the hexadecimal value on the data bus.
- STATUS is the type of processor activity.
- SEGMENT is the type of memory segment used. (8086/88 only)
- IE is the status of interrupt. (8086/88 only)
- SPARE is the status of the headers on the personality (CEP) board.
- DMA indicates a DMA cycle in the 80186/188 if a "1" appears (for C/L commands).
- FLUSH indicates that the instruction queue of the emulation processor has been flushed if a "1" appears (for C/L commands). (8086/88 and 80186/188 only)

The specific type of processor activity that can be displayed in the status column includes:

A - Interrupt Acknowledge
H - Program Halt
I - Port Input
O - Port Output
P - Passive (not applicable for 80286)
R - Memory Read Cycle
S - Instruction Fetch Cycle
W - Memory Write Cycle

Trace point connections are also provided for monitoring external signals. There is a 7 or 8 post stick header on the CEP board designated XO-X6/X7 from right to left. Any cable or wiring can be used to connect from these test points to the target. The monitored signals are called spare bits and support concurrent trace with the address, data and status bus. Any of these bits can be monitored by trace commands to provide hardware status for display (under the SPARE column) in List, Cycle Step and Instruction Step commands; where "1" represents a high (or floating) signal and "0" indicates a low signal.

7.1 Go/Execution Command - G

G[[segment:]address]

G is the command keyword for Go/Execution.

segment is the segment value of a hexadecimal address in the target processor's program memory where emulation is to begin. (The default for memory type is the current CS.)

address is the offset (instruction pointer) of a hexadecimal address of the target processor's program memory where emulation will begin. (The default for the offset address is the current instruction pointer.)

The target processor's program counter is first set to the address indicated. MICE-II then starts realtime emulation of the target processor. If no address is specified, program emulation will start at the current program counter.

Example: Display register content (PC=00000H); then begin emulation from address 100H.

```
>R
 AX= FFFF BX= FFFF CX= FFFF DX= FFFF SP= FFF1 BP= FFFF SI= FFFF DI= FFFF
 DS= 0000 SS= 0000 ES= 0000 CS= 0000 IP= 0000 FS= F002 PC= 00000
>G100
>
```

Example: Display register content, and resume emulation from the current PC.

```
>R
 AX= 0000 BX= 225F CX= F110 DX= E156 SP= 01CA BP= F110 SI= FFFF DI= DC88
 DS= FFFF SS= E117 ES= 0000 CS= 8013 IP= 00D9 FS= F417 PC= 80209
>G
>
```

Note: ?/!/D/E/L/BS* commands may be input when the processor is executing G, FR (section 7.3) or BR (section 7.4) commands, without halting emulation. (*BS is a BPP command.)

7.2 Halt/Breakpoint Set Command - H

```
H[0[1|2]|1 [addx[ count[ qualifier]]]|2 [address-2]]
```

H is the command keyword for Halt/Breakpoint Set.

0 is the command to clear a breakpoint.

1 is breakpoint 1.

2 is breakpoint 2. (Count, qualifier and wildcard do not apply.)

addx is an address setting for breakpoint 1. Either absolute or logical addresses can be used. A wildcard byte pair of "XX" or a wildcard nibble of "X" [only for the fifth digit (e.g. X2345)] can be used for physical addresses.

count is a hexadecimal value from 1 to 4000H which specifies the number of times the indicated condition must be matched before emulation stops.

qualifier is a single alphabetic character indicating the type of processor activity to be selected for the breakpoint (page 7-1).

address-2 is a hexadecimal address setting for breakpoint 2. Physical or logical addresses can be used; but no wildcard, count or qualifier is permitted.

7.2.1 Halt

H

Input "H" during emulation or tracing to stop execution immediately and display the current address.

Example: Stop emulation and display the current address.

```
>G
>H
PROGRAM BROKE AT ADDRESS 00003
>
```

Note: Trace commands do not display the program address for the 80186/188 or 80286. Instead, MICE displays "PROGRAM BROKE AT BREAKPOINT-X", where X(1 or 2) stands for the breakpoint number.

7.2.2 Set Breakpoint

H 1[addx[count [qualifier]]]|2 address

MICE-II supports two realtime breakpoints. Breakpoint 1 can be used to specify a program address where emulation will stop. A count may also be specified to define the number of times the indicated condition is to be matched before emulation stops. If no count is specified, the default is one and emulation stops at the first breakpoint address matched. A qualifier can also be selected to choose the type of processor activity which must be matched along with the breakpoint address and count to stop emulation. If no qualifier is specified, all machine cycles are chosen. To enter a qualifier, a count must first be defined.

Breakpoint 2 can be used to specify a program address where emulation or tracing will stop. When setting breakpoint 2, set the breakpoint address only.

Example: Set breakpoint 1 to address 100H and count to 5.

```
>H1 100 5
>
```

Example: Set breakpoint 2 to address 200H.

```
>H2 200
>
```

Breakpoints 1 and 2 only set breakpoint conditions. Unless the target CPU is in execution, an emulation or trace command must be input to start the CPU. If H1 or H2 is matched, MICE-II will stop the target CPU and display the current address (at the specified breakpoint address), but if the breakpoint is an odd address for a word operation, emulation will stop at the end of the next cycle. Breakpoint 1 will reset automatically after the Trace command since it shares the same logic as the Backward/Forward Trace commands. Note that breakpoints are not active in Cycle Step and Instruction Step commands.

Note: When either H1 or H2 is matched for the 8086/88(MAX), 80186/188 or 80286, the CPU actually stops one or two cycles beyond the specified breakpoint.

Example: Set an emulation program at 6210H; then set breakpoint 1 to 6216H, count to 1, qualifier for instruction fetch cycle (S), and breakpoint 2 to 6213H.

```
>J600:210
>H1 600:216 1 S
>H2 600:213
>G
PROGRAM BROKE AT ADDRESS 06216
>
```

Example: Set the breakpoint address to XX12H, the count to one and qualifier for memory write cycle (W).

```
>H1 XX12 1 W
>G
PROGRAM BROKE AT ADDRESS 03012
>
```

7.2.3 Display Breakpoint

H 1|2

Input H1 or H2 to display breakpoint 1 or breakpoint 2 messages respectively.

Example: Display breakpoint 1 and 2 messages. Note that the address, count and qualifier are displayed for breakpoint 1; and only address is displayed for breakpoint 2.

```
>H1
H1=XX12 1 W
>H2
H2=600:213
>
```

7.2.4 Clear Breakpoint

H 0[1|2]

MICE-II can clear all breakpoints concurrently or separately.

Example: Clear all breakpoints concurrently.

```
>H0
>
```

Example: Clear breakpoint 1 only.

```
>H01
>
```

7.3 Forward Trace Command - F

F[R]addx[count[qualifier]]

F is the command keyword for Forward Trace.

R continues running the target CPU after the trace stops.

addx is a trigger address where MICE-II begins recording trace information. Either absolute or logical addresses can be used. A wildcard byte pair of "XX" or a wildcard nibble of "X" [only for the fifth digit (e.g. X2345)] can be used for physical addresses.

count is a hexadecimal value from 1 to 4000H which specifies the number of times the target condition must be matched before the trace records information.

qualifier is a single alphabetic character indicating the type of processor activity that must be matched with the trigger address (page 7-1).

Forward tracing starts realtime emulation of the target and begins recording target status information when the trigger condition is matched. Forward trace will stop when the trace buffer is full or breakpoint 2 is reached. Forward trace can record up to 2048 machine cycles. (However if the last cycle recorded is followed by passive bus cycles, then only a maximum of 2047 cycles can be recorded.)

When the trace stops, MICE-II halts the target CPU at the break address, but if the break is an odd address for a word operation, emulation will stop at the end of the next cycle. The recorded information can be examined by using the List Trace Buffer command.

Note: When the trace stops for the 8086/88(MAX), 80186/188 or 80286, the CPU actually stops one or two cycles beyond the break address.

Option R allows the target CPU to continue running after the trace stops but breakpoint 2 will force the CPU to stop. Without option R, the CPU will stop whenever the trace stops. If only the trigger address is specified, all machine cycles after that trigger address is encountered are recorded. If a count is specified, the trace will record information after the number of matches of the indicated trigger condition occur. If no count is specified, the trace will record information as soon as the trigger condition is matched.

A single qualifier may be specified to choose the type of processor activity associated with the trigger address; default is for all machine cycles. To enter a qualifier, a count must first be defined. When specified, the trace will start recording only when the trigger address together with the qualifier occurs the number of times defined by count.

Example: Begin recording program status from the trigger address of 06214H. In the following message, STEP indicates the last trace frame recorded. The trace count begins at zero, so the actual value is one more than the step number shown. The trace buffer is completely filled if STEP 07FF is displayed.

```
>F6214
THE TRACE STOPS AT STEP 07FF
>
```

Depending on command specifications, the elapse time before the buffers fill may be long. Entering <ESC> terminates the trace, yet retains the information already recorded.

Example: Specify a trigger address of 0095H which the program never reaches; the trace buffer is emptied. Note that if breakpoint 2 is encountered before the trigger address is reached, the message will be displayed without entering <ESC>.

```
>F95
<ESC>
FORWARD TRACE FAILS
>
```

Example: Begin tracing when memory location 6200H is addressed and stop at breakpoint 2 before the trace buffer is filled.

```
>J210
>H2 600:21A
>F600:200
THE TRACE STOPS AT STEP 0005
PROGRAM BROKE AT ADDRESS 0621A
>
```

Example: Begin recording status information the fifth time the program writes to an address in the range of 3000 to 30FF. Continue running after the trace stops until the program addresses 401EH. Note that qualifiers are only associated with the trigger address; all machine cycles are recorded after the trace has begun.

```
>J210
>H2 401E
>FR 30XX 5 W
THE TRACE STOPS AT STEP 0015
PROGRAM BROKE AT ADDRESS 0401E
>
```

7.4 Backward Trace Command - B

B[R]addx[count[qualifier]]

B is the command keyword for Backward Trace.

R continues running the target CPU after the trace stops.

addx is a trigger address where emulation is to stop. Either absolute or logical addresses can be used. A wildcard byte pair of "XX" or a wildcard nibble of "X" [only for the fifth digit (e.g. X2345)] can be used for physical addresses.

count is a hexadecimal value from 1 to 4000H which specifies the number of times the target condition must be matched before the trace stops recording information.

qualifier is a single alphabetic character indicating the type of processor activity that must be matched with the trigger address (page 7-1).

Backward tracing starts realtime emulation of the target and immediately begins recording target status information until the trigger address or breakpoint 2 is reached. When the trace is matched, MICE-II halts the target CPU at the specified break address, but if the break is an odd address for a word operation, emulation will stop at the end of the next cycle.

Note: When the trace is matched for 8086/88(MAX), 80186/188 or 80286, the CPU actually stops one or two cycles beyond the specified breakpoint.

Backward trace can record up to 2048 cycles and the recorded information can be examined using the List Trace Buffer command. If more than 2048 cycles elapse before backward tracing ends, only the last 2048 cycles are recorded. Option R allows the target CPU to continue running after the trace stops, however breakpoint 2 will force the CPU to stop and display a breakpoint match message.

Without option R, the CPU will stop whenever the trace stops. If only the trigger-address is specified, all machine cycles are recorded up to and including the trigger address encountered. If a count is specified, backward tracing will record all information before the number of matches of the indicated trigger address. If no count is specified, the trace will stop after the first match.

A single qualifier may be specified to choose the type of processor activity associated with the trigger address. To enter any qualifier, a count must first be defined.

Example: Begin tracing immediately and stop when memory location F000H is addressed. Note that the trace will also stop when breakpoint 2 is encountered.

```
>J210
>B600:214
THE TRACE STOPS AT STEP 02
>
```

In the above example, the status line following the command line indicates that 02H cycles have been recorded. The counter is a hexadecimal value from 0 to 7FFH. The trace count begins at zero, so the actual value is one more than the step number shown.

Again, depending on the specification, elapse time before the trigger address is encountered may be long. Entering an <ESC> terminates the trace, yet retains the information already recorded.

Example: Specify a trigger address of 0095H which the program never reaches. The trace buffer is filled with the cycles immediately before <ESC> is received.

```
>B95
<ESC>
THE TRACE STOPS AT STEP 07FF
>
```

Example: Begin tracing immediately and stop at the 16th write operation into address range 3000H-30FFH. Continue running after the trace stops, until the program addresses 401EH. Note that qualifiers are only associated with the trigger address; all machine cycles are recorded once the trace begins.

```
>J210
>H2 401E
>BR 30XX 10 W
THE TRACE STOPS AT STEP 0026
PROGRAM BROKE AT ADDRESS 0401E
>
```

7.5 Execution Breakpoint Command - BP

BP<addr1 [addr2 [addr3 [addr4]]]> [IDT = a1]

BP is the command keyword for Execution Breakpoint tracing.

addr1-4 are up to four logical addresses that specify breakpoints where emulation is to stop. Note that both the segment value and offset must be specified. If more than one breakpoint is specified, angular brackets < > must enclose the addresses to indicate a logical OR construct.

All four addresses serve as software breakpoints, except when using addr1 for the 80286, where it serves as a hardware breakpoint without any limitation on execution in either RAM or ROM.

a1 is the base value (in the range of 0H-FFFFFFH) for the Interrupt Descriptor Table (IDT) register. This value is only required for input if it will be changed by the user's program. If this parameter is omitted, default is for the current base value when tracing is initiated. (This parameter is only valid for the 80286.)

The Execution Breakpoint command starts real time emulation of the target; it immediately begins recording target status information, and stops tracing when any of up to four permissable breakpoints is executed. The trace can record up to 2048 machine cycles. If more than 2048 cycles elapse before the trace ends, only the last 2048 cycles are recorded. The recorded information can be examined by using the List Trace Buffer command.

The CPU stops whenever the trace stops. All machine cycles are recorded up to the trigger-address, including prefetch cycles (although the prefetch cycles do not influence program execution).

Although the trace buffer records all bus activity including prefetch cycles, the BP command causes the CPU to break at a point in the program where a specified address is actually to be executed and not just prefetched.

A realtime Execution Breakpoint is achieved by inserting a software interrupt (ie. code INT 3 [0CCH]) into memory at the specified breakpoint address. A Halt breakpoint is then set at the interrupt vector fetch address. This ensures that the execution breakpoint is triggered only if the interrupt vector fetch cycle is executed, ignoring all prefetch cycles.

Example: The following program segment illustrates the difference between the Execution Breakpoint command and traditional breakpoints based on bus activity.

```
   :        :
   :        :
BRANCH: MOV SI,#4000  ;Source pointer for data read.
        MOV DI,#3000  ;Destination pointer for data write.
        MOV CX,#10    ;Move 10 words of data.
        REP           ;From 4000H to 3000H.
        MOVS, WORD
  DONE: JMP BRANCH    ;Finish and restart.
   :        :
   :        :
```

Setting a traditional breakpoint at DONE in the above example would cause the program to break at the bus prefetch cycle, even though program execution has not reached the breakpoint. Also note that if WORD access is used, and a traditional breakpoint is set at an odd address, the program may never break because program data is fetched by word addresses.

If the same trigger-address is set using the BP command, then execution will jump to another area without reaching the breakpoint. The program will break only if execution reaches the breakpoint address (either odd or even).

Example: Messages for the Execution Breakpoint command are illustrated below.

```
>X0                                          ;Default is for CS=100, IP=0
>M0 FF 90                                    ;Fill memory 0-FF with NOP
>BP <600:6,0:8>                              ;
TRACE STOPS AT 0600:0006, STEP 0005          ;Current register values are
                                             ; CS=100, IP=8
>BP <600:B,600:0                             ;Right bracket > is optional
TRACE STOPS AT 0600:000B, STEP 0025          ;
>BP 1000:560                                 ;
ADDRESS 1000:0560 PROTECTED, COMMAND ABORTED! ;Memory protected address
>BP 600:B                                    ;
PC=0600:000B, TRACE ABORTED!                 ;PC=1050 already
>
```

APPLICATION NOTES

1) During execution of the BP command, breakpoints H1-6 are all disabled automatically; other than H1 they will be reenabled when execution has completed. (H3-6 are BPP options.)

2) Stack contents from SP-1 thru SP-6 will be modified if the breakpoint (INT 3 code) is executed; but the SS and SP registers will not be affected.

3) Memory for breakpoint addresses is modified to OCCH during command execution*; the List command will therefore display this data, instead of the user's code, for instruction prefetch cycles. This has no effect on program execution, though, as the original data is restored after the interrupt instruction is executed.

   * Because the first breakpoint address of the 80286 is a hardware execution breakpoint, the data at this memory address will not be modified. The List command will display the user's code when this address is located in HUEM and OFFH when it is located in the target.

4) When the program stops at the breakpoint address, the user's program instruction at that location has not yet been executed. The same trigger-address cannot be specified again. Setting any breakpoint where the PC equals the trigger's physical address will generate an error message: "PC=trigger address, TRACE ABORTED!"

5) For the 80186/188, Memory Chip Select logic will be active only after the initialization program is executed. Therefore if target memory is connected with chip select from the 80186/188, whenever the CPU is reset, the initialization program should first be executed before a breakpoint command is entered.

6) INT 3 instructions defined in the user's program are executed correctly without any conflict with internal INT 3 codes generated by BP commands.

7) The right angular bracket ">" is optional (illustrated in the following example, line 5). However, this bracket is mandatory if the base value of the IDT is specified for the 80286.

LIMITATIONS

The following limitations apply to the Execution Breakpoint:

1) The emulation program must be placed in RAM, where it will be tested before the trace starts; except when using addr1 for the 80286, which may be placed in either RAM or ROM.

2) If any memory read cycle occurs at 0000:000CH (or an INT 3 instruction in the user's program is executed), a few non-realtime cycles will be inserted to process this code.

3) Breakpoints must be set at the first byte of the instruction.

4) To avoid illegal operation with string instructions (e.g. MOVS, SCAS, OUTS, etc.), treat repetitive instructions (e.g. REPE, REPNE, REPNZ, etc.) and string instuctions as one instruction. Set the trigger address at repetitive instructions only, and never at string instructions. In the above example, the trigger address should never be set at instruction MOVS; set it at instruction REP instead.

In addition to those listed above, the following limitations also apply to the 80286:

1) The BP command can only be used in MICE Debug Mode.

2) The limit of the IDT cannot be set to less than 0FH, otherwise the emulation processor would hang up.

7.6 List/Display Trace Buffer Command - L

```
L[step[ address-1[ address-2[ qualifier(s)]]]|S[step]|Z[step]|N]
```

L — is the command keyword for List/Display Trace Buffer.

step — is a hexadecimal value from 0 to 7FFH indicating the initial cycle step to display.

address-1 — is an absolute hexadecimal address indicating the starting address of the range to be listed.

address-2 — is an absolute hexadecimal address indicating the ending address of the range to be listed.

qualifier(s) — are single characters indicating the type of processor activities to list (page 7-1). Note that qualifiers H and S are not displayed for the 8086(MIN).

S — lists the trace buffer using mnemonic code. Not applicable for MICE-II 8086(MIN).

Z — lists the trace buffer displaying mnemonic code and all machine statuses. Not applicable for MICE-II 8086(MIN).

N — displays the frame number where the trace stopped.

Status information recorded during a trace command is displayed using this command. If a step is specified, MICE-II lists the trace buffer from the specified step to the last recorded step. If no step is specified, MICE-II will list all records in the trace buffer. An optional address range can be entered to specify the range to be listed. If no address range is specified, then the display range is from 00000H to FFFFFH (000000~FFFFFFH for 80286).

The type of information to be listed is specified by entering qualifier(s). If no qualifier is entered, all machine cycles within the specified address range are listed. To enter a qualifier, an address range must first be defined. To enter an address range, a step must first be entered. The List command accepts multiple qualifiers, and displays only machine cycles that match the specified qualifiers. If the buffer is listed with the LS or LZ command, only "step" can be specified.

Example: List all records in the trace buffer. Note that the list operation can be terminated by entering an <ESC>. Note that FRAME is the sequence number of the trace step in hexadecimal; the range is 0 to 2047 (7FFH).

```
>L
   FRAME ADDRESS DATA STATUS SEG IE  SPARE  FLUSH
   0000   06210  00BE   S     CS  0 1111111   0
   0001   06212  BF40   S     CS  0 1111111   0
   0002   06214  3000   S     CS  0 1111111   0
   0003   06216  10B9   S     CS  0 1111111   0
   0004   06218  F300   S     CS  0 1111111   0
   0005   0621A  CCA5   S     CS  0 1111111   0
   0006   0621C  00F3   S     CS  0 1111111   0
   0007   0621E  2402   S     CS  0 1111111   0
   0008   04000  4241   R     DS  0 1111111   0
   0009   03000  4241   W     ES  0 1111111   0
   000A   04002  2043   R     DS  0 1111111   0
   000B   03002  2043   W     ES  0 1111111   0<ESC>
>
```

Example: List the trace buffer from the 6th step to the last step, and terminate the operation before the listing is completed.

```
>L5
   FRAME ADDRESS DATA STATUS SEG IE  SPARE  FLUSH
   0005   0621A  CCA5   S     CS  0 1111111   0
   0006   0621C  00F3   S     CS  0 1111111   0
   0007   0621E  2402   S     CS  0 1111111   0
   0008   04000  4241   R     DS  0 1111111   0
   0009   03000  4241   W     ES  0 1111111   0
   000A   04002  2043   R     DS  0 1111111   0
   000B   03002  2043   W     ES  0 1111111   0<ESC>
>
```

Example: List the trace buffer write cycles in the range 621CH to 621EH starting at step 2.

```
>L2 621C 621E
   FRAME ADDRESS DATA STATUS SEG IE  SPARE  FLUSH
   0006   0621C  00F3   S     CS  0 1111111   0
   0007   0621E  2402   S     CS  0 1111111   0
>
```

Example: List the trace buffer from 3002H to 4004H, with qualifiers R and W.

```
>LO 3002 4004 R W
   FRAME ADDRESS DATA STATUS SEG IE  SPARE  FLUSH
   0008   04000  0AB9   R     DS  0 1111111   0
   000A   04002  A708   R     DS  0 1111111   0
   000B   03002  A708   W     ES  0 1111111   0
   000C   04004  49F5   R     DS  0 1111111   0
   000D   03004  49F5   W     ES  0 1111111   <ESC>
>
```

Example: List trace message with source code; the operation is terminated before the listing is completed. Note that the command also ends if the trace buffer is empty or if nothing is recorded within the specified range.

```
>LS
    FRAME ADDRESS DATA  SOURCE CODE
     0000  06210  00BE  MOV    SI,#4000
           06213    BF  MOV    DI,#3000
     0003  06216  10B9  MOV    CX,#0010<ESC>
>
```

Example: List trace messages with mnemonic code and all machine statuses; then display the frame number of the last trace.

```
>LZ
    FRAME ADDRESS DATA STATUS SEG IE  SPARE  FLUSH SOURCE CODE
     0000  06210  00BE   S     CS  0 1111111   0   MOV    SI,#4000
     0001  06212  BF40   S     CS  0 1111111   0   MOV    DI,#3000
     0002  06214  3000   S     CS  0 1111111   0
     0003  06216  10B9   S     CS  0 1111111   0   MOV    CX,#0100
     0004  06218  F300   S     CS  0 1111111   0   REP<ESC>
>LN
THE TRACE STOPS AT STEP 0004
>
```

Specifying an LS or LZ command with a step that starts in the middle of a two or three byte instruction, will cause several machine cycles to be disassembled incorrectly.

Example: List trace messages with a step that starts at an improper address.

```
>LZ1
    FRAME ADDRESS DATA STATUS SEG IE  SPARE  FLUSH SOURCE CODE
     0001  06212  BF40   S    CS  0 1111111   0   INC    AX
                                                 MOV    DI,#3000
     0002  06214  3000   S    CS  0 1111111   0
     0003  06216  10B9   S    CS  0 1111111   0   MOV    CX,#0100
     0004  06218  F300   S    CS  0 1111111   0   REP
     0005  0621A  EBA5   S    CS  0 1111111   0   MOVS   WORD
                                                 JMP<ESC>
>
```

Note: 80186/188 data bus messages for internal control block access are not valid because data can not be externally observed while accessing the internal control blocks.

CHAPTER 8
STEPPED EMULATION COMMANDS

These commands perform cycle-stepped or instruction-stepped emulation of the target processor in real time. Refer to page 7-1 for a description of the displayed status information. Note that breakpoints are not active for C/S commands.

Note: A blank space (ASCII 20H) is sent at the end of each displayed line immediately before the cursor for C/S commands. This code serves as a an end of data message to the host computer; and is provided to help in implementing symbolic debuggers.

8.1 Cycle Step Command - C

C[W|count]

C is the command keyword for Cycle Step.

W stops the CPU at WAIT state.

count specifies the machine cycle interval between displayed messages. The range is 0-FFFFH. Note that an input value of "0" represents 10000H.

The Cycle Step command first stops the target processor, steps the program one cycle, and then halts the processor in Hold Acknowledge state (HLDA). MICE-II displays target status for the cycle just executed, and awaits a <CR> or <LF> to advance the target processor to the next machine cycle. The new status is then displayed and MICE-II again waits. Enter a <CR> or <LF> to repeat the entire sequence, or an <ESC> to terminate the single-cycle mode of emulation.

In the C command, the displayed cycle status has already been executed. However, a CW command can be used to force the CPU to stop at WAIT state; where the displayed status has not yet been executed. Because the signals are still active, this makes it much easier to debug associated hardware signals. But for some target designs which do not allow the CPU to stop at WAIT state very long, the CW command will make the target system fail (e.g. dynamic RAM refresh).

Note: When 80286 is stopped at wait state, the status is floating on the target.

Note that if a word operand operation requires two cycles to execute (e.g. read an odd word address [for the 8086, 80186 or 80286] or read a word address [for the 8088]), MICE will display both cycles simultaneously. However, in the CW command only one cycle is displayed per step. Also, remember that breakpoints are not active for the C command.

Example: Cycle step the following program. Note that diplay for C command with WAIT option would be identical with the example below.

```
>J600:210
>C
    ADDRESS    DATA   STATUS    SEG    IE    SPARE    FLUSH
     06210     00BE      S       CS     0   1111111     0 <CR>
     06212     BF40      S       CS     0   1111111     0 <CR>
     06214     3000      S       CS     0   1111111     0 <CR>
     06216     10B9      S       CS     0   1111111     0 <CR>
     06218     F300      S       CS     0   1111111     0 <CR>
     0621A     EBA5      S       CS     0   1111111     0 <CR>
     04000     9894      R       DS     0   1111111     0 <CR>
     03000     9894      W       ES     0   1111111     0 <CR>
     04002     9894      R       DS     0   1111111     0 <CR>
     03002     9894      W       ES     0   1111111     0 <CR>
     04004     9894      R       DS     0   1111111     0 <ESC>
>
```

Example: Cycle step the above program using a count of 3.

```
>J600:210
>C3
    ADDRESS    DATA   STATUS    SEG    IE    SPARE    FLUSH
     06214     3000      S       CS     0   1111111     0 <CR>
     0621A     EBA5      S       CS     0   1111111     0 <CR>
     04002     9894      R       CS     0   1111111     0 <ESC>
>
```

8.2 Instruction Step Command - S

S[S|R][count] [Z]|[start-address end-address]

S	is the command keyword for Instruction Step.
S(option)	displays messages in mnemonic form.
R	displays register contents in addition to mnemonic code.
count	is an hexadecimal value from 0000H to FFFFH specifying the step interval between status display. Note that 0000H indicates 65536 steps.
Z	displays cycle status in addition to mnemonic code.
start-address	is a hexadecimal address of the emulation CPU indicating a memory location of the specified range.
end-address	is a hexadecimal address of the emulation CPU indicating the last memory location of the specified range.

8.2.1 Instruction Step

S

If count is not specified for instruction step, the default is one. In this mode, MICE-II first displays the current instruction to be executed and waits for input. To cause the target processor to execute this instruction, a <CR> or <LF> should be entered; the new status is then displayed and MICE-II again waits. Enter a <CR> or <LF> to repeat the entire sequence, displaying the new status after every instruction. Enter an <ESC> to terminate the single step mode of emulation. Note that breakpoints are not active for the S command.

Instruction step only displays "S" status to indicate an instruction fetch cycle; no other status types are displayed. If a more careful examination of a program is required, first Instruction Step to the problem area, and then Cycle Step through the problem area for a more detailed display.

Example: Instruction step the previous program. Note that the processor does not begin emulation until the current status displays and either a <CR> or <LF> is entered.

```
>J210
>S
    ADDRESS    DATA   STATUS    SEGMENT   IE   SPARE (7 BITS)
     06210     00BE      S         CS      0   1111111 <CR>
     06213       BF      S         CS      0   1111111 <CR>
     06216     10B9      S         CS      0   1111111 <CR>
     06219       F3      S         CS      0   1111111 <CR>
     06219       F3      S         CS      0   1111111 <CR>
     06219       F3      S         CS      0   1111111 <ESC>
>
```

8.2.2 Instruction Step in Mnemonic Form

SS

Example: Instruction step with S option.

```
>J210
>SS
    ADDRESS    DATA   SOURCE   CODE
     06210     00BE    MOV     SI,#4000 <CR>
     06213       BF    MOV     DI,#3000 <CR>
     06216     10B9    MOV     CX,#0010 <CR>
     06219       F3    REP
                       MOVS    WORD <CR>
     06219       F3    REP
                       MOVS    WORD <CR>
     06219       F3    REP
                       MOVS    WORD <ESC>
>
```

8.2.3 Instruction Step with Register Content

SR

The R option displays register content along with mnemonic information. All register contents associated with the current step in program execution can be observed; but remember that the contents displayed are prior to executing the instruction.

Example: Instruction step with R option.

```
>J210
>SR
          ADDRESS DATA      SOURCE CODE
  AX= FF86 BX= 6E86 CX= 000D DX= 8686 SP= 9762 BP= 9760 SI= 4006 DI= 3006
  DS= 0000 SS= 0000 ES= 0000 CS= 0600 IP= 0210 FS= F002 PC= 06210
        06210   00BE      MOV      SI,#4000 <CR>
  AX= FF86 BX= 6E86 CX= 000D DX= 8686 SP= 9762 BP= 9760 SI= 4000 DI= 3006
  DS= 0000 SS= 0000 ES= 0000 CS= 0600 IP= 0213 FS= F002 PC= 06213
        06213     BF      MOV      DI,#3000 <CR>
  AX= FF86 BX= 6E86 CX= 000D DX= 8686 SP= 9762 BP= 9760 SI= 4000 DI= 3000
  DS= 0000 SS= 0000 ES= 0000 CS= 0600 IP= 0216 FS= F002 PC= 06216
        06216   10B9      MOV      CX,#0010 <CR>
  AX= FF86 BX= 6E86 CX= 0010 DX= 8686 SP= 9762 BP= 9760 SI= 4000 DI= 3000
  DS= 0000 SS= 0000 ES= 0000 CS= 0600 IP= 0219 FS= F002 PC= 06219
        06219     F3      REP
                          MOVS     WORD <CR>
  AX= FF86 BX= 6E86 CX= 000F DX= 8686 SP= 9762 BP= 9760 SI= 4002 DI= 3002
  DS= 0000 SS= 0000 ES= 0000 CS= 0600 IP= 0219 FS= F002 PC= 06219
        06219     F3      REP
                          MOVS     WORD <ESC>
>
```

8.2.4 Instruction Step with Count

S[S|R] count

If an instruction count other than one is entered, MICE-II immediately begins emulation from the current program counter until the count of the next instruction to be executed equals the number specified. The current status is then displayed and MICE-II waits for a <CR> or <LF> before executing the next "count" instruction. Enter an <ESC> to terminate the multi-step mode of emulation. For large intervals, emulation may be terminated by entering an <ESC> before the specified number of instructions are executed.

Instruction Step executes "count-1" instructions before the first status line is displayed and then executes "count" instructions between subsequent displays.

Example: Multi-step the program and compare results with the display in the first example.

```
>J210
>S3
    ADDRESS    DATA   STATUS   SEGMENT   IE    SPARE (7 BITS)
     06216     10B9     S         CS      0    1111111 <CR>
     06219       F3     S         CS      0    1111111 <CR>
     06219       F3     S         CS      0    1111111 <CR>
     06219       F3     S         CS      0    1111111 <ESC>
>
```

8.2.5 Instruction Step with Cycle Status

SZ

The SZ option includes a memory and I/O access message that provides data for bus cycles R/W/I/O/A, e.g. 01200-W(DS)-2000. The first 5 digits (01200) indicate the memory address; the capital letter (W) represents the type of bus cycle; the 2 letters in parentheses (DS) indicate memory type; and the last 4 digits (2000) are the data transferred. Note that the 80186/188 does not display memory type for this message.

Example: Instruction step the program displaying Cycle status together with mnemonic code.

```
>J210
>SZ
      ADDRESS DATA     SOURCE CODE
       06210  00BE     MOV    SI,#4000<CR>
       06213    BF     MOV    DI,#3000<CR>
       06216  10B9     MOV    CX,#0010<CR>
       06219    F3     REP
                       MOVS   WORD<CR>
        04000-R(DS)-80FF
        03000-W(ES)-80FF
       06219    F3     REP
                       MOVS   WORD<CR>
        04002-R(DS)-0020
        03002-W(ES)-0020
       06219    F3     REP
                       MOVS   WORD<ESC>
>
```

8.2.6 Instruction Step Within a Specified Range

S start-address end-address

After accept this command, MICE will first execute current instruction then check whether the PC falls into the specified address range. If PC is within the address range, MICE continues to execute, the cursor locates at the first position of the next line. At this time, execution can be aborted by inputting an <ESC>, PC is then stopped and a prompt is printed. If PC is outside the address range, execution stops and prints a prompt.

This function is useful for implementing Higer-Level-Language debugger (HLLD), Available for the following firmware revisions: MICE-II 8086MAX (V 3.3 and later), MICE-II 8086MIN (V 3.3 and later), MICE-II 80186 (V 3.3 and later) and MICE-II 80286 (V 4.0 and later).

Example: Instruction step the program within the specified range. PC is shown by issuing an R command.

```
>Z
LOC         OBJ        LINE LABEL       SOURCE CODE
0000:1000   F9         0001             STC
0000:1001   90         0002             NOP
0000:1002   FD         0003             STD
0000:1003   FC         0004             CLD
0000:1004   CE         0005             INTO
0000:1005   90         0006             NOP
0000:1006   B80080     0007             MOV       AX,#8000
0000:1009   01C0       0008             ADD       AX,AX
0000:100B   90         0009             NOP
0000:100C   CE         0010             INTO
0000:100D   E9FDFF     0011 B100D       JMP       100D
0000:1010   FFFF       0012                    ERROR CODE
>R
 AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFA BP=0000 SI=000F DI=0000
 DS=0000 SS=0000 ES=0000 CS=0000 IP=1000 FS=F002 PC=01000
>S 1000 1000
>R
 AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFA BP=0000 SI=000F DI=0000
 DS=0000 SS=0000 ES=0000 CS=0000 IP=1001 FS=F003 PC=01001
>S 1001 1002
```

```
>R
 AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFA BP=0000 SI=000F DI=0000
 DS=0000 SS=0000 ES=0000 CS=0000 IP=1003 FS=F403 PC=01003
>S 0 0
>R
 AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFA BP=0000 SI=000F DI=0000
 DS=0000 SS=0000 ES=0000 CS=0000 IP=1004 FS=F003 PC=01004
>S 1000 1007
>R
 AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFA BP=0000 SI=000F DI=0000
 DS=0000 SS=0000 ES=0000 CS=0000 IP=1009 FS=F003 PC=01009
>S 1000 1010 <CR>
<ESC>
>
```

8.3 Application Notes for Stepped Emulation Commands

1) To make debugging easier, there are two available methods for slowing down the CPU's execution speed. (The following description assumes a CPU clock of 8MHz.)

   a) Use the C command with a "Ø" count to slow execution speed as follows:

MICE-II	Speed
8086/88(MIN/MAX)	1/260
80186/188	1/210
80286	1/1250

   b) Use the S command with a "Ø" count to slow execution speed as follows:

MICE-II	Speed
8086/88(MIN/MAX)	1/25000
80186/188	1/30000
80286	1/63800

   c) Typing ahead with <CR> will keep the processor running.

```
Example: >CØ
         ><CR>
         ><CR>
         ><CR>
```

2) When executing C or S commands, or whenever the CPU stops, the bus request signal $\overline{RQ0}/\overline{RQ1}$ for the 8086/88(MAX) will not be accepted, even when the "EH" command is keyed in, until it is free-running. However, it will latch the RQ pulse.

3) 80186/188 data bus messages for internal control block access are not valid because data can not be observed externally while reading the internal control blocks.

4) If the program contains an "STI CLI" code sequence, where the location for STI is an odd address and the INTR pin is high, MICE-II will recognize the interrupt in S[S|R|Z] commands. However, it will not recognize the interrupt in the Cycle Step command. Therefore to properly execute this program sequence, first use the Disable Interrupt command before initiating the Instruction Step command.

   However, for the 80286, note that there is no limitation on executing this instruction sequence in S[S|R|Z] commands. MICE-II can execute this sequence up to 11 consecutive times for the 80286.

5) If S[S|R|Z] commands are used, the CPU will not be halted by a HLT instruction even though there is no interrupt. However, if a halt instruction is encountered during G/F/B/BP/C commands, the CPU will be halted.

6) Non-Maskable Interrupt for the 80186/188 and 8086 MAX/MIN is not accepted in the executing interval (message displaying interval) of S[S|R|Z] commands, but can be accepted one time while in the non-executing interval (awaiting key-in interval).

7) If DMA cycles occur for the 80186/188 in S[S|R|Z] commands, the current instruction will not execute until the DMA activity is over. Typing <CR> will redisplay this instruction until the DMA activity is completed.

8) MICE-II can execute the code sequence "STI CS: DS: ES: SS: LOCK REP" up to 11 times for the 80286.

CHAPTER 9
UTILITY COMMANDS INVOLVING A SYSTEM

These commands are only applicable when MICE-II is connected to a host system. Programs and data can be downloaded from a file in a host computer to memory in the target system. Conversely, information can also be uploaded from the target back to the host.

Intel and Tektronix loading formats are recognized by MICE-II. Each format is described in detail in the following pages.

Note: Because the MICE-II 80286 only supports Real Address Mode for Download and Upload commands (section 2.9), the sixth digit of the PC is always set to "0" during data transfer.

9.1 Download Command (Intel Format) - :

:load-record

: is the command keyword for Download.

load-record is a string of ASCII data containing up to 128 bytes of program information.

Each record transferred contains the record type, length, memory load address, and checksum in addition to data. Each transfer is limited to 128 bytes of program data. The general format of a record, shown with spaces separating each field, is:

RECORD MARK	RECORD LENGTH	LOAD ADDRESS	RECORD TYPE	PROGRAM DATA	CHECK-SUM
:	##	aaaa	tt	dd...dd	cc

where:

: is the keyword used to signal start of record.

is a two ASCII hexadecimal value indicating the record length, the number of data bytes in the record.

aaaa is a four ASCII hexadecimal value indicating the program memory load-address, the address at which the first byte is to be loaded. (For record types 01-03 [next item], this field contains "0000".) Successive data bytes are stored in the following memory locations.

tt is a two ASCII hexadecimal value representing the record type:

tt	##
00 - data record	actual data length
01 - end of file record	00
02 - extended address	02
03 - start address record	04

dd...dd is a two ASCII hexadecimal value per byte representation of the program. When the record type (tt) is extended address (02) the following four ASCII hexadecimal value (dddd) represents the Code Segment base for the subsequent data record. For each record type the data is as follows:

tt	dd...dd
00	A pair of hex digits representing the ASCII code for each data byte, where the high order digit is the 1st digit of each pair.
01	none
02	The Upper Segment Base Address (USBA) is a four ASCII hexadecimal value.
03	CS and IP (8 digits)

cc is a two ASCII hexadecimal value representing the negative sum of the record. Beginning with the record length "##" and ending with the checksum "cc", the hexadecimal sum, taken two at a time, modulo 256 should be zero.

When MICE-II receives a ":", it reads the record length (##). The rest of the record is then read and verified. If the incoming checksum agrees with the computed checksum, MICE-II stores the data into program memory of the target system and reprompts after the following acknowledgement response <ACK sequence> is sent:

ACK LF CR NUL NUL NUL NUL NUL NUL

If the checksum does not agree, MICE-II also reprompts but the alternate negative acknowledgement response <NAK sequence> is sent:

NAK LF CR NUL NUL NUL NUL NUL NUL

Example: Download with an end-record into a target system using Intel format. Note that the received characters are not echoed. A <CR> is not required at the end of the input line.

```
>:060000002300A8A917046B<ACK sequence>
>:00000001FF<ACK sequence>
>
```

In the above example, the first load record is

```
##      = 06H
aaaa    = 0000H
tt      = 00H
dd...dd = 23H, 00H, A8H, A9H, 17H, 04H
cc      = 6BH
```

where: 06H+00H+00H+00H+23H+00H+A8H+A9H+17H+04H+6BH = 00H

For the end-record,

```
##      = 00H
aaaa    = 0000H
tt      = 01H
cc      = FFH
```

where: 00H+00H+00H+01H+FFH = 00H

Example: Download with an end-record into a target system using Intel type 2 format.

```
>:020000020020DC<ACK sequence>
>:10000000004992DB246DB6FF4891DA263CB5FE47E5<ACK sequence>
>:0600100090D9226BB4FD43<ACK sequence>
>:020000020036C6<ACK sequence>
>:100000000062C42688EA4CAE1072D43698FA5CBE00<ACK sequence>
>:080010002082E446A80A6CCE30<ACK sequence>
>:00000001FF<ACK sequence>
>
```

The download command must first transmit 02 to define the Code Segment base of the subsequent data record. End-of-record must then be specified by 01 in the last line of transmission.

9.2 Download Command (Tektronix Format) - /

/load-record

/ is the command keyword for Download.

load-record is a string of ASCII data containing up to 128 bytes of program information.

Each record transferred contains the record type, length, memory load address and checksum in addition to data. Each transfer is limited to 128 bytes of program data. The general format of a record, shown with spaces separating each field, is:

RECORD MARK	LOAD ADDRESS	RECORD LENGTH	LOAD SUM	PROGRAM DATA	CHECK-SUM
/	aaaa	##	ss	dd...dd	cc

where:

/ is the keyword used to signal start of record.

aaaa is a four ASCII hexadecimal value indicating the program memory load-address, the address at which the first byte is to be loaded. Successive data bytes are stored in the following memory locations.

is a two ASCII hexadecimal value indicating record length, the number of data bytes in the record. A record length of zero indicates end-of-file.

ss is a two ASCII hexadecimal value representing the sum of the preceding six digits, load-address and record length.

dd...dd is a two ASCII hexadecimal value per byte representation of the program.

cc is a two ASCII hexadecimal value representing the sum of the digits comprising the data, modulo 256.

When MICE-II receives a "/", input is read until a <CR> terminates the line. Checksums are then computed and compared. If the incoming checksum agrees with the computed checksum, MICE-II stores the data into program memory of the target system and reprompts after the following acknowledgement response <ACK sequence> is sent:

ACK LF CR NUL NUL NUL NUL NUL NUL

If the checksum does not agree, MICE-II also reprompts but the alternate negative acknowledgement response <NAK sequence> is sent:

NAK LF CR NUL NUL NUL NUL NUL NUL

Example: Download with an end-record into a target system using Tektronix format.

```
>/0000006062300A8A9170436<CR>
<ACK sequence>
>/00000000<CR>
<ACK sequence>
>
```

In the above example, the load record is

```
aaaa     = 0000H
##       = 06H
ss       = 06H
dd...dd = 23H, 00H, A8H, A9H, 17H, 04H
cc       = 36H
```

where: ss = 0H+0H+0H+0H+0H+6H = 06H
cc = 2H+3H+0H+0H+AH+8H+AH+9H+1H+7H+0H+4H = 36H

For the end-record,

```
aaaa     = 0000H
##       = 00H
ss       = 00H
```

where: ss = 0H+0H+0H+0H+0H+0H = 00H

9.3 Upload Command - U

U [type] start-address end-address [format]

U — is the command keyword for Upload.

type — are the alphabetic characters indicating the type of memory to be used - CS, DS, ES, SS or a four digit hexadecimal segment value followed by a colon (:); default is for CS.

start-address — is a logical hexadecimal address of the target processor's memory where data transfer begins.

end-address — is the last logical hexadecimal address of the target processor's program memory where data transfer ends.

format — is a single alphabetic character [I|T] indicating the load-record format to be used.

I - Intel
T - Tektronix

Memory contents defined by the memory range start-address through end-address are transferred to the host system using either Intel or Tektronix format. If the format is not specified, then Intel is used. Also, the end-address must be greater than or equal to the start-address or an error message is sent and the command ended.

MICE-II first sends the current segment value record to the host system. It then reads data from memory and sends a maximum of 32 bytes (depending on data length) to the host system using one of the above formats. Data transmission continues in this manner until an end address is sent. MICE-II then sends an end-of-record file and prompts for the next command.

If anything other than an <ACK sequence> is received when MICE-II is waiting for an acknowledgement response, the same block is retransmitted. If after five retries transmission is still unsuccessful, the command is aborted, and an error message is sent to the host system before MICE-II prompts for the next command. The command can also be aborted by the host system when MICE-II receives an <ESC> character from the console.

Example: Upload from MICE-II using Intel format. Note that load records are shown on separate lines for clarity. No <CR> or <LF> characters are generated by MICE-II, and anything received other than an <ACK sequence>, is treated the same as a <NAK sequence>.

```
>U 0 6 I<CR>
:020000020000FC<ACK sequence>
:060000002300A8917046E8<ACK sequence>
:00000001FF<ACK sequence>
>
```

Example: Upload using Tektronix format.

```
>U 0 6 T<CR>
/0000006062300A8A9170436<CR>
<NAK sequence>
/0000006062300A8A9170436<CR>
/00000000
<ACK sequence>
>
```

In the above example, the host system did not acknowledge the first transmission when Tektronix format was used. Hence, the first line was retransmitted until an <ACK sequence> was received.

Appendix A
Summary of MICE-II Commands

MICE-II UTILITY COMMANDS

? - Help Command

! - Attention Command

K - Clock Selection (80286 only)
K[I|T]

!M - Coprocessor Mode Selection (80286 only)
!M[87|287]

Mode - Real/Protected Virtual Address Mode Selection (for 80286 only)

MEMORY, PORT AND REGISTER COMMANDS

M - Memory Display/Examine/Modify/Fill/Search Command
M[[W][type]start-address[end-address[data-1...data-8]][S]]

T - Memory Checksum/Test/Transfer/Compare Command
T[type]start-address end-address {S|M|address-3[V]}

A - Line Assembly command
A[[type]start-address]

Z - Disassembly Command
Z[[type]start-address[end-address]]

I - Port Input Command
I[W] port[count[duration]]

O - Port Output Command
O[W] port data-1[...data-8]

X - Reset/Initialization Command
X[segment[:address]]

R - Register Display/Modify Command
R[register]

J - Jump/Branch Command

CONTROL SIGNAL COMMANDS

D - Disable/Display Control Signal Command
D[control signal]

E - Enable/Display Control Signal Command
E[control signal]

EMULATION AND TRACE CONTROL COMMANDS

G - Go/Execution Command
G[segment[:address]]

H - Halt/Breakpoint Set Command
H[0[1|2]|1[address-1[count[qualifier]]]|2 [address-2]]

F - Forward Trace Command
F[R]trigger-address[count[qualifier]]

B - Backward Trace Command
B[R]trigger-address[count[qualifier]]

BP - Execution Breakpoint Command
BP<address-1 [address-2 [address-3 [address-4]]]> [IDT = a1]

L - List/Display Trace Buffer Command
L[step[address-1[address-2[qualifier(s)]]]|S[step]|Z[step]|N]

STEPPED EMULATION COMMANDS

C - Single Cycle, Step Command
C[W|count]

S - Instruction Step Command
S[S|R][count][Z]

UTILITY COMMANDS INVOLVING A SYSTEM

: - Download Command (Intel Format)
    :load-record

/ - Download Command (Tektronix Format)
    /load-record

U - Upload Command
    U[type]start-address end-address [I|T]

Appendix B
Hexadecimal-Decimal Conversion

To find the decimal equivalent of a hexadecimal number, first locate your hex number in the correct position (1st through 4th place digit) in the following table. Note the decimal equivalent for each hex digit in your number and then add up these decimal values.

To find the hexadecimal equivalent of a decimal number, locate the next lower decimal number in the table, write down the hex equivalent and its position (1st through 4th place digit). Subtract this decimal number from yours, take the difference and continue this process until conversion is completed.

BYTE				BYTE			
4th place digit		3rd place digit		2nd place digit		1st place digit	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0
1	4096	1	256	1	16	1	1
2	8192	2	512	2	32	2	2
3	12288	3	768	3	48	3	3
4	16384	4	1024	4	64	4	4
5	20480	5	1280	5	80	5	5
6	24576	6	1536	6	96	6	6
7	28672	7	1792	7	112	7	7
8	32768	8	2048	8	128	8	8
9	36864	9	2304	9	144	9	9
A	40960	A	2560	A	160	A	10
B	45056	B	2816	B	176	B	11
C	49152	C	3072	C	192	C	12
D	53248	D	3328	D	208	D	13
E	57344	E	3584	E	224	E	14
F	61444	F	3840	F	240	F	15

Appendix C
ASCII Code Lists and Definitions

Table C-1
ASCII Code List

HEX	DEC	CHAR	HEX	DEC	CHAR
00	0	NUL	22	34	"
01	1	SOH	23	35	#
02	2	STX	24	36	$
03	3	ETX	25	37	%
04	4	EOT	26	38	&
05	5	ENQ	27	39	'
06	6	ACK	28	40	(
07	7	BEL	29	41	)
08	8	BS	2A	42	*
09	9	HT	2B	43	+
0A	10	LF	2C	44	,
0B	11	VT	2D	45	-
0C	12	FF	2E	46	.
0D	13	CR	2F	47	/
0E	14	SO	30	48	0
0F	15	SI	31	49	1
10	16	DLE	32	50	2
11	17	DC1	33	51	3
12	18	DC2	34	52	4
13	19	DC3	35	53	5
14	20	DC4	36	54	6
15	21	NAK	37	55	7
16	22	SYN	38	56	8
17	23	ETB	39	57	9
18	24	CAN	3A	58	:
19	25	EM	3B	59	;
1A	26	SUB	3C	60	<
1B	27	ESC	3D	61	=
1C	28	FS	3E	62	>
1D	29	GS	3F	63	?
1E	30	RS	40	64	@
1F	31	US	41	65	A
20	32	SP	42	66	B
21	33	!	43	67	C

Appendix C
ASCII Code Lists and Definitions

Table C-1
ASCII Code List (Cont.)

HEX	DEC	CHAR	HEX	DEC	CHAR
44	68	D	62	98	b
45	69	E	63	99	c
46	70	F	64	100	d
47	71	G	65	101	e
48	72	H	66	102	f
49	73	I	67	103	g
4A	74	J	68	104	h
4B	75	K	69	105	i
4C	76	L	6A	106	j
4D	77	M	6B	107	k
4E	78	N	6C	108	l
4F	79	O	6D	109	m
50	80	P	6E	110	n
51	81	Q	6F	111	o
52	82	R	70	112	p
53	83	S	71	113	q
54	84	T	72	114	r
55	85	U	73	115	s
56	86	V	74	116	t
57	87	W	75	117	u
58	88	X	76	118	v
59	89	Y	77	119	w
5A	90	Z	78	120	x
5B	91	[	79	121	y
5C	92	\	7A	122	z
5D	93	]	7B	123	{
5E	94	^	7C	124	|
5F	95	-	7D	125	}
60	96	`	7E	126	~
61	97	a	7F	127	DEL

Appendix C
ASCII Code Lists and Definitions

Table C-2
ASCII-Code Definitions

ABB	CTRL	MEANING	HEX
NUL		NULL Character	00
SOH	A	Start of Heading	01
STX	B	Start of Text	02
ETX	C	End of Text	03
EOT	D	End of Transmission	04
ENQ	E	Enquiry	05
ACK	F	Acknowledge	06
BEL	G	Bell	07
BS	H	Backspace	08
HT	I	Horizontal Tabulation	09
LF	J	Line Feed	0A
VT	K	Vertical Tabulation	0B
FF	L	Form Feed	0C
CR	M	Carriage Return	0D
SO	N	Shift Out	0E
SI	O	Shift In	0F
DLE	P	Data Link Escape	10
DC1	Q	Device Control 1	11
DC2	R	Device Control 2	12
DC3	S	Device Control 3	13
DC4	T	Device Control 4	14
NAK	U	Negative Acknowledgement	15
SYN	V	Synchronous Idle	16
ETB	W	End of Transmission Block	17
CAN	X	Cancel	18
EM	Y	End of Medium	19
SUB	Z	Substitute	1A
ESC		Escape	1B
FS		File Separator	1C
GS		Group Separator	1D
RS		Record Separator	1E
US		Unit Separator	1F
SP		Space	20
DEL		Delete	7F

Appendix D
Writing a Driver Program

Driver programs for various computer systems have already been written. However, the driver programs available may not run or be suitable for your particular host computer.

The most important consideration in writing a driver program is the availability of an RS-232C port. Without it, a driver program has no means of communicating with MICE-II. The programming language used to implement the driver program is not important as long as access to the console, the MICE-II communication port and the file system is available.

Eight modules are necessary to establish a communication link between the console and MICE-II.

```
InitMICE-II  - initialize MICE-II
TestMICE-II  - test if MICE-II sent a character
ReadMICE-II  - read a character from MICE-II
WriteMICE-II - send a character to MICE-II
InitCons     - initialize baud rate and parity of console
TestCons     - test if console has a character
ReadCons     - read a character from console
WriteCons    - send a character to console
```

These eight modules are the necessary ingredients in creating a program to replace the simple terminal.

```
call InitCons
call InitMICE-II
do forever
     if TestCons then do
          Char = ReadCons
          call WriteMICE-II(CHAR)
     end
     if TestMICE-II then do
          Char = ReadMICE-II
          call WriteCons(CHAR)
     end
end
```

If the RS-232C handshaking signals are not monitored, the baud rate selection for MICE-II must be less than or equal to the console baud rate to avoid overrun errors, where MICE-II is sending too fast for the display console. Refer to Chapter 2 for configuring MICE-II with the appropriate baud rate, parity and interface (DTE or DCE, with or without handshaking).

Interface to the file system requires similar but more complicated modules, which vary for different computer systems. The general concept is to have the program loop intercept the download and upload commands, not retransmit them to MICE-II. The program then queries for the file to be transmitted or received. The appropriate command must then be created and transmitted to MICE-II.

Remember that the maximum amount of data that can be transferred at one time is limited to 128 bytes for download and 32 bytes for upload. The program may be required to reformat data to one of the acceptable formats required by MICE-II. Finally, to add hardcopy for CRT consoles, an additional write to a remote printer can be added by calling the printer output module each time a write to MICE-II or a write to the console is performed.

Appendix E
MICE Driver and Symbolic Debug Software

The following is a listing of currently available MICE software.

List of MICE Software Support

Host Computer	Operating System	Driver	Symbolic Debug	Company
Altos 8000	MP/M-80	Yes	No	Microtek International Inc.
Apple II/II+/IIe	Apple DOS 3.3	Yes	No	Microtek International Inc.
DEC VAX-11 750/780	VMS	Yes	No	Microtek International Inc.
DEC VAX-11 750/780	VMS	Yes	Yes	ARS Microsystems Ltd.*
DEC MicroVAX	VMS	Yes	Yes	ARS Microsystems Ltd.
DEC Rainbow	CP/M-86 & MS-DOS	Yes	Yes	ARS Microsystems Ltd.
IBM PC/XT/AT	CP/M-86 & MS-DOS	Yes	No	Microtek International Inc.
IBM PC/XT/AT	MS-DOS	Yes	Yes	ARS Microsystems Ltd.
IBM PC/XT/AT	MS-DOS	Yes	Yes	Microtek International Inc.
Intel-MDS 220/225/286	ISIS-II/III	Yes	No	Microtek International Inc.
Intel-MDS 220/225/286	ISIS-II/III	Yes	Yes	ARS Microsystems Ltd.
Intel-MDS 310/380	iRMX-86	Yes	Yes	RTCS Corp.
Intel-PDS	ISIS-II	Yes	No	Microtek International Inc.
8080/8085/Z80 systems	CP/M Version 2.2	Yes	Yes	ARS Microsystems Ltd.

* ARS Microsystems Ltd.
Doman Road, Camberly
Surrey, GU15 3DF, England
Tel:(0276)64341 Tlx:858779

Real-Time Computer Science Corp.
1390 Flynn Road
Camarillo, CA 93010, U.S.A.
Tel: (805)987-9781 Tlx:467897

APPENDIX F

High Performance Universal Emulation Memory Board(HUEM)

The High Performance Universal Emulation Memory (HUEM) board is the bottom board in the conventional three board MICE-II module. Each memory board supports 128K bytes of emulation memory.

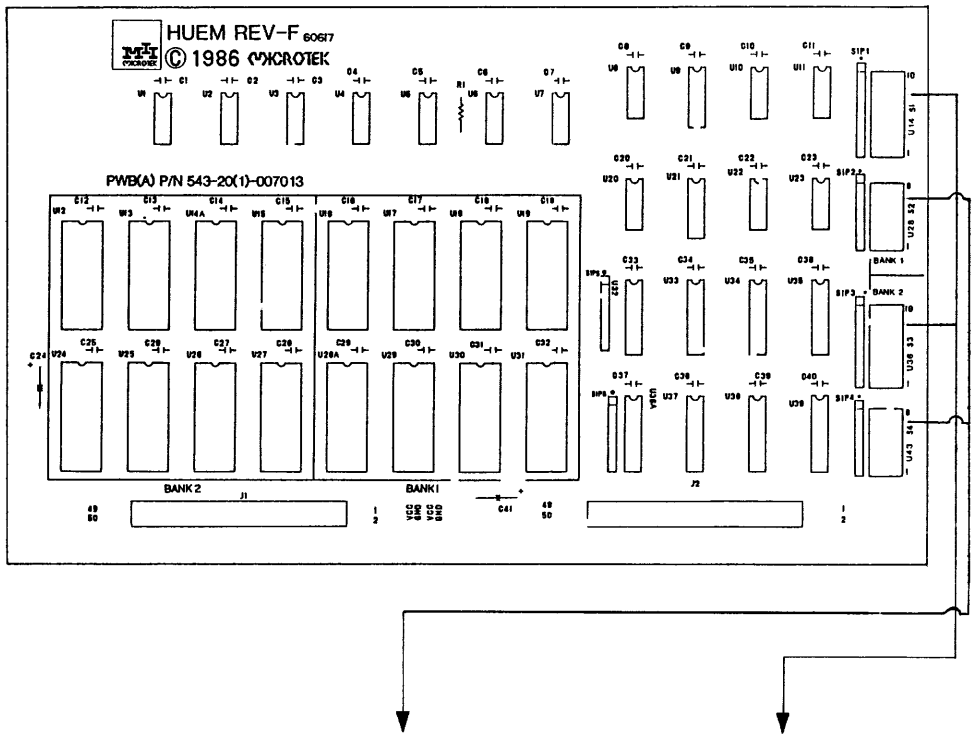
Memory can be expanded with the addition of other HUEM boards. Memory boards can be added simply by the addition of six brass spacers for each board and by replacing the existing 50 pin flat wire assembly with enough connections to join the Control Emulation Processor board, Realtime Trace board and the number of memory boards required. An additional board can be assembled into the MICE-II case by using smaller spacers. However, we recommend leaving the cover of the MICE-II case off while using four boards, otherwise overheating may cause intermittent problems in operation.

Remember, each additional memory board requires 800 mA current at +5 volts DC which may require a larger power supply to support the expanded memory in your MICE-II.

A memory expansion package including the HUEM board, spacers and interconnection cable is optionally available from Microtek.

APPENDIX F

High Performance Universal Emulation Memory Board (HUEM) Placement Chart

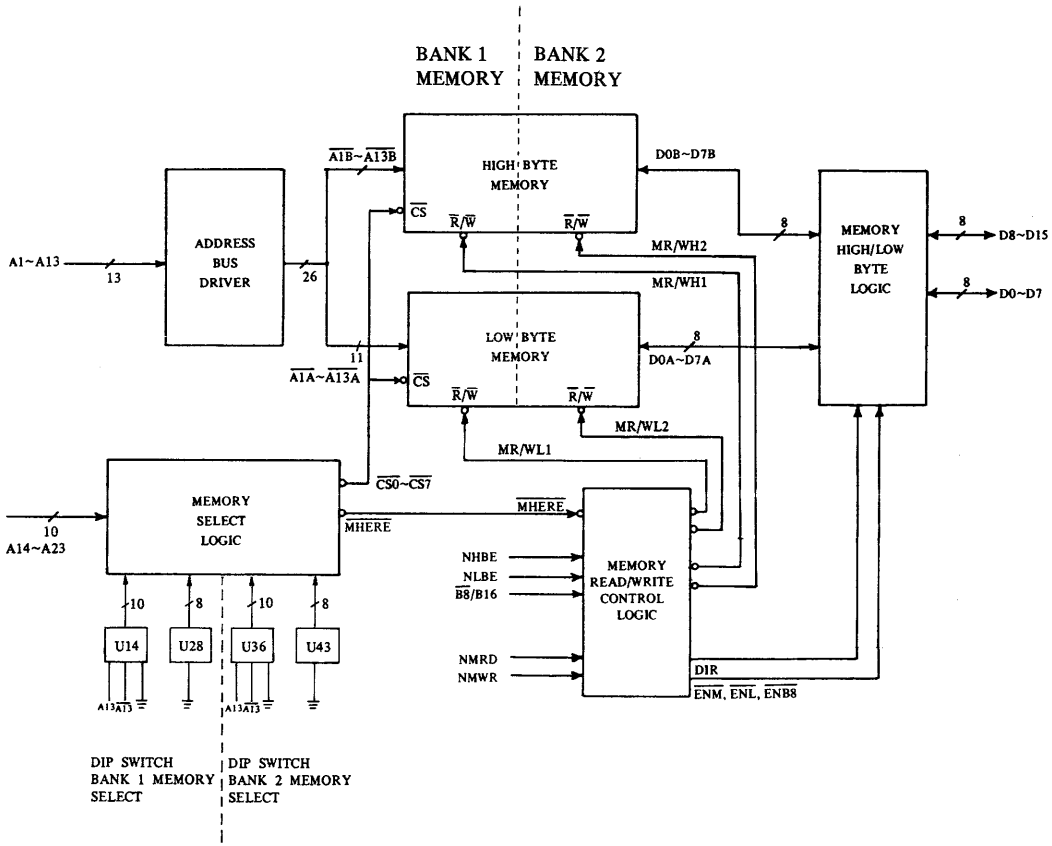


DIP Switch U28, U43
Memory Segment
Select

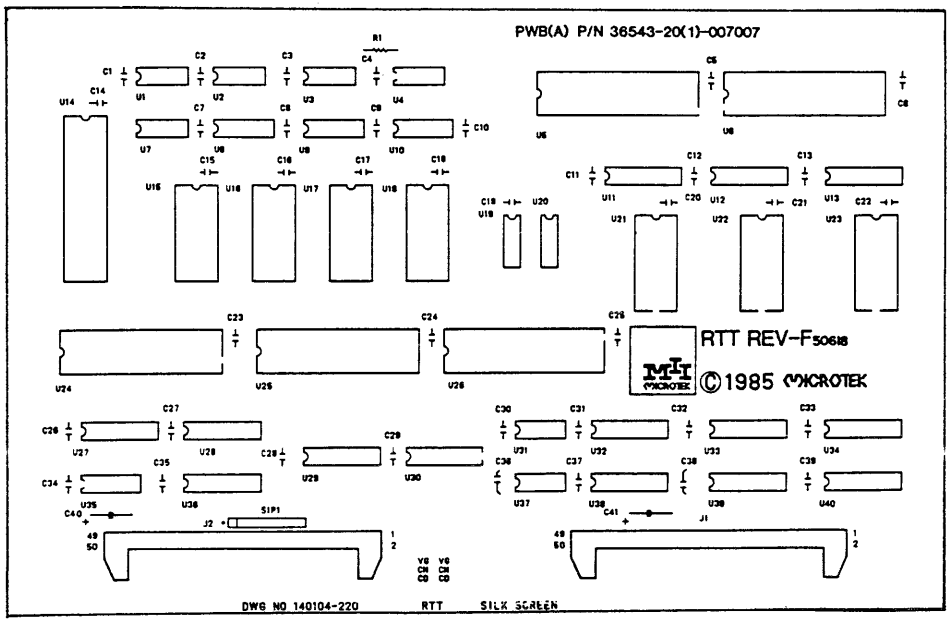
DIP Switch U14, U36
Emulation Memory
Enable/Disable

APPENDIX F

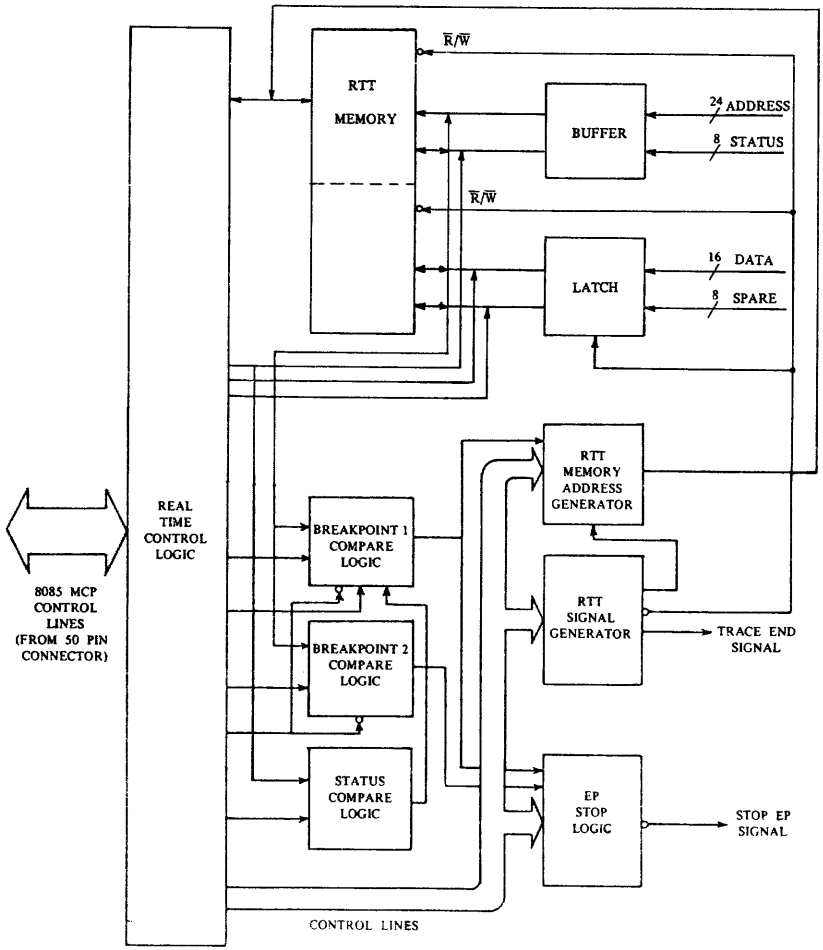
High Performance Universal Emulation Memory Board (HUEM) Block Diagram



Appendix G
Reatime Trace Board (RTT)
Placement Chart

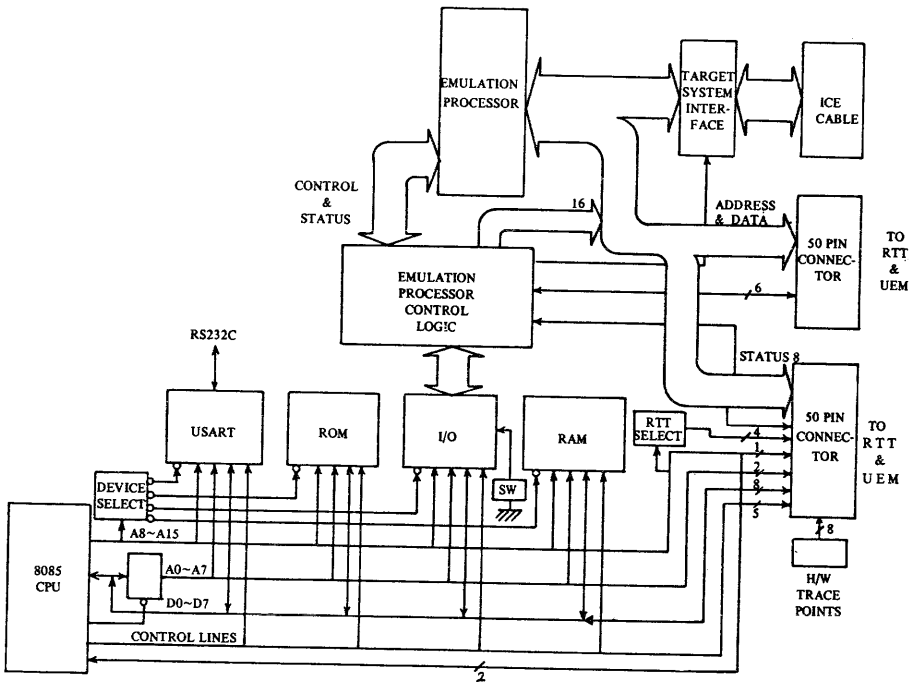


Appendix G
Reatime Trace Board (RTT) Block Diagram



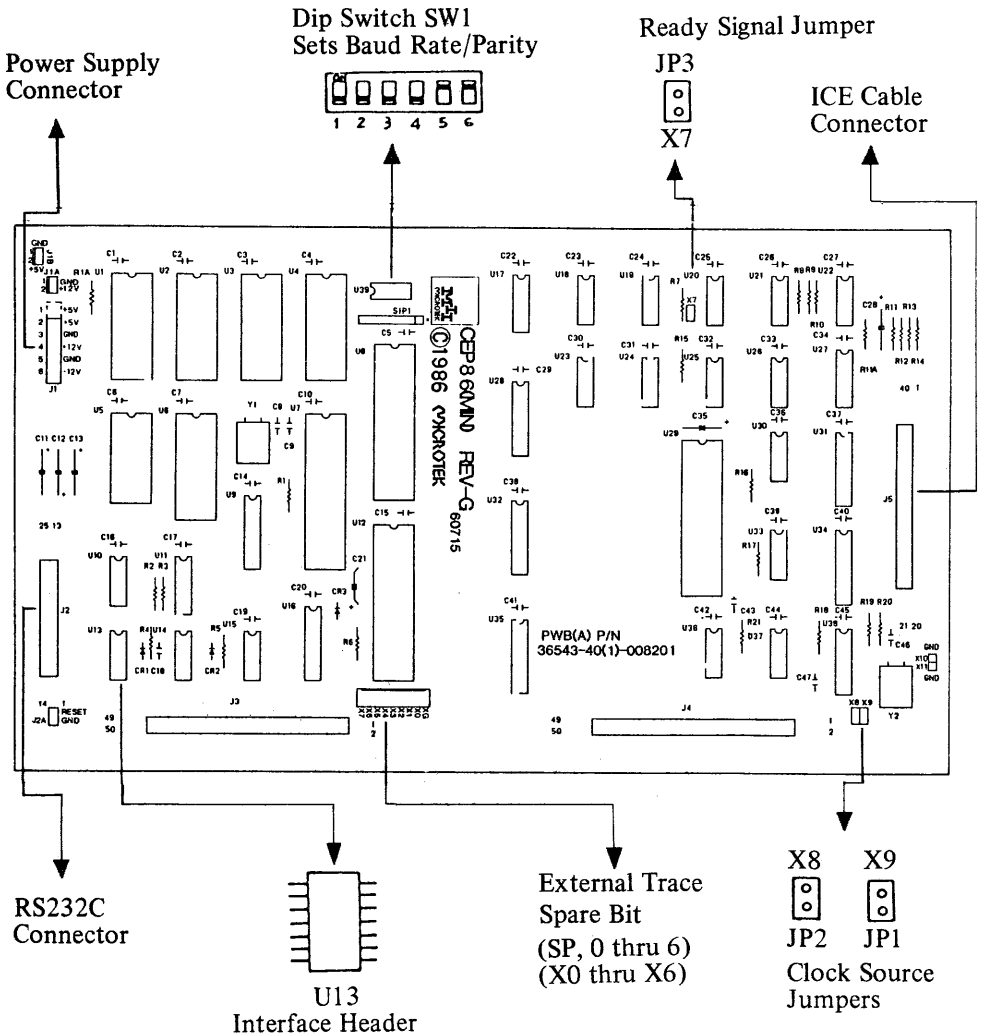
Appendix H
Control Emulation Processor Board (CEP)

H.1 Control Emulation Processor Board (CEP) Block Diagram

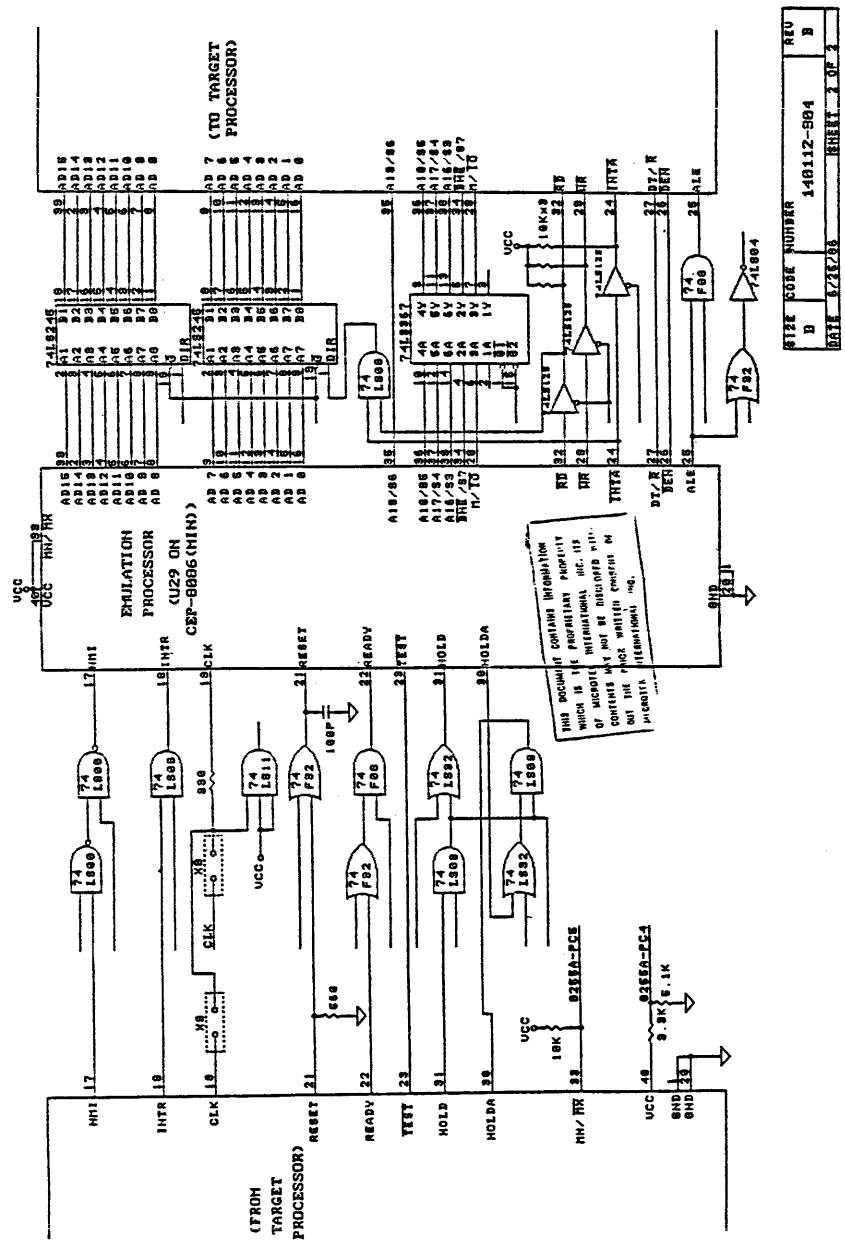


H.2 8086/88(MIN) Hardware

H.2.1 Control Emulation Processor Board, CEP-8086(MIN) Placement Chart



H.2.2 Interface Diagram from Target Processor to CEP-8086(MIN)



H.2.3 Switch Settings

1) The target's clock can be selected by placing the jumper on the personality board at X9; or the on-board 8 MHz clock can be selected by placing the jumper at X8. The CEP board has a 330 ohm damping resistor (R21) for the clock signal.

	X8	X9
INT	C	O
EXT	O	C

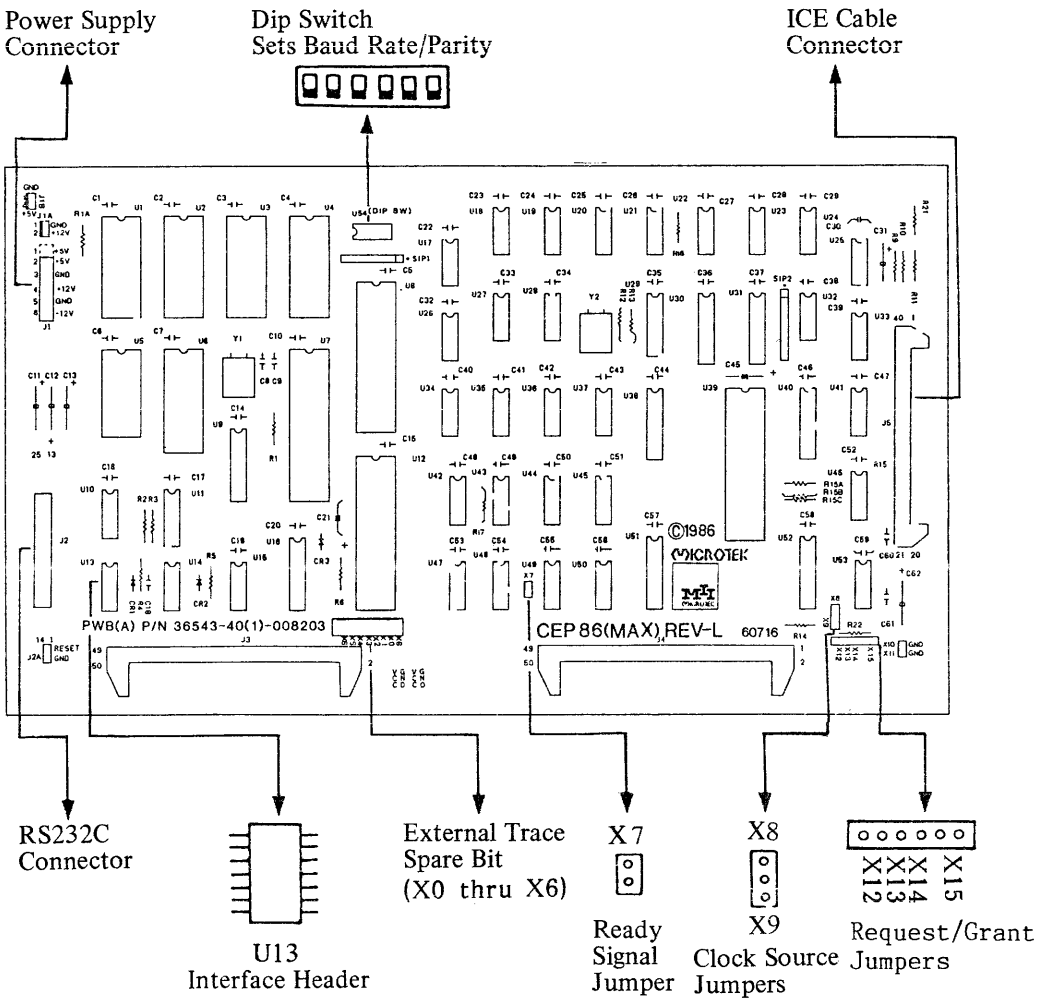
C: Close
O: Open

Internal/External Clock Selection

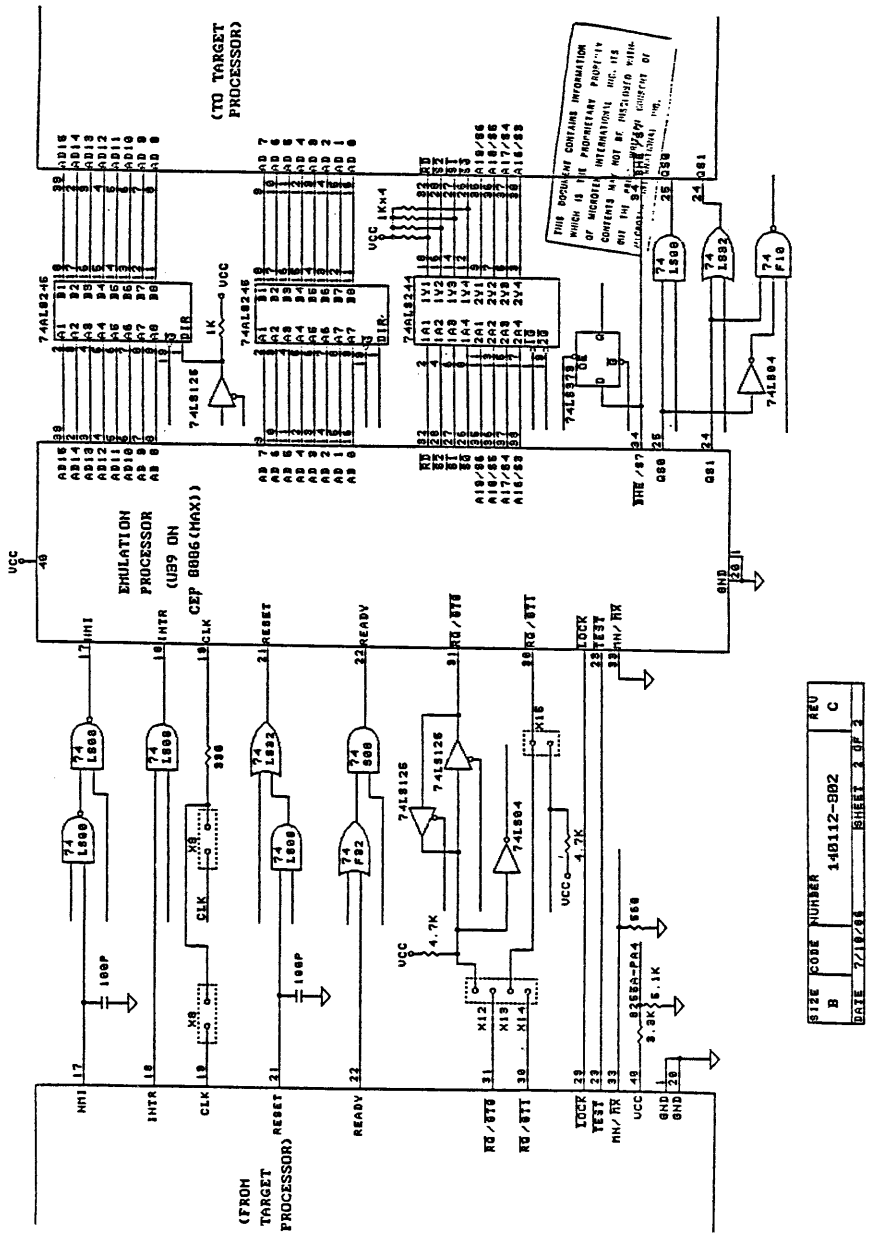
2) The target ready signal can be selected by removing jumper X7; or if memory selection is at HUEM, the on-board ready signal can be used by connecting jumper X7.

H.3 8086/88(MAX) Hardware

H.3.1 Control Emulation Processor Board, CEP-86(MAX) Placement Chart



H.3.2 Interface Diagram from Target Processor to CEP-86(MAX)



H.3.3 Switch Settings

1) The target's clock can be selected by placing the jumper on the personality board at X8; or the on-board 8 MHz clock can be selected by placing the jumper at X9. The CEP board has a 33 ohm damping resistor (R14) for the clock signal.

	X8	X9
INT	O	C
EXT	C	O

C: Close
O: Open

Internal/External Clock Selection

2) The target ready signal can be selected by removing jumper X7; or if memory selection is at HUEM, the on-board ready signal can be used by connecting jumper X7.

H.3.4 Coprocessor Support

1) Basic Features of the COB-8687

The COB-8687 multi-processor board, containing both the 8086/88 and 8087 microprocessors, is installed at U39 on the CEP-86(MAX), Revision I2 and later*. The TGB-8087 adaptor board, which includes a 4-wire cable, is installed in the 8087 socket on the target board to facilitate communication between the internal 8087 coprocessor on the COB-8687 and the target system.

* The COB-8687 may also be installed on the CEP-8086(MAX), Revision G-I1, by contacting your nearest Microtek distributor. Remember that firmware version 3.2 or later must be used with the coprocessor board. Note that after modification to Revision I1 or later, $\overline{RQ}/GT1$ of the 8086 is disconnected from the target and dedicated to $\overline{RQ}/GT0$ of the 8087 (on the COB-8687) instead.

The COB-8687 provides the following key features:

a) Full support for 8087 instructions in all emulation/debug commands.
b) Programs with 8087 numeric instructions can be located on the target or in MICE emulation memory; however, programs with 8089 I/O controller instructions must be located and executed on the target.
c) All bus activity (address/data/status) for the 8087 can be recorded in the trace buffer by B/F/BP/BT commands or displayed by stepped emulation commands C/CW/SZ.
d) Bus status for the 8087 can also be specified in breakpoints and trigger conditions.

User can also place 8087 on target, but advantages of items b,c,d described above will be lost.

For CEP 8086 MAX revision K and later, jumpers X12-X15 are provided for more flexibility to implement 8087.

		X12	X13	X14	X15
8087 locate inside MICE		C	O	O	O
8087 locate on target side	8087 uses RQ/GT1 as I/P	C	O	C	C
	8087 uses RQ/GT0 as I/P *	O	C	O	C

* In this mode, the RQ/GT1 pin of the target is disconnected to the MICE. Therefore, function of RQ/GT1 in the target is not available.

2) COB-8687 Cable and TGB-8087 Installation

a) When emulating an 8087 on the target, replace the target's 8087 with the TGB-8087 adaptor and connect the 4-wire cable from this adaptor to the COB-8687, then plug your 8087 into U1 of the COB-8687 board.

b) If an 8089 is used in the target system, then we recommend connecting RQ/GT of the 8089 to $\overline{RQ}/GT1$ of the TGB-8087 on the target board.

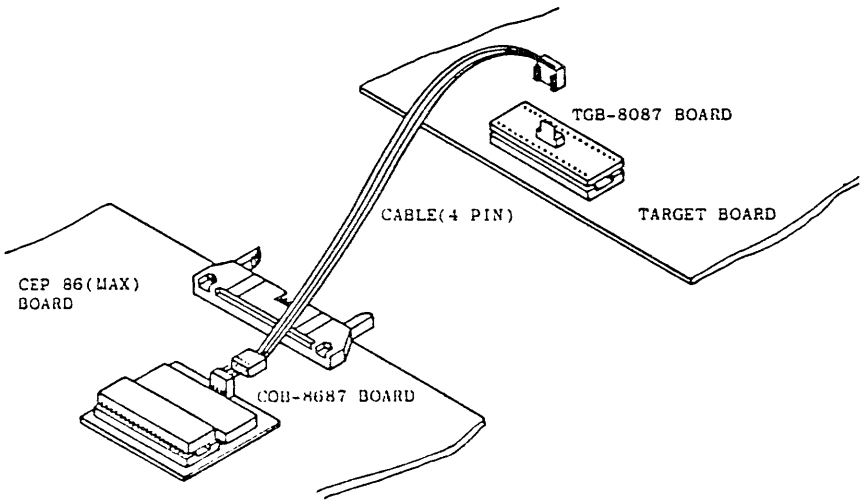
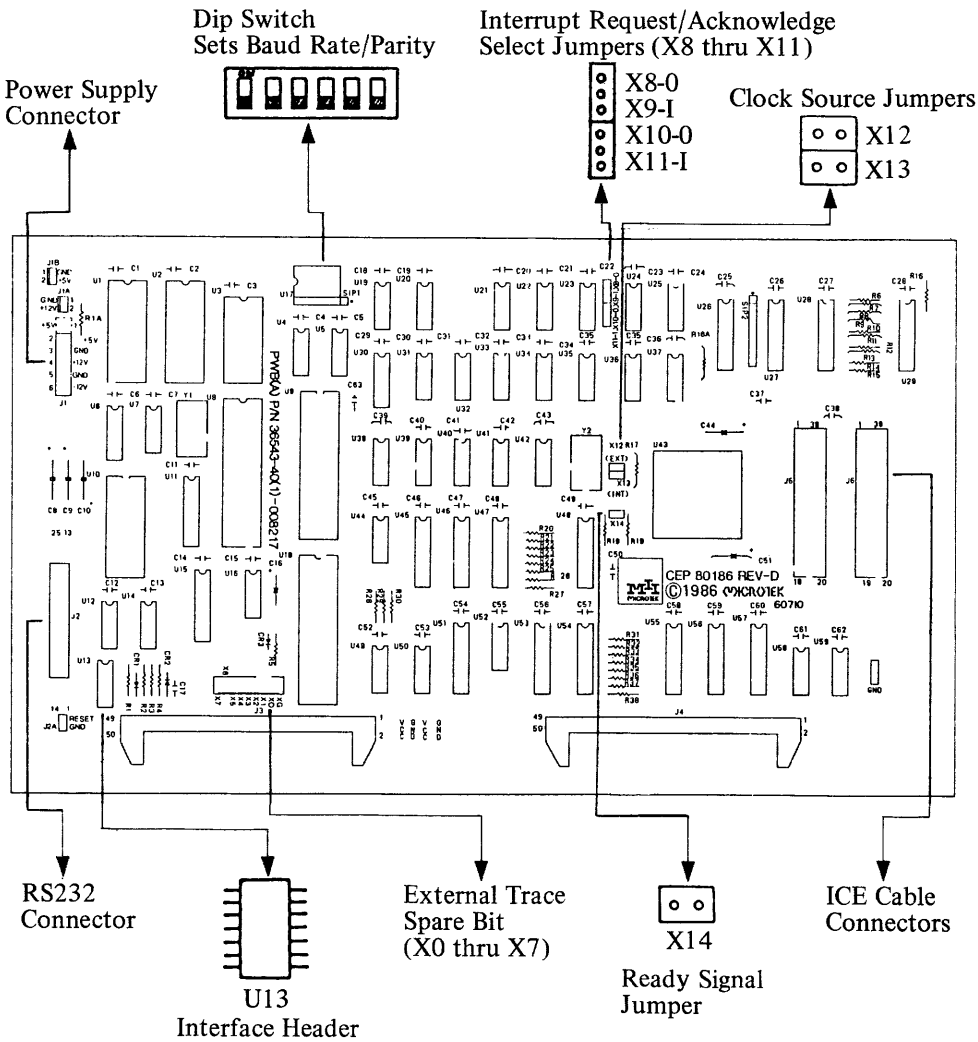


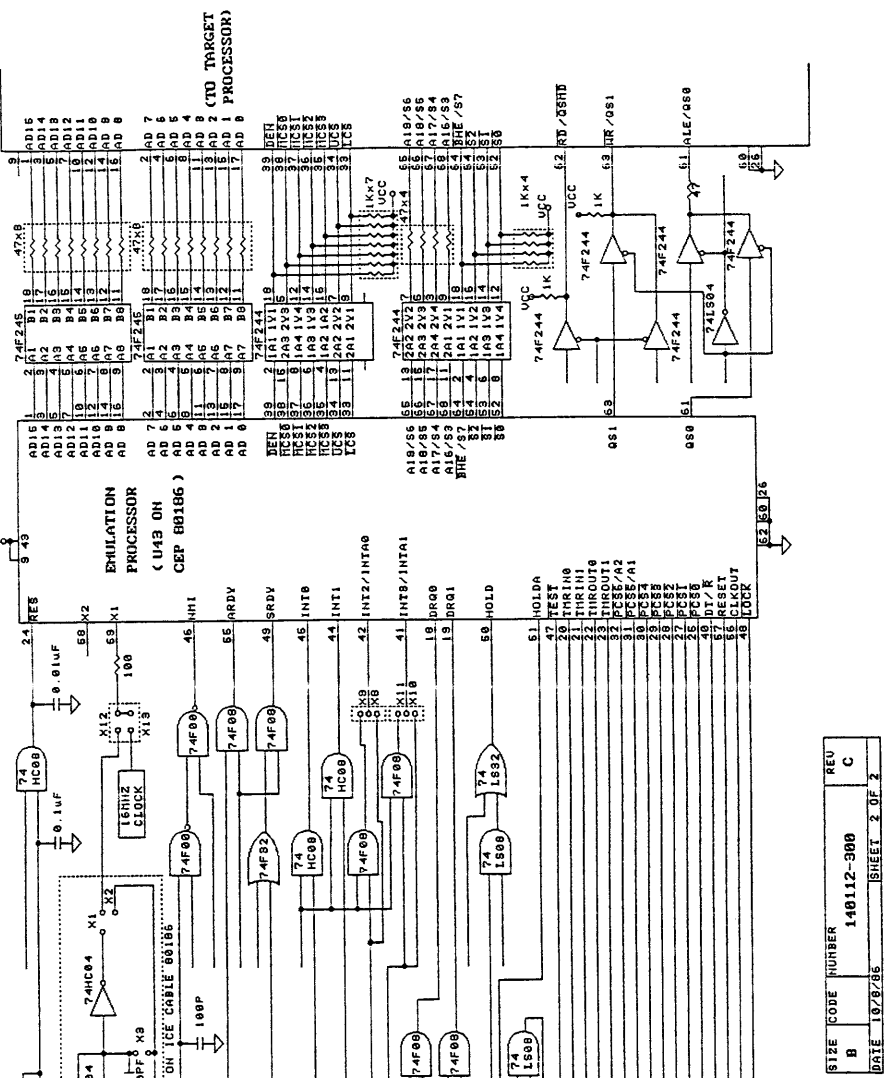
Figure H3-1
COB-8687 Cable and TGB-8087 Installation

H.4 80186/188 Hardware

H.4.1 Control Emulation Processor Board, CEP-80186 Placement Chart



H.4.2 Interface Diagram from Target Processor to CEP-80186



H.4.3 Switch Settings

1) The 8 MHz internal clock is selected by placing a jumper at X13 on the personality board. The CEP board has a 100 ohm damping resistor (R17) for the clock signal. To select the external clock set a jumper at X12 and place a jumper for the OAB adapter on the ICE cable at X2. To run on the external crystal set a jumper at X12 and place a jumper for the OAB adapter at X1 (or X1 and X3).

(i)

old OAB Adapter

	X12	X13	X1	X2
INT	O	C	O	O
EXT CLOCK	C	O	O	C
EXT CRYSTAL	C	O	C	O

C: Close
O: Open

Internal/External Clock Selection

(ii)

new XOA Adapter

	X12	X13	X1	X2	X3
INT	O	C	O	C	O
EXT CLOCK	C	O	O	C	O
EXT CRYSTAL	C	O	C	O	C

C: Close
O: Open

Internal/External Clock Selection

2) Dual function pins 41 and 42, INT3/$\overline{\text{INTA1}}$ and INT2/$\overline{\text{INTA0}}$ respectively, may be selected as either Input or Output pins for the target system. Input is selected by placing jumpers on X9, X11 or both to function as Interrupt Request pins. Output is selected by placing jumpers on X8, X10 or both to function as Interrupt Acknowledge pins.

Pin No.	Jumper	Function	
		Interrupt ACK INTA	Interrupt Request (INTR)
42	X8	C	O
	X9	O	C
41	X10	C	O
	X11	O	C

C: Close

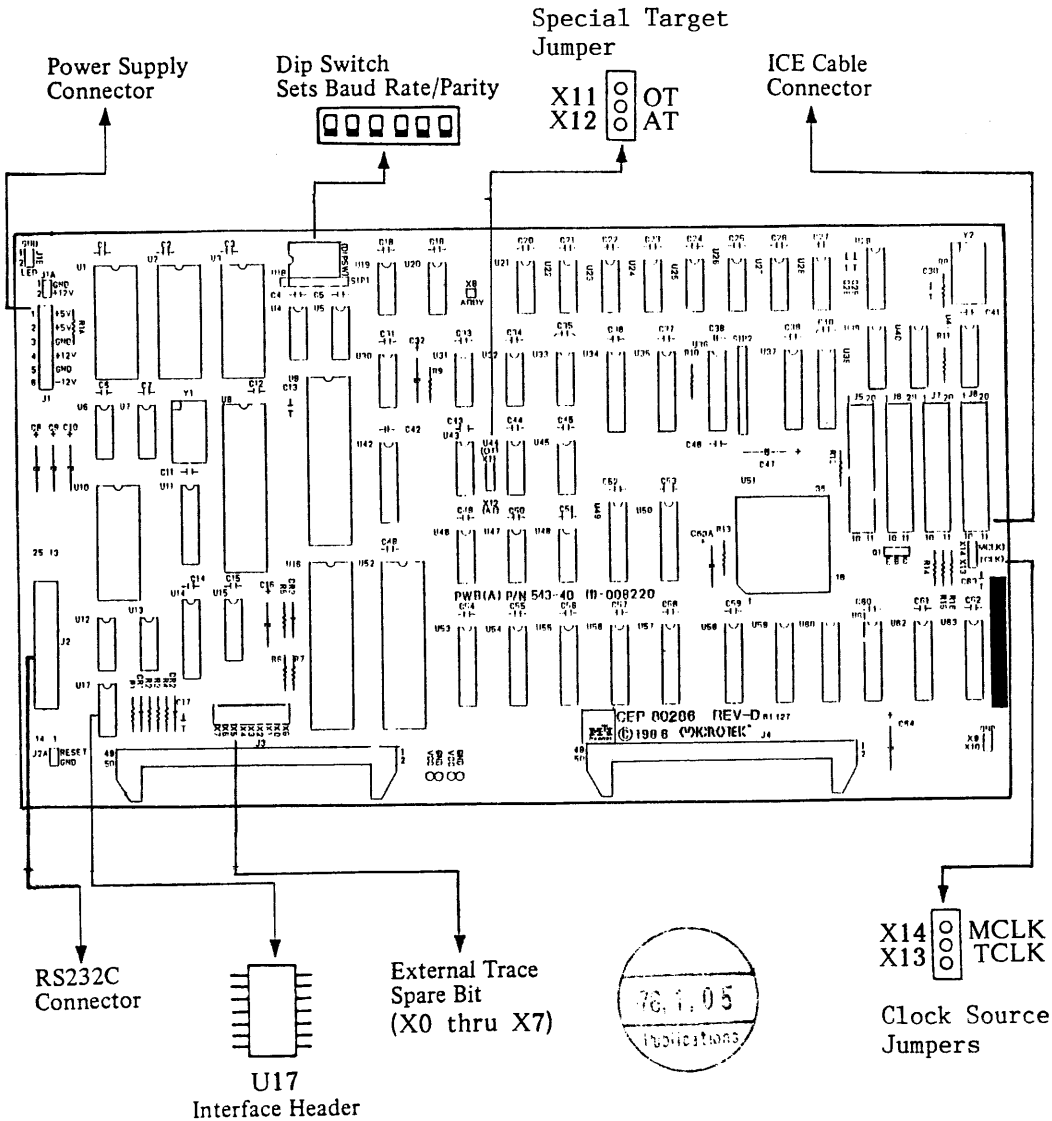
O: Open

Request/Interrupt Acknowledge Pinout Selection

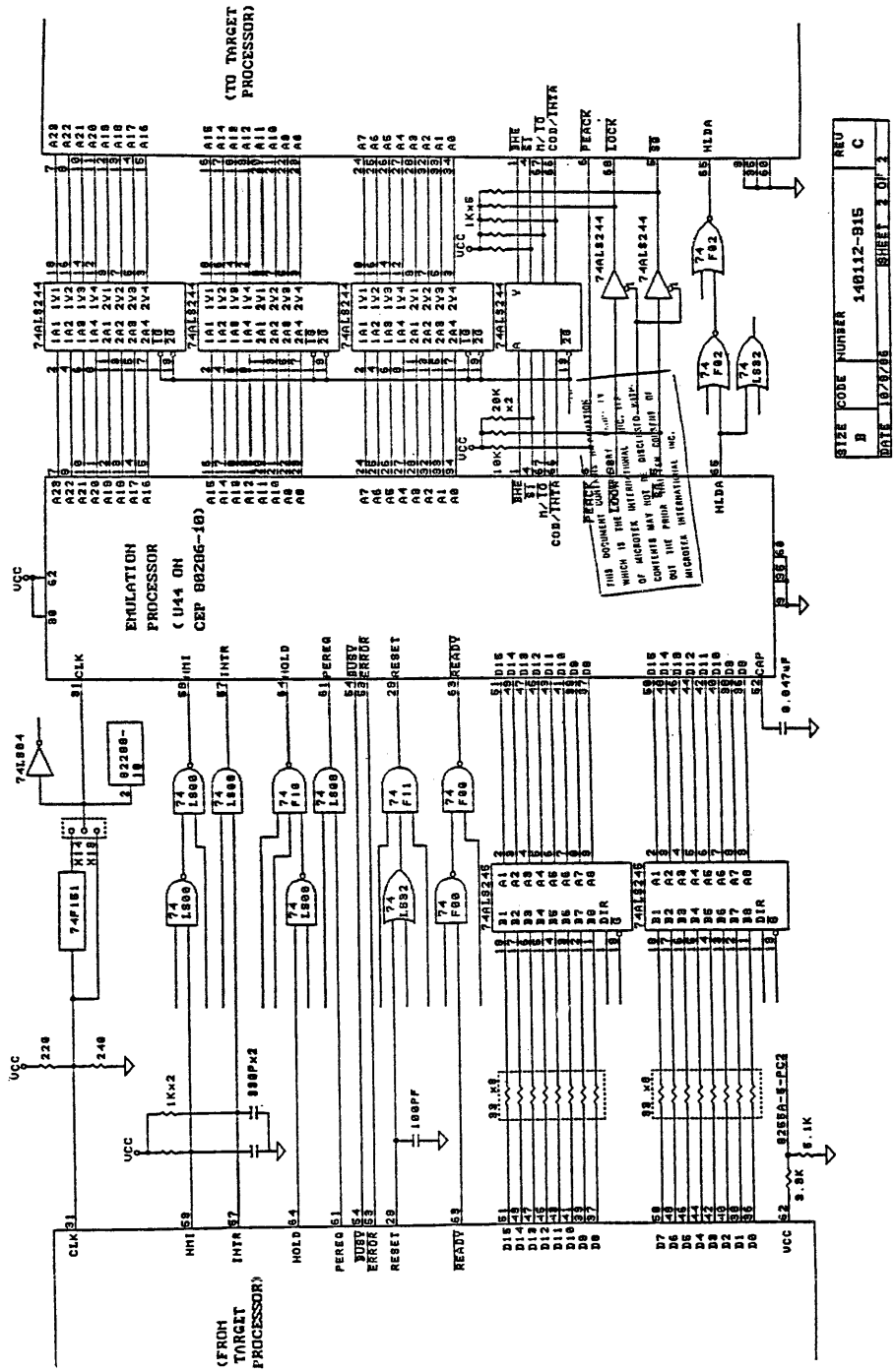
3) If memory selection is at HUEM, the on-board ready signal may be selected by placing a jumper at X14. The target ready signal is selected by removing the jumper at X14.

H.5 80286 Hardware

H.5.1 Control Emulation Processor Board, CEP-80286 Placement Chart



H.5.2 Interface Diagram from Target Processor to CEP-80286



H.5.3 Switch Settings: (For 80286 CEP revision C only)

1) In protected mode, if target is IBM PC-AT, to avoid destroying the system DRAM refresh circuit function, place mini-jumper on X12.

   Default mini-jumper setting is on X11.

	IBM PC-AT	Other Targets
X11	O	C
X12	C	O

C: Close
O: Open

2) Two system clock options are provided:

   a) Set the mini-jumper on X13 to choose the clock source supplied by target directly.

   b) Place the mini-jumper on X14. At this option, clock is set by first passing through F151, then modified by a pull-up circuit. Note that clock selection is set by inputting K command.

	Clock Source
X13	Exact clock signal from target directly
X14	Determined by K command (Default)

H.6 Application Note for 80286 Commands

The MICE-II 80286P, firmware revision 4.0 and later supports both the i80286 microprocessor's Real Address Mode and Protected Virtual Address Mode, while the earlier MICE 80286 model only supports Real Address Mode.

MICE-II 80286P can automatically detect the i80286's current operating mode. However, due to the complexity of the microprocessor's Protected Mode, two new MICE commands have been added - "MODE" and "IDT". Remember that these commands are only required when the emulation processor (EP) is operated in Protected Mode. Also remember that the value specified in the IDT command is only used as a reference for the MICE control program, i.e. the contents of the EP registers are not affected.

Notes 1) When the 80286P is powered up, the following message will display:

"DEBUG MODE, MODE R; NMI, AND PEREQ DISABLED!"

2) Input "?P" to display the Help screen for Protected Mode commands:

```
>?P
MP [W]a1[ a2[ d1[..d8]][S]]
R S|DT|TR|dtable
R DT(selector|sel_reg)[.[component]]
R dtable(index)[.[component]]
R TSS[(selector)][.[component]]

* dtable  = GDT, LDT and IDT
  sel_reg = CS, DS, ES, SS, LDT and TR
>
```

3) After executing any of the following commands (while operating in i80286 Real Address Mode), the sixth digit of the PC is forced to zero (i.e. OxxxxxH): A/I/J/M/O/R/S/T/U/Z and download commands. Therefore, take special precaution when working with a target- decoding circuit where the sixth digit of the PC must be interpreted.

H.6.1 Specify EP Operation Mode

MODE [R|P]

MODE is the command to specify the EP's current operation mode. If no other parameters are defined, inputting "MODE<CR>" will display the current mode selection.

R means that the EP is always run in Real Address Mode.

P means that EP may be run in Real Address Mode and/or Protection Mode.

After a cold or warm start, MICE is automatcally set for Real Address Mode. Selection of the microprocessor's operation mode will affect the MICE control program as indicated below.

Example: Display current mode and then specify Protected Mode.

```
>r     *** MICE-II 80286 V4.0 ***          ; Warm start (or cold start)
DEBUG MODE, MODE R; NMI AND PEREQ DISABLED!
NO TARGET VCC; INTR AND RESET DISABLED!
>MODE
MODE REAL
>MODE P
>MODE
MODE PROTECTIVE
>
```

1) A/BP/I/J/M/O/R/S/T/U/Z and Download commands:

R Mode - The IDT value must be specified for the BP command if it will be changed by the user's program (refer to Section H.6.8.2).

P Mode - The IDT value must be specified (refer to Section H.6.2).

- An incorrect value may be read when accessing word data from the address IDT+14H. If this occurs, an error message will display -

  "WARNING - WORD ACCESS AT IDT+14H MAY GIVE INCORRECT DATA!"

  To access the correct data, first abort the current command, set the PC to the preceding instruction address and use the Cycle Step, Trace or Run command to reexecute the required instruction.

- User must select "MODE P" when emulation processor is operating in protectecd mode; otherwise, the EP will hang up and MICE displays the message- "TARGET CAN'T STEP!!". Nothing but only the reset command can recover EP!.

H.6.1.1 B/BP/BT/F/H Commands:

If the trigger is specified with an absolute address (not applicable to BP command), no further modification is required, and the EP is kept running. However, if the trigger is specified with a logical address, then the following conditions apply:

R Mode - If a logical address is entered (i.e. segment:offset), set the trigger address to -

(Segment Value x 10H) + (Instruction Pointer)

without stopping the EP.(This type of address specification may be used for realtime target systems.)

P Mode a) The base value of the IDT must be defined prior to specifying a trigger address; the EP will stop when the trigger address is inputted.

b) If the EP is currently operating in Real Address Mode and a logical address is entered (i.e. segment:offset),set the trigger to -

(Segment Value x 10H) + (Instruction Pointer)

c) If the EP is currently operating in Protected Mode and the selector refers to a correct descriptor (TSS, LDT, ESEG and DSEG), then set the trigger to-

(Physical Base in Descriptor) + (Offset)

But if the selector refers to an incorrect descriptor, then "INVALID SELECTOR!" will be displayed.

H.6.2 Set Base Value of Interrupt Descriptor Table Register (IDT)

IDT=address

IDT is the command to set the base value of the IDT register. If no other parameters are defined, inputting "IDT<CR>" will display the current base value of the IDT register.

address is the base value of the IDT register (OH-FFFFFEH). Note that this address must be of an even value, otherwise MICE will display an error message - "IDT.BASE MUST BE AN EVEN VALUE!".

This command only defines a reference value for MICE; the emulation processor's status and registers are not affected. When "MODE R" is specified, the IDT command is not required. However, when "MODE P" is specified, the base value of the IDT must be set prior to entering A/I/J/M/O/R/S/T/U/Z or Download commands; otherwise "PLEASE INPUT IDT.BASE!" will be displayed.

Example: Display the general registers without first initializing the base value of the IDT; then specify the current value and try again.

```
>R
PLEASE INPUT IDT.BASE!
>IDT=0
>R
 AX=FFOF  BX=FFFF  CX=FFFF DX=FFFF SP=FFFF BP=FFFF SI=FFFF   FS   -NPLODITSZ-A-P-C
 DI=FFFF  DS=0000  SS=0000 ES=0000 CS=F000 IP=FFF0 MS=FFF0  0002  0000000000000010
 GDT.BASE=FFFFF0   GDT.LIMIT=FFFF   IDT.BASE=000000  IDT.LIMIT=FFFF  PC=0FFFF0
>
```

Example: Initialize the base value of the IDT, and show that the IDT command does not affect the EP's registers. Then use the "RIDT" command to change the current value of EP's IDT register.

```
>IDT=800
>IDT
IDT.BASE=000800
>R
 AX=FFOF BX=FFFF CX=FFFF DX=FFFF SP=FFFF BP=FFFF SI=FFFF  FS  -NPLODITSZ-A-P-C
 DI=FFFF DS=0000 SS=0000 ES=0000 CS=FOOO IP=FFFO MS=FFFO 0002 0000000000000010
 GDT.BASE=FFFFFO GDT.LIMIT=FFFF  IDT.BASE=000000 IDT.LIMIT=FFFF  PC=0FFFFO
>RIDT
IDT.BASE 000000 800<CR>
   .LIMIT  FFFF <CR>
AX FFOF <CR>
>R
 AX=1234 BX=FFFF CX=FFFF DX=FFFF SP=FFFF BP=FFFF SI=FFFF  FS  -NPLODITSZ-A-P-C
 DI=FFFF DS=0000 SS=0000 ES=0000 CS=FOOO IP=FFFO MS=FFFO 0002 0000000000000010
 GDT.BASE=FFFFFO GDT.LIMIT=FFFF  IDT.BASE=000800 IDT.LIMIT=FFFF  PC=0FFFFO
>
```

H.6.3 Physical Memory Display/Examine/Modify/Fill/Search Commands - MP

```
MP[W] start-address[ end-address[ data-1...data-8][S]]
```

MP — are the keywords for memory display/examine/modify/fill/search when EP is operating in protected mode.

W — specifies a memory operation with memory content organized in word format.

start-address — is a physical hexadecimal address (OH - FFFFFFH) of the target processor indicating a memory starting point where the operation is to begin.

end-address — is a physical hexadecimal address (OH - FFFFFFH) of the target processor indicating the last memory location of the specfied range. Such range must not exceed 64K (FFFFH), otherwise, MICE will convert the end-address to a new value (start-address+FFFFH).

data-1...8 are hexadecimal or ASCII data. Data in ASCII must be enclosed in apostrophes (e.g. 'AR'). Where a "block fill" or "block" search" operation is specified, the block may be of 8 bytes in hex or 8 characters in ASCII. Combined use of hex and ASCII is permitted. Whenever the block-fill/search command is executed, while the address range (from start to end address) is smaller than the block (data 1 to 8), the excess trailing data will be ignored. Note that the apostrophe ('); lower case "r" and MICE editing/control characters are not applicable for ASCII input.

S defines a "search" option which looks for specified data.

This command is useful when EP is operating in protected mode and user wants to access the memory with physical address. To access the memory with logical address, please refer to Section H.6.5.

H.6.4 Display/Modify Registers and Descriptors

This section includes display/modify for -

a) General Registers
b) Segment Selectors*
c) Descriptor Table (DT) components*
c) Global Descriptor Table (GDT) components*
d) Local Descriptor Table (LDT) components*
e) Interrupt Descriptor Table (IDT) components*
f) Task State Segment (TSS) components*

* Display/modification for these items is only permitted when the emulation processor is operating in Protected Mode.

The components associated with each descriptor type are listed in table H-1, the descriptor types are defined in table H-2, and a brief description of the descriptor components is provided in table H-3. These tables are shown below:

Descriptor	Components													
	.BASE	.LIMIT	.WCNT	.SSEL	.SOFF	.SR	.DPL	.ED	.W	.A	.C	.R	.P	.B
DSEG	X	X				X	X	X	X	X			X	
ESEG	X	X				X	X			X	X	X	X	
CALLG			X	X	X	X	X						X	
TRAPG				X	X	X	X						X	
INTG				X	X	X	X						X	
TASKG				X		X	X						X	
TSS	X	X				X	X						X	X
LDT	X	X				X	X						X	

Table H-1 Components Associated with each Descriptor Type, where 'X' means available type.

Mnemonic	Meaning	Access Rights Byte
DSEG	data segment	S=1, E=0
ESEG	executable segment	S=1, E=1
CALLG	call gate	S=0, Type=4
TRAPG	trap gate	S=0, Type=7
INTG	interrupt gate	S=0, Type=6
TASKG	task gate	S=0, Type=5
TSS	task state segment	S=0, Type=1 or 3
LDT	descriptor table	S=0, Type=2

Table H-2 Descriptor Types

Component	Description	Size
.BASE	segment (or table) base	3 bytes
.LIMIT	segment (or table) limit	1 word
.WCNT	word count	5 bits
.SSEL	segment selector	1 word
.SOFF	segment offset	1 word
.SR	software reserved word	1 word
Components of the Access Right (AR) Field		
.DPL	descriptor privilege level	2 bits
.ED	expand down	1 bit
.W	wirtable	1 bit
.A	accessed	1 bit
.C	conforming	1 bit
.R	readable	1 bit
.P	present	1 bit
.B	busy	1 bit

Table H-3 Descriptor Components

The syntax for these commands depends on the currently specified operation mode of the emulation processor, i.e. Real Address or Protected. Syntax for Real Address Mode is described in Section 5.8 of the User's Guide while syntax for Protected Mode is described below. When operating in Protected Mode, basic syntax for register access commands is organized into -

a) General Registers (Section H.6.4.1)
b) Segment Selectors (Section H.6.4.2)
c) Descriptor Table registers (Section H.6.4.3)
d) Descriptor Table components (Section H.6.4.4)
e) Task State components (Section H.6.4.5)

H.6.4.1 Display/Modify General Registers

R [AX|BX|CX|DX|SP|BP|SI|DI|DS|SS|ES|CS|IP|FS|MS]

When the EP is operating in Protected Mode and segment selectors CS/SS point to an illegal descriptor, MICE will modify the selector to 08H and display one of the following error messages.

"CS=XXXXH - INVALID SELECTOR!" and/or "SS=XXXXH - INVALID SELECTOR!"

If an invalid selector is indicated, the physical base in the descriptor is ignored and the base value is set to a new value (original selector value *10H).

When modifying selectors (CS/DS/ES/SS), MICE uses an algorithm very similar to the one i80286 uses when changing a selector value. Any selector change that is detected to violate any part of the segment descriptor attributes, MICE will display "INVALID SELECTOR" and then display the previous value for prompt modification into a legal value. An example of the detection algorithm used by MICE is shown below for the Code Segment.

Detection algorithm of CS:

```
    if {new-cs} is null selector then displays "NULL SELECTOR"
    {new-cs}'s RPL* must equal CPL* else ERROR
    {new-cs} must be within descriptor table limit else ERROR
    if {new-cs} refers to LDT then
    {new-cs} must be within LDT.LIMIT else ERROR
    {new-cs} descriptor AR* byte must indicate present code segment else ERROR
    if conforming code then CPL must be ≥ DPL else ERROR
    if non_conforming code then CPL must be = DPL else ERROR
    CPU current IP must be within {new-cs} descriptor limit else ERROR
```

where ERROR means that "INVALID SELECTOR!" will be displayed.

*RPL - Requested Privilege Level
*CPL - Current Privilege Level
*AR - Access Rights Byte

Notes:
1. The general register modify (RMS) command does not support mode switching of EP.
2. The trap flag (TF) always reset when EP is operating in protected mode.

Example: Display general registers with an incorrect CS and SS value.

```
>R
 AX=0001 BX=FFFF CX=FFFF DX=FFFF SP=0400 BP=FFFF SI=FFFF  FS  -NPLODITSZ-A-P-C
 DI=FFFF DS=0000 SS=0008 ES=0000 CS=0008 IP=0006 MS=FFF1 0002 0000000000000010
 PC=001006
CS=0100H -- INVALID SELECTOR!
SS=0000H -- INVALID SELECTOR!
>
```

Example: Create a code segment and data segment, then change the CS/SS selectors to point to a correct descriptor.

```
>RDT(10).
TYPE=ESEG  ESEG <CR>
BASE=00FF08 2000 <CR>
LIMIT=FFA0 7FF <CR>
P=1 <CR>
DPL=3 0 <CR>
 C=1 0 <CR>
R=1 1 <CR>
A=1 1 <CR>
SR=FF22 0 <CR>
TYPE=ESEG  <ESC>
CS=0100H -- INVALID SELECTOR!
SS=0000H -- INVALID SELECTOR!
>RDT(20).
TYPE=?     DSEG <CR>
BASE=FF009F 3000 <CR>
LIMIT=00FD 7FF <CR>
P=0 1 <CR>
DPL=0 0 <CR>
ED=0 0 <CR>
W=0 1 <CR>
A=0 1 <CR>
SR=00FF 0 <CR>
TYPE=DSEG  <ESC>
CS=0100H -- INVALID SELECTOR!
SS=0000H -- INVALID SELECTOR!
>RSS
 SS 0008 10 <CR>
INVALID SELECTOR!
 SS 0008 20 <CR>
 ES 0000 <CR>
 CS 0008 20 <CR>
INVALID SELECTOR!
 CS 0008 10 <CR>
 IP 0006 <ESC>
>
```

Example: Display general registers with a correct CS and SS value.

```
>R
 AX=0001 BX=FFFF CX=FFFF DX=FFFF SP=0400 BP=FFFF SI=FFFF  FS  -NPLODITSZ-A-P-C
 DI=FFFF DS=0000 SS=0020 ES=0000 CS=0010 IP=0006 MS=FFF1 0002 0000000000000010
 PC=002006
>
```

H.6.4.2 Display Segment Selectors

R S

This command displays descriptor components of the current on-chip segment selectors. However, note that if a segment selector refers to an illegal descriptor, then "CS(selector) - INVALID SELECTOR!" will be displayed.

Example: Attempt to display a descriptor component of current on-chip segment selectors. Then modify the null selector to an acceptable value and redisplay.

```
>RS
CS (0010)  TYPE=ESEG  BASE=002000 LIMIT=07FF P=1 DPL=0  C=0 R=1 A=1 SR=0000
DS (0000)  NULL SELECTOR!
ES (0000)  NULL SELECTOR!
SS (0020)  TYPE=DSEG  BASE=003000 LIMIT=07FF P=1 DPL=0 ED=0 W=1 A=1 SR=0000
>RDT(18).
TYPE=ESEG  DSEG <CR>
BASE=40FF03 6000 <CR>
LIMIT=FF00 2000 <CR>
P=1 <CR>
DPL=3 0 <CR>
ED=1 0 <CR>
W=1 1 <CR>
A=1 1 <CR>
SR=FF00 0 <CR>
TYPE=DSEG <ESC>
>RDS
 DS 0000 18 <CR>
 SS 0020 <CR>
 ES 0000 18 <CR>
 CS 0010 <ESC>
>RS
CS (0010)  TYPE=ESEG  BASE=002000 LIMIT=07FF P=1 DPL=0  C=0 R=1 A=1 SR=0000
DS (0018)  TYPE=DSEG  BASE=006000 LIMIT=2000 P=1 DPL=0 ED=0 W=1 A=1 SR=0000
ES (0018)  TYPE=DSEG  BASE=006000 LIMIT=2000 P=1 DPL=0 ED=0 W=1 A=1 SR=0000
SS (0020)  TYPE=DSEG  BASE=003000 LIMIT=07FF P=1 DPL=0 ED=0 W=1 A=1 SR=0000
>
```

H.6.4.3 Display/Modify Descriptor Table Registers

Display - R DT
Modify - R GDT|IDT|LDT|TR

If the LDTR (Local Descriptor Table Register) or TR (Task Register) refers an incorrect or null descriptor, then MICE will display -

"LDT(selector) - INVALID SELECTOR!" or "LDT(Selector) - NULL SELECTOR !"
"TR(selector) - INVALID SELECTOR!" or "TR(Selector) - NULL SELECTOR !"

If the GDT or IDT limit is less than 10H or 20H respectively, then MICE will display -

"GDT.LIMIT CAN'T BE LESS THAN 10H!" or
"IDT.LIMIT CAN'T BE LESS THAN 20H!".

If an odd value is inputted for the IDT.BASE, then MICE will display -

"IDT.BASE MUST BE AN EVEN VALUE!".

Example: Attempt to display descriptor table registers referring to the null descriptor. Then modify the selector to an acceptable value and redisplay.

```
>RDT
GDT                    BASE=001400 LIMIT=07FF
IDT                    BASE=000000 LIMIT=FFFF
LDT(0000)  NULL SELECTOR!
TR (0000)  NULL SELECTOR!
>RDT(30).
TYPE=DSEG  LDT <CR>
BASE=00FF18 3800 <CR>
LIMIT=FF00 7FF <CR>
P=1 1 <CR>
DPL=0 0 <CR>
SR=FF02 0 <CR>
TYPE=LDT <ESC>
>RLDT
LDT=0000 30 <CR>
>
```

```
>RDT(40).
TYPE=?      TSS <CR>
BASE=BF00FF 1C00 <CR>
LIMIT=00FF 3F <CR>
P=1 1 <CR>
DPL=0 0 <CR>
B=0 0 <CR>
SR=00FF 0 <CR>
TYPE=TSS <ESC>
>RTR
TR=0000 40 <CR>
>RDT
GDT                    BASE=001400 LIMIT=07FF
IDT                    BASE=000000 LIMIT=FFFF
LDT(0030)  TYPE=LDT    BASE=003800 LIMIT=07FF P=1 DPL=0 SR=0000
TR (0040)  TYPE=TSS    BASE=001C00 LIMIT=003F P=1 DPL=0 B=1 SR=0000
>
```

Example: Attempt to modify the limit of the Global (or Interrupt) Descriptor Table to an illegal value, then change it to acceptable value.

```
>RGDT
GDT. BASE = 000000 123456<CR>
    LIMIT =   FFFF 9<CR>
GDT.LIMIT CAN'T BE LESS THAN 10H!
    LIMIT =   FFFF 3FF<CR>
GDT.BASE  = 123456 <ESC>
>RIDT
IDT. BASE = 000000 123456<CR>
    LIMIT =   FFFF 4<CR>
IDT.LIMIT CAN'T BE LESS THAN 20H!
    LIMIT =   FFFF 3FF<CR>
IDT.BASE  = 123456 <ESC>
>
```

H.6.4.4 Display/Modify/Create Descriptor Table Components

1) Display Descriptor Table Components

R DT(selector)
R DT(sel_reg)
R dtable(index)

DT refers to either the GDT or LDT, depending on the TI (table indicator) bit of the selector or sel_reg (Selector Register)-

when TI=0, then the descriptor table is displayed for the GDT,
when TI=1, then the descriptor table is displayed for the LDT.

selector is a hex value (OH-FFFFH) that points to a descriptor(<LIMIT).

sel_reg indicates the selector register, which includes the CS, DS, ES, SS, LDT and TR (Task Register).

dtable is one of the descriptor tables - GDT, LDT or IDT.

index is an hex value (OH-1FFFH) that points to a dtable address. (INDEX*8+7<=LIMIT)

a) The selector or sel_reg contain an index, a table indicator and an RPL value. Note that the RPL value is masked for display operations.
b) When executing the display command, MICE displays only one descriptor at a time and waits for a <CR> to display the next descriptor. An <LF> will display the previous descriptor, or an <ESC> to terminate the command.
c) If the selector points to an invalid selector, then the 8 bytes of hex code at this location will be displayed from high to low byte without being interpreted.
d) RPL bits of the selector or selector register are ignored. Before displaying the selector, these bits are masked to zero.

Example: Display descriptor table components by specifying a selector.

```
>RDT(10)
DT(0010) TYPE=ESEG  BASE=002000 LIMIT=07FF P=1 DPL=0  C=0 R=1 A=1 SR=0000 <CR>
DT(0018) TYPE=DSEG  BASE=006000 LIMIT=2000 P=1 DPL=0 ED=0 W=1 A=1 SR=0000 <CR>
DT(0020) TYPE=DSEG  BASE=003000 LIMIT=07FF P=1 DPL=0 ED=0 W=1 A=1 SR=0000 <CR>
DT(0028) TYPE=ESEG  BASE=00FF00 LIMIT=FF00 P=1 DPL=3  C=1 R=1 A=1 SR=FF00 <CR>
DT(0030) TYPE=LDT   BASE=003800 LIMIT=07FF P=1 DPL=0 SR=0000 <CR>
DT(0038) 00 FF 00 FF 00 FF 00 FF <CR>
DT(0040) TYPE=TSS   BASE=001C00 LIMIT=003F P=1 DPL=0 B=1 SR=0000 <LF>
DT(0038) 00 FF 00 FF 00 FF 00 FF <LF>
DT(0030) TYPE=LDT   BASE=003800 LIMIT=07FF P=1 DPL=0 SR=0000 <LF>
DT(0028) TYPE=ESEG  BASE=00FF00 LIMIT=FF00 P=1 DPL=3  C=1 R=1 A=1 SR=FF00 <LF>
DT(0020) TYPE=DSEG  BASE=003000 LIMIT=07FF P=1 DPL=0 ED=0 W=1 A=1 SR=0000 <LF>
DT(0018) TYPE=DSEG  BASE=006000 LIMIT=2000 P=1 DPL=0 ED=0 W=1 A=1 SR=0000 <LF>
DT(0010) TYPE=ESEG  BASE=002000 LIMIT=07FF P=1 DPL=0  C=0 R=1 A=1 SR=0000 <ESC>
>
```

Example: Display descriptor table components by specifying a selector register.

```
>RDT(CS)
 DT(0010) TYPE=ESEG  BASE=002000 LIMIT=07FF P=1 DPL=0  C=0 R=1 A=1 SR=0000 <ESC>
>RDT(SS)
 DT(0020) TYPE=DSEG  BASE=003000 LIMIT=07FF P=1 DPL=0 ED=0 W=1 A=1 SR=0000 <ESC>
>RDT(LDT)
 DT(0030) TYPE=LDT   BASE=003800 LIMIT=07FF P=1 DPL=0 SR=0000 <ESC>
>RDT(TR)
 DT(0040)   TYPE=TSS    BASE=001C00  LIMIT=003F  P=1  DPL=0  B=1  SR=0000 <ESC>
>
```

Example: Display descriptor table components by specifying the descriptor table along with an index.

```
>RGDT(2)
GDT(0002) TYPE=ESEG  BASE=002000 LIMIT=07FF P=1 DPL=0  C=0 R=1 A=1 SR=0000 <CR>
GDT(0003) TYPE=DSEG  BASE=006000 LIMIT=2000 P=1 DPL=0 ED=0 W=1 A=1 SR=0000 <CR>
GDT(0004) TYPE=DSEG  BASE=003000 LIMIT=07FF P=1 DPL=0 ED=0 W=1 A=1 SR=0000 <CR>
GDT(0005) TYPE=ESEG  BASE=00FF00 LIMIT=FF00 P=1 DPL=3  C=1 R=1 A=1 SR=FF00 <CR>
GDT(0006) TYPE=LDT   BASE=003800 LIMIT=07FF P=1 DPL=0 SR=0000 <CR>
GDT(0007) 00 FF 00 FF 00 FF 00 FF <CR>
GDT(0008) TYPE=TSS   BASE=001C00 LIMIT=003F P=1 DPL=0 B=1 SR=0000 <CR>
GDT(0009) TYPE=ESEG  BASE=00FF00 LIMIT=FF00 P=1 DPL=3  C=1 R=1 A=1 SR=FF00 <CR>
GDT(000A) 00 FF 00 FF 00 FF 00 FF <ESC>
>
```

2) **Modify/Create Descriptor Table Components**

R DT(selector)[.[component]]
R DT(sel_reg)[.[component]]
R dtable(index)[.[component]]

DT refers to either the GDT or LDT, depending on the TI (table indicator) bit of the selector or sel_reg (Selector Register)-

when TI=0, then the descriptor table is displayed for the GDT,
when TI=1, then the descriptor table is displayed for the LDT.

selector is a hex value (OH-FFFFH) that points to a descriptor.

component is one of the descriptor components prefixed by a period. The components for each descriptor type are listed in Table H-1.

sel_reg indicates the selector register, which includes the CS, DS, ES, SS, LDT and TR.

dtable is one of the descriptor tables - GDT, LDT or IDT.

index is an hex value (OH-1FFFH) that points to a dtable address.

After entering a modify/create command, MICE will display the indicated type of descriptor and wait for a <CR> to display the next component or an <ESC> to terminate the command. If an illegal descriptor type is referred, then MICE will display "?" to indicate that the type is unknown.

There are two messages which will indicate incorrect data:

"ILLEGAL DESCRIPTOR TYPE!" for incorrect type
"ILLEGAL COMPONENT DATA!" for incorrect data

Example: Display Descriptor Table with an index that points to an illegal descriptor type, then modify it to a correct one.

```
>RGDT(10).
TYPE=?     INTG <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     TRAPG <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     TASKG <CR>
SSEL=00EB 40 <CR>
P=0 1 <CR>
DPL=0 0 <CR>
SR=00FF 0 <CR>
TYPE=TASKG <ESC>
>RGDT(10)
GDT(0010) TYPE=TASKG SSEL=0040 P=1 DPL=0 SR=0000 <ESC>
>RIDT(11).
TYPE=?     TSS <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     LDT <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     CALLG <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     DSEG <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     ESEG <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     INTG <CR>
SSEL=00FC 10 <CR>
SOFF=002F 0 <CR>
P=0 1 <CR>
DPL=0 <CR>
SR=00FF 0<CR>
TYPE=INTG <ESC>
>RIDT(11)
IDT(0011) TYPE=INTG  SSEL=0010 SOFF=0000 P=1 DPL=0 SR=0000 <ESC>
>
```

```
>RLDT(12).
TYPE=?    TSS <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?    LDT <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?    INTG <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?    TRAPG <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?    CALLG <CR>
SSEL=BCF4 10 <CR>
SOFF=2189 0 <CR>
P=0 <CR>
DPL=2 0 <CR>
WCNT=0B 20 <CR>
ILLEGAL COMPONENT DATA!
WCNT=0B 1F <CR>
SR=5091 0 <CR>
TYPE=CALLG <ESC>
>RLDT(12)
LDT(0012) TYPE=CALLG SSEL=0010 SOFF=0000 P=1 DPL=0 WCNT=1F SR=0000 <ESC>
>
```

H.6.4.5 Display/Modify Task State Segment Components

R TSS
R TSS(selector)
R TSS.component
R TSS(selector).component

---

selector is an hex value (0H-FFFFH) that points to a descriptor.

component is the name of a task state segment component, prefixed by a period. Any of the following components may be specified:

LNK - Back Link Selector to TSS	AX	ES - ES Selector
SP0 - SP for CPL 0	CX	CS - CS Selector
SS0 - SS for CPL 0	DX	SS - SS Selector
SP1 - SP for CPL 1	BX	DS - DS Selector
SS1 - SS for CPL 1	SP	LDT - Task LDT Selector
SP2 - SP for CPL 2	SI	
SS2 - SS for CPL 2	DI	
IP - (Entry Point)		
FS - Flag Word		

This command will only display/modify the components of the descriptor table; i.e. it will not affect EP register contents. If no selector is specified the current task register (TR) is used as selector. The selector must refer to the TSS descriptor, otherwise "INVALID SELECTOR!" will be displayed.

The specified selector must refer to the GDT, and the index of the selector must be less than GDT.limit, otherwise "INVALID SELECTOR!" will be displayed.

After entering the modify command, MICE displays the specified component and waits for a <CR> to display the next item, a <LF> to display the preceding item or an <ESC> to terminate the command.

Example: Display/Modify the current Task State Segment components.

```
>RTSS
LNK=00F7 SP0=00BF SS0=00F7 SP1=009F SS1=00FE SP2=00FF SS2=00FE  IP=00DF
 FS=FF00  AX=FF00  CX=FF80  DX=FF00  BX=7F00  SP=FF00  BP=FF30  SI=FF08
 DI=00F9  ES=207F  CS=00FF  SS=00CF  DS=00FE LDT=00AF
>RTSS.IP
 IP=00DF 0 <CR>
 FS=FF00 2 <CR>
 AX=FF00 1111 <CR>
 CX=FF80 2222 <CR>
 DX=FF00 3333 <CR>
 BX=7F00 4444 <CR>
 SP=FF00 200 <CR>
 BP=FF30 6666 <CR>
 SI=FF08 7777 <CR>
 DI=00F9 8888 <CR>
 ES=207F 0 <CR>
 CS=00FF 10 <CR>
 SS=00CF 20 <CR>
 DS=00FE 18 <CR>
LDT=00AF 30 <CR>
LNK=00F7 0 <CR>
SP0=00BF 200 <CR>
SS0=00F7 28 <CR>
SP1=009F 2000 <CR>
SS1=00FE 49 <CR>
SP2=00FF 300 <CR>
SS2=00FE 5A <CR>
 IP=0000 <ESC>
>RTSS
LNK=0000 SP0=0200 SS0=0028 SP1=2000 SS1=0049 SP2=0300 SS2=005A  IP=0000
 FS=0002  AX=1111  CX=2222  DX=3333  BX=4444  SP=0200  BP=6666  SI=7777
 DI=8888  ES=0000  CS=0010  SS=0020  DS=0018 LDT=0030
>
```

H.6.5 Memory Manipulation Commands

This section describes the following memory manipulation commands -

a) Memory Display/Modify/Fill/Search
b) Memory Test/Transfer/Checksum/Compare
c) Assembly/Disassembly
d) Download/Upload

1) If the command operands refer to correct descriptors:

   a) Memory Display/Modify/Fill/Search/Test/Checksum, Assembly and Disassembly commands will display -

                              SEGMENT BASE = XXXXXX

   to indicate the physical base value of the accessed selector.

   Example:

```
>M0,F, A1 34 12 A3 34 12 90 90
SEGMENT BASE = 002000
>Z0,F
SEGMENT BASE = 002000
LOC         OBJ             LINE LABEL      SOURCE CODE
0010:0000   A13412          0001            MOV     AX,1234
0010:0003   A33412          0002            MOV     1234,AX
0010:0006   90              0003            NOP
0010:0007   90              0004            NOP
0010:0008   A13412          0005            MOV     AX,1234
0010:000B   A33412          0006            MOV     1234,AX
0010:000E   90              0007            NOP
0010:000F   90              0008            NOP
     DISASSEMBLY COMPLETED
>
```

   b) Transfer/Compare commands will display -

                                  SOURCE SEGMENT BASE = XXXXXX
                             DESTINATION SEGMENT BASE = XXXXXX

   Example:

```
>T0,F 18:0
     SOURCE SEGMENT BASE = 002000
DESTINATION SEGMENT BASE = 006000
>T0,F,18:0,V
     SOURCE SEGMENT BASE = 002000
DESTINATION SEGMENT BASE = 006000
 COMPARISON OK !
>
```

   c) Upload/Download commands will not display the preceding message.

2) If the command operands refer to incorrect descriptors:

a) Memory Display/Modify/Fill/Search/Test/Checksum, Assembly and Disassembly commands will display -

```
SEGMENT NOT PRESENT
WARNING: DISPLAY IN PHYSICAL ADDRESS FORM? (Y/N)
```

Enter "Y" to set the physical base value to the "selector x 10H", or "N" to abort to the command prompt.

Example:

```
>M100:0,F,90
 SEGMENT NOT PRESENT ERROR
 WARNING:DISPLAY IN PHYSICAL ADDRESS FORMS?(Y/N)Y
>
```

b) Transfer/Compare commands will display either of the following messages:

If the source selector refers to an invalid descriptor, then MICE will display -

```
SOURCE SEGMENT NOT PRESENT
WARNING: DISPLAY IN PHYSICAL ADDRES FORM? (Y/N)
```

If the destination selector refers to a wrong descriptor, then MICE will display -

```
DESTINATION SEGMENT NOT PRESENT
WARNING: DISPLAY IN PHYSICAL ADDRESS FORM? (Y/N)
```

c) Upload/Download commands will not display the preceding message; they will automatically use physical address form.

3) Upload/Download commands using Intel format support record types 0-3 for both Read Address and Protection mode.

Note: There are some differences between Upload/Download commands for 8086(80186) and 80286:

- Type 2 will not modify the current CS register (because the CS must refer to an executable segment descriptor, otherwise EP will generate an Exception for protection).

- When user wants to implement the driver of 80286, the Upload command must follow the syntax shown below, as soon as each different segment is executed-

  U segment: start-address, end-address[format]

Example: Display memory using a correct selector.

```
>M20:0 1F
SEGMENT BASE = 00A4E0
           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0020:0000  B8 00 00 C1 E8 04 8E D8 8E D0 BC 00 04 8C D8 8E  ................
0020:0010  C0 BF 48 10 8B F3 81 C6 02 00 8B C2 B1 0C D3 E0  ..H.............
>
```

Example: First display the general registers, execute a memory fill and then display the results.

```
>R
 AX=FF00 BX=03AA CX=0000 DX=0021 SP=03A6 BP=03C4 SI=0080  FS  -NPLODITSZ-A-P-C
 DI=0000 DS=0018 SS=0048 ES=0018 CS=0020 IP=01EC MS=FFF1 0002 0000000000000010
 PC=00A6CC
>MDS 0 1F 55 66 77 88
SEGMENT BASE = 000000
>MDS 0 1F
SEGMENT BASE = 000000
           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0018:0000  55 66 77 88 55 66 77 88 55 66 77 88 55 66 77 88  Ufw.Ufw.Ufw.Ufw.
0018:0010  55 66 77 88 55 66 77 88 55 66 77 88 55 66 77 88  Ufw.Ufw.Ufw.Ufw.
>
```

Example: Attempt to display a memory range, using an invalid selector type in the command. Note that the CS and SS contained in displayed data range also point to an invalid selector type.

```
>M100:0 1F
SEGMENT NOT PRESENT
WARNING: DISPLAY IN PHYSICAL ADDRESS FORM? (Y/N)Y
           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0100:0000  00 00 00 00 00 00 00 00 FF 07 00 10 00 42 00 00  ..............B..
0100:0010  FF 07 00 60 00 98 00 00 FF 07 00 68 00 92 00 00  ...........h....
>
```

Example: Disassemble using a correct selector.

```
>ZCS 0 1F
SEGMENT BASE = 00A4E0
LOC         OBJ             LINE LABEL      SOURCE CODE
0020:0000   B80000          0001            MOV     AX,#0000
0020:0003   C1E804          0002            SHR     AX,04
0020:0006   8ED8            0003            MOV     DS,AX
0020:0008   8ED0            0004            MOV     SS,AX
0020:000A   BC0004          0005            MOV     SP,#0400
0020:000D   8CD8            0006            MOV     AX,DS
0020:000F   8EC0            0007            MOV     ES,AX
0020:0011   BF4810          0008            MOV     DI,#1048
0020:0014   8BF3            0009            MOV     SI,BX
0020:0016   81C60200        0010            ADD     SI,#0002
0020:001A   8BC2            0011            MOV     AX,DX
0020:001C   B10C            0012            MOV     CL,#0C
0020:001E   D3E0            0013            SAL     AX,CL
     DISASSEMBLY COMPLETED
>
```

Example: Disassemble using an incorrect selector.

```
>Z600:10 23
INVALID SELECTOR
WARNING: DISPLAY IN PHYSICAL ADDRESS FORM? (Y/N)Y
LOC         OBJ             LINE LABEL      SOURCE CODE
0600:0010   EB00            0001            JMP     6012
0600:0012   B85000          0002 B6012      MOV     AX,#0050
0600:0015   0F00D8          0003            LTR     AX
0600:0018   B84000          0004            MOV     AX,#0040
0600:001B   0F00D0          0005            LLDT    AX
0600:001E   B81400          0006            MOV     AX,#0014
0600:0021   8ED0            0007            MOV     SS,AX
0600:0023   BC0000          0008            MOV     SP,#0000
     DISASSEMBLY COMPLETED
>
```

H.6.6 Trace Commands

H.6.6.1 B/F/BT Commands

a) Refer to Section H.6.1 for the detailed description of the algorithm governing segment value modification.

b) When the EP is currently operating in Protected Mode, MICE is operating in "P" mode, and the selector points to a logical trigger address (selector:offset), then only the discriptor types ESEG, DSEG, TSS and LDT are acceptable. Also note that the offset must be within the range of the accessed descriptor.

H.6.6.2 BP Command

a) Execution

BP<addr1 [addr2 [addr3 [addr4]]]> [IDT = a1]

BP is the command keyword for Execution Breakpoint tracing.

addr1-4 are up to four logical addresses that specify breakpoints where emulation is to stop. Note that both the segment value and offset must be specified. If more than one breakpoint is specified, angular brackets < > must enclose the addresses to indicate a logical OR construct. All four addresses serve as software breakpoints, except when using "addr1" where it serves as a hardware breakpoint without any limitation on execution in either RAM or ROM.

a1 is the base value (in the range of OH - FFFFFFH) for the Interrupt Descriptor Table (IDT) register. This value is only required for input if it will be changed by the user's program. If this parameter is omitted, default is for the current base value when tracing is initiated.

The Execution Breakpoint command starts real time emulation of the target; it immediately begins recording target status information, and stops tracing when any of up to four permissable breakpoints is executed. The trace can record up to 2048 machine cycles. If more than 2048 cycles elapse before the trace ends, only the last 2048 cycles are recorded. The recorded information can be examined by using the List Trace Buffer command.

The CPU stops whenever the trace stops. All machine cycles are recorded up to the trigger-address, including prefetch cycles (although the prefetch cycles do not influence program execution).

Although the trace buffer records all bus activity including prefetch cycles, the BP command causes the CPU to break at a point in the program where a specified address is actually to be executed and not just prefetched.

A realtime Execution Breakpoint is achieved by inserting a software interrupt (ie. code INT 3 [OCCH]) into memory at the specified breakpoint address. A halt breakpoint is then set at the interrupt vector fetch address. This ensures that the execution breakpoint is triggered only if the interrupt vector fetch cycle is executed, ignoring all prefetch cycles.

Example: The following program segment illustrates the difference between the Execution Breakpoint command and traditional breakpoints based on bus activity.

```
   :        :
   :        :
BRANCH: MOV SI,#4000  ;Source pointer for data read.
        MOV DI,#3000  ;Destination pointer for data write.
        MOV CX,#10    ;Move 10 words of data.
        REP           ;From 4000H to 3000H.
        MOVS, WORD
  DONE: JMP BRANCH    ;Finish and restart.
   :        :
   :        :
```

Setting a traditional breakpoint at DONE in the above example would cause the program to break at the bus prefetch cycle, even though program execution has not reached the breakpoint. Also note that if WORD access is used, and a traditional breakpoint is set at an odd address, the program may never break because program data is fetched by word addresses.

If the same trigger-address is set using the BP command, then execution will jump to another area without reaching the breakpoint. The program will break only if execution reaches the breakpoint address (either odd or even).

Example: Messages for the Execution Breakpoint command are illustrated below.

```
>X100                                          ;Default is for CS=100, IP=0.
>MO FF 90                                      ;Fill memory 0-FF with NOP.
>BP <100:8,0:8>                                ;
TRACE STOPS AT 0100:0008, STEP 0006            ;Current register values are
                                               ; CS=100, IP=8.
>BP <100:B,100:0                               ;Right bracket > is optional.
TRACE STOPS AT 0100:000B, STEP 0004            ;
>BP <100:0,1000:560>                           ;
ADDRESS 1000:0560 PROTECTED, COMMAND ABORTED!  ;Memory protected address.
>BP 100:B                                      ;
PC=0100:000B, TRACE ABORTED!                   ;PC=100B already.
>
```

b) APPLICATION NOTES

1) During execution of the BP command, breakpoints H1-6 are all disabled automatically; other than H1 they will be reenabled when execution has completed. (H3-6 are BPP options.)

2) Stack contents from SP-1 thru SP-6 will be modified if the breakpoint (INT 3 code) is executed; but the SS and SP registers will not be affected.

3) Memory for breakpoint addresses is modified to OCCH during command execution*; the List command will therefore display this data, instead of the user's code, for instruction prefetch cycles. This has no effect on program execution, though, as the original data is restored after the interrupt instruction is executed.

   * Because the first breakpoint address of the 80286 is a hardware execution breakpoint, the data at this memory address will not be modified. The List command will display the user's code when this address is located in HUEM and OFFH when it is located in the target.

4) When the program stops at the breakpoint address, the user's program instruction at that location has not yet been executed. The same trigger-address cannot be specified again. Setting any breakpoint where the PC equals the trigger's physical address will generate an error message-

   "PC=trigger address, TRACE ABORTED!"

5) INT 3 instructions defined in the user's program are executed correctly without any conflict with internal INT 3 codes generated by BP commands.

6) The right angular bracket ">" is optional (illustrated in line 5 of the above example). However, this bracket is mandatory if the base value of the IDT is specified.

c) LIMITATIONS

The following limitations apply to the Execution Breakpoint:

1) The emulation program must be placed in RAM, where it will be tested before the trace starts; except when using addr1, which may be placed in either RAM or ROM.

2) If any memory read cycle occurs at 0000:000CH when EP is running in real-address mode(or an INT 3 instruction in the user's program is executed), a few non-real time cycles will be inserted to process this code.

3) If any memory read cycle occurs at IDT BASE + 1CH, MICE will display-

"WORD ACCESS AT IDT + 1CH MAY GIVE INCORRECT DATA !"

and stop tracing when the current EP is operating in protected mode.

4) Breakpoints must be set at the first byte of the instruction.

5) To avoid illegal operation with string instructions (e.g. MOVS, SCAS, OUTS, etc.), treat repetitive instructions (e.g. REPE, REPNE, REPNZ, etc.) and string instuctions as one instruction. Set the trigger address at repetitive instructions only, and never at string instructions. In the above example, the trigger address should never be set at instruction MOVS; set it at instruction REP instead.

6) When both EP and MICE are operating in protected mode, the selector of a logical trigger address must point to a executable segment descriptor.

7) Whenever any offset is outside the "LIMIT" component of the segment value, MICE will stop EP and display (for Protected Mode only)-

"OFFSET ERROR !"

8) The BP command can only be used in MICE Debug Mode.

9) The limit of the IDT cannot be set at less than 0FH when EP is running in Real-Mode, or less than 20H when EP is running in Protected-Mode and GDT cannot be less than 10H in Protected Mode, otherwise the emulation processor would hang up.

10) The EP cannot operate in switching mode when BP command is executed.

H.6.7 Program Examples:

HOW TO TAKE ADVANTAGE OF MICE TO CREATE A PROTECTED TASK SYSTEM

```
>r     *** MICE-II 80286 V4.0 ***                ; Warm start (or cold start)
DEBUG MODE, MODE R; NMI AND PEREQ DISABLED!      ;
NO TARGET VCC; HOLD, INTR AND RESET DISABLED!    ; Set the HUEM address from 000H to
>RGDT                                            ; OFFFFH (set S1_10 off, others on)
 GDT.BASE 001800 1400 <CR>                       ; Locate the GDT on the available
    .LIMIT  FFFE 7FF <CR>                        ;  memory spaces.
 IDT.BASE 000000 0 <CR>                          ; Locate the IDT on the available
    .LIMIT  FFFF 7FF <CR>                        ;  memory spaces.
 AX 0001 <ESC>                                   ;
>RSP                                             ; Set SP to a fixed number.
 SP 03F4 400 <CR>
 BP FFFF <ESC>
>MODE P                                          ; Set emulation mode to protective.
>IDT=0                                           ; Input the IDT.BASE to MICE when user
>X100                                            ;  wants to emulate protective mode.
>A
LOC         OBJ             LINE LABEL      SOURCE CODE
0100:0000   B80100          0001            MOV AX,#1    ; The program which allow EP
0100:0003   0F01F0          0002            LMSW AX      ;  to switch from real mode to
0100:0006   EA00001000      0003            JMP 10:0     ;  protected mode.
0100:000B                   0004
>SZ
       ADDRESS DATA     SOURCE CODE          ; Instruction steps from real mode
        001000 01B8     MOV     AX,#0001     ;  to protected mode.
        001003   0F     LMSW    AX           ; The current EP mode is in the
        001006 00EA     JMP     0010:0000    ;  protected mode.
CS=0100H -- INVALID SELECTOR!                ; The current values of CS and SS
SS=0000H -- INVALID SELECTOR!                ;  are defined in real address mode.
                                             ; But the descriptors in the descriptor
                                             ;  table are still undefined.
```

```
>RDT(10).                                    ; Create an executable segment
TYPE=ESEG  DSEG <CR>                        ;  descriptor on GDT.
BASE=00FF00 2000 <CR>
LIMIT=FF00 <CR>
P=1 1 <CR>
DPL=3 0 <CR>
ED=1 0 <CR>
W=1 1 <CR>
A=1 1 <CR>
SR=FF00 0 <CR>
TYPE=DSEG  ESEG <CR>
BASE=002000 <CR>
LIMIT=FF00 7FF <CR>
P=1 <CR>
DPL=0 0 <CR>
C=0 0 <CR>
R=1 1 <CR>
A=1 1 <CR>
SR=0000 0 <CR>
TYPE=ESEG <ESC>
CS=0100H -- INVALID SELECTOR!
SS=0000H -- INVALID SELECTOR!
>RDT(20).                                   ; Create a data segment descriptor
TYPE=?     DSEG <CR>                        ;  on GDT.
BASE=FF00FF 3000 <CR>
LIMIT=00FF 7FF <CR>
P=0 1 <CR>
DPL=0 0 <CR>
ED=0 0 <CR>
W=0 1 <CR>
A=0 1 <CR>
SR=00FF 0 <CR>
TYPE=DSEG <ESC>
CS=0100H -- INVALID SELECTOR!
SS=0000H -- INVALID SELECTOR!
>SZ
      ADDRESS DATA     SOURCE CODE
       001006 00EA     JMP     0010:0000
CS=0100H -- INVALID SELECTOR!
SS=0000H -- INVALID SELECTOR!
         001414-R-9B00
         001410-R-07FF
         001412-R-2000
         001414-W-9B00
        002000 00FF     INC     WORD [BX][SI]
SS=0000H -- INVALID SELECTOR!
```

```
>RSS                                          ; Modifies SS selector and refers
 SS 0008 20                                   ;  to correct descriptor.
 ES 0000 <ESC>
>MO FF 90
SEGMENT BASE = 002000
>Z
SEGMENT BASE = 002000
LOC         OBJ             LINE LABEL      SOURCE CODE
0010:0000   90              0001            NOP
0010:0001   90              0002            NOP
0010:0002   90              0003            NOP
0010:0003   90              0004            NOP
0010:0004   90              0005            NOP
0010:0005   90              0006            NOP
0010:0006   90              0007            NOP
0010:0007   90              0008            NOP
0010:0008   90              0009            NOP
0010:0009   90              0010            NOP
0010:000A   90              0011            NOP
0010:000B   90              0012            NOP
0010:000C   90              0013            NOP
0010:000D   90              0014            NOP
0010:000E   90              0015            NOP
0010:000F   90              0016            NOP
     DISASSEMBLY COMPLETED

>RS
CS (0010)  TYPE=ESEG  BASE=002000 LIMIT=07FF P=1 DPL=0  C=0 R=1 A=1 SR=0000
DS (0000)  NULL SELECTOR!                     ; DS and ES are non-defined.
ES (0000)  NULL SELECTOR!
SS (0020)  TYPE=DSEG  BASE=003000 LIMIT=07FF P=1 DPL=0 ED=0 W=1 A=1 SR=0000
>RDT(18).
TYPE=ESEG  DSEG <CR>
BASE=00FF00 6000 <CR>
LIMIT=FF00 2000 <CR>
P=1 1 <CR>
DPL=3 0 <CR>
ED=1 0 <CR>
W=1 1 <CR>
A=1 1 <CR>
SR=FF00 0 <CR>
TYPE=DSEG <ESC>
```

```
>RDS
 DS 0000 18 <CR>
 SS 0020 <CR>                                   ; Initializes the DS and ES 2
 ES 0000 18 <CR>                                ;  selectors.
 CS 0010 <ESC>
>RS
CS (0010)  TYPE=ESEG  BASE=002000 LIMIT=07FF P=1 DPL=0  C=0 R=1 A=1 SR=0000
DS (0018)  TYPE=DSEG  BASE=006000 LIMIT=2000 P=1 DPL=0 ED=0 W=1 A=1 SR=0000
ES (0018)  TYPE=DSEG  BASE=006000 LIMIT=2000 P=1 DPL=0 ED=0 W=1 A=1 SR=0000
SS (0020)  TYPE=DSEG  BASE=003000 LIMIT=07FF P=1 DPL=0 ED=0 W=1 A=1 SR=0000
>RDT
GDT                   BASE=001400 LIMIT=07FF
IDT                   BASE=000000 LIMIT=07FF
LDT(0000)  NULL SELECTOR!                      ; LDT and TR are non-defined.
TR (0000)  NULL SELECTOR!
>RDT(30).                                      ; Creates a LDT descriptor
TYPE=ESEG  LDT <CR>                            ;  on GDT.
BASE=00FF00 3800 <CR>
LIMIT=FF00 7FF <CR>
P=1 <CR>
DPL=0 <CR>
SR=FF00 0 <CR>
TYPE=LDT <ESC>
>RLDT
LDT=0000 30 <CR>
>RDT(40).                                      ; Creates a TSS descriptor
TYPE=?     TSS <CR>                            ;  on GDT.
BASE=FF00FF 1C00 <CR>
LIMIT=00FF 3F <CR>
P=1 1 <CR>
DPL=0 0 <CR>
B=0 0 <CR>
SR=00FF 0 <CR>
TYPE=TSS <ESC>
>RTR
TR=0000 40 <CR>
>RDT
GDT                   BASE=001400 LIMIT=07FF
IDT                   BASE=000000 LIMIT=07FF
LDT(0030)  TYPE=LDT   BASE=003800 LIMIT=07FF P=1 DPL=0 SR=0000
TR (0040)  TYPE=TSS   BASE=001C00 LIMIT=003F P=1 DPL=0 B=1 SR=0000
```

```
>RDT(0)                                        ; "0" is a selector.
 DT(0000) 00 FF 00 FF 00 FF 00 FF <CR>
 DT(0008) 00 FF 00 FF 00 FF 00 FF <CR>
 DT(0010) TYPE=ESEG  BASE=002000 LIMIT=07FF P=1 DPL=0  C=0 R=1 A=1 SR=0000 <CR>
 DT(0018) TYPE=DSEG  BASE=006000 LIMIT=2000 P=1 DPL=0 ED=0 W=1 A=1 SR=0000 <CR>
 DT(0020) TYPE=DSEG  BASE=003000 LIMIT=07FF P=1 DPL=0 ED=0 W=1 A=1 SR=0000 <CR>
 DT(0028) 00 FF 00 FF 00 FF 00 FF <CR>
 DT(0030) TYPE=LDT   BASE=003800 LIMIT=07FF P=1 DPL=0 SR=0000 <CR>
 DT(0038) TYPE=ESEG  BASE=00FF00 LIMIT=FF00 P=1 DPL=3  C=1 R=1 A=1 SR=FF00 <CR>
 DT(0040) TYPE=TSS   BASE=001C00 LIMIT=003F P=1 DPL=0 B=1 SR=0000 <CR>
 DT(0048) 00 FF 00 FF 00 FF 00 FF <CR>
 DT(0050) TYPE=ESEG  BASE=00FF00 LIMIT=FF00 P=1 DPL=3  C=1 R=1 A=1 SR=FF00 <CR>
 DT(0058) TYPE=ESEG  BASE=00FF00 LIMIT=FF00 P=1 DPL=3  C=1 R=1 A=1 SR=FF00 <ESC>
>RGDT(010).                                    ; "10" is the index of GDT.
TYPE=?     INTG <CR>                           ;
ILLEGAL DESCRIPTOR TYPE!                       ; GDT may contain all descriptor types
TYPE=?     TRAPG <CR>                          ;  except interrupt and trap gate.
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     TASKG <CR>
SSEL=00FF 40 <CR>
P=0 0 <CR>
DPL=0 <CR>
SR=00FF 0 <CR>
TYPE=TASKG <CR>
SSEL=0040 <CR>
P=0 1 <CR>
DPL=0 <ESC>
>RGDT(10)
GDT(0010) TYPE=TASKG SSEL=0040 P=1 DPL=0 SR=0000<ESC>
>RIDT(11).                                     ; The IDT may only contain task gate,
TYPE=?     TSS <CR>                            ;  interrupt gate and trap gate.
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     LDT <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     CALLG <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     DSEG <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     ESEG <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=?     INTG <CR>
SSEL=00FF 10 <CR>
SOFF=00FF 0 <CR>
P=0 1 <CR>
DPL=0 <CR>
SR=00FF 0 <CR>
TYPE=INTG  <ESC>
```

```
>RIDT(11)
IDT(0011) TYPE=INTG  SSEL=0010 SOFF=0000 P=1 DPL=0 SR=0000 <ESC>
>RLDT(12).                          ; The LDT may only contain segment, task
TYPE=ESEG  TSS <CR>                 ;  gate and call gate descriptors.
ILLEGAL DESCRIPTOR TYPE!
TYPE=ESEG  LDT <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=ESEG  INTG <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=ESEG  TRAPG <CR>
ILLEGAL DESCRIPTOR TYPE!
TYPE=ESEG  CALLG <CR>
SSEL=FF00 10 <CR>
SOFF=FF00 0 <CR>
P=1 1 <CR>
DPL=3 2 <CR>
WCNT=10 20 <CR>                     ; The maximun for word count(WCNT)
ILLEGAL COMPONENT DATA!             ;  must be less than or equal to 31
WCNT=10 1F <CR>                     ;  (or 01FH).
SR=0000 <ESC>
>RLDT(12)
LDT(0012) TYPE=CALLG SSEL=0010 SOFF=0000 P=1 DPL=2 WCNT=1F SR=0000 <ESC>
>RTSS                               ; Displays the current task state segment contents.
LNK=00FF SP0=00FF SS0=00FF SP1=00FF SS1=00FF SP2=00FF SS2=00FF  IP=00FF
 FS=FF00  AX=FF00  CX=FF00  DX=FF00  BX=FF00  SP=FF00  BP=FF00  SI=FF00
 DI=00FF  ES=00FF  CS=00FF  SS=00FF  DS=00FF LDT=00FF
```

```
>RTSS.LDT
LDT=00FF 30 <CR>                    ; Modifies the current task state
LNK=00FF 0 <CR>                     ;  segment contents.
SP0=00FF 200 <CR>
SS0=00FF 28 <CR>
SP1=00FF 200 <CR>
SS1=00FF 48 <CR>
SP2=00FF 200 <CR>
SS2=00FF 58 <CR>
 IP=00FF 0 <CR>
 FS=FF00 2 <CR>
 AX=FF00 1111 <CR>
 CX=FF00 2222 <CR>
 DX=FF00 3333 <CR>
 BX=FF00 4444 <CR>
 SP=FF00 400 <CR>
 BP=FF00 6666 <CR>
 SI=FF00 7777 <CR>
 DI=00FF 8888 <CR>
 ES=00FF 0 <CR>
 CS=00FF 10 <CR>
 SS=00FF 20 <CR>
 DS=00FF 18 <CR>
LDT=0030 <ESC>
>RTSS                               ; Displays the modified task state segment.
LNK=0000 SP0=0200 SS0=0028 SP1=0200 SS1=0048 SP2=0200 SS2=0058 . IP=0000
 FS=0002  AX=1111  CX=2222  DX=3333  BX=4444  SP=0400  BP=6666  SI=7777
 DI=8888  ES=0000  CS=0010  SS=0020  DS=0018 LDT=0030
>M0,F
SEGMENT BASE = 002000
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0010:0000  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
>TO,F 18:0
     SOURCE SEGMENT BASE = 002000
DESTINATION SEGMENT BASE = 006000
>M18:0,F
SEGMENT BASE = 006000
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F     ASCII-CODE
0018:0000  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
>TO,F 18:0 V
     SOURCE SEGMENT BASE = 002000
DESTINATION SEGMENT BASE = 006000
 COMPARISON OK !
>
```

H.7 Assembly Mnemonic Code Summary, with Disassembly Examples

H.7.1 8086/8088

```
LOC           OBJ             LINE LABEL      SOURCE CODE
0000:0000     37              0001            AAA
0000:0001     D50A            0002            AAD
0000:0003     D40A            0003            AAM
0000:0005     3F              0004            AAS
0000:0006     11D8            0005            ADC     AX,BX
0000:0008     11DF            0006            ADC     DI,BX
0000:000A     10DD            0007            ADC     CH,BL
0000:000C     13160020        0008            ADC     DX,2000
0000:0010     13840020        0009            ADC     AX,[SI]2000
0000:0014     124820          0010            ADC     CL,[BX][SI]20
0000:0017     13880020        0011            ADC     CX,[BX][SI]2000
0000:001B     119D0020        0012            ADC     [DI]2000,BX
0000:001F     11B80023        0013            ADC     [BX][SI]23000,DI
0000:0023     11060030        0014            ADC     3000,AX
0000:0027     12060300        0015            ADC     AL,0003
0000:002B     1430            0016            ADC     AL,#30
0000:002D     13060012        0017            ADC     AX,1200
0000:0031     157698          0018            ADC     AX,#9876
0000:0034     819100400080    0019            ADC     WORD [BX][DI]4000,#8000
0000:003A     12366500        0020            ADC     DH,0065
0000:003E     130E3204        0021            ADC     CX,0432
0000:0042     81D30033        0022            ADC     BX,#3300
0000:0046     01D8            0023            ADD     AX,BX
0000:0048     01D1            0024            ADD     CX,DX
0000:004A     01F7            0025            ADD     DI,SI
0000:004C     01EB            0026            ADD     BX,BP
0000:004E     030E0030        0027            ADD     CX,3000
0000:0052     03060623        0028            ADD     AX,2306
0000:0056     034430          0029            ADD     AX,[SI]30
0000:0059     03910020        0030            ADD     DX,[BX][DI]2000
0000:005D     015B24          0031            ADD     [BP][DI]24,BX
0000:0060     01850005        0032            ADD     [DI]0500,AX
0000:0064     010E0020        0033            ADD     2000,CX
0000:0068     003E4000        0034            ADD     0040,BH
0000:006C     02060300        0035            ADD     AL,0003
0000:0070     03065604        0036            ADD     AX,0456
0000:0074     0430            0037            ADD     AL,#30
0000:0076     057689          0038            ADD     AX,#8976
0000:0079     21D8            0039            AND     AX,BX
0000:007B     21F9            0040            AND     CX,DI
0000:007D     20CF            0041            AND     BH,CL
0000:007F     23360030        0042            AND     SI,3000
```

```
LOC          OBJ          LINE LABEL      SOURCE CODE
0000:0083    23978023     0043            AND    DX,[BX]2380
0000:0087    235820       0044            AND    BX,[BX][SI]20
0000:008A    234590       0045            AND    AX,[DI]90
0000:008D    22360060     0046            AND    DH,6000
0000:0091    212E3000     0047            AND    0030,BP
0000:0095    214587       0048            AND    [DI]87,AX
0000:0098    217130       0049            AND    [BX][DI]30,SI
0000:009B    20060012     0050            AND    1200,AL
0000:009F    E85E2F       0051            CALL   3000
0000:00A2    FF1E0020     0052            CALL   DWORD 2000
0000:00A6    FF5F40       0053            CALL   DWORD [BX]40
0000:00A9    98           0054            CBW
0000:00AA    F8           0055            CLC
0000:00AB    FC           0056            CLD
0000:00AC    FA           0057            CLI
0000:00AD    39D0         0058            CMP    AX,DX
0000:00AF    39EE         0059 B00AF      CMP    SI,BP
0000:00B1    38CF         0060            CMP    BH,CL
0000:00B3    39360020     0061            CMP    2000,SI
0000:00B7    382E0030     0062            CMP    3000,CH
0000:00BB    39888012     0063            CMP    [BX][SI]1280,CX
0000:00BF    3B9A5634     0064            CMP    BX,[BP][SI]3456
0000:00C3    3A060030     0065            CMP    AL,3000
0000:00C7    3C20         0066            CMP    AL,#20
0000:00C9    3D0909       0067            CMP    AX,#0909
0000:00CC    3B069909     0068            CMP    AX,0999
0000:00D0    83B900302B   0069            CMP    WORD [BX][DI]3000,#2B
0000:00D5    80B9004032   0070            CMP    BYTE [BX][DI]4000,#32
0000:00DA    A6           0071            CMPS   BYTE
0000:00DB    A7           0072            CMPS   WORD
0000:00DC    99           0073            CWD
0000:00DD    27           0074 B00DD      DAA
0000:00DE    2F           0075            DAS
0000:00DF    FE0E0020     0076            DEC    BYTE 2000
0000:00E3    FF08         0077            DEC    WORD [BX][SI]
0000:00E5    FECB         0078            DEC    BL
0000:00E7    F636003B     0079            DIV    BYTE 3B00
0000:00EB    F736         0080            DIV    WORD F6F4
0000:00EF    F4           0081            HLT
0000:00F0    F67F30       0082            IDIV   BYTE [BX]30
0000:00F3    F7BC0090     0083            IDIV   WORD [SI]9000
0000:00F7    F6AF0040     0084            IMUL   BYTE [BX]4000
0000:00FB    F7AC5643     0085            IMUL   WORD [SI]4356
0000:00FF    ED           0086            IN     AX,DX
0000:0100    EC           0087 B0100      IN     AL,DX
0000:0101    40           0088            INC    AX
```

```
LOC           OBJ              LINE  LABEL     SOURCE  CODE
0000:0102     42               0089            INC     DX
0000:0103     47               0090            INC     DI
0000:0104     CD02             0091            INT     02
0000:0106     CD67             0092            INT     67
0000:0108     CE               0093            INTO
0000:0109     CF               0094            IRET
0000:010A     77FE             0095  B010A     JNBE    010A
0000:010C     7704             0096            JNBE    0112
0000:010E     77CD             0097            JNBE    00DD
0000:0110     730B             0098            JNB     011D
0000:0112     72F6             0099  B0112     JB      010A
0000:0114     7617             0100            JBE     012D
0000:0116     E31E             0101            JCXZ    0136
0000:0118     74E6             0102            JE      0100
0000:011A     7FF1             0103            JNLE    010D
0000:011C     7D91             0104            JNL     00AF
0000:011E     7ED1             0105            JLE     00F1
0000:0120     E98A4F           0106            JMP     50AD
0000:0123     EA00401000       0107            JMP     0010:4000
0000:0128     EBFE             0108  B0128     JMP     0128
0000:012A     FFEB             0109            JMP     DWORD BX
0000:012C     FFE6             0110            JMP     WORD SI
0000:012E     76D5             0111            JBE     0105
0000:0130     7204             0112            JB      0136
0000:0132     73A9             0113            JNB     00DD
0000:0134     77BB             0114            JNBE    00F1
0000:0136     75FE             0115  B0136     JNE     0136
0000:0138     7ED3             0116            JLE     010D
0000:013A     7C0A             0117            JL      0146
0000:013C     7D7E             0118            JNL     01BC
0000:013E     7FD1             0119            JNLE    0111
0000:0140     71D1             0120            JNO     0113
0000:0142     7932             0121            JNS     0176
0000:0144     75A7             0122            JNE     00ED
0000:0146     7064             0123  B0146     JO      01AC
0000:0148     7A63             0124            JP      01AD
0000:014A     7871             0125            JS      01BD
0000:014C     74CF             0126            JE      011D
0000:014E     74AD             0127            JE      00FD
0000:0150     9F               0128            LAHF
0000:0151     C59C0005         0129            LDS     BX,[SI]0500
0000:0155     8D03             0130            LEA     AX,[BP][DI]
0000:0157     C4BF0009         0131            LES     DI,[BX]0900
0000:015B     F0               0132            LOCK
0000:015C     AC               0133            LODS    BYTE
0000:015D     AD               0134            LODS    WORD
```

```
LOC           OBJ             LINE LABEL      SOURCE CODE
0000:015E     E2AD            0135            LOOP   010D
0000:0160     E1FF            0136            LOOPZ  0161
0000:0162     E1C9            0137            LOOPZ  012D
0000:0164     E1C7            0138            LOOPZ  012D
0000:0166     D104            0139            LOOPZ  016C
0000:0168     E083            0140            LOOPNZ 00ED
0000:016A     E081            0141            LOOPNZ 00ED
0000:016C     A10030          0142 B016C      MOV    AX,3000
0000:016F     8A063000        0143            MOV    AL,0030
0000:0173     8EC2            0144            MOV    ES,DX
0000:0175     8ED8            0145            MOV    DS,AX
0000:0177     8ED3            0146            MOV    SS,BX
0000:0179     368E060030      0147            MOV    ES,SS:3000
0000:017E     8910            0148            MOV    [BX][SI],DX
0000:0180     88F1            0149            MOV    CL,DH
0000:0182     A17700          0150            MOV    AX,0077
0000:0185     8B3E1806        0151            MOV    DI,0618
0000:0189     8B1E8400        0152            MOV    BX,0084
0000:018D     C70600200030    0153            MOV    WORD 2000,#3000
0000:0193     C606001280      0154            MOV    BYTE 1200,#80
0000:0198     A4              0155            MOVS   BYTE
0000:0199     A5              0156            MOVS   WORD
0000:019A     F6260006        0157            MUL    BYTE 0600
0000:019E     F61E0060        0158            NEG    BYTE 6000
0000:01A2     90              0159            NOP
0000:01A3     F716            0160            NOT    WORD 060B
0000:01A7     0B066842        0161            OR     AX,4268
0000:01AB     09D6            0162            OR     SI,DX
0000:01AD     0A0E3700        0163 B01AD      OR     CL,0037
0000:01B1     E744            0164            OUT    44,AX
0000:01B3     E606            0165            OUT    06,AL
0000:01B5     9D              0166            POPF
0000:01B6     50              0167            PUSH   AX
0000:01B7     FFF7            0168            PUSH   DI
0000:01B9     9C              0169            PUSHF
0000:01BA     D0D4            0170            RCL    AH
0000:01BC     D1D2            0171 B01BC      RCL    DX
0000:01BE     D1D9            0172            RCR    CX
0000:01C0     F3              0173            REP
0000:01C1     F3              0174            REP
0000:01C2     F2              0175            REPNE
0000:01C3     C3              0176            RET
0000:01C4     C20200          0177            RET    0002
0000:01C7     CB              0178            RET    I
0000:01C8     CA0400          0179            RET    I,0004
0000:01CB     D0C4            0180            ROL    AH
```

```
LOC           OBJ          LINE LABEL      SOURCE CODE
0000:01CD     D1C9         0181            ROR    CX
0000:01CF     9E           0182            SAHF
0000:01D0     D1E2         0183            SAL    DX
0000:01D2     D1FE         0184            SAR    SI
0000:01D4     D1FF         0185            SAR    DI
0000:01D6     1A060400     0186            SBB    AL,0004
0000:01DA     1B1E0120     0187            SBB    BX,2001
0000:01DE     AE           0188            SCAS   BYTE
0000:01DF     AF           0189            SCAS   WORD
0000:01E0     D0E4         0190            SAL    AH
0000:01E2     D0E8         0191            SHR    AL
0000:01E4     F9           0192            STC
0000:01E5     FD           0193            STD
0000:01E6     FB           0194            STI
0000:01E7     AA           0195            STOS   BYTE
0000:01E8     AB           0196            STOS   WORD
0000:01E9     29DB         0197            SUB    BX,BX
0000:01EB     2B1E0060     0198            SUB    BX,6000
0000:01EF     85CB         0199            TEST   BX,CX
0000:01F1     9B           0200            WAIT
0000:01F2     87C3         0201            XCHG   AX,BX
0000:01F4     86C3         0202            XCHG   AL,BL
0000:01F6     D7           0203            XLAT
0000:01F7     30DC         0204            XOR    AH,BL
0000:01F9     31DE         0205            XOR    SI,BX
0000:01FB     320E3700     0206            XOR    CL,0037
```

H.7.2 8087

```
LOC           OBJ             LINE LABEL      SOURCE CODE
0000:0000     9BD9F0          0001            F2XM1
0000:0003     9BD9E1          0002            FABS
0000:0006     9BDCC2          0003            FADD
0000:0009     9BD8C1          0004            FADD   ST(2),ST
0000:000C     9BD800          0005            FADD   ST,ST(1)
0000:000F     9BDEC6          0006            FADDP  DWORD [BX][SI]
0000:0012     9BDF6020        0007            FBLD   ST(6),ST
0000:0016     9BDFB70026      0008            FBSTP  [BX][SI]20
0000:001B     9BD9E0          0009            FCHS   [BX]2600
0000:001E     9BDBE2          0010            FCLEX
0000:0021     DBE2            0011            FNCLEX
0000:0023     9BD8D5          0012            FCOM   ST(5)
0000:0026     9BD8D1          0013            FCOM   ST(1)
0000:0029     9BD8160020      0014            FCOM   DWORD 2000
0000:002E     9BD8DA          0015            FCOMP  ST(2)
0000:0031     9BD8D9          0016            FCOMP  ST(1)
0000:0034     9BDC1E0003      0017            FCOMP  QWORD 0300
0000:0039     9BDED9          0018            FCOMPP
0000:003C     9BD9F6          0019            FDECSTP
0000:003F     9BDBE1          0020            FDISI
0000:0042     9BD8F7          0021            FDIV   ST,ST(7)
0000:0045     9BD8F1          0022            FDIV   ST,ST(1)
0000:0048     9BDEF1          0023            FDIVP  ST(1),ST
0000:004B     9BD8F9          0024            FDIVR  ST,ST(1)
0000:004E     9BDEF9          0025            FDIVRP ST(1),ST
0000:0051     9BDBE0          0026            FENI
0000:0054     DBE0            0027            FNENI
0000:0056     9BDDC1          0028            FFREE  ST(1)
0000:0059     9BDDC5          0029            FFREE  ST(5)
0000:005C     9BDE068030      0030            FIADD  WORD 3080
0000:0061     9BDA5090        0031            FICOM  DWORD [BX][SI]90
0000:0065     9BDE1E0040      0032            FICOMP WORD 4000
0000:006A     9BDE365634      0033            FIDIV  WORD 3456
0000:006F     9BDE3E0045      0034            FIDIVR WORD 4500
0000:0074     9BDF01          0035            FILD   WORD [BX][DI]
0000:0077     9BDE0E0003      0036            FIMUL  WORD 0300
0000:007C     9BD9F7          0037            FINCSTP
0000:007F     9BDBE3          0038            FINIT
0000:0082     DBE3            0039            FNINIT
0000:0084     9BDF160004      0040            FIST   WORD 0400
0000:0089     9BDF3E0005      0041            FISTP  QWORD 0500
0000:008E     9BDE268040      0042            FISUB  WORD 4080
0000:0093     9BDA6930        0043            FISUBR DWORD [BX][DI]30
0000:0097     9BD9C1          0044            FLD    ST(1)
```

```
LOC          OBJ           LINE LABEL      SOURCE CODE
0000:009A    9BD9E8        0045            FLD1
0000:009D    9BD929        0046            FLDCW  [BX][DI]
0000:00A0    9BD920        0047            FLDENV [BX][SI]
0000:00A3    9BD9EA        0048            FLDL2E
0000:00A6    9BD9E9        0049            FLDL2T
0000:00A9    9BD9EC        0050            FLDLG2
0000:00AC    9BD9ED        0051            FLDLN2
0000:00AF    9BD9EB        0052            FLDPI
0000:00B2    9BD9EE        0053            FLDZ
0000:00B5    9BD8C9        0054            FMUL   ST,ST(1)
0000:00B8    9BDEC9        0055            FMULP  ST(1),ST
0000:00BB    D9D0          0056            FNOP
0000:00BD    9BDD31        0057            FSAVE  [BX][DI]
0000:00C0    DD30          0058            FNSAVE [BX][SI]
0000:00C2    9BD97980      0059            FSTCW  [BX][DI]80
0000:00C6    D938          0060            FNSTCW [BX][SI]
0000:00C8    9BD931        0061            FSTENV [BX][DI]
0000:00CB    D9B00090      0062            FNSTENV [BX][SI]9000
0000:00CF    9BDD39        0063            FSTSW  [BX][DI]
0000:00D2    DD3F          0064            FNSTSW [BX]
0000:00D4    9BD9F3        0065            FPATAN
0000:00D7    9BD9F8        0066            FPREM
0000:00DA    9BD9F2        0067            FPTAN
0000:00DD    9BD9FC        0068            FRNDINT
0000:00E0    9BDD6694      0069            FRSTOR [BP]94
0000:00E4    9BD9FD        0070            FSCALE
0000:00E7    9BD9FA        0071            FSQRT
0000:00EA    9BDDD1        0072            FST    ST(1)
0000:00ED    9BD9D9        0073            FSTP   ST(1)
0000:00F0    9BD8E1        0074            FSUB   ST,ST(1)
0000:00F3    9BD8260040    0075            FSUB   DWORD 4000
0000:00F8    9BDEE1        0076            FSUBP  ST(1),ST
0000:00FB    9BD8E9        0077            FSUBR  ST,ST(1)
0000:00FE    9BDEE9        0078            FSUBRP ST(1),ST
0000:0101    9BD9E4        0079            FTST
0000:0104    9BD9E5        0080            FXAM
0000:0107    9BD9C9        0081            FXCH   ST(1)
0000:010A    9BD9F4        0082            FXTRACT
0000:010D    9BD9F1        0083            FYL2X
0000:0110    9BD9F9        0084            FYL2XP1
```

H.7.3 8089

```
LOC           OBJ          LINE LABEL    SOURCE CODE
0000:0000     01A1         0001          ADD    GA,[GB]
0000:0002     A1D2         0002          ADD    [GC],IX
0000:0004     00A2         0003          ADDB   GA,[GC]
0000:0006     40D1         0004          ADDB   [GB],GC
0000:0008     682030       0005          ADDBI  BC,#30
0000:000B     08207F       0006          ADDBI  GA,#7F
0000:000E     31200020     0007          ADDI   GB,#2000
0000:0012     19C28001     0008          ADDI   [GC],#80
0000:0015     01A9         0009          AND    GA,[GB]
0000:0017     A1DA         0010          AND    [GC],IX
0000:0019     20AA         0011          ANDB   GB,[GC]
0000:001B     60D8         0012          ANDB   [GA],BC
0000:001D     282812       0013          ANDBI  GB,#12
0000:0020     08CA86       0014 S0020    ANDBI  [GC],#86
0000:0023     11280060     0015          ANDI   GA,#6000
0000:0027     11C90030     0016          ANDI   [GB],#3000
0000:002B     919ED11F     0017          CALL   [GC],2000
0000:002F     899EEE       0018          CALL   [GC],0020
0000:0032     A0F8         0019          CLR    [GA],5
0000:0034     A2F917       0020          CLR    [GB].17,5
0000:0037     603C         0021          DEC    BC
0000:0039     07ED         0022          DEC    [GB+IX+]
0000:003B     04EC         0023 B003B    DECB   [CA+IX]
0000:003D     2048         0024          HLT
0000:003F     0038         0025          INC    GA
0000:0041     01E9         0026          INC    [GB]
0000:0043     00EA         0027          INCB   [GC]
0000:0045     882000       0028          JMP    0048
0000:0048     10B2B41F     0029 B0048    JMCE   [GC],2000
0000:004C     10B4B811     0030          JMCNE  [GA],1208
0000:0050     88205F       0031          JMP    0048
0000:0053     50B9A92F     0032          JNBT   [GB],2,3000
0000:0057     7040A50F     0033          JNZ    BC,1000
0000:005B     40A3         0034          ADDB   GC,[PP]
0000:005D     09E2FA       0035          JNZ    [GC],005A
0000:0060     08E11D       0036          JNZB   [GB],0080
0000:0063     7044995F     0037          JZ     BC,6000
0000:0067     11E69522     0038          JZ     [GC],2300
0000:006B     08E4D2       0039          JZB    [GA],0040
0000:006E     919CBE1F     0040          CALL   [GA],2030
0000:0072     90BD8A3F     0041          JBT    [GB],4,4000
0000:0076     10B08611     0042          JMCE   [GA],1200
0000:007A     10B68267     0043          JMCNE  [GC],6800
0000:007E     91207E3F     0044          JMP    4000
```

```
LOC           OBJ            LINE LABEL      SOURCE CODE
0000:0082     F0B97A6F       0045            JNST   [GB],7,7000
0000:0086     10407601       0046            JNZ    GA,0200
0000:008A     11E2922F       0047            JNZ    [GC],3020
0000:008E     10E26E32       0048            JNZB   [GC],3300
0000:0092     C8446B         0049            JZ     CC,0100
0000:0095     11E6673F       0050            JZ     [GC],4000
0000:0099     038B12         0051            LPD    GA,[PP].12
0000:009C     110800010002   0052            LPDI   GA,0200:0100
0000:00A2     6186           0053            MOV    [GC],BC
0000:00A4     2180           0054            MOV    GB,[GA]
0000:00A6     019001CE       0055            MOV    [GC],[GA]
0000:00AA     0082           0056            MOVB   GA,[GC]
0000:00AC     E085           0057            MOVB   [GB],MC
0000:00AE     009100CC       0058            MOVB   [GA],[GB]
0000:00B2     C83030         0059            MOVBI  CC,#30
0000:00B5     084D20         0060            MOVBI  [GB],#20
0000:00B8     71309028       0061            MOVI   BC,#2890
0000:00BC     114D3412       0062            MOVI   [GB],#1234
0000:00C0     839812         0063            MOVP   [GA].12,TP
0000:00C3     0199           0064            MOVP   [GB],GA
0000:00C5     018E           0065            MOVP   GA,[GC]
0000:00C7     0000           0066            NOP
0000:00C9     002C           0067            NOT    GA
0000:00CB     602C           0068            NOT    BC
0000:00CD     01DD           0069            NOT    [GB]
0000:00CF     41AC           0070            NOT    GC,[GA]
0000:00D1     02DE60         0071            NOTB   [GC].60
0000:00D4     40AD           0072            NOTB   GC,[GB]
0000:00D6     61A4           0073            OR     BC,[GA]
0000:00D8     E1D5           0074            OR     [GB],MC
0000:00DA     80A5           0075            ORB    TP,[GB]
0000:00DC     00D5           0076            ORB    [GB],GA
0000:00DE     0BC47628       0077            ORI    [GA].76,#28
0000:00E2     0BC50010       0078            ORI    [GB].00,#10
0000:00E6     4000           0079            SINTR
0000:00E8     18952215       0080            TSL    [GB],#22,0101
0000:00EC     0FE002         0081            JNZ    [GA+IX+],00F1
0000:00EF     8000           0082            WID    8,8
0000:00F1     6000           0083 B00F1      XFER
```

H.7.4 80186/80188

The following summary is a listing of additional 80186/80188 assembly mnemonics not included for the 8086/8088.

```
LOC          OBJ          LINE LABEL     SOURCE CODE
0000:0000    680003       0001           PUSH   #0300
0000:0003    60           0002           PUSHA
0000:0004    61           0003           POPA
0000:0005    69C020D1     0004           IMUL   AX,AX,#D120
0000:0009    6B5250C2     0005           IMUL   DX,[BP][SI]50,#-3E
0000:000D    C1C020       0006           ROL    AX,20
0000:0010    C1401224     0007           ROL    WORD [BX][SI]12,24
0000:0014    C1CB58       0008           ROR    BX,58
0000:0017    C1483040     0009           ROR    WORD [BX][SI]30,40
0000:001B    C1D148       0010           RCL    CX,48
0000:001E    C1513456     0011           RCL    WORD [BX][DI]34,56
0000:0022    C1DA86       0012           RCR    DX,86
0000:0025    C1580282     0013           RCR    WORD [BX][SI]02,82
0000:0029    C1E540       0014           SAL    BP,40
0000:002C    C1624050     0015           SAL    WORD [BP][SI]40,50
0000:0030    C1E47E       0016           SAL    SP,7E
0000:0033    C1613224     0017           SAL    WORD [BX][DI]32,24
0000:0037    C1E9AE       0018           SHR    CX,AE
0000:003A    C16A0660     0019           SHR    WORD [BP][SI]06,60
0000:003E    C1FB70       0020           SAR    BX,70
0000:0041    C1781234     0021           SAR    WORD [BX][SI]12,34
0000:0045    6C           0022           INS    BYTE
0000:0046    6D           0023           INS    WORD
0000:0047    6E           0024           OUTS   BYTE
0000:0048    6F           0025           OUTS   WORD
0000:0049    F2           0026           REPNE
0000:004A    6C           0027           INS    BYTE
0000:004B    F2           0028           REPNE
0000:004C    6D           0029           INS    WORD
0000:004D    F2           0030           REPNE
0000:004E    6E           0031           OUTS   BYTE
0000:004F    F2           0032           REPNE
0000:0050    6F           0033           OUTS   WORD
0000:0051    C83FDE00     0034           ENTER  DE3F,00
0000:0055    C9           0035           LEAVE
0000:0056    6200         0036           BOUND  AX,[BX][SI]
```

H.7.5 80286

The following summary is a listing of additional 80286 assembly mnemonics not included for the 8086/8088 or 80186/80188.

```
LOC           OBJ           LINE  LABEL     SOURCE CODE
0100:0257     0F06          0244            CLTS
0100:0259     0F0110        0245            LGDT    [BX][SI]
0100:025C     0F0111        0246            LGDT    [BX][DI]
0100:025F     0F0102        0247            SGDT    [BP][SI]
0100:0262     0F0103        0248            SGDT    [BP][DI]
0100:0265     0F011C        0249            LIDT    [SI]
0100:0268     0F011D        0250            LIDT    [DI]
0100:026B     0F010E3412    0251            SIDT    1234
0100:0270     0F010F        0252            SIDT    [BX]
0100:0273     0F005012      0253            LLDT    [BX][SI]12
0100:0277     0F005134      0254            LLDT    [BX][DI]34
0100:027B     0F004256      0255            SLDT    [BP][SI]56
0100:027F     0F004378      0256            SLDT    [P][DI]78
0100:0283     0F005C9A      0257            LTR     [SI]9A
0100:0287     0F005DBC      0258            LTR     [DI]BC
0100:028B     0F004EDE      0259            STR     [BP]DE
0100:028F     0F004FF0      0260            STR     [BX]F0
0100:0293     0F01B03412    0261            LMSW    [BX][SI]1234
0100:0298     0F01B17856    0262            LMSW    [BX][DI]5678
0100:029D     0F01A2BC9A    0263            SMSW    [BP][SI]9ABC
0100:02A2     0F01A32143    0264            SMSW    [BP][DI]4321
0100:02A7     0F02C1        0265            LAR     AX,CX
0100:02AA     0F021E3412    0266            LAR     BX,1234
0100:02AF     0F030C        0267            LSL     CX[SI]
0100:02B2     6353AB        0268            ARPL    [BP][DI]AB,DX
0100:02B5     0F00A42211    0269            VERR    [SI]1122
0100:02BA     0F00A54433    0270            VERR    [DI]3344
0100:02BF     0F00AE6655    0271            VERW    [BP]5566
0100:02C4     0F00AF8877    0272            VERW    [BX]7788
```

H.7.6 80287

The following summary is a listing of additional 80287 assembly mnemonics not included for the 8087.

```
LOC           OBJ           LINE  LABEL     SOURCE CODE
0100:0000     DBE4          0001            FSETPM
0100:0002     DFE0          0002            FSTSW AX
```

Appendix I
Breakpoint Processor Board (BPP)

I.1 Introduction

MICROTEK's BreakPoint Processor (BPP) is a comprehensive breakpoint control unit for all MICE-II emulators. The BPP is an optional single card PCB that features sophisticated breakpoint logic. It includes up to 120 new breakpoint constructs for more flexibility in target system debug and development. The BPP also includes external hardware triggering and an execution interval timer. This powerful triggering system helps you to quickly solve the most difficult software and hardware problems.

Breakpoint conditions designating precise events can be logically joined with ARM/AND/OR connectives, permitting you to -

* begin or end tracing
* activate another event group
* specify delayed trigger count
* initiate event counter
* use external signal as trace trigger
* send trigger signal to external device
* record execution time between 2 events
* break emulation

Flexible trigger constructs are used to define single events, multiple activities or hardware sync signals. These powerful triggers make it easy to specify complex processor and hardware events. Data breakpoints can be set for address, data, status, count and delay. While external hardware breakpoint can specify up to 2 triggers (each with any combination of 2 signals).

The breakpoint system combines a large number of trigger events, permitting you to begin trace or break emulation on the following activities -

* Address
* Data bus
* Delay Count
* Event Counter (trigger match)
* Microprocessor Status Lines (Fetch, Read, Write, Input, Output, Interrupt Acknowledge)
* Logically Joined Breakpoints
* Sequence Levels
* Logic State Input

I.2 Breakpoint Characteristics

The BPP includes 4 breakpoints (H3-H6) and 2 triggers for transmitting a sync signal to an external device (BS1,BS2). MICE-II's other two breakpoints (H1,H2) can also be combined. H3-4 are data breakpoints (sync triggers), H5-6 are the external hardware breakpoints, and BS1-2 are sync signal triggers.

I.2.1 Breakpoint Syntax

```
H3 [addx [data [<dbwc> ][status [count]]]]
H4 [addx [data [<dbwc> ][status [delay]]]]
H5 [xx] - 2 spare bits (X1,X0)
H6 [xx] - 2 spare bits (X3,X2)
BS1 [addx [data [<dbwc> ][status]]]
BS2 [addx [data [<dbwc> ][status]]]
```

addx - up to 5 digit hex address with wildcard byte pair of "XX", or with wildcard nibble of "X" for the fifth digit only (i.e. X2345).
data - up to 4 digit hex data.
dbwc - up to 4 digit hex setting with binary "0" as wildcard for corresponding data bit.
status - type of processor activity with wildcard "X".
count - trigger count; range 1-4000H.
delay - cycle count delay; range 2-FFFFH.
spare bits - enabled at HIGH, LOW or wildcard (1/0/X).

I.2.2 Data Breakpoints

These breakpoints are latched on the rising edge of a $\overline{\text{STROBE}}$ signal from the CEP board. When the breakpoint is matched, emulation stops and a SYNC signal is transmitted. If a trigger count or cycle delay is defined in H3 or H4, then the input value must be matched before the breakpoint is activated. This method is most useful for tracing data after all other trigger conditions have been matched.

I.2.3 External Hardware Breakpoints

H5 and H6 use two external signal groups on the CEP board (X1,X0 and X3,X2 respectively). HIGH, LOW or "don't care" can be defined for these signals. Hardware breakpoints are latched when the specified logic state occurs.

I.2.4 Halt Breakpoints

H1-6 can serve as individual Halt breakpoints for B, F or G commands. H1 and H2 (defined prior to BT) can also be used to add another logical OR to the trigger construct.

I.3 Breakpoint Trace Logic

Breakpoints H3-H6 are used in logical combination with the "BT" command. Tracing begins at the current address as soon as the command is input (Backward Tracing) and continues until the specified breakpoints are matched. Each logical connective is used as follows:

1) ARM Trigger Trace

```
>BT A[ B[ C[ D]]]*
```

Breakpoints can be logically joined in any armed sequence (except that they can't be repeated). There are 64 different ARM trigger combinations.

*A-D can represent any of H3~6.

2) AND Trigger Trace

```
>BT (A B C) [D]
>BT (A B) [C [D]]
```

Breakpoints A-D can be combined as indicated above in any combination. Sequence breakpoints may also be specified after the AND trigger construct. There are 38 different AND trigger combinations.

3) OR Trigger Trace

```
>BT <A B C>
>BT <A B> [C [D]]
```

The OR trigger may use any breakpoint combination. Sequence breakpoints may also be specified after the OR trigger construct. When a Sequence trigger is used with the OR trigger construct, the ORed pairs must be either data (H3,H4) or external breakpoints (H5,H6). There are 18 different OR trigger combinations.

I.4 Breakpoint Interval Timer

A timer is also supported for trigger trace mode which displays the interval from the initial trigger until emulation stops. (The interval will display only where a sequenced breakpoint is set.) Time is displayed down to microseconds for up to 35.39 minutes before the counter is reset.

I.5 External Hardware Trigger Output

The BS command can be used to trigger a SYNC signal to an external device e.g. oscilloscope or logic analyzer. When the specified parameters are matched, a SYNC signal is transmitted and emulation continues.

Appendix J
Error Messages

1. ADDRESS xxxx PROTECTED, COMMAND ABORTED!

   Condition: Memory at trigger address for BP command cannot be modified to interrupt code, causing the command to be aborted.

   Recovery: Verify that the specified break address is RAM.

2. ADDRESS xxxx RAM ERROR!

   Condition: Memory addressed in Memory Test (T) command failed.

   Recovery: Verify that specified address is write enabled RAM; and then see if the address bus and/or data bus are open or shorted.

3. BACKWARD TRACE FAILS!

   Condition: Trace buffer is empty when the breakpoint is matched, or <ESC> is input, or the processor is stopped without stepping any machine cycles.

   Recovery: See if the READY or HOLD signal is always inactive. Then see is the trigger address or a specified breakpoint (H1~H6) was matched immediately.

4. BC TABLE FULL

   Condition: Instruction branches and calls in disassembly command exceed 900.

6. BUS REQUESTING!

   Condition: Target bus requesting signal always active.

   Recovery: Check $\overline{RQ}/\overline{GT0}$ and $\overline{RQ}/\overline{GT1}$ lines [for 8086/88(MAX)]; and verify that the target has a bus request signal.

7. COMMAND TOO LONG!

   Condition: Instruction string in assembly command exceeds 50 characters.

8. **ERROR!**

   Condition: Command syntax error.

9. **ERROR CODE!**

   Condition: Instruction error in disassembly command.

10. **ERROR CODE, TRY AGAIN!**

    Condition: Instruction error in assembly command.

11. **FORWARD TRACE FAILS!**

    Condition: Trace buffer is empty when the breakpoint is matched, or <ESC> is input, or the processor is stopped without stepping any machine cycles.

    Recovery: See if the trigger address was invalid, or a specified breakpoint was matched before the trace started.

12. **Hx NOT SET!**

    Condition: Specified breakpoint not set prior to executing BT command.

13. **IDT.LIMIT CAN'T BE LESS THAN OFH!!**

    Condition: An attempt was made to set the IDT.LIMIT to less that 10H.

    Recovery: Modify the value of this register to 10H or larger.

14. **MEMORY SEARCH FAILURE!**

    Condition: Data string not found within specified address range for Memory Search (M) command.

15. **MEMORY WRITE FAILURE!**

    Condition: Data cannot be written to addressed memory in assembly command.

    Recovery: Verify that specified address is valid and write enabled.

16. MEMORY VERIFICATION FAILURE!

Condition: Data cannot be written to address memory in Modify (M) command.

Recovery: Verify that specified address is valid and write enabled.

17. NO BPP CARD!

Condition: BPP commands cannot be executed without a BPP card.

18. PC=address, TRACE ABORTED!

Condition: Trace aborted because PC equals trigger address in the BP command, causing the breakpoint to be matched immediately with no data recorded.

19. PROGRAM HALT!

Condition: Emulation processor halted by instruction.

Recovery: Use an interrupt or reset to recover; then perform a trace to see where the HALT instruction occurred.

20. SP CAN'T BE LESS THAN 6H!!

Condition: An attempt was made to set the SP to less than 6H with the R command.

Recovery: MICE will not accept any input less than 6H. Reinput another value for this register of 6H or larger.

21. SP CAN'T BE LESS THAN 6H; SP HAS BEEN INCREASED TO 6H!!

Condition: An attempt was made to set the SP to less than 6H with the X command.

Recovery: MICE will automatically reset the SP to 6H. If this default setting is not acceptable, then modify the value of this register to 7H or larger.

22. **TARGET CAN'T STEP! ***

Condition: Emulation processor does not respond after applying power or inputting the software reset command "r" (see Section 2.9). There is a problem with target processor that prevents a program from executing; there are several possibilities which can generate this message:

1) There is no clock.
2) There is a problem with the address line.
3) Target does not support the "ready" signal.
4) There is a problem with the RTT, CEP or HUEM board.
5) Other component failure.

Recovery: Check to be sure the target processor clock signal is correct and verify that the clock source selection switch is properly adjusted (see CEP Placement Chart in Appendix H for switch location).

23. **TARGET IS NOT READY!**

Condition: Target memory or input/output ready signal is inactive.

Recovery: Use a valid physical location in the command specification.

24. **Ux — FAILURE! ***

Condition: MICE selftest indicates location number for failure device.

Recovery: Contact your nearest Microtek distributor for assistance.

25. **WARNING: ONLY 6 BYTES ARE VALID**

Condition: Define byte (DB) in assembly command exceeds 6 bytes.

26. **WHAT? ***

Condition: RS-232C communication with parity or framing error.

Recovery: Check interface setting for communications at U13.

* Potentially fatal error; if problem cannot be resolved, contact your nearest Microtek distributor for assistance.

Appendix K
LIMITED WARRANTY; Service

Microtek International Inc. ("Microtek") warrants MICE-II for 8086/88(MIN), 8086/88(MAX), 80186/188 and 80286 MICROPROCESSORS (the "Product") and the user's manual for the Product to be free from physical defects for a period of twelve (12) months from the date of the original retail purchase. This warranty applies only to the original retail purchaser who bought these Products from an authorized Microtek representative. This warranty is void if the Product is damaged by improper or abnormal use or by accident, if the Product is altered or modified in any way, or if any attempt is made to repair the Product without authorization from Microtek.

If you find a Product to be defective, contact your authorized Microtek representative or Microtek. When you receive authorization to do so, return the Product as directed; include proof of purchase and purchase date. Microtek will correct physical defects in Products under warranty at no charge to you by repairing or, at its option, replacing such Products. THIS IS THE SOLE AND EXCLUSIVE REMEDY AVAILABLE FOR BREACH OF WARRANTY OR UNDER ANY OTHER LEGAL THEORY WITH RESPECT TO MICROTEK PRODUCTS. Product repairs not covered by warranty, and Product updates, are provided at a set rate.

ALL OTHER WARRANTIES AND REPRESENTATIONS, ORAL OR WRITTEN, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, ARE EXCLUDED AND DO NOT APPLY. Except as set forth in this limited warranty, the purchaser assumes the risk as to these Products' quality, accuracy, design, and performance.

It is solely the purchaser's responsibility to determine the suitability of these Products for each particular application. Microtek Products are in all events not suitable, and are not authorized, for use in connection with life support devices or systems, or in other devices or systems potentially injurious to life or health.

IN NO EVENT WILL MICROTEK BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY BREACH OF WARRANTY OR UNDER ANY OTHER LEGAL THEORY, even if advised of the possibility of such damages. Microtek is thus not liable for lost profits or goodwill; downtime; damage or destruction of any program, data, equipment, or other property; costs of recovering, reprogramming, or reproducing any program, data, or equipment; personal injury or loss; or any other damages.

Except as required by law, no representative, agent, or employee of Microtek is authorized to commit Microtek to warranties, representations, or obligations inconsistent with or in addition to those set forth in this limited warranty.

Please complete and return the registration section of the warranty card (provided in the packing box) to Microtek.

Appendix L
MICE International Distributors

(As of March 1, 1987)

ARGENTINA

MICROTEC INGENIERIA S.R.L.
VIAMONTE 1167, 8° P. OF. 31 Y 32
(1053) BUENOS AIRES
ARGENTINA
TEL: 54 (1) 46-8749/45-1934
TLX: 22496 TECNOS AR

BRAZIL

DIGIBYTE SISTEMAS DIGITAIS LTDA.
RUA PALACETE DAS AGUIAS, 494
CEP 04635
SAO PAULO, S.P., BRAZIL
TEL: 55 (011) 241-3611
TLX: 1153972 DGBT BR

AUSTRALIA

MACRO DYNAMICS PTY LTD.
80 LEWIS RD., WANTIRNA SOUTH
P.O. BOX 336, BAYSWATER 3153
VICTORIA, AUSTRALIA
TEL: 61 (03) 2207260
TLX: 30674 METO AA

CANADA

TRACAN ELECTRONICS CORP.
1200 AEROWOOD DR., #4
MISSISSAUGA, ONTARIO
L4W 2S7 CANADA
TEL: 1 (416) 625-7752
TLX: 961366

AUSTRIA

ALLMOS ELECTRONIC GMBH.
3RD FL., TROSTSTRASSE 50
1100 WIEN
VIENNA, AUSTRIA
TEL: 43 (222) 627 19 54
TLX: 112757

CHILE

EQUIPOS INDUSTRIALES S.A.C.I.
MONEDA 812, OF. 905
CASILLA 13550
SANTIAGO, CHILE
TEL: 56 (2) 382942
TLX: 241282 FLOBR CL, 340987 FLOBRA CK

BELGIUM

SIMAC ELECTRONICS B.V.
RUE DU PROGRES 52 BTE 3
1210 BRUSSELS, BELGIUM
TEL: 32 (02) 219 24 53
TLX: 23662

DENMARK

METRIC A/S
SKODSBORGVEJ 305
DK-2850 NAERUM
DENMARK
TEL: 45 (02) 80 42 00
TLX: 37163

FINLAND

NURMIKRO KY
KESKUSTIE 4
01900 NURMIJARVI
FINLAND
TEL: 358 (0) 204981
TLX: 15132 PERO SF

FRANCE

MICRO TECHNOLOGIE ELECTRONIQUE
68, RUE DE PARIS
93804 EPINAY-SUR-SEINE
FRANCE
TEL: 33 (1) 823.15.24
TLX: 613615 MTE F

HONG KONG

CHAI LUK COMPUTER LTD.
16TH FLOOR, UNIT A-B
VICWOOD BUILDING
128 GLOUCESTER ROAD
HONG KONG
TEL: 852 (5) 751151/752172
CBL: "HKCHAILUK"
TLX: 84960 CHAIL HX

IDEALAND ELECTRONICS, LTD.
9TH FL., BLOCK D
HOP HING INDUSTRIAL BLDG.
702 CASTLE PEAK ROAD
LAI CHI KOK, KOWLOON
HONG KONG
TEL: 852 (3) 7443516-9
TLX: 37155 IDEA HX
CABLE ADDRESS: IDINTLIM
FAX: 852 (3) 7441354

ISRAEL

ARITMOR
54, JABOTINSKY ST.
RAMAT-GAN 52 462
ISRAEL
TEL: 972 (03) 727317
TLX: 342369 CVSIL

ITALY

PRATICA S.R.L.
CORSO RE UMBERTO, 79
10128 TORINO
ITALY
TEL: 39 (011) 50.34.27
TLX: 212163

ITALY (EDUCATIONAL MARKET ONLY)

TEKNO SCUOLA SRL
VIA DELLA BONIFICA N. 104-106
65100 PESCARA
ITALY
TEL: 39 (085) 693510
TLX: 600878

NETHERLANDS

SIMAC ELECTRONICS B.V.
HIGH TECH PARK
5503 HP VELDHOVEN
THE NETHERLANDS
TEL: 31 (040) 533725
TLX: 51037

NORWAY

MORGENSTIERNE & CO. A/S
POSTBOKS 6688-RODELØKKA
0502 OSLO 5
NORWAY
TEL: 47 (472) 3561 10
TLX: 71719 MOROF

SINGAPORE

EPE COMPUTRONICS PTE LTD.
35, TANNERY ROAD, #10-04
TANNERY BLOCK
RUBY INDUSTRIAL COMPLEX
SINGAPORE 1334
TEL: 65 7468182
TLX: RS 37998 EPECOM

SPAIN

NOVATRONIC S.A.
POLIGONO INDUSTRIAL NEINVER
48016 DERIO-VIZCAYA
SPAIN
TEL: 34 (4) 4520811
TLX: 34163 NVAM E

SWEDEN

NORDISK ARRAYTEKNIK AB
BOX 1410
HUVUDSTAGATAN 1
S-17127 SOLNA
SWEDEN
TEL: 46 (08) 7349935
TLX: 14932 NORTEC S
FAX: 46 8272204

SWITZERLAND

TRACO ELECTRONIC AG
JENATSCHSTRASSE 1
8002 ZURICH
SWITZERLAND
TEL: 41 (01) 201 07 11
TLX: 815 570 TRCO CH
FAX: 41 (01) 201 11 68

UNITED KINGDOM

ARS MICROSYSTEMS LTD.
DOMAN ROAD
CAMBERLEY, SURREY, GU15 3DF
ENGLAND
TEL: 44 (0276) 685005
TLX: 858779 ARSM G
FAX: 44 (0276) 61524

U.S.A.

NEW MICRO INC.
16901 SOUTH WESTERN AVE.
GARDENA, CA 90247
U.S.A.
TEL: 1 (213) 321-2121
TLX: 797880 MICRO
FAX: 1 (213) 538-1193

WEST GERMANY

ALLMOS ELECTRONIC
FRAUNHOFERSTRASSE 11A
8033 MARTINSRIED
WEST GERMANY
TEL: 49 (089) 8572086-89
TLX: 5215111 ALEC D
FAX: 49 (089) 857 37 02

Manual Revision Notice
MICE-II User's Guide
For 16-Bit Intel Series Microprocessors
(3rd Edition, Oct. 1986)
Manual Revision: B (2/12/87)

Document No. I49-000033

MICE-II 80286 New ICE Cable Installation Instructions

The MICE-II 80286 ICE cable installation instructions are changed due to Engineering Change Order (ECO No. C138). This manual revision notice will be included in the next printing of MICE-II User's Guide.

The ICE (In-Circuit-Emulator) cable assembly consists of four flat-wire connectors at one end and an IC header at the other end. To install the ICE cable after setup has been completed, remove the cover of MICE-II, also remove the retainer (if exists). There are two latest versions for MICE-II 80286 ICE cable assembly: 80286A (Rev: C), the current one; and 80286A (Rev: B), the former one.

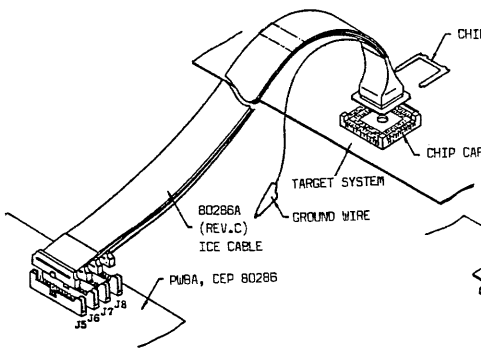
For 80286A (Rev: C) -

To install the 80286A, thread the four flat-wire connectors through the rectangular opening, attach the four connectors at positions J5-J8, slide and shut cover of the MICE-II case (see fig. 1, 3).

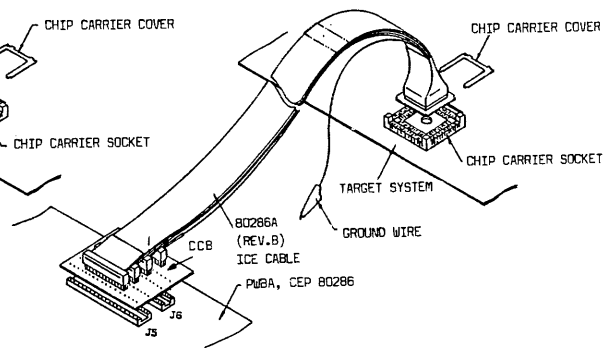
For 80286A (Rev: B) -

To install the 80286A, first separate the four flat-wire cable with CCB board, connect the CCB board to J5 and J6 on the CEP board, thread the four flat-wire connectors through the rectangular opening, attach them at related positions on CCB, then slide and shut cover of the MICE-II case (see fig. 2, 4).

The alligator clip on the ICE cable should be attached to a ground point on the target board, as close to the IC header as possible. Note that the ICE cable for MICE-II 80286 cannot be interchanged with other MICE-II models.

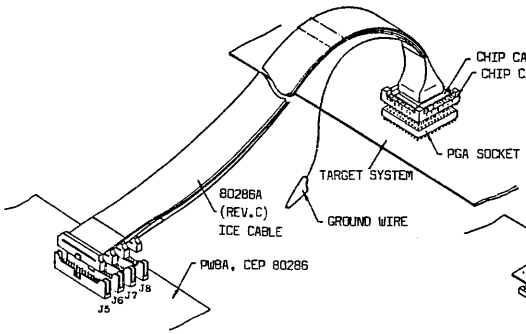


(Fig 1)

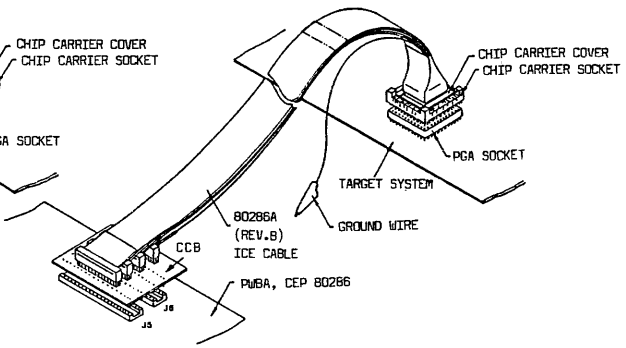


(Fig 2)

There is no clock adaptor mounted on the 80286 ICE Cable since the 80286 CPU will not accept input from a crystal oscillator for the clock source. The two different connector options are supported for target interface, depending on the user's requirements. Cable assemblies with connectors for the standard chip carrier type socket and for the pin grid array type socket are indicated in figures 1, 2 and 3, 4 respectively.



(Fig 3)



(Fig 4)

For CEP80286 revisions A-C, a CCB-286 is required to be connected between four flat-wire connectors and two 38-pin connectors. For revision D and later, the CCB-286 is not required because layout of the CEP board has been changed. Some glue is applied to the new ICE Cable, improving the cable strength.

Notes: 1. Don't remove the insulated tape on the chip carrier cover because the tape is used to prevent the signals circuit short.

2. A chip carrier socket is added on one end of the ICE cable. For targets which already have chip carrier socket, user must remove the chip carrier socket from ICE cable before connecting ICE cable to the target. Refer to fig. 1, 2.

Microtek International, Inc.
No. 6, Industry East Road 3
Science-based Industrial Park
Hsinchu, Taiwan 30077, R.O.C.
Tel: (035) 772155
Tlx: 32169 MICROTEK
Fax: 35-772598