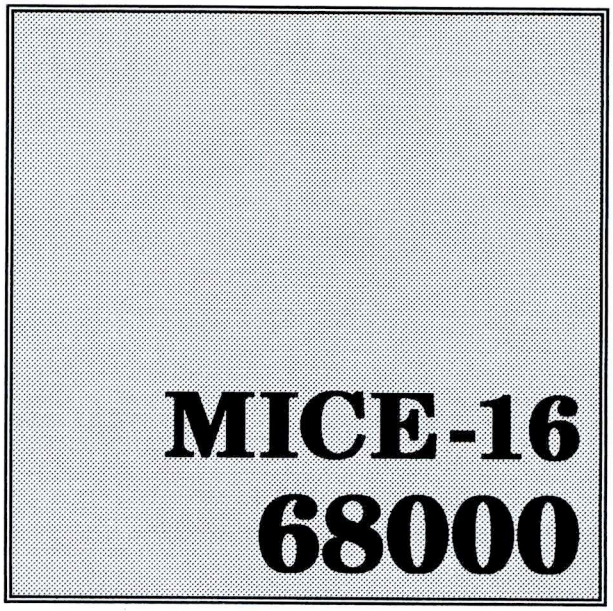


USER'S MANUAL

Doc No I49-000321
Second Edition
December 1988

READ ME FIRST!!

This manual contains information applicable to MICE equipped with LAM. For information unique to MICE with LAM-II, please refer to the *"User's Manual Addendum"* located at the bottom of this manual.

MICROTEK INTERNATIONAL INC.

6, Industry East Road 3
Science-based Industry Park
Hsinchu 300, Taiwan, ROC
Tel: (35) 772155
Fax: (35) 772598
Tlx: 32169 Microtek

Development Systems Division
3300 N.W. 211th Terrace
Hillsboro, Oregon 97124-7136
U.S.A.
Tel: (503) 645-7333
Fax: (503) 629-8460

Trademarks Acknowledgement

IBM,PC,XT, AT and **PS/2** are trademarks of International Business Machines.
MS-DOS and **MS-WINDOWS** are trademarks of Microsoft Corporation
Sun Microsystems is a trademark of Sun Microsystems, Inc.
UNIX is a trademark of AT&T Bell Laboratories.
MRI is a trademark of Microtec Research Inc.
NEC is a trademark of NEC Corporation.

Printed in Taiwan, ROC 12/88

CONTENTS

APPENDICES

DIRECTORY OF FIGURES

FIGURES

DIRECTORY OF TABLES

TABLES

PREFACE

This manual provides all the information necessary to install and operate your MICE-16 68000. Each command syntax is explained in detail in Chapter 5 with execution examples provided.

To fully understand and properly utilize the MICE command syntax and syntax notations used in this manual, user must be familiar with the Convention of Notations described in Chapter 4.

Interface requirement between MICE and terminal or PC/XT/AT is provided in Chapter 2 while MICE system setup and initialization are given in Chapter 3. However, user is expected to provide his own source of the installation and operating instructions for terminal, host computer and/or non-Microtek software packages linked with MICE.

Interface specification with other industry-standard host systems, i.e., VAX, MicroVAX, Apollo, Sun Microsystems, etc., and user's guide for the MICE standard software tool, are provided in the Universal Symbolic Debugger (USD) User's Manual. This manual is furnished with the MICE package.

For a quick browse of MICE-16 68000 features and specification, a "MICE-16 68000 AT A Glance" is provided in the next pages.

To facilitate finding of the figures and tables, they are sequenced under a single series of serial number.

Microtek provides full range of customer support worldwide to keep its products well maintained and continue to meet customer's ever growing expectations. Microtek has representatives worldwide to oversee customer satisfaction of its products. Inquiries for hardware repair/update, software support, training and/or consulting services may be forwarded to the nearest Microtek representative as listed in the last pages of this manual, or directly to Microtek.

Microtek also value and welcome all information and user's feedback on whatever comments they have on this manual and the product described herein. Please write to :

MICROTEK INTERNATIONAL INC.
6 Industry East Road 3
Science-based Industrial Park
Hsinchu, Taiwan 30077, ROC.
Tel: 35 772155
Tlx: 32169 MICROTEK
Fax: 886 35 772598

MICE-16 68000 AT A GLANCE

MICROTEK INTERNATIONAL INC.

ITEM	SPECIFICATION
Emulation CPU	MC68000, MC68010
Clock Source	Internal and external clocks are automatically sourced by auto-detection of target power: Without target: Internal (12MHz) is switched on. With powered target: External (up to 16.67 MHz) is swithed on.
Emulation Memory	EMM: 256KB static RAM at four 64K segment settings with 4KB mapping resolution Break by guard and write protect. HEMM*:1MB static RAM at four 256K segment settings with 4KB mapping resolution Break by guard and write protect.
File Transfer	High-speed up/downloading of object code in Motorola S-format.
Breakpoints (Trigger events)	Total of 6 real-time breakpoints: -3 execution breakpoints -2 bus breakpoints (parameters: address, data, status and N-time count) Status:-Supervisor program access -Supervisor data read -Supervisor data write -User program access -User data read -User data write -Interrupt acknowledge -1 external hardware breakpoint (AND/OR/IF-THAN combined with bus breakpoint.)
Trace	Real-Time Trace: address, data, status, hardware timer & external trace bits. Trace Buffer: 2048 cycles deep and 80 bits wide. Trace with cycle qualify.

* The optional HEMM will be available with Firmware Version 3.0 to be released soon.

ITEM	SPECIFICATION
Register	Display/Modify
Single Step	One instruction step/Step with skipping call routines/step range/ step with count/Auto step.
Cycle Step	One machine cycle step/step with count.
Execution Time Measurement	Measured by self-contained hardware timer (1 micro-seconds~44 hours).
Trace Start	When Emulation Start or when trigger condition matches.
Trace Stop	By Emulation Stop/Breakpoints matched/Optional delay N cycles (64K max) after breakpoints.
Memory Commands	Modify/Fill/Dump/CHecksum/Move/COMpare/SEarch/ TEst/Line Assembler/Disassembler.
Communication Ports	Parallel/Serial Module (PSM) Provides: -1 RS232C serial DCE port -1 8-bit parallel port. Serial/Serial Module (SSM) Provides: -2 RS232C serial ports. RS232C ports support: -Auto baud rate detection -Baud rate: 110 to 19200 bps -Data bits: 7 & 8 bits/character -Stop bits: 1 bit/character -Parity : Even, odd and no parity
Hosts	IBM PC/XT/AT or compatibles, VAX, MicroVAX, Apollo, Sun Microsystems.
Drivers	Universal Symbolic Debugger (USD)* (for both parallel and serial ports) Non-Microtek drivers

* Parallel port supported only on MICE-16 68000 with Firmware Version 1.1 or later.

ITEM	SPECIFICATION
Power Requirements	100 to 120V or 200 to 240VAC 47 to 63Hz Consumption: 100W Maximum (approx) Fuse Rating: 1.5A/250V slow blow
Environment	Operating Temp: 0° to 50° C (32° to 122° F) Storage Temp: -10° to 65° C (14° to 149° F) Relative Humidity: 20 to 80%
Physical:	
Main Case	Length: 32.5cm (13.0 in.) Width : 26.0cm (10.4 in.) Height: 12.5cm (5.0 in.) Net Weight: 7.0kg (15.4 lbs.)
Emulation Pod	Length: 18.6cm (7.3 in.) Width : 12.0cm (4.7 in.) Height: 2.5cm (1.0 in.) Net Weight: 0.4kg (0.9 lbs.)

Chapter 1

GENERAL INTRODUCTION

The MICE-16 68000 is a member of the third generation MICE compact in-circuit emulators. It offers advanced emulation and debugging capabilities to support hardware/software design, development and maintenance of **Motorola 68000/68010** microprocessors based products. It is capable of linking with host through its parallel or serial (RS-232C) ports. It features a very high downloading speed and supports programs written in "C" and assembly language.

This chapter introduces in details the significant qualities of MICE-16 68000 in the following areas:

- Key features (Section 1.1)
- Physical description (Section 1.2)
- System operating configuration (Section 1.3)
- Software tools (Section 1.4)

1.1 KEY FEATURES OF MICE-16 68000

1) **Convertible Emulation Processor:**
   The MICE-16 68000 is shipped from the factory with (16-MHz) 68HC000P CPU emulation processor. The emulation processor may be replaced with another CPU of different type (HMOS) or with 68010 to accommodate other target system requirement.

2) **Target Proximate Emulation Processor**
   The MICE-16 68000 emulation CPU is installed on the In-Circuit Emulation (ICE) cable target header or right on top of the target. With the emulation processor piggy-backed on top of the target, propagation delays associated with long cables between emulation CPU and target are optimally minimized.

3) **Remotely Extended In-Circuit Emulation (ICE) Cable**
   The ICE cable, which is umbilically connected to the MICE through a flat cable and buffer box (EPOD-68000) assembly, extends approximately 1 meter (3.3 feet) from the main MICE case. The added length improves ICE cable's flexibility in accessing larger system targets.

4) **Clock Sourcing by Auto-Detection of Target Power:**
   MICE automatically detects target power and switches to external clock (up to 16.17 MHz) if it is connected to a powered target. If no target power is detected (or no target is connected), MICE automatically switches to internal 12 MHz clock.

5) **Retainable Memory and I/O Space:**
   Target processor can retain its entire memory and I/O space.

6) **Keyboard Execution of Enable/Disable Control Signals to the CPU**

The following hardware control signals may be enabled/disabled from the keyboard-

$\overline{BERR}$ (bus error input signal)
$\overline{BR}$ (bus request signal)
IPL0-2 (interrupt request signal)

7) **High-Speed Memory Manipulation:**

MICE commands such as Memory Test, Download, Upload, Block Fill, Memory Search, etc., are executed at high-speed mode

MICE also supports high speed binary download (MICROTEK) format under Universal Symbolic Debugger (**USD version 2.3 or later**). Approximate downloading performance (in minutes) are illustrated in the example below:

NOTE

To support binary download format, MICE communication baud rate configuration should be set at 8 data bits per character.

HEX FILE	HOST(MHz)	EP(MHz)	PORT	
			SSM/PSM Serial	PSM Parallel
64K	PC-AT (8-MHz)	68000/10 (12-MHz)	1:27 (9600bps)* 0:52 (19200bps)*	0:26

*Under the specified baud rate

Table 1-1 Download Execution Speed

8) **Real-Time Emulation for Multi-processor Target System:**

A multi-emulator synchronization command and associated signal ports support real-time emulation for multi-processor target systems.

9) **Powerful Execution Breakpoint Processing:**

The MICE-16 68000 provides three execution breakpoints which stop emulation only if program execution reaches the specified instruction.

10) **User-selectable "Insert Wait State" Feature:**

More flexible command to emulate the system with programmable 0 or 1 wait state for all machine cycle (see Section 5.3.18)-

WAit On - 1 wait state, with target $\overline{DTACK}$ in use.
WAit OFf - 0 wait state, with target $\overline{DTACK}$ in use.

11) **Real-Time Trace:**

The trace buffer is 2K bytes deep and 80 bits wide. Signals monitored by the trace feature are:

EP address bus (24 bits)
EP data bus (16 bits)
EP status signals (8 bits)
External hardware signals (up to 8 bits)
Timer signals (24 bits)

12) **Trace with Cycle Qualifiers:**

Qualifiers can be set so that only cycles with the specified address and/or processor status are recorded in the trace buffer. The address may include wildcard bits or a wildcard nibble; any combination of processor states may be specified.

13) **Multiple (Six) Real-Time Breakpoints:**

The MICE-16 68000 provides three execution breakpoints (with address qualify), two bus breakpoints (with options for address, data, status and count qualify), and one external hardware breakpoint. All executions and bus breakpoints are hardware breakpoints for programs located in either RAM or ROM.

14) **Flexible Trigger Constructs:**

Single events, multiple activities and external hardware signals can be defined alone or in combination as trace-stop conditions. A cycle delay feature is also included, allowing trace/emulation to continue 0-65535 cycles after all trigger conditions have been met. (Note that tracing begins as soon as the emulation processor begins running the user's program.)

15) **Uninterrupted Run Option:**

As a special feature for real-time on-line debugging, the Emulation Processor can continue running the user's program after the trace has stopped. The trace buffer can then be examined without stopping the processor.

16) **Individual or Cumulative Execution Timer:**

Machine cycle timing can be listed under either individual or cumulative basis when the trace buffer is displayed. The time base is user selectable.

17) **Event Match Pulse Output:**

Whenever a bus breakpoint is encountered, a sync signal is asserted at the appropriate sync output port for use with an oscilloscope, logic analyzer, etc.

18) **Flexible Memory Access Management:**

The whole 16M bytes memory of 68000/68010 are divided into blocks of 4K bytes each. Any 4K-byte block can be enabled, disabled, set as write-protected, non-existence or as following special memory access type assignment:

SP (supervisor program)
UP (user program)
SD (supervisor data)
UD (user data)

19) **Optional High Capacity High Performance Emulation Memory:**

Where a larger emulation memory is desired, enhanced high performance emulation memory module (HEMM*) is available as an option to replace the EMM. The HEMM* consists of 1M bytes of static RAM. Four independent 256K segment settings are provided, and any 4K-byte block may be enabled, disabled, set as write-protected or as non-existence, or as following special memory access type assignments:

SP (supervisor program)
UP (user program)
SD (supervisor data)
UD (user data)

HEMM therefore is capable of mapping up to 1M of emulation memory anywhere within 1M bytes. (See Section 5.3.8 for more details on memory access type assignment command).

20) **NOVRAM Saved Emulation Mapping and information:**

Non-volatile RAM (NOVRAM) on the CPM (Control Processor Module) are used to save user-defined emulation memory mapping and information. Save and Recall commands are provided for storage and retrieval respectively, of these information during emulation.

21) **Push Button Hardware Reset:**

A reset button is provided on the CPM board, and can be accessed at the front panel of the MICE-16 68000 when a hardware reset is required.

22) **Serial and Parallel Communication Interfaces:**

The MICE-16 68000 communicates with the outside world through the communication channel(s) on a "baby" board module attached to CPM. Two such interface boards are available:

a) **Parallel/Serial Module (PSM):**

This board is installed in each MICE-16 68000 from the factory to provide one RS-232C serial communication channel (designated as Channel A) and one 8-bit parallel communication channel (designated Channel B).

NOTE

The parallel Channel B of PSM communicates only with IBM PC/XT/AT, IBM PS/2 or NEC PC-9801 equipped with the MICROTEK MS-PCE, MS-MCE or MS-JPC interface card respectively (see Section 2.3.3).

b) **Serial/Serial Module (SSM):**

The SSM board is included in the accessories package of the MICE-16 68000 and may be installed (in place of PSM board) to provide two RS-232C serial communication channels, designated as Channel A and Channel B respectively.

* The optional HEMM will be available with Firmware Version 3.0 to be released soon.

Both serial Channel A's of PSM and SSM feature an automatic detection of baud rate, data bits, stop bits and parity setting on the controlling system or terminal. The following interface parameters are supported:

Baud rate: 110 to 19200 bps
Data bits: 7 and 8 bits/character
Stop bits: 1 bit/character
Parity : Even, odd and no parity

SSM Channel B interface parameters are user programmable (see Section 5.3.1 for further detail).

23) **Popular Host System Supported :**

MICE-16 68000 is supported by popular industry standard hosts, i.e. IBM PC/XT/AT, IBM PS/2 and compatible, *NEC* PC-9801, Apollo/AEGIS, VAX/VMS, MicroVAX/ULTRIX and Sun Microsystems/UNIX to perform symbolic debugging and other enhanced function with a driver such as MICROTEK,s Universal Symbolic Debugger (USD).

24) **User Friendly Command Syntax:**

Plain-English-Language command keywords and parameters that can be entered in their full word form or in easy-to-remember shorthand (abbreviated) form are used.

25) **Powerful Commands for Multiple Applications:**

a) Command 'Help' provides either a quick menu of all commands (keywords only or the whole command syntax), or command syntax of a particular group of commands, or just an individual command. In either case, pertinent command description and syntax definitions will also display on the screen.

b) Memory-related functions are offered with such commands as Compare, Search, Fill, Test, Copy, Checksum, Byte, Word and several others.

c) Resident line Assembler and Disassembler are available for 68000 /68010 CPU instructions.

d) Instruction Step and Cycle commands are provided for hardware debug.

e) Tracing function are offered with Event, List, Trace, Trigger and other commands.

1.2 PHYSICAL DESCRIPTION

1.2.1 MAJOR HARDWARE

The MICE-16 68000 comes from the factory with its main modules completely assembled in a sturdy metal case, fully tested and preset. The EPOD flat cable assembly has its one end factory installed to the case. The buffer box EPOD-68000 and ICE cable assembly EB-68000H/L are shipped uninstalled.

1.2.1.1 Main MICE Case Assembly

The main MICE consists of the following components:

a) chassis
b) motherboard
c) power supply
d) cooling fan
e) EPOD-68000 flat cable assembly (connects EPOD-68000 to MICE)
f) the following main module boards:

- Emulation Processor Module (EPM)
- Logic Analyzer Module (LAM)
- Emulation Memory Module (EMM)
- Control Processor Module (CPM)*

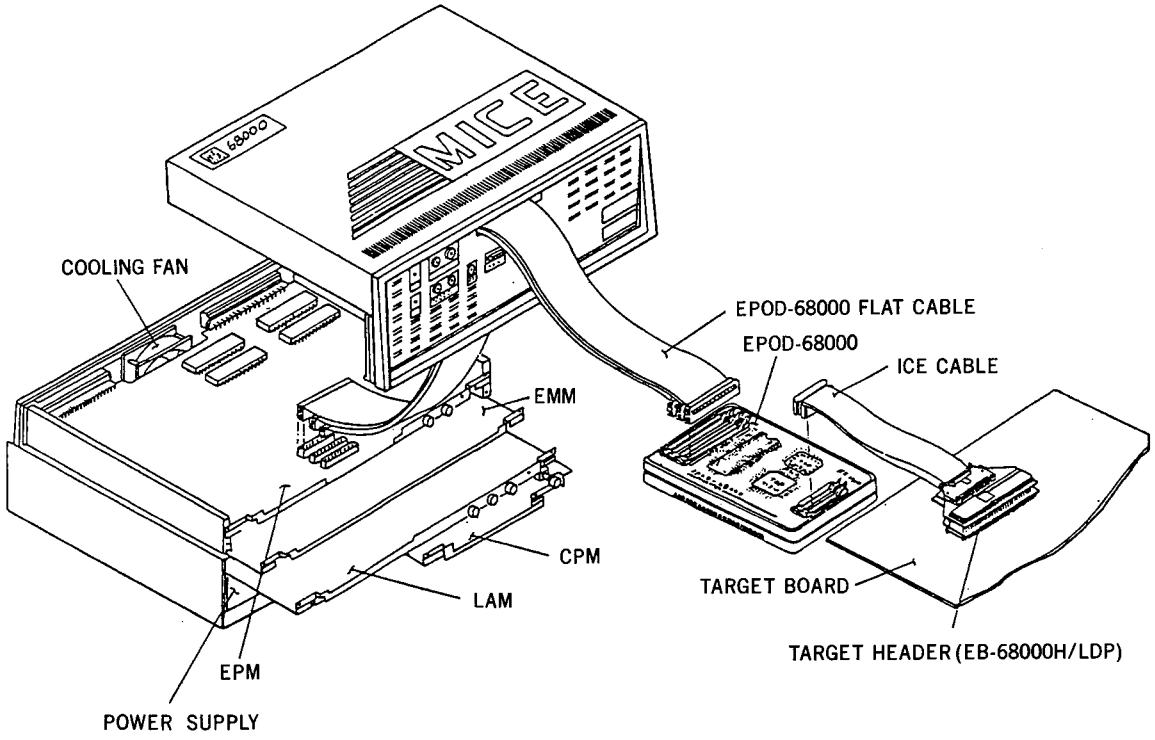


Figure 1-2 MICE-16 68000 Main Hardware Introduction

*Piggy-backed with Parallel/Serial Module (PSM) "baby" board.

1.2.1.2 EPOD-68000 and In-Circuit Emulator (ICE) Cable Assembly

The EPOD-68000 and ICE cable is packed as a separate assembly (Figure 1-3) in the MICE-16 68000 shipment package. The EPOD-68000 houses the control board (CB-68000) from which the ICE cable is connected through a twin 50-pin connectors. The target header of the ICE cable houses the emulation processor boards; EB-68KHDP and EB-68KLDP. A dual-in-line (DIP*) emulation CPU is installed on the EB-68KHDP board.

When connected to the MICE through the flat cable assembly and EPOD-68000, the ICE cable target header stretches approximately one meter (3.3 feet) from the main MICE case. The ICE Cable Assembly has the following electrical characteristics:

Capacitance : 15 pf/ft (i.e., between adjacent signals)
Propagation delay: 1.31 ns/ft (in each direction)
Impedance : 109 ohms/1kft

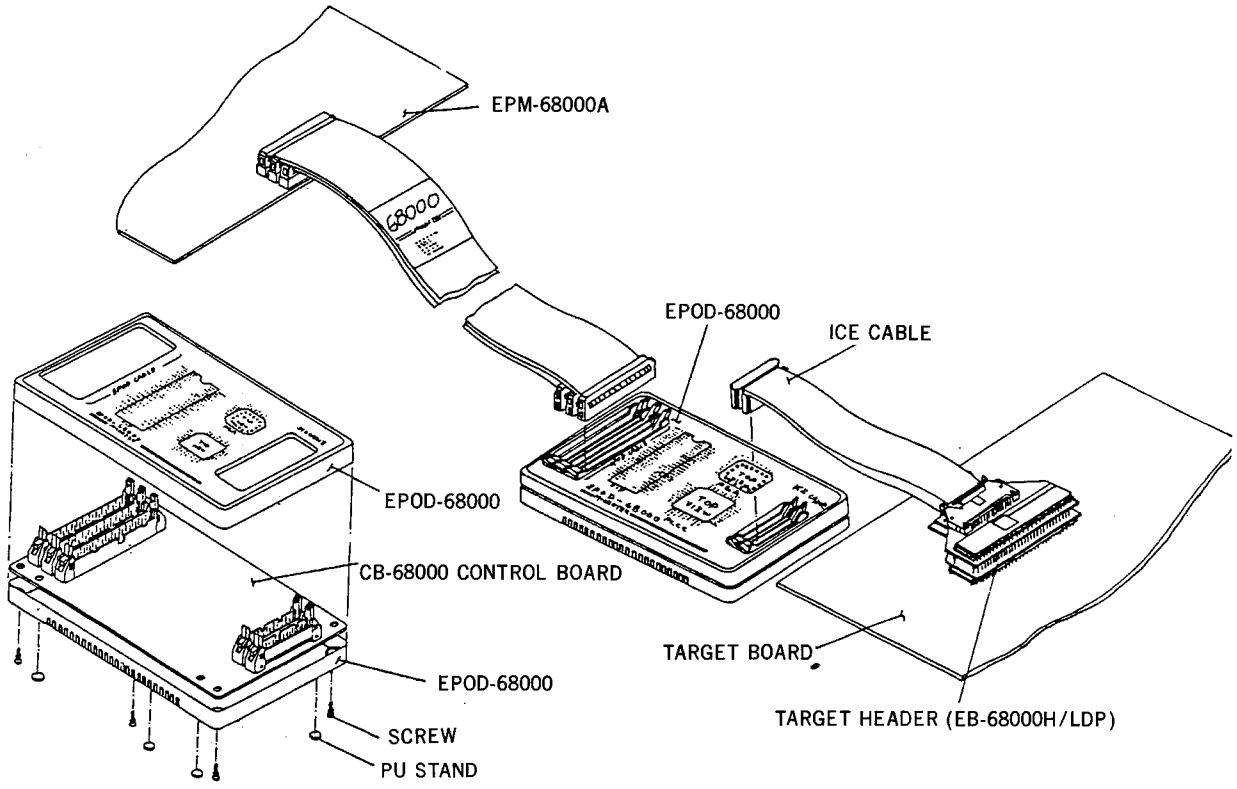


Figure 1-3 EPOD-68000 and ICE Cable Assembly

* Optional ICE Cable for pin grid array (PGA) and plastic leaded chip carrier (PLCC) will be available in the near future.

1.2.2 ACCESSORIES

The complete MICE-16 68000 package also includes the following accessories:

a) An AC power cable.
b) Three BNC coaxial cables.
c) A 9-pin D-type connector trace cable.
d) Serial/Serial Module (SSM)
e) IBM PC/XT/AT Parallel Interface Card (MS-PCE)
f) A MICE-16 68000 User's Manual.
g) Driver (Universal Symbolic Debugger (USD)) diskette with user's manual.

NOTE

For complete list of items which are provided with every shipment of a MICE-16 68000 system, refer to the Package Checklist packed and attached to MICE-16 68000 package.

1.3 SYSTEM OPERATING CONFIGURATION

The MICE-16 68000 may be operated in several configurations as described in the following sections. This manual however, only describes the details on how to operate the MICE-16 68000 under Terminal Mode, i.e., when MICE-16 68000 is connected to a data terminal or host computer through the RS-232C serial communication port (Channel A) on the PSM or SSM, and through the parallel port (Channel B) of the PSM.

For details of other operational modes, e.g., System/Debug Mode (MICE SSM Channel A connected to a terminal and Channel B to a host, and running under USD), refer to the Universal Symbolic Debugger (USD) User's Manual. A brief overview of the System/Debug Mode is given in Section 5.1.

NOTE

The RS-232C Channel A serial ports of both PSM and SSM are wired for DCE interface, while Channel B of SSM is wired for DTE interface. User must determine the correct interface requirement of their controlling device before hooking it up with MICE-16 68000.

1.3.1 MICE WITH TERMINAL CONFIGURATION

The MICE-16 68000 basically operates under Terminal Mode while interfaced with a terminal. In this simplest configuration, a data terminal with an RS-232C communication port is connected to Channel A of the MICE PSM or SSM communication ports. The Terminal is used as the display console and command entry device (see Figure 1-4).

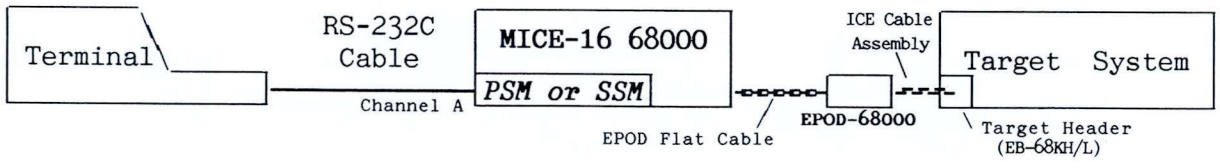


Figure 1-4 Configuring the MICE-16 68000 with a Terminal

1.3.2 MICE WITH HOST COMPUTER CONFIGURATION

1.3.2.1 Connected Through Serial Channel A Port

Serial port Channel A of the PSM or SSM communication ports can also be connected to a host computer (e.g. IBM PC/XT/AT, IBM PS/2 or compatible, Apollo, VAX, MicroVAX, Sun Microsystems, etc.) running under either terminal emulation software tool or a driver, such as the MICROTEK's Universal Symbolic Debugger (USD). Figure 1-5 below, illustrates the arrangement of such configuration.

Under this configuration, MICE-16 68000 operates in Terminal Mode with USD enhanced features, such as symbolic debugging, logic state analysis, software performance analysis, etc.

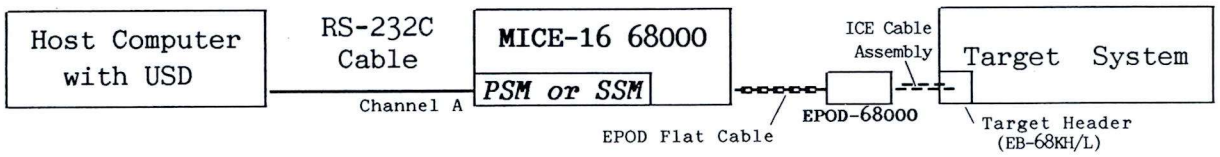


Figure 1-5 Configuring the MICE-16 68000 with a Host Computer Through Channel A Serial Port

1.3.2.2 Connected Through Parallel Channel B Port

A host computer **(restricted to IBM PC/XT/AT, IBM PS/2, *NEC* PC-9801 or compatible only)** may be also connected to parallel Channel B of the PSM through a host resident Interface Card provided with the MICE package. Under this configuration (Figure 1-6), MICE-16 68000 will also be able to operate in Terminal Mode as explained above. The USD version 2.3 or later should be used with this particular configuration.

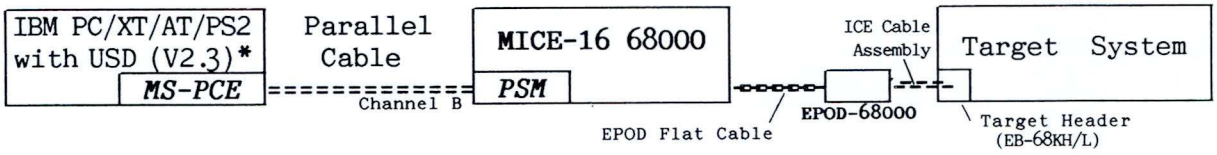


Figure 1-6 Configuring the MICE-16 68000 with IBM PC/XT/AT/PS2 Through Channel B Parallel Port and Interface Card

* USD Version 2.3 and MS-PCE Interface Card are applicable only to IBM PC/XT/AT and IBM PS/2 Model 30. Use USD Version 2.4 and MS-MCE or MS-JPC Interface Card if IBM PS/2 Models 50/60/70/80 or NEC PC-9801 is the host computer respectively.

1.3.3 MICE WITH MULTI-USER HOST COMPUTER CONFIGURATION

For more sophisticated use of the MICE-16 68000, it can also be connected with a terminal through the Channel A of SSM serial port, and a multi-user host computer (e.g. VAX, MicroVAX, etc.) through Channel B of the same serial port as illustrated in Figure 1-7 below. Under this configuration, Terminal, System and Debug Modes may be accessed or exited with no hardware adjustment involved.

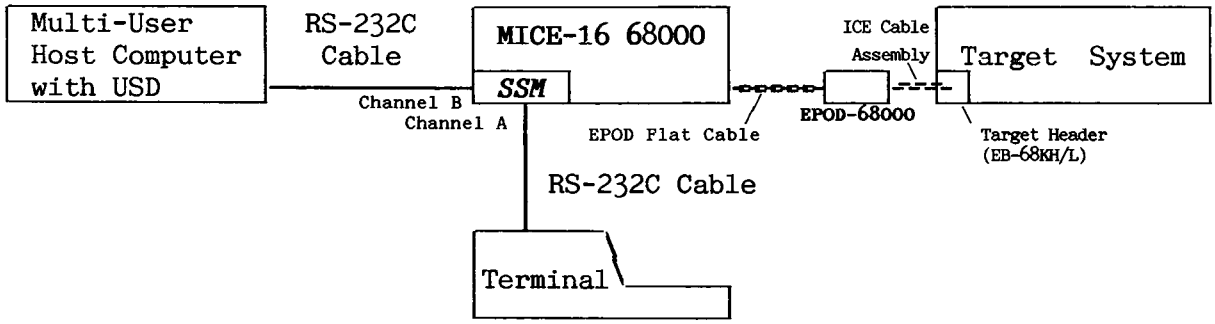


Figure 1-7 Configuring the MICE-16 68000 with Both Terminal and Multi-User Host Computer

1.3.4 MICE-16 68000 OPERATING CONFIGURATION SUMMARY

The following table summarizes the available operational configurations of MICE-16 68000:

I/O PORT	CHANNEL	OPERATING MODE	HOST CONFIGURATION	SOFTWARE TOOL	REFERENCE DOCUMENTATION
Parallel (PSM)	B	Terminal	1.IBM PC/XT/AT with MS-PCE. 2.IBM PS/2* with MS-MCE. 3.*NEC* PC-9801 with MS-JPC.	USD	USD and MICE-16 68000 User's Manuals
Serial (SSM or PSM)	A		1. same with above. 2. Sun and Apollo.		
			Data Terminal	N/A	MICE-16 68000 User's Manual
Serial (SSM)	A and B	Terminal /System /Debug	VAX, MicroVAX etc.	USD	USD and MICE-16 68000 User's Manuals

* Applies to micro-channel models 50, 60, 70 and 80 only. For Model 30, MS-PCE is used.

Table 1-8 MICE-16 68000 Operating Configuration Summary

1.4 MICE SOFTWARE TOOLS

When the MICE is connected to a host computer; tools such as cross-assemblers, compilers, linkers, high level language debugger (HLLD), symbolic translator drivers, and hardware/software analysis can be applied to generate/debug code for the target system. Complete sets of development software are available from MICROTEK, third party software houses and its distributors worldwide. Please contact your nearest MICROTEK representative for further information.

1.4.1 UNIVERSAL SYMBOLIC DEBUGGER (USD)

The USD (included in the MICE shipment package) is a fully integrated software package from MICROTEK designed for use with MICE. When USD is loaded to the host computer and linked with MICE, it becomes a self contained development system ready for assembly level symbolic debugging. USD runs under PC-DOS, VMS, AEGIS and ULTRIX-32 operating systems. It can be hosted by VAX mainframe, Sun Microsystems, Apollo and MicroVAX workstations in addition to *NEC* PC-9801, IBM PS/2. IBM PC/XT/AT and compatibles. It is capable of interfacing with both parallel (through *NEC* PC-9801, IBM PS/2, IBM PC/XT/AT and compatibles only) and serial ports of MICE (see Appendix E).

Major features of USD include File Management, Command File Facility, Symbolic Debug, Logic State Analysis, Software Performance Analysis, High Speed Binary Download etc. Refer to the USD User's Manual for further details.

1.4.2 HIGH LEVEL LANGUAGE DEBUGGER (HLLD)

High Level Language Debugger (HLLD) packages are also available from third-party software houses as option to users who need to debug target systems in high or assembly level. User can list, perform stepping or set breakpoints on source line. Display/set variables according to type, powerful macro and window display are also available from HLLD. For further details, refer to the pertinent vendor's manual.

Chapter 2

INSTALLATION AND USER SERVICEABLE PARTS

Detailed instructions for interfacing with a target system through the ICE cable and the MICE-16 68000, and user level maintenance are provided in this chapter, i.e.:

- Ice cable setup with target (Section 2.1)
- External signal cables connection (Section 2.2)
- Connecting with terminal or host system through SSM and PSM (Section 2.3)
- User serviceable parts (Section 2.4)

2.1 MICE-16 68000 ICE CABLE SETUP WITH TARGET

MICE-16 68000 is shipped from the factory with one end of the EPOD flat cable assembly attached to its Emulation Processor Module (EPM). The three 50-pin connectors at the outer end of the flat cable should be initially plugged into the emulation pod EPOD-68000 (Figure 1-3) before proceeding to connect ICE cable with the target.

WARNING

Power must be OFF while connecting the EPOD flat cable assembly to the EPOD-68000. The ICE cable for MICE-16 68000 is not interchangeable with those used for other MICE models.

2.1.1 CONNECTING THE ICE CABLE HEADER TO TARGET

Check to ensure that the ICE cable connectors are properly connected to the EPOD-68000. To install ICE cable to the target, remove the protective padding (sponge) from the bottom of the IC header (EB-68KH/L) and gently insert the header into the target CPU socket.

WARNING

To prevent possible damage to equipment, ensure that pin numbers of the target header and that of the target CPU socket, correspond with each other and are properly aligned before attempting to mate them. Note that the header is equipped with a dual-in-line (DIP or DP-64) emulation CPU and that pin No. 1 position is indicated on top of the header.

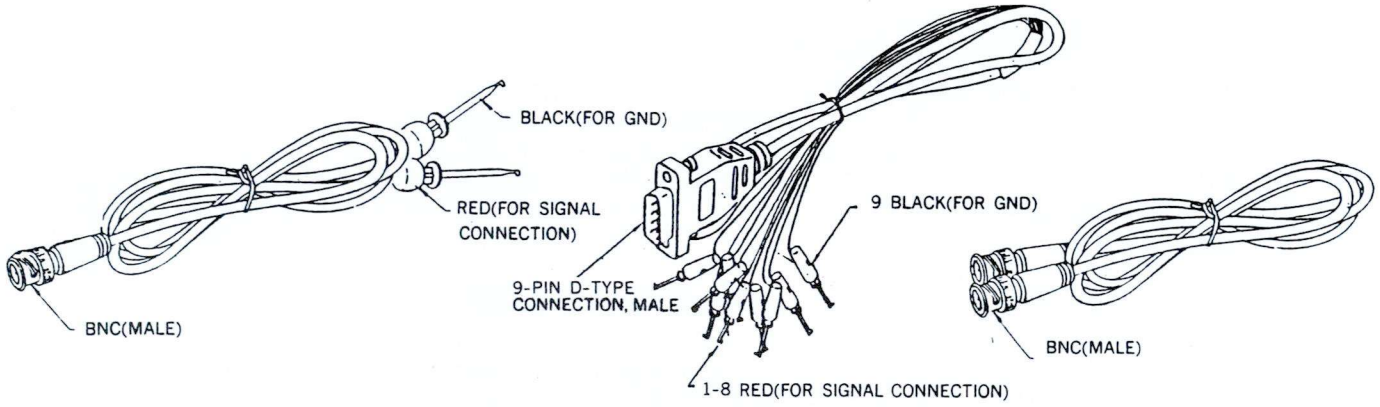
2.2 CONNECTING THE EXTERNAL SIGNAL CABLES

Three coaxial cables with BNC connectors and one cable with a 9-pin D-shell connector are provided for input and output of external hardware signals. The cable types are listed below and shown in the Figure 2-1:

a) Signal I/O cables : two sync signal output cables (used with Events 1 and 2) and one external breakpoint signal input cable (used with Event 3).

b) **Signal monitor cable:** one trace bits input cable (carries data that are displayed at the List and Cycle commands).

c) **Sync cable** : one multiprocessor synchronization cable.



Signal I/O Cable Signal Monitor Cable Sync Cable

Figure 2-1 External Cables

Attach cables as needed to their proper connectors on the MICE-16 68000 front panel as shown in the following figure.

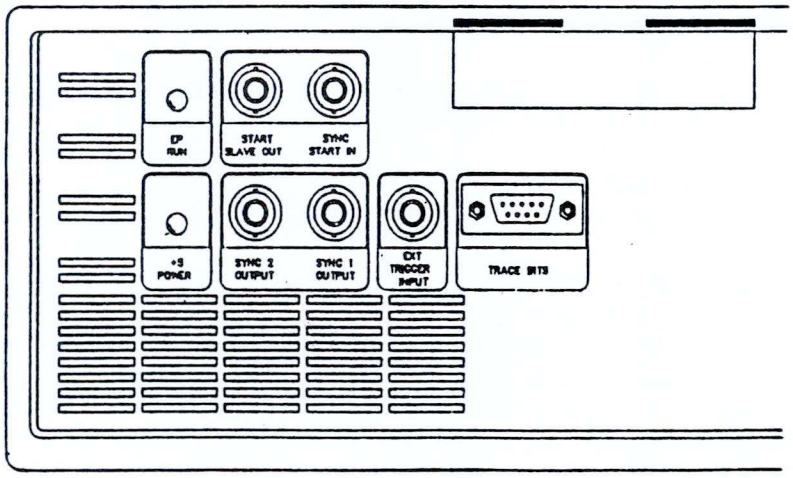


Figure 2-2 External Signal Cable Ports

2.3 CONNECTING MICE-16 68000 WITH TERMINAL OR HOST SYSTEM

The following section describes the details of connecting MICE-16 68000 with data terminal, IBM PS/2, IBM PC/XT/AT, *NEC* PC9801 and compatible only. Refer to the USD User's Manual for the details of connecting MICE-16 68000 with SUN, Apollo, VAX and MicroVAX host computer systems.

2.3.1 IBM PC/XT/AT HOST REQUIREMENTS

The IBM PC/XT/AT used with MICE-16 68000 should have the following specifications:

- PC DOS version 2.2 or later.
- Serial channels (COM 1 or 2)
- 512K bytes of available memory

2.3.2 INTERFACING THROUGH RS-232C SERIAL PORTS

The RS-232C communication ports on the SSM and PSM (designated Channel A and B on SSM and Channel A on the PSM), terminates in a DB-25S D-shell connectors at the back of the MICE-16 68000 (Figure 2-3). The pin assignment used on this connector is shown in Figures 2-4a/b. A suitable cable may be fabricated by referring to these figures and to the pin assignment used on the terminal or host system; the MICE end of the cable should consist of a DB-25P (male) D-shell connector with signals fixed for DCE if cable is intended for Channel A use. Cable intended for serial Channel B use, should consist of a DB-25P (female) D-shell connector with signals fixed for DTE.

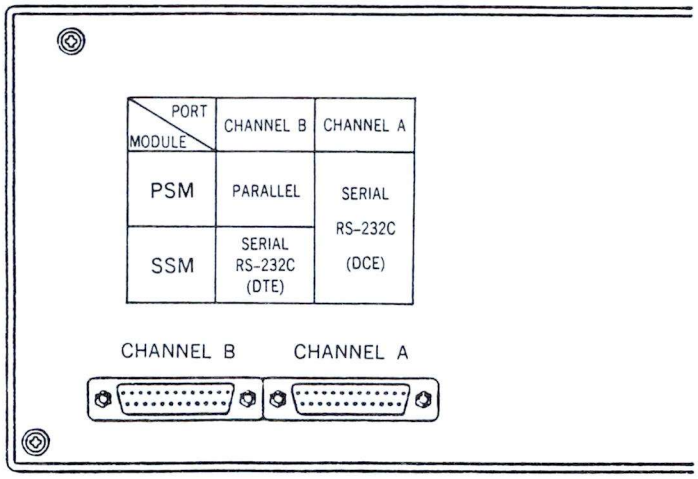


Figure 2-3 Communication Interface Connectors

An alternative setup is to connect only pins 2, 3 and 7 on the MICE-16 68000 side. In this configuration, floating input on pins 4 and 20 is interpreted by the SSM (or PSM) as always active (high).

MICE	Terminal or Host Computer	Pin Definition
Pin 2	<--------o	Transmit Data
3	-------->--------o	Receive Data
4	<--------o	Request To Send
5	-------->--------o	Clear To Send
6	-------->--------o	Data Set Ready
7	----------o	Signal Ground
20	<--------o	Data Terminal Ready

Figure 2-4a Pin Assignment for Serial Interface (Channel A Port, DCE Type)

MICE	Terminal or Host Computer	Pin Definition
Pin 2	>---------o	Transmit Data
3	<---------o	Receive Data
4	>---------o	Request To Send
5	<---------o	Clear To Send
6	<---------o	Data Set Ready
7	----------o	Signal Ground
20	>---------o	Data Terminal Ready

Figure 2-4b Pin Assignment for Serial Interface (Channel B Port, DTE Type)

NOTE

Pins not indicated in the above Figures, are not to be connected. User must correctly determine the right type of interface requirement for their controlling device.

Set baud rate and other interface parameters on the terminal or host computer as described in the documentation for the unit. The MICE-16 68000 will automatically adjust to the settings on the terminal or host when initialization is carried out (Section 3.2).

2.3.3 INTERFACING THROUGH PARALLEL PORT

NOTE

Communication through the parallel interface is currently supported by Universal Symbolic Debugger (USD version 2.3 or later). For detailed information on the USD, refer to the USD User's Manual.

The parallel port on the PSM must connect to an appropriate interface card installed in either IBM PS/2, IBM PC/AT/XT, *NEC* PC-9801 or compatibles. Appropriate interface card for each model of host computer is summarized in the following table:

HOST COMPUTER	IBM PC/AT/XT or compatibles	IBM PS/2*	*NEC* PC-9801
INTERFACE CARD	MS-PCE**	MS-MCE	MS-JPC

* Applies to Models 50, 60, 70, and 80 only. For Model 30, use MS-PCE.

** Also applicable to IBM PS/2 Model 30. Unless otherwise specified, this board (MS-PCE) is shipped with MICE.

Table 2-5 Required Interface Card for Parallel Port Communication

The 8-bit parallel port on the PSM terminates in a DB-25P (male) D-shell connector at the back of the MICE-16 68000. Its pin assignment and signal timing characteristics are shown below.

PIN	SIGNAL	DIRECTION	DESCRIPTION
1	GND	----	Signal ground
2	$\overline{\text{SEND BUFFER FULL}}$	from MICE-16 68000	A "LOW" signal indicates that the MICE-16 68000 has already sent data to the outgoing mailbox. (The host computer then uses the $\overline{\text{READ DATA}}$ signal to fetch it.)
3	$\overline{\text{RECV BUFFER EMPTY}}$	from MICE-16 68000	A "LOW" signal indicates that the incoming mailbox is empty and the host computer can send the next data transmission to MICE-16 68000
4	$\overline{\text{READ DATA}}$	to MICE-16 68000	A "LOW" signal enables data in the outgoing mailbox to be available on the data bus.
5	$\overline{\text{WRITE DATA}}$	to MICE-16 68000	A "LOW" signal enables data on the data bus to be latched into the incoming mailbox.
6	DATA 8	Bi-directional	These signals interface the data bus. Data 8 is the most significant bit (MSB).
7	DATA 7	Bi-directional	
8	DATA 6	Bi-directional	
9	DATA 5	Bi-directional	
10	DATA 4	Bi-directional	
11	DATA 3	Bi-directional	
12	DATA 2	Bi-directional	
13	DATA 1	Bi-directional	
14-15	GND	---	
16-25	not used		

Table 2-6 Pin Assignment and Signals for Parallel Interface

2.3.3.1 IBM-PC/XT/AT MS-PCE Parallel Interface Card Installation and Setup

The MS-PCE Interface card is required for communications between IBM-PC/XT/-AT (including compatibles and IBM PS/2 Model 30) and parallel port (PSM Channel B) of MICE-16 68000. It is a half-card that plugs into any expansion slot except the eighth in the PC or PC/XT, or in the 8-bit bus slot for the PC/AT. Refer to the host computer's manual for instructions on installing auxiliary cards. Port selection and definition are discussed below:

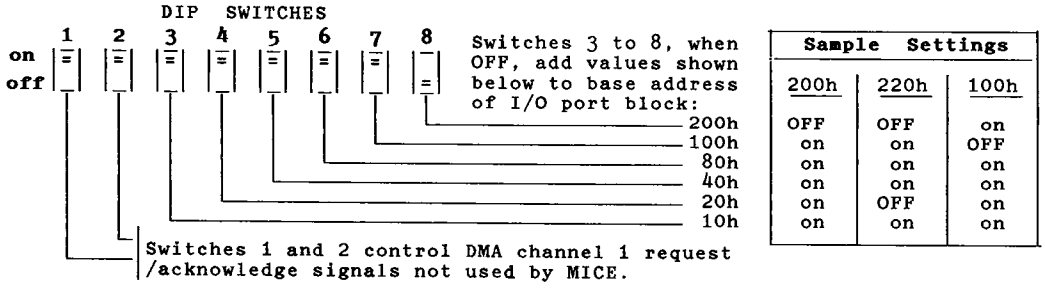
1) Port Selection

Two adjacent I/O ports (data and status) are used as mandatory defaults on the interface card to facilitate parallel communications. Switches 3 to 8 on the DIP switch are used to specify the six Most Significant Bits (A9-A4) of the first I/O port address. The card is shipped from the factory with switches 3 to 7 on and switch 8 off, which sets the port address to be used for parallel data communications at 200H and 201H.

To set the software tool to use I/O port addresses which correspond with the switch setting, the user should specify the parallel port option value in the installation file of the tool (e.g., "P:" option in the USD installation file, "USD.INS").

Address bits A3-A1 are not decoded by the interface card so the actual address space is 16 I/O ports. Data port or status port selection is indicated by bit A0. The data port is activated if A0 is low (logic 0); and the status port is activated if A0 is high (logic 1).

The following figure indicates how the port address is specified by switch position, with the switches preset at 200H.



Sample Settings		
200h	220h	100h
OFF	OFF	on
on	on	OFF
on	on	on
on	on	on
on	OFF	on
on	on	on

Figure 2-7 MS-PCE Port Selection Example

NOTES

a) Make sure the port address does not conflict with any other expansion card residing in the system. For example, the game control adapter on the AST multifunction card must be disabled since it also uses port address 200H.

b) Certain port addresses are reserved for special expansion cards or host-system functions and cannot be used for MICE I/O; check the I/O address map in the manual for the host computer before selecting an address.

c) The address selected on the card must be followed by 15 other currently unused addresses.

d) The MICE does not make use of the DMA request and acknowledge lines controlled by switches 2 and 1; the presence of the MS-PCE will not affect the function of any other expansion card which uses DMA channel 1.

2) Port Definition

One Data Port and one Status Port can be defined on the MS-PCE interface card:

a) The first even address port specified (e.g. 200H) is used as a bi-directional read/write Data Port.

b) The first odd address port specified (e.g. 201H) is used as a read-only Status Port. The first status bit lets the IBM-PC know when data in the Data Port can be read; and the second status bit lets it know when the Data Port is ready to accept data from the IBM-PC.

Status Port bit assignment is as follows:

Status Port bit assignment:

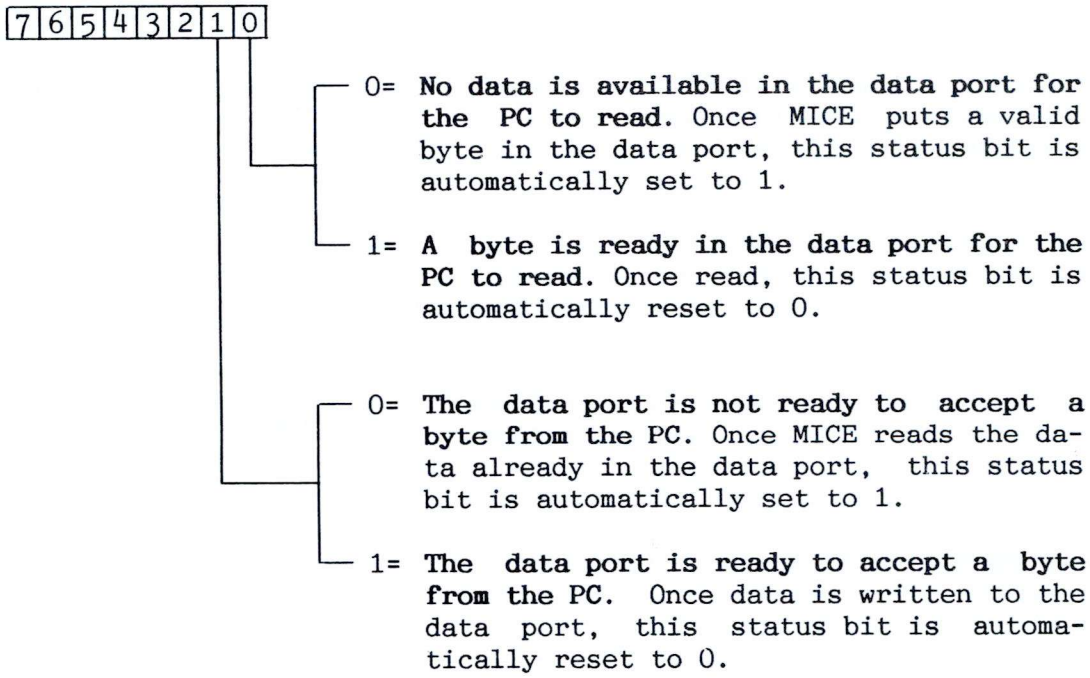


Figure 2-8 MS-PCE Status Port Bit Assignment

2.3.3.2 IBM PS/2 MS-MCE Parallel Interface Card Installation and Setup

The MS-MCE Interface card is required for high-speed bi-directional "MAILBOX" communications between IBM PS/2 and parallel port (PSM Channel B) of MICE-16 68000. The card can be installed into any expansion slot of the PS/2 (Model 50, 60, 70 or 80). Refer to "MS-MCE Interface Card Operation Manual" and the host computer's manual for instructions on auxiliary card installation.

After performing the MS-MCE setup diagnostic with file "@60AC.ADF", initiate PS/2 parallel communication port address setting with file "MCE.EXE" by executing the following command. Note that the above mentioned files are provided with the USD package:

>MCE [port-address]

Where [port-address] is the I/O port address setting to be used for parallel data communication. If no port address is specified, setting will default at 200H.

2.3.3.3 *NEC* PC-9801 MS-JPC Parallel Interface Card Installation and Setup

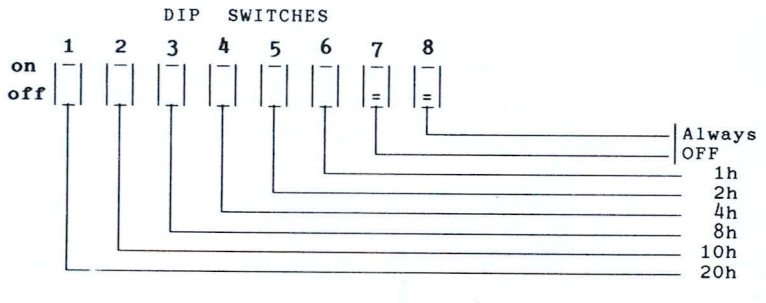
The MS-JPC interface card provides bi-directional "MAILBOX" communication between MICE-16 68000 and *NEC* PC-9801 series computer. It can be plugged in any expansion slot of the PC-9801. Refer to the computer manual for expansion card installation.

1) Port Selection

MS-JPC interface card provides two DIP switches from which I/O port (data and status) address is selected. The I/O port address range may be specified from C0 to FF (hexadecimal).

2) Port Definition

The DIP switch U4 is the data port and the DIP switch U5 is the status port. These DIP switches are defined as follows:



Sample Settings	
DATA PORT	STATUS PORT
C0	C1
OFF	OFF
OFF	OFF
on	OFF
on	on
on	on
on	on
on	on
on	on

Figure 2-9 MS-JPC DIP Switches U4 and U5 Port Selection Example

3) Port Bit Assignment

Status Port bit assignment is the same with MS-PCE (Figure 2-8).

NOTE

The Data Port address and Status Port address can be set at any address between C0 and FF, but both ports should not have the same address value at the same time.

When using USD with NEC PC-9801 through MS-JPC card, "P: port-address" option should be specified in the USD's "USD.INS" file, e.g., the port-address as data port, and the adjacent port address as status port. For example; setting "P: C0" means the data port is C0, and the status port is C1. The switches on MS-JPC should be set to correspond with the value specified in "P:" option.

2.4 USER SERVICEABLE PARTS

The MICE-16 68000 is housed in a four-piece, high-impact metal case. The case may be opened in order to perform the following user level services.

1) Inspect or replace defective module boards and EPOD flat cables.
2) Swap EMM with HEMM* modules and vise-versa.

Likewise the emulation pod (EPOD-68000) may be also opened to inspect, remove or change defective CB-68000 control board.

The cooling fan also required to be cleaned periodically as described in Section 2.4.4, but the case need not be opened.

2.4.1 OPENING/CLOSING THE MICE-16 68000 CASE

To open the case, remove the three retaining screws with Allen wrench as shown below.

WARNING

Make sure that power is turned off before attempting to open the case.

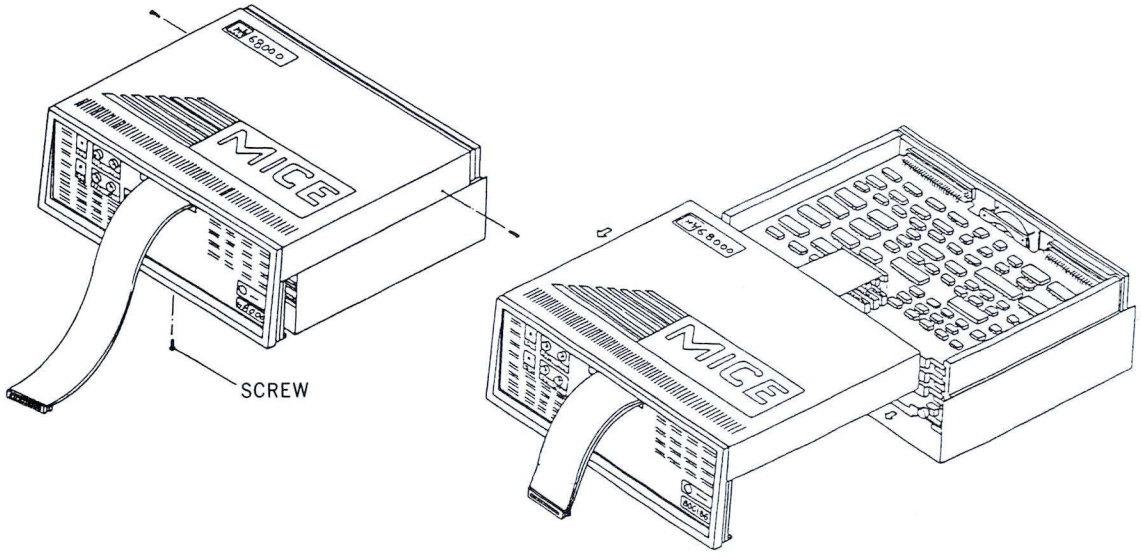


Figure 2-10 Opening the MICE-16 68000 Case Assembly

When closing the MICE-16 68000 case, make sure that the flat cable (leading to EPOD-68000) passes through the opening in the front panel, then align the screw holes of the cover assembly with that on the chassis and secure it with the cover retaining screws.

* The optional HEMM will be available with Firmware Version 3.0 to be released soon.

2.4.2 CHANGING MODULE BOARDS

This section describes the procedure for replacing MICE modules for repair, module update, etc.

WARNING

Defective boards are to be replaced only as complete module and returned to authorized service centers for repair. Any users attempt to perform unauthorized repair on the boards may nullify the warranty covering the MICE as a complete equipment, or the module as a part.

1) First detach the cover assembly (top cover and front panel). Next, use an (2.0 mm) Allen wrench to remove the two retaining screws (which secures the boards to the chassis) as shown in Figure 2-11a.

2) Loosen the module board by pulling the two ejector latches outward (Figure 2-11b) and carefully slide the board out (Figure 2-11c).

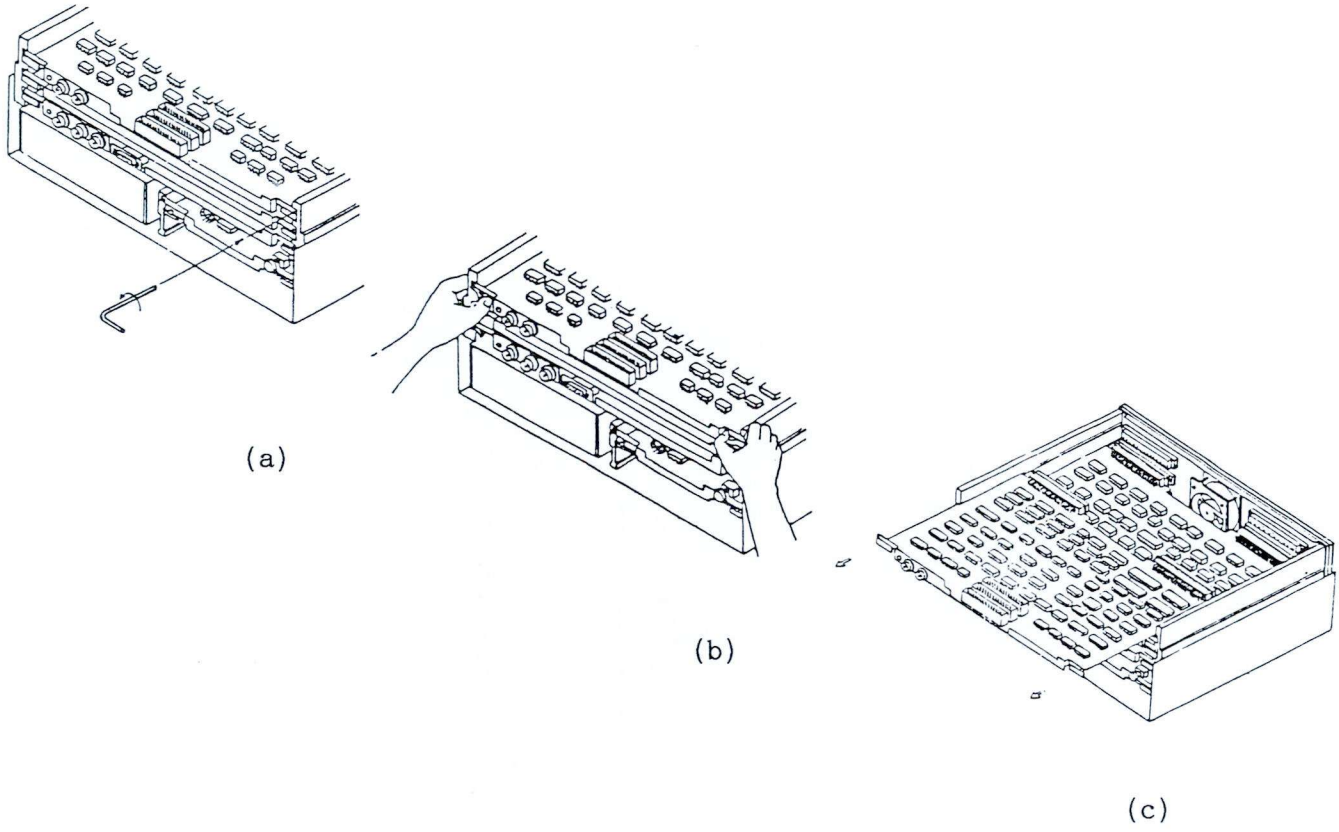


Figure 2-11 Changing a Module Board

3) Install the new module board by sliding it into the unit, carefully aligning its interface connectors with those on the motherboard, and pushing on the ejector latches until the board is firmly seated. Secure to the chassis with the retaining screws.

2.4.3 Changing EPOD-68000 Flat Cable

The EPOD-68000 flat cable assembly is made up of three flexible flat cables, with a 50-pin connector at each end. When found defective, the cable is to be removed and replaced as a complete assembly as follows:

1) Disconnect the flat cable from EPOD-68000 by unplugging the three 50-pin connectors from the EPOD.

2) Open the MICE case and unplug the other set of 50-pin cable connectors from the EPM board and carefully slip them out through the cable slot on the MICE case front panel (Figure 2-12).

3) To install a new EPOD-68000 cable, plug the cable connectors from either end of the cable to the EPM board. Pass the free end of the cable through the cable slot before closing the MICE case.

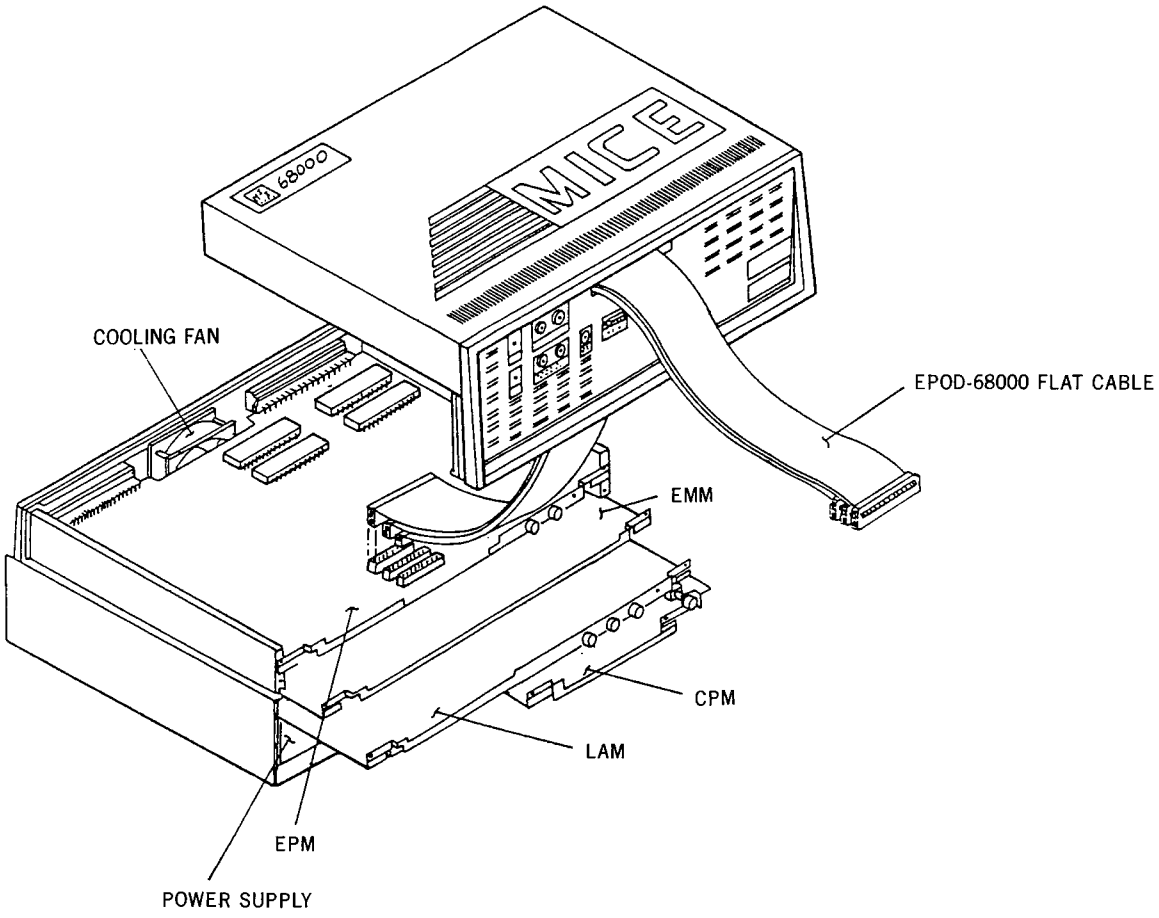


Figure 2-12 Removal/Installation of EPOD-68000 Flat Cable

2.4.4 Changing ICE Cable Assembly and EPOD-68000 Buffer Box

The ICE cable assembly, consisting of a target header (EB-68KH/L) and a 15-cm (6 inches) cable may be removed and replaced by simply plugging or unplugging its two half-pitch connectors from the EPOD-68000. Defective ICE cable should be replaced and return to authorized service center for repair as a complete assembly.

The buffer box (EPOD-68000), when found defective, should also be removed and replaced as a complete unit.

2.4.5 Changing the Cooling Fan Filter

NOTE

Filter should be cleaned or replaced every 6 months.

To protect the module boards from dust, MICE-16 68000 is equipped with a polyester air filter (I57- 000178) mounted at the cooling air intake vent of the rear panel. This filter must be inspected and cleaned every six months. If found worn, filter must be replaced.

Remove the filter by removing the two screws which secure the filter cover to the rear panel as shown in the following figure. Wash filter with a non-toxic cleaning solvent, blow dry with warm air, and reinstall.

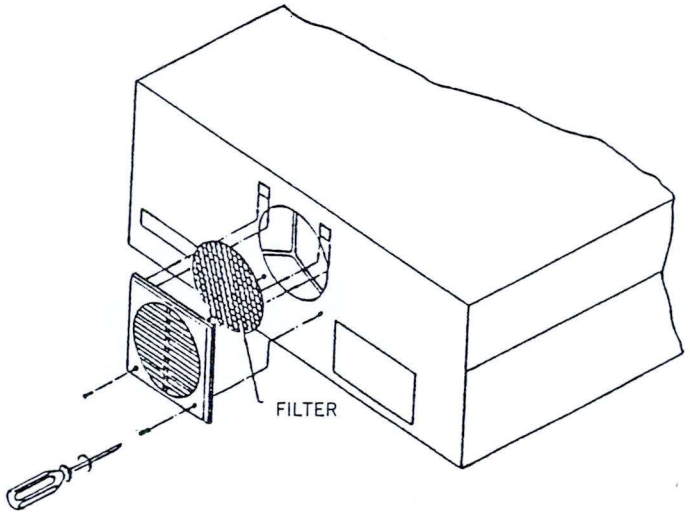


Figure 2-13 Removal/Installation of the Cooling Air Filter

Chapter 3

POWER UP AND INITIALIZATION

This chapter covers the details of the following :

- Applying power to the MICE-16 68000 (Section 3.1)
- Initialization (Section 3.2)
- Default Setup Parameters (Section 3.3)
- Hardware Reset (Section 3.4)
- Configuring MICE-16 68000 for Target (Section 3.5)
- Trouble Shooting Guide in Case of No Response (Section 3.6)

3.1 APPLYING POWER TO THE MICE-16 68000

WARNING

Check voltage selector switches of all equipment for correct voltage settings and safety grounding.

3.1.1 PRE-POWER UP CHECK LIST

1) Set up the host computer or terminal according to its user's guide.

2) Set up the MICE-16 68000 as explained in Sections 2.1 to 2.3. Power and environmental requirement of MICE is described in the "PREFACE" section of this manual

3) Set up the interface cable to be used between the terminal or host computer and the MICE (Section 2.3) and plug it into the female D-shell connector at the back of the MICE (Figure 3-1).

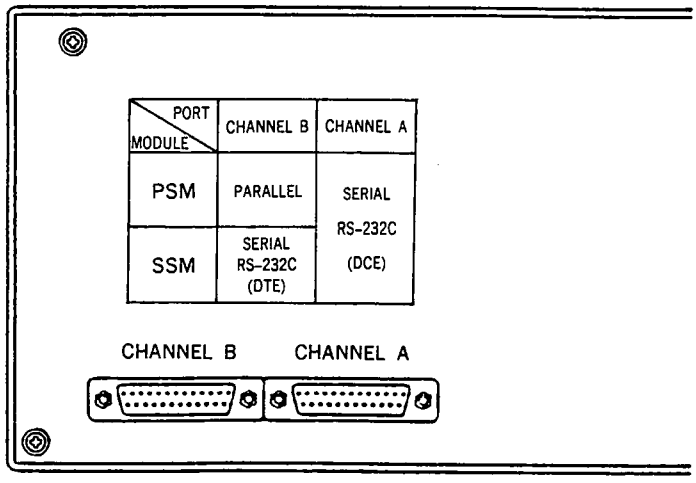


Figure 3-1 Communication Interface Connection

4) Plug the power cord into the D-shell power input connector (Figure 3-2) at the back of the MICE. Check MICE power setting.

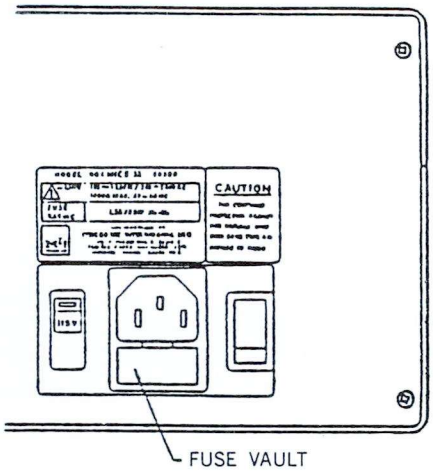


Figure 3-2 Input Power Panel

WARNING

The MICE-16 68000 must be set for 115V or 220V AC power. Before applying power, ensure that the power source selector switch at the input power panel (Figure 3-2) is set to match the available current supply.

3.1.2 MICE POWER UP/DOWN SEQUENCE

The recommended power up sequence is to apply power to target system first and then to MICE. When turning the power off, MICE and other equipment are to be turned off first and target system the last.

NOTE

1) *When power is applied or a hardware reset (Section 3.4) or software reset command is executed, the MICE-16 68000 reads the first four words of memory (bytes 000000-000007H) and loads the supervisor stack pointer (SSP) and program counter (PC) from these locations. If either the SSP or PC is initially loaded with an odd address (as specified in memory bytes 000000-000007H), the system will automatically set the PC to 400H and the SSP to 1FFF0H. If both the SSP and PC are initially loaded with even addresses (as specified in memory bytes 000000-000007H), the system will accept the values without adjustment.*

2) *If the first four words of memory (bytes 000000-000007H) is mapped as internal memory (I or IR) the datum 0001H, FFF0H, 0000H and 0400H will be written into this location after power up.*

3) *At power-up, the MICE-16 68000 automatically performs a self-test and checks to determine whether power is supplied to the target. If target power is detected, MICE will auto-switch to external clock. Otherwise, the internal (12 MHz) clock will switch on.*

3.2 INITIALIZATION

This section describes how to initialize the MICE-16 68000 for the terminal or host computer and for the target system. Also refer to the Universal Symbolic Debugger (USD) User's Manual for the initialization of MICE-16 68000 with USD in a host computer.

3.2.1 POWER UP SELF-TEST

After powering up MICE with its serial (Channel A) port connected to a terminal or host computer, input "M<CR>" (upper-case "M" is required) from the keyboard to initiate auto-interfacing. The interface parameters on the MICE (baud rate, data bits, stop bits and parity) are then automatically adjusted to match the current settings of the host computer or terminal.

If MICE is connected through its parallel (Channel B of PSM) port, MICE will automatically perform self-test and establish communication link with its host immediately after "M<CR>" is entered.

3.2.2 POWER UP SCREEN DISPLAY

When proper communications are established, the results of the MICE's power-on self-test will then display on screen. Figure 3-3a below shows the screen display when MICE is equipped with EMM and Figure 3-3b when it has the HEMM* installed. If there is no response, refer to Section 3.6.

The display will also show the version number of the controlling firmware on the CPM at the top of the screen. At the same time, the type of CPU (68000 or 68010) installed on the ICE header (EB-68KH) is automatically detected and displayed.

```
      *** MICE-16 68000/68010, Firmware V#.# ***
        MICROTEK International Inc., 1987, 1988

  The Emulation CPU is 68000

  Self-test in progress...
  CPM ROM    OK
  CPM RAM    OK
  EPM RAM    OK
  LAM        OK
  EMM RAM    OK
  Self-test completed

>
```

Figure 3-3a Status Screen for Terminal Mode

* The optional HEMM will be available with Firmware Version 3.0 to be released soon.

```
        *** MICE-16 68000/68010, Firmware V#.# ***
          MICROTEK International Inc., 1987, 1988

  The Emulation CPU is 68000

  Self-test in progress...
  CPM ROM    OK
  CPM RAM    OK
  EPM RAM    OK
  LAM        OK
  HEMM RAM BANK1  OK
  HEMM RAM BANK2  OK
  HEMM RAM BANK3  OK
  HEMM RAM BANK4  OK
  Self-test completed

 >
```

Figure 3-3b Status Screen for Terminal Mode with HEMM

NOTE

1) *If any failure is detected, it will be listed in the following format:*

   "<board> <device> FAILURE! (location)"

   If a failure occurs on the EPROM however, it may not be possible to determine the precise location; in such cases, more than one board location may be indicated.

2) *If the type of CPU can not be recognized due to hardware problem, the following message will display and the program hangs-on until <ESC> is entered.*

   "CPU is not ready"

3.2.3 POWER UP DATA CHECK

MICE continues to check the data (memory map, timebase, SSM Channel B baud rate, and select codes) stored in NOVRAM to determine their compatibility with MICE-16 68000 Identification Code (ID).

If the medium (EMM or HEMM) of memory map stored in NOVRAM is inconsistent with what is actually installed, mapping to external memory will default.

Additional messages will also display near the bottom of the screen (Figure 3-3c for EMM and 3-3d for HEMM) if:

a. The ID Code does not match with the NOVRAM data.

Under this condition MICE power up setup is accomplished with the system default setup parameters and the following message will display-

"NOVRAM parameters inconsistent!
Proper default parameters have been set"

When this occur, the data stored in the NOVRAM is not changed and it is recommended that the NOVRAM data are updated using the SAve command (Section 5.3.14).

b. Power is not supplied to the target at the time of the self-test.

When this condition occurs, the following message will also display on the same screen. Perform warm reset (Section 3.4) after target is powered up.

"No target VCC; BERR, BR, INTR (IPL0-2) disabled"

c. The SSP or PC are initially loaded with odd addresses.

If the SSP or PC are initially loaded with odd addresses (versus the even addresses specified in memory bytes 000000-000007), the following message will also appear on the screen.

"Odd PC/SSP values found in memory locations 0-7
EP set by default to PC=400H and SSP=1FFF0H"

```
      *** MICE-16 68000/68010, Firmware V#.# ***
        MICROTEK International Inc., 1987, 1988

  The Emulation CPU is 68000

  Self-test in progress...
  CPM ROM    OK
  CPM RAM    OK
  EPM RAM    OK
  LAM        OK
  EMM RAM    OK
  Self-test completed

  NOVRAM parameters inconsistent!
  Proper default parameters have been set

  No target VCC; BERR, BR, INTR (IPL0-2) disabled

  Odd PC/SSP values found in memory locations 0-7
  EP set by default to PC=400H and SSP=1FFF0H

>
```

Figure 3-3c Status Screen for Terminal Mode with EMM Showing Warning Messages

```
      *** MICE-16 68000/68010, Firmware V#.# ***
        MICROTEK International Inc., 1987, 1988

  The Emulation CPU is 68000

  Self-test in progress...
  CPM ROM    OK
  CPM RAM    OK
  EPM RAM    OK
  LAM        OK
  HEMM RAM BANK1  OK
  HEMM RAM BANK2  OK
  HEMM RAM BANK3  OK
  HEMM RAM BANK4  OK
  Self-test completed

  NOVRAM parameters inconsistent!
  Proper default parameters have been set

  No target VCC; BERR, BR, INTR (IPL0-2) disabled

  Odd PC/SSP values found in memory locations 0-7
  EP set by default to PC=400H and SSP=1FFF0H

>
```

Figure 3-3d Status Screen for Terminal Mode with HEMM Showing Warning Messages

3.3 MICE-16 68000 DEFAULT SETUP PARAMETERS

The following setup defaults are in effect whenever the MICE-16 68000 is powered up:

1) Handshaking Code - 03H

2) Timer Interval Setting - INTerval OFf

3) EMM Ready Signal Selection - REAdy ($\overline{\text{DTACK}}$) Internal

4) Memory Access Size - SIze Word

5) Memory Verification Switch - Verify On

6) Insert Wait State - Wait OFf (0 wait state insert)

7) Events 1 to 6 - Events 1, 2, 4, 5 and 6 Clear Event 3 Low

8) Cycle Qualifier - (trace set for all cycles)

9) Trigger Condition	- Trigger Event 1 OR Event 2 (cycle delay count = 0)
10) Sync Input/Output	- OFf
11) Baud (Serial Channel B)	- Recall from NOVRAM (factory NOVRAM setting: Baud is 9600 bps; Even parity; 7 data bits and 1 stop bits)
12) Select	- Recall from NOVRAM (factory NOVRAM setting: SELect device leading code=01H SELect route leading code=15H)
13) Router	- Recall from NOVRAM (factory NOVRAM setting: Route is in terminal mode)
14) Timebase Selection	- Recall from NOVRAM (factory NOVRAM setting: Timebase is 1 *us*; timer starts when emulation CPU begins to run)
15) Emulation Memory Mapping	- Recall from NOVRAM (factory NOVRAM setting: **with EMM-** All memory mapped to external. **with HEMM*-** All memory mapped to external.

These settings serve as defaults and are effective until modified by the user. They are not affected by hardware or software reset commands. Current settings for all the parameters listed above can be viewed with the MAp, SETup and TRAce commands (Sections 5.3.8, 5.3.15 and 5.7.8, respectively).

Baud (11), Select (12), Router (13), Timebase Selection (14) and Emulation Memory Mapping (15) had their default values saved in the NOVRAM. These values are preset in the factory. User may change these values and save the new values in the NOVRAM using the SAve command (Section 5.3.13).

* The optional HEMM will be available with Firmware Version 3.0 to be released soon.

3.4 HARDWARE RESET

The MICE-16 68000 may be reset by either of the following methods:

1) Cold Reset

   By turning power to the MICE-16 68000 off and then on again. (Be sure, however, to wait a few seconds and let the capacitors discharge fully before restoring power to the unit. Note that whenever the MICE-16 68000 is powered up, it automatically performs a self-test after "M<CR>" is keyed in).

2) Warm Reset

   By pressing the manual RESET switch on the front panel as shown in Figure 3-4 below. The warm reset switch is used to reset the control and emulation processor. Note that the contents of EMM memory are not affected by this type of reset.

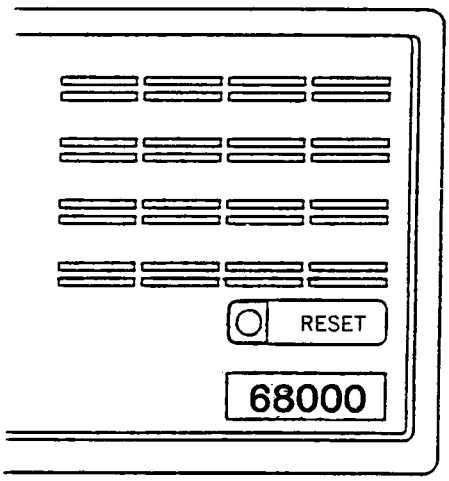


Figure 3-4 Hardware Warm Reset Switch

User must re-enter "M<CR>" after a warm reset to initiate auto-interfacing again as described in Section 3.2.1. The screen will then display the following:

```
        *** MICE-16 68000/68010, Firmware V#.# ***
           MICROTEK International Inc., 1987, 1988

    The Emulation CPU is 68000

    No target VCC; BERR, BR, INTR (IPL0-2) disabled
>
```

Figure 3-5 Status Screen after Warm Reset

3.5 CONFIGURING MICE-16 68000 FOR TARGET

After power up processes are completed successfully, a right angle-bracket prompt [>] will appears on the screen to indicate that Terminal Mode commands may now be input to start configuring MICE-16 68000 for target requirements as described below.

a) Emulation Memory Ready Signal Selection - REAdy [Internal|External]

When memory on the EMM is used, the emulation memory "ready" signals ($\overline{\text{DTACK}}$ for 68000) can be generated either internally (by the MICE) or externally (by the target). Refer to Section 5.3.10 for selection command details.

b) Control Signal Selection - CONtrol [[Berr] [Intr] [BR] {Enable|Disable}]

Certain 68000 control signals may be enabled or disabled at the input pins of the emulation CPU (see Section 5.3.3 for a description of the command syntax). Only input at the emulation CPU is affected; the setting is transparent to circuits which generate the signals in the target system. Note that when the emulation CPU is reset (Section 5.6.4), these signals are disabled if power is not applied at the target.

Table 3-6 below explains the control signal parameters used with the Control command. Pin numbers are for a dual-in-line (DIP) device.

Parameter	Signal Type	Name	Pin No.
Berr	Bus Error	$\overline{\text{BERR}}$	22
Intr	Interrupt Request* (including non maskable interrupt)	$\overline{\text{IPL0}}$ $\overline{\text{IPL1}}$ $\overline{\text{IPL2}}$	25 24 23
BR	Bus Request	$\overline{\text{BR}}$	13

* Disabling the interrupt signals will not affect the interrupt mask list in the processor status register.

Table 3-6 Parameters for Control Signal Enable/Disable

c) Memory Mapping

MAp [space-adr adr {{I|IR} {1|2|3|4}|E|ER|G}|ALl {E|G}]

1) With EMM

256K bytes of emulation memory can be mapped anywhere within 16M bytes for a 24 bit wide bus. (Refer to Section 5.3.8 for the detailed description of the command syntax.) Any 4K-byte block* of internal (EMM) memory can be set to-

writable (I)
read-only (IR)
external memory (E)
external read-only (ER)
guarded memory (G)

according to the amount of emulation memory required or as special memory access type assignments described below. (Remember that external and guarded memory are not mapped to the emulation memory.)

supervisor program (SP)
user program (UP)
supervisor data (SD)
user data (UD)

2) With HEMM**

Where the larger emulation memory HEMM (consisting of 1M bytes static RAM) is used, four independent 256K segment settings are provided, and any 4K-byte block may be enabled, disabled, set as write-protected or as non-existence or as special memory access type assignments, as defined above.

HEMM therefore is capable of mapping up to 1M of emulation memory anywhere within 16M bytes. See Section 5.3.9 for more details on memory access type assignment command.

3.6 IN CASE OF NO RESPONSE

If there is no response from the MICE-16 68000 when the setup procedures are performed, check the following items:

1) Power supply and voltage.

2) Interface cable and connections (make sure that the cable is connected to the correct port and that the port is working).

3) Request to Send and Data Terminal Ready signals (pins 4 and 20). These must be either at +12V DC or disconnected (Section 2.3.2).

4) Voltage levels at the terminal's or host computer's RS-232C port.

* 4K-byte block for MICE-16 68000 with EPOD (F/W Versions 2.0 or later). Previous versions is 16k-byte block.

** The optional HEMM will be available with Firmware Version 3.0 to be released soon.

5) Connections between the module boards and the motherboard. Reinsert properly if required to establish good contact.

6) Driver or terminal emulation program, if using a host system. Make sure that the software is running properly.

NOTE

If RS-232C interface parameters on the terminal or host computer are changed, communication with the MICE may fail. These parameters are checked only on power-up or MICE reset.

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

Chapter 4

CONVENTIONS OF NOTATION AND COMMANDS SUMMARY

This chapter presents the basic principles of the command language used when operating the MICE-16 68000 in Terminal Mode. i.e.:

- MICE-16 68000 command syntax notation (Section 4.1)
- Listings of all commands complete with required and optional parameters (Section 4.2)

The commands themselves are individually described in details in Chapter 4 where examples are also given.

4.1 CONVENTIONS OF NOTATION

4.1.1 RULES OF INPUT

4.1.1.1 Legal Command Input

The MICE indicates that it is ready to accept a command by displaying an angle bracket prompt [>] on a new line. Any legal command may then be entered. The general syntax format for MICE commands is:

command [parameter(s)]<CR>

where: **command** is the **command syntax keyword**, a character or combination of characters invoking a specific function.

[parameter(s)] is one item or more of information, in some cases required and some cases optional, further defining or extending the operation to be carried out.

<CR> is carriage return key [↵] which must be entered at the end of every command line.

Multiple parameters must be input in the order shown in the command syntax. A space or comma <,> must be entered between the command keyword and any subsequent parameter entry, as well as between parameters whenever more than one parameter is required.

Command keywords and parameters need not be entered in its full word format. Shorthand format is available and are denoted in upper-case characters in each command syntax listed throughout this manual, e.g.:

SEarch

"SE" (upper or lower-case) is the shorthand or the minimum characters which user may enter to execute the "SEARCH" legal command, i.e.,

>SE<CR>

Keying "SEA", "SEAR", "SEARC" or "SEARCH" are equally acceptable. Pressing the space bar after a shorthand entry, will auto-expand the shorthand into its full word format.

4.1.1.2 Input in ASCII

Where input in ASCII (instead of hexadecimal values) is required or permitted, it must be enclosed in apostrophes, e.g., 'AR'. Note that the apostrophe ['] itself need not be input in ASCII.

4.1.1.3 Wildcard Nibbles Input

Wildcard nibbles may be specified in certain parameters ("adx", "datum" and "trace_bits"). To specify a wildcard nibble, replace the corresponding hexadecimal digit with an "X". For example, "12X456" means 16 values: 120456H, 121456H, 122456H, and so forth, up to and including 12F456H.

4.1.1.4 Wildcard Bits Input

To specify wildcard bits, enter a four-bit binary number, enclosed in parentheses, in place of the corresponding hexadecimal digit, using "X" to indicate "don't care". For example, "12(00XX)456" means 120456H, 121456H, 1223456H or 123456H.

4.1.2 DEFINITIONS OF SYNTAX AND COMMAND EXAMPLES NOTATION

The following conventions are used in this manual to define MICE command syntax notations as used in the individual command execution examples shown in the next chapter:

4.1.2.1 Parameter(s) Without Enclosures

List ***Number***

REQUIRED parameter are shown without any enclosures. Hence. in the example above, user **must** enter "N(umber)" right after the command keyword "L(ist)" in order to display the last frame number in the trace buffer and the total elapsed time since timer start.

4.1.2.2 Parameter(s) in Square Brackets ([])

HANdshake [*code*]

OPTIONAL parameter(s) are enclosed in square brackets. As shown in the above example, the brackets enclosing "code" mean that a hexadecimal value indicating a handshaking code is optional.

"HAN(dshake)" or "HAN(dshake) + hex value" are both legal commands but each will execute different results, i.e., when "HAN(dshake)<CR>" alone is entered, the current MICE handshaking code will display on screen. When "HAN(dshake) 03<CR>" is input, a one-byte hexadecimal value of 03 will be sent to the terminal or host computer to indicate that MICE is ready to accept input.

4.1.2.3 Parameter(s) in Brackets ([]) and Vertical Bars (|)

TImebase [*Go*|*Event*] [*1*|*10*|*100*|*1000*|*10000*]

Where there are several choices for an **OPTIONAL** parameter, they are enclosed in brackets and separated by vertical bars. The above example indicates that it is permissible to enter just the keyword "TI(mebase), or the keyword followed by either "G(o)" or "E(vent)" as first optional entry and "1", "10", "100", "1000" or "10000" as second input option. Again, each combination of legal entry will have different results.

4.1.2.4 Parameter(s) in Braces ({}) and Vertical Bars (|)

Help [GRoup {*Emulation*|*Memory*|*Port*|*Setup*|*Trace*}|command|AL1]

Where there are several choices for a **REQUIRED** parameter, they are enclosed in braces and separated by vertical bars. In the above example, it indicates that if the optional parameter "GR(oup)" is input, it **must** be followed by either E(mulation), M(emory), P(ort), S(etup) or T(race); e.g.-

"H(elp) GR(oup) E(mulation)", "H(elp) GR(oup) M(emory)", etc.

"H(elp) GR(oup)" alone is invalid.

4.1.2.5 Parameter(s) in Lower-Case

Qualify *adx [status]*

Any parameter shown entirely in lower-case represents a collective nomenclature, location, setting, values, etc. of the parameter. User should **never literally input** the characters itself. The particular string , actual value or setting so described in the command syntax should be entered. In the example above, where the lowercase items "adx" and "status" appears, the MICE will only accept a hexadecimal address setting in the cycle qualifier (e.g., 1234, 1XX4, etc.) in lieu of "adx" and a processor status code (e.g., SP, SR, SW, etc.) for "status". Hence actual input should look like this:

>Qualify *1234 SP SR*<CR>

The following lower case parameters and their definitions are used in the MICE-16 68000 command syntax:

adr is a hexadecimal address setting.

adx is a hexadecimal address setting, with option to include wildcard nibbles/bits, e.g. 1234, 1XX4, 1(X1X0)X4, 1(X1X0X0XX)4.

baud-rate is baud rate of serial channel B (110 150 300 600 1200 2400 4800 9600 19200).

code is a one-byte hexadecimal value.

command is a specific MICE command.

count is a hexadecimal value (from 0H to 0FFFFH).

data is 1 to 32 bytes of data in hex and/or ASCII.

data-bits is either 7 or 8 data bits of serial channel B.

datum is a byte, word or long-word in hexadecimal, with wildcard nibbles/bits option, e.g. 1234, 1XX4, 1(X1X0)X4, 1(X1X0X0XX)4.

frame is a hexadecimal value (from 0H, to 07FFH) indicating the frame at which the listing is to begin.

in-length is a hexadecimal value (from 01H to 0FFFFFFH).

length is a hexadecimal value (from 01H to 0FFFFFFH) specifying the number of location to access.

parity is E (even), O (odd) or N (none) parity of serial channel B.

pc is a hexadecimal address to load in place of the current PC.

register is the standard Motorola-68000/68010 designation for the register, the contents of which are to be displayed or modified.

space-adr is a hexadecimal address setting, with option to specify memory type: "[UP|UD|SP|SD] adr".

ssp is a hexadecimal address to load in place of the current ssp.

status is 1 to 7 processor status codes (SP SR SW UP UR UW AK).

string is a 1-8 character command prompt, input in ASCII.

trace-bits is a one-byte hexadecimal setting indicating signal levels for trace bits input from the external port. Includes option to specify wildcard nibbles/bits.

value is a hexadecimal value of vector base register.

4.1.2.6 Underlined Spaces and Characters in the Command Entry Examples

>Help GRoup Port<CR>

Characters and spaces underlined in the command entry examples indicate the minimum keyboard entry user must input to execute a given command syntax.

4.1.3 EDITING KEYS

4.1.3.1 Special Editing Keys

All keyboard input is stored in a line editing buffer until <CR> is entered. The contents of this buffer can be edited or entirely deleted with the following special editing keys:

BACKSPACE deletes the preceding character from the line buffer and from the display. Repeated usage is allowed. On some terminals and computers this key may be labeled "Bs" or indicated with a left arrow.

ESCAPE cancels 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 expressed as <ESC> in this manual. On most keyboards this key is labeled "ESC" or "Esc."

4.1.3.2 Control Characters

Control characters are input by simultaneously holding down the Ctrl (Control) key and pressing the appropriate letter key. They are expressed in this manual as "Ctrl-" and an alphabetic character, together enclosed in angle brackets. The following control characters have special meaning for the MICE:

<Ctrl-H> is the same as BACKSPACE.
<Ctrl-J> is the same as <LF>.
<Ctrl-M> is the same as <CR>.
<Ctrl-Y> is the same as <ESC>.
<Ctrl-S> suspends data transmission from the MICE.
<Ctrl-Q> resumes data transmission from the MICE.

NOTE

In most instances, <Ctrl-J> or <LF> performs the same function as <CR>, except when executing the following:

For the Assembly, Cycle and Step commands, <CR> is used to advance to the next line. For the Memory Modify (Byte|Word|LOng) and Register Modify commands, a <CR> is used to advance to the next register or location and <Ctrl-J> is used to go back to the preceding one.

4.2 SUMMARY OF COMMANDS

4.2.1 COMMAND OVERVIEW

1) Setup Commands

BAud	IDentify	RECall	SIze
CLock	INTerval	ROute	Verify
CONtrol	MAp	SAve	WAit
HANdshake	Prompt	SELect	
Help	REAdy	SETup	

2) Memory Commands

Assemble	COMpare	Fill	SEarch
Byte	COpy	LOng	TEst
CHecksum	Disassemble	Memory	Word

3) Port Commands

Input	Output

4) Emulation Commands

Cycle	Register	RESet	Step
Jump			

5) Trace Commands

Event	List	SYnc	TRAce
Go	Qualify	TImebase	Trigger
HAlt			

4.2.2 COMMAND SYNTAX LISTING

1) Setup Commands Group

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Port B Baud Rate Setting	BAud	[baud-rate [parity] [data-bits]]
Display Clock Source	CLock	
Control Signal Display/En/Disable	CONtrol	[[Berr] [Intr] [BR] {Enable|Disable}]
Set Handshaking Code	HANdshake	[code]
Command Help	Help	[GRoup {Emulation|Memory|Port|Setup |Trace}|command|ALl]
Identify Emulator	IDentify	
Timer Interval Switch	INTerval	[On|OFf]
Memory Mapping	MAp	[space-adr adr {{I|IR} {1|2|3|4}|E|ER|G} |ALl {E|G}]
Change Command Prompt	Prompt	[string]
Ready Signal Selection	REAdy	[Internal|External]
Recall Status from NOVRAM	RECall	
Change Operational Mode	ROute	[System]

Setup Commands Group (continued)

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Save Status to NOVRAM	SAve	
Select Leading Code	SELect	[{Route|DEVice} [code]]
Display Setup Parameters	SETup	
Memory Access Size	SIze	[Byte|Word]
Memory Verification Switch	Verify	[On|OFf]
Insert Wait State	WAit	[On|OFf]

2) Memory Commands

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Line Assembly	Assemble	[space-adr]
Memory Modify	Byte	[space-adr [data]]
Memory Checksum	CHecksum	space-adr {adr|Length length}
Memory Compare	COMpare	space-adr1 {adr|Length length} space-adr2
Memory Copy	COpy	space-adr1 {adr|Length length} [space-adr2]
Disassembly	Disassemble	[space-adr [adr|Length length]]
Memory Fill	Fill	space-adr {adr|Length length} data
Memory Modify	LOng	[space-adr [data]]
Memory Dump	Memory	[space-adr [adr|Length length]]
Memory Search	SEarch	space-adr {adr|Length length} data
Memory Test	TEst	space-adr {adr|Length length}
Memory Modify	Word	[space-adr [data]]

3) Port Commands

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Port Input	Input	space-adr [Length in-length]
Port Output	Output	space-adr data

4) Emulation Commands

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Cycle Step	Cycle	[Wait|count]
Jump	Jump	adr
Register Display/Modify	Register	[register]
Reset Emulation Processor	RESet	[pc [ssp]]
Instruction Step	Step	[adr1 adr2 [CALl]|count|CALl]

5) Trace Commands

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Display Events	Ev[ent]	
Set Bus Event 1	Ev[ent]1	adx [datum] [status] [COunt count]
Set Bus Event 2	Ev[ent]2	adx [datum] [status]
Set External Trigger	Ev[ent]3	{High|Low}
Set Execution Event 1	Ev[ent]4	adr [SP|UP]*
Set Execution Event 1	Ev[ent]4	adr [SP|UP] [Vector-Base value]**
Set Execution Event 2	Ev[ent]5	adr [SP|UP]*
Set Execution Event 2	Ev[ent]5	adr [SP|UP] [Vector-Base value]**
Set Execution Event 3	Ev[ent]6	adr [SP|UP]*
Set Execution Event 3	Ev[ent]6	adr [SP|UP] [Vector-Base value]**
Clear Event	Ev[ent]	[[1|2|3|4|5|6] CLear]
Execute User Program	Go	[Run] [adr]
Halt Emulation Processor	HAlt	
List Trace Buffer	List	[frame [adr1 adr2] [status] [trace-bits]]
List with Source Code	List	{Source|Instruction} [frame]
List Trace Frame Total	List	Number
Display Trace Cycle Qualifier	Qualify	adx [status]
Display/Clear Qualifier	Qualify	[CLear]
Display Sync Setting	SYnc	
Set Sync In/Out	SYnc	[{Input|OUtput} {On|OFf}]
Timebase Selection	TImebase	[Go|Event] [1|10|100|1000|10000]
Display Trace Parameters	TRAce	
Set "Then" Trigger	Trigger	[Run] # [Then # [Then #]] [BACKward |CENter|FOrward|DELay count]
Set "And" Trigger	Trigger	[Run] # And # [And #] [BACKward|CENter |FOrward|DELay count]
Set "Or" Trigger	Trigger	[Run] # Or # [Or #] [BACKward|CENter |FOrward|DELay count]
Set "And-Then" Trigger	Trigger	[Run] # And # Then # [BACKward|CENter |FOrward|DELay count]
Set "Or-Then" Trigger	Trigger	[Run] # Or # Then # [BACKward|CENter |FOrward|DELay count]
Display/Clear Trigger	Trigger	[CLear]

* Applies to 68000 CPU only.
** Applies to 68010 CPU only.

Chapter 5

MICE-16 68000 COMMAND DESCRIPTION AND EXAMPLES

This chapter describes in detail each of the command syntax used to operate the MICE-16 68000 in Terminal Mode. One or more typical examples are provided for each command description.

For greater convenience in interpreting the symbols and notations used in each command syntax and in the given examples, user should first become familiar with the Conventions of Notation (Section 4.1) used in this manual.

The command syntax are listed in the following functional groups, with the command keywords in each group presented in alphabetical order.

- Setup Commands (Section 5.3)
- Memory Commands (Section 5.4)
- Port Commands (Section 5.5)
- Emulation Commands (Section 5.6)
- Trace Commands (Section 5.7)

5.1 MULTIPLE MODES OF OPERATION SETUP

When arranged in Multi-User Computer Configuration (Figure 5-1), MICE-16 68000 may also operate in System and Debug Modes. These are in addition to the Terminal Mode available under this and other configurations discussed in Section 1.3. Details of all of these modes are as follows:

1. **Terminal Mode**

   In this mode, MICE-16 68000 works the same way as if it were connected to a terminal as described in Section 1.3.1

2. **System Mode**

   In this mode, MICE acts as a transparent buffer between terminal and host computer where user can perform all functions expected from the host computer, such as editing, compiling source files, etc., while completely ignoring the presence of MICE.

3. **Debug Mode**

   Input from the terminal is received by Channel A of the MICE serial port and is passed to the host computer (through Channel B of the same serial port) for translation. The translated data is then dispatched to MICE through the same channel for execution. Similarly, output from MICE is passed to the host computer through the same path where it is transferred back to terminal for display.

With a user developed driver or Microtek Universal Symbolic Debugger (USD), this mode offers the optimum functions that MICE could make available to user. This could include such sophisticated functions as software development, symbolic debugging, logic state analysis, software performance analysis, etc.

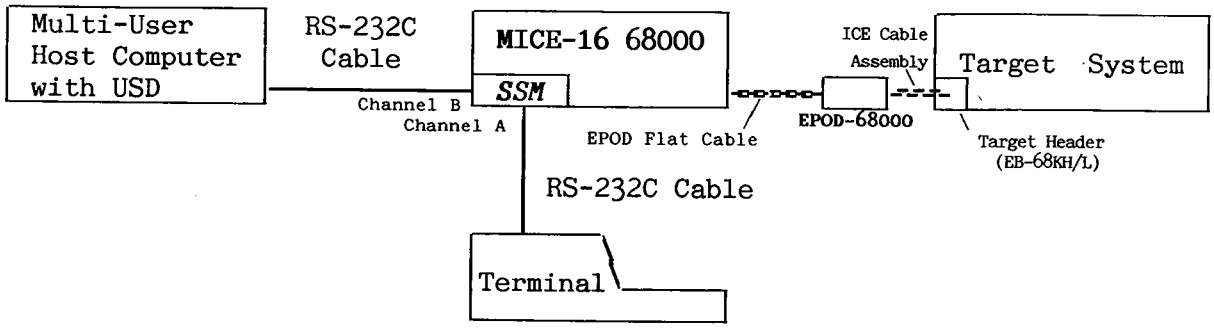


Figure 5-1 Configuring the MICE-16 68000 with Both Terminal and Multi-User Host Computer

When utilizing the above configuration, user should assign baud-rate, parity and data-bits for Serial Channel B through the BAud command described in Section 5.3.1.

Access to and from each of the above mentioned modes could be done any time and are accomplished easily by simply keying ROute and SELect commands described in Sections 5.3.12 and 5.3.14 respectively. Description on operational mode changes are also explained in detail in Section 5.1.14. With this configuration, only one serial port at multi-user host computer will be dedicated to MICE, leaving the other ports free to serve other users.

5.2 TRACING AND FREE-RUN EMULATION

Tracing and free-run emulation are in realtime. Information monitored during tracing and emulation are displayed in the trace buffer listing and at the Cycle and Step commands under the column headings defined in Table 5-2.

The kinds of processor activities which are displayed under STATUS column and which can be specified as conditions for trace-buffer recording/listing and bus breakpoints 1 and 2, are shown in Table 5-3 below. The two-letter codes at the left column of the table, can be used to specify that only machine cycles with certain types of processor activity are to be recorded in the trace buffer and displayed in the trace buffer listing. They can also be used in the Event 1 and Event 2 commands so that a breakpoint is encountered at the specified address only when the indicated type of processor activity is matched.

Note that there is no limitation on the number of status types that may be specified. If no status is specified, the default is for all machine cycles.

COLUMN HEADING	DEFINITION
FRAME	is the sequential position of the displayed execution cycle; the range is 0-7FFH (0-2047).
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.
TRACE BITS	indicates signal levels input from miniprobes connected to the external signal input port.
TIMER	is elapsed time from trace/timer/cycle start (depending on command specification).

Table 5-2 Information Recorded in the Trace Buffer

SYMBOL	PROCESSOR STATUS
SP	Supervisor Program Memory Access
SR	Supervisor Data Memory Read
SW	Supervisor Data Memory Write
UP	User Program Memory Access
UR	User Data Memory Read
UW	User Data Memory Write
AK	CPU Space Read (e.g. interrupt acknowledge or CPU read from coprocessor)

Table 5-3 Processor Status Codes

5.3 SETUP COMMANDS

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Port B Baud Rate Setting	BAud	[baud-rate [parity] [data-bits]]
Display Clock Source	CLock	
Control Signal Display/En/Disable	CONtrol	[[Berr] [Intr] [BR] {Enable|Disable}]
Set Handshaking Code	HANdshake	[code]
Command Help	Help	[GRoup {Emulation|Memory|Port|Setup |Trace}|command|ALl]
Identify Emulator	IDentify	
Timer Interval Switch	INTerval	[On|OFf]
Memory Mapping	MAp	[space-adr adr {{I|IR} {1|2|3|4}|E|ER|G] |ALl {E|G}]
Change Command Prompt	Prompt	[string]
Ready Signal Selection	REAdy	[Internal|External]
Recall Status from NOVRAM	RECall	
Change Operational Mode	ROute	[System]
Save Status to NOVRAM	SAve	
Select Leading Code	SELect	[{Route|DEVice} [code]]
Display Setup Parameters	SETup	
Memory Access Size	SIze	[Byte|Word]
Memory Verification Switch	Verify	[On|OFf]
Insert Wait State	WAit	[On|OFf]

Table 5-4 Setup Group Full Command Syntax Summary

The Setup Commands provide the configuration required to initialize the MICE-16 68000 before emulation or debugging could start for a specific target system, such as determining sources for clock, ready signal and control signal.

5.3.1 PORT B BAUD RATE SETTING - BAud

BAud [baud-rate [parity] [data-bits]]

BAud is the command to use to assign baud rate, parity, data bits of Serial Channel B. "BAud" alone without any parameter will display the last assigned settings.

baud-rate is baud rate of Serial Channel B (110 150 300 600 1200 2400 4800 9600 19200).

parity is E (even), O(odd) or N (none) parity of serial channel B.

data-bits is either 7 or 8 data bits of Serial Channel B.

The factory set defaults for baud rate, data-bits and parity are explained in Section 3.3.

Example:
```
>BAud 9600,E,7<CR>
 Serial channel B baud rate = 9600 , 7, E
>BAud 4800,8,0<CR>
 Serial channel B baud rate = 4800 , 8, 0
>
```

5.3.2 DISPLAY CLOCK SOURCE - CLock

CLock

CLock is the command to display current clock source.

MICE automatically sources external clock from the target if it is connected to a pre-powered target during MICE power-up. If MICE is powered up without a target or the connected target has no power, the internal clock (12 MHz) will default.

Example: Display current clock source.

```
>CLock<CR>
 Internal clock
>
```

5.3.3 DISPLAY/ENABLE/DISABLE CONTROL SIGNALS - CONtrol

CONtrol [[Berr] [Intr] [BR] {Enable|Disable}]

CONtrol is the command to Display/Enable/Disable Control Signals. "CONtrol <CR>" without any parameters displays the current status for all the control signals listed below.

Berr,Intr,BR, specifies a control signal to the emulation processor.

Enable|Disable sets the status of the specified control signal to "enabled" or "disabled." If only Enable|Disable is input after the Control command, all three affected control signals are enabled or disabled.

This command permits specific control signals going to the emulation processor to be enabled or disabled at the corresponding CPU input pins. Only input to the processor is affected; the setting is transparent to circuits which generate the control signals in the target system.

NOTE

When the emulation processor is reset (Section 5.6.4), the control signals previous to "RESet" command execution are retained if the MICE is connected to a power-applied target system, otherwise, these signals are disabled.

Table 5-5 explains the control signal parameters used with the Control command. Pin numbers are for a Dual-in-Line (DIP) device.

Parameter	Signal Type	Name	Pin No.
Berr	Bus Error	$\overline{BERR}$	22
Intr	Interrupt Request* (including non-maskable interrupt)	$\overline{IPL0}$ $\overline{IPL1}$ $\overline{IPL2}$	25 24 23
BR	Bus Request	$\overline{BR}$	13

* Disabling the interrupt request signal will not affect the interrupt mask list in the processor status register.

Table 5-5 Parameters for Control Signal Enable/Disable

Example 1: Display control signal status, then activate the bus error and interrupt request control signals.

```
>CONtrol<CR>
 BERR INTR (IPL0-2) BR disabled
>CONtrol Berr Enable<CR>
 BERR enabled
 INTR (IPL0-2) BR disabled
>CONtrol Intr Enable<CR>
 BERR INTR (IPL0-2) enabled
 BR disabled
>
```

Example 2: Deactivate the interrupt request control signal so that the EP will not process any interrupts which may occur. Note that this command does not affect the interrupt mask list in the processor status register.

```
>CONtrol Intr Disable<CR>
 BERR enabled
 INTR (IPL0-2) BR disabled
>
```

Example 3: First disable, then enable, all selectable control signals.

```
>CONtrol Disable<CR>
 BERR INTR (IPL0-2) BR disabled
>CONtrol Enable<CR>
 BERR INTR (IPL0-2) BR enabled
```

5.3.4 SET HANDSHAKING CODE - HANdshake

HANdshake [code]

HANdshake is the command for displaying or changing the handshaking code. Inputting "HANdshake<CR>" alone, without any parameters, will cause the current handshaking code to be displayed.

code is a one-byte hexadecimal value indicating the handshaking code to be sent to the terminal or host computer when the MICE-16 68000 is ready to accept input. For example, when using the Universal Symbolic Debugger (USD) software tool, the handshaking code is 03H.

Example: Display the current handshaking code and change it to 3H (ETX).

```
>HANdshake<CR>
 Handshaking code is FF    ;Handshake code is FF
>HANdshake 3<CR>
>HANdshake<CR>
 Handshaking code is 03    ;Handshake code is 03
>
```

5.3.5 COMMAND HELP - Help

Help [GRoup {Emulation|Memory|Port|Setup|Trace}|command|ALl]

Help is the keyword for Command Help. Inputting "Help" alone, without any parameter, causes the MICE to display a list of command keywords arranged in group according to its functions.

GRoup is the prefix to use to display a specific functional group of commands. When followed by the desired group's name, the syntax for all the commands in that particular group will display. Definitions of typical parameters used in such group, will also appear on screen. MICE commands are divided into the following groups:

Setup Commands	(see Section 5.3)
Memory Commands	(see Section 5.4)
Port Commands	(see Section 5.5)
Emulation Commands	(see Section 5.6)
Trace Commands	(see Section 5.7)

command is any of the specific MICE command keywords. Inputting "Help" followed by a specific command keyword, will display the full syntax for that particular command and the definitions for the typical parameters used by such command.

ALl is the parameter to display full syntax and typical parameter definitions for all available MICE-16 68000 commands.

In order to make efficient use of screen space, special conventions of notation are used in Command Help displays. These conventions are explained in Section 4.1.

Example 1: Display a command overview.

```
>Help<CR>
  Setup Group:
   BAud              IDentify          RECall            SIze
   CLock             INTerval          ROUte             Verify
   CONtrol           MAp               SAve              WAit
   HANdshake         Prompt            SELect
   Help              REAdy             SETup
  Memory Group:
   Assemble          COMpare           Fill              SEarch
   Byte              COpy              LOng              TEst
   CHecksum          Disassemble       Memory            Word
  Port Group:
   Input             Output
  Emulation Group:
   Cycle             Register          RESet             Step
   Jump
  Trace Group:
   Event             List              SYnc              TRAce
   Go                Qualify           Timebase          Trigger
   HAlt
>
```

Example 2: Display full syntax for all MICE-16 68000 (with 68010 CPU) commands. (When the message "**[More]**" appears, press any key to scroll for the next screen.)

```
>Help AL1<CR>
 Line Assembly              Assemble [space-adr]
 Port B Baud rate Setting   BAud [baud-rate [parity] [data-bits]]
 Memory Modify              Byte [space-adr [data]]
 Memory Checksum            CHecksum space-adr {adr|Length length}
 Display Clock Source       CLock
 Memory Compare             COMpare space-adr1 {adr|Length length} space-adr2
 Control Signal En/Disable  CONtrol [[Berr] [Intr] [BR] {Enable|Disable}]
 Memory Copy                COpy space-adr1 {adr|Length length} [space-adr2]
 Cycle Step                 Cycle [Wait|count]
 Disassembly                Disassemble [space-adr [adr|Length length]]
 Set Bus Event 1            Ev[ent ]1 adx [datum] [status] [COunt count]
 Display/Clear Event 1      Ev[ent ]1 [CLear]
 Set Bus Event 2            Ev[ent ]2 adx [datum] [status]
 Display/Clear Event 2      Ev[ent ]2 [CLear]
 Set External Trigger       Ev[ent ]3 {High|Low}
 Display/Clear Ext-trigger  Ev[ent ]3 [CLear]
[ More ]
 Set Execution Event 1      Ev[ent ]4 adr [SP|UP] [Vector-Based value]
 Display/Clear Exe-event 1  Ev[ent ]4 [CLear]
 Set Execution Event 2      Ev[ent ]5 adr [SP|UP] [Vector-Based value]
 Display/Clear Exe-event 2  Ev[ent ]5 [CLear]
 Set Execution Event 3      Ev[ent ]6 adr [SP|UP] [Vector-Based value]
 Display/Clear Exe-event 3  Ev[ent ]6 [CLear]
 Memory Fill                Fill space-adr {adr|Length length} data
 Execution                  Go [Run] [adr]
 Halt Emulator              HAlt
 Set Handshaking Code       HANdshake [code]
 Command Help               Help [GRoup {Emulation|Memory|Port|Setup|Trace}
                                 |command|AL1]
 Identify Emulator          IDentify
 Port Input                 Input space-adr [Length in-length]
 Timer Interval Switch      INTerval [On|OFf]
[ More ]
 Jump                       Jump adr
 List Trace                 List [frame [adr1 adr2] [status] [trace-bits]]
 List Source Code           List {Source|Instruction} [frame]
 List Frame Total           List Number
 Memory Modify              LOng [space-adr [data]]
 Memory Map                 MAp [space-adr adr {{I|IR} {1|2|3|4|}|E|ER|G}
                                |AL1 {E|G}]
 Memory Dump                Memory [space-adr [adr|Length length]]
 Port Output                Output space-adr data
 Change Prompt              Prompt [string]
 Set Trace Cycle Qualifier  Qualify adx [status]
 Display/Clear Qualifier    Qualify [CLear]
 Ready Signal Selection     REAdy [Internal|External]
 Recall Status from NOVRAM  RECall
 Register Display/Modify    Register [register]
 Reset Emulation Processor  RESet [pc [ssp]]
[ More ]
```

```
 Change Operational Mode    ROute [System]
 Save Status to NOVRAM      SAve
 Memory Search              SEarch space-adr {adr|Length length} data
 Select leading code        SELect [{Route|DEVice} [code]]
 Display Setup Parameters   SETup
 Memory Access Size         SIze [Byte|Word]
 Instruction Step           Step [adr1 adr2 [CALl]|count|CALl]
 Synchronization            SYnc [{Input|OUtput} {On|OFf}]
 Memory Test                TEst space-adr {adr|Length length}
 Timebase Selection         TImebase [Go|Event] [1|10|100|1000|10000]
 Display Trace Parameters   TRAce
 Set "Then" Trigger         Trigger [Run] # [Then # [Then #]] [BACKward
                                    |CENter|FOrward|DELay count]
 Set "And" Trigger          Trigger [Run] # And # [And #] [BACKward
                                    |CENter|FOrward|DELay count]
 Set "Or" Trigger           Trigger [Run] # Or # [Or #] [BACKward
                                    |CENter|FOrward|DELay count]
[ More ]
 Set "And-Then" Trigger     Trigger [Run] # And # [Then #] [BACKward
                                    |CENter|FOrward|DELay count]
 Set "Or-Then" Trigger      Trigger [Run] # Or # [Then #] [BACKward
                                    |CENter|FOrward|DELay count]
 Display/Clear Trigger      Trigger [CLear]
 Memory Verification        Verify [On|OFf]
 Insert Wait State          WAit [On|OFf]
 Memory Modify              Word [space-adr [data]]

 --- Definition ----------------------------------------------------------
 #           is EV1, EV2 or EV3 (do not specify same event twice in one
             command).
 adr         is a hexadecimal address setting.
 adx         is a hexadecimal address setting, with option to include
             wildcard nibbles/bits, e.g. 1234, 1XX4, 1(X1X0)X4, 1(X1X0X0XX)4.
[ More ]
 ALl         is the parameter to display full syntax for all available
             commands.
 BACKward    is the delay count = 0.
 baud-rate   is baud rate of serial channel B (110 150 300 600 1200 2400 4800
             9600 19200).
 CALl        is the keyword for skipping subroutines.
 CENter      is the delay count = 3FFH.
 code        is a one-byte hexadecimal value.
 command     is a specific MICE command.
 count       is a hexadecimal value (from 0H to 0FFFFH).
 data        is 1 to 32 bytes of data in hex and/or ASCII.
 data-bits   is either 7 or 8 data bits of serial channel B.
 datum       is a byte, word or long-word in hexadecimal, with wildcard
             nibbles/bits option, e.g. 1234, 1XX4, 1(X1X0)X4, 1(X1X0X0XX)4.
 FOrward     is the delay count = 7FFH.
 frame       is a hexadecimal value (from 0H, to 07FFH) indicating the frame
[ More ]
             at which the listing is to begin.
 in-length   is a hexadecimal value (from 01H to 0FFFFFFH).
```

```
 Instruction is the parameter to list the trace buffer containing frame,
             address, data and assembly source-code, but do not include
             status, trace-bits, timer and read/write cycle.
 length      is a hexadecimal value (from 01H to 0FFFFFFH) specifying the
             number of location to access.
 parity      is E (even), O (odd) or N (none) parity of serial channel B.
 pc          is a hexadecimal address to load in place of the current PC.
 register    is the standard Motorola-68000 CPU designation for the register,
             the contents of which are to be displayed or modified.
 Source      is the parameter to list the trace buffer containing frame,
             address, data, status, trace-bits, timer and assembly
             source-code.
 space-adr   is a hexadecimal address setting, with option to specify
[ More ]
             memory type: "[UP|UD|SP|SD] adr".
 ssp         is a hexadecimal address to load in place of the current SSP.
 status      is 1 to 7 processor status codes (SP SR SW UP UR UW AK).
 string      is a 1-8 character command prompt, input in ASCII.
 System      is the path of operational mode from Terminal to System mode.
 trace-bits  is a one-byte hexadecimal setting indicating signal levels for
             trace bits input from the external port. Includes option to
             specify wildcard nibbles/bits.
 value       is a hexadecimal value of vector base register.
 Wait        stops the CPU in a wait state.

 Note: Uppercase character(s) in the command keyword and command parameters
       represent the minimum abbreviated (shorthand) input that would remain
       transparent to MICE. Lowercase character(s) are optional. A space
       should be entered between a command keyword and a subsequent parameter,
       as well as between parameters.
[ More ]
>
```

Example 3: Display a single command syntax.

```
>Help BAud<CR>
 Port B Baud rate Setting   BAud [baud-rate [parity] [data-bits]]

 --- Definition ---------------------------------------------------------
 baud-rate   is baud rate of serial channel B (110 150 300 600 1200 2400 4800
             9600 19200).
 data-bits   is either 7 or 8 data bits of serial channel B.
 parity      is E (even), O (odd) or N (none) parity of serial channel B.
```

Example 4: Display the "Setup" group command syntax.

```
>Help GRoup Setup<CR>
 Port B Baud rate Setting   BAud [baud-rate [parity] [data-bits]]
 Display Clock Source       CLock
 Control Signal En/Disable  CONtrol [[Berr] [Intr] [BR] {Enable|Disable}]
 Set Handshaking Code       HANdshake [code]
 Command Help               Help [GRoup {Emulate|Memory|Port|Setup|Trace}
                                 |command|ALl]
 Identify Emulator          IDentify
 Timer Interval Switch      INTerval [On|OFf]
 Memory Mapping             MAp [space-adr adr {{I|IR} {1|2|3|4|}|E|ER|G}
                                |ALl {E|G}]
 Change Prompt              Prompt string
 Ready Signal Selection     REAdy [Internal|External]
 Recall Status from NOVRAM  RECall
 Change Operational Mode    ROute [System]
 Save Status to NOVRAM      SAve
 Select leading code        SELect [{Route|DEVice} [code]]
[ More ]
 Display Setup Parameters   SETup
 Memory Access Size         SIze [Byte|Word]
 Memory Verification        Verify [On|OFf]
 Insert Wait State          WAit [On|OFf]

 --- Definition ----------------------------------------------------------
 ALl         is the parameter to display full syntax for all available
             commands.
 baud-rate   is baud rate of serial channel B (110 150 300 600 1200 2400 4800
             9600 19200).
 code        is a one-byte hexadecimal value.
 command     is a specific MICE command.

 data-bits   is either 7 or 8 data bits of serial channel B.
 parity      is E (even), O (odd) or N (none) parity of serial channel B.
 string      is a 1-8 character command prompt, input in ASCII.
 System      is the path of operational mode from Terminal to System mode.
[ More ]
```

5.3.6 IDENTIFY EMULATOR TYPE -IDentify

IDentify

IDentify is the command to identify the emulator model and version number currently being used. The MICE model name and EPM firmware revision number will display on the screen.

Example: Display the type of emulator currently being used.

```
>IDentify<CR>
 *** MICE-16 68000, Firmware V#.# ***
>
```

5.3.7 EXECUTION INTERVAL DISPLAY SETTING - INTerval

INTerval [On|OFf]

INTerval is the command to display, or to change, the execution interval display setting. This determines whether the frames recorded in the trace buffer (Section 5.7.4) are listed showing the execution interval between cycles or the overall execution time. Inputting "INTerval<CR>" alone, without specifying any parameters, causes the current setting to be displayed.

On|OFf switches the display between "cycle-to-cycle interval" and "overall execution time" for each trace frame displayed at the List command. For further information concerning timer initialization and format of timer messages, refer to Section 5.7.7.

Example: Show the current status for the List display and then change it.

```
>INTerval<CR>
 Time interval OFF in trace buffer timer display
>INTerval On<CR>
>INTerval<CR>
 Time interval ON in trace buffer timer display
>
```

5.3.8 MEMORY MAPPING - MAp

MAp [space-adr adr {{I|IR} {1|2|3|4}|E|ER|G}|AL1 {E|G}]

MAp is the command for Memory Mapping. Inputting "MA<CR>" alone, without specifying any parameters, causes the current memory map setting to be displayed.

space-adr is the setting indicating the starting point of the memory block to be mapped. It is a hexadecimal address in the memory space of the emulation CPU optionally preceded by specification type- "[SP|SD|UP|UD] adr",

where: **SP** is the supervisor program
SD is the supervisor data
UP is the user program
UD is the user data

If none of the memory types is specified, all four memory types will default. The address specified is automatically adjusted into 4K-byte* block boundary (0H,1000H, etc.), the lower 12 bits are set 0H.

adr is a hexadecimal address indicating the end point of the memory block to be mapped. As in adr, any specified address will also be automatically adjusted into the end of the 4K-byte block boundary value (0FFFH, 1FFFH, etc.).

I|IR|E|ER|G indicates the attribute for the mapped memory block. Possible attributes are:

I	Internal (on the EMM), writable
IR	Internal (on the EMM), read-only
E	External (on the target), writable
ER	External (on the target), read-only
G	Guarded memory, non-existence

1|2|3|4 indicates the four memory banks in the EMM , with 64K-byte each (256K-byte each with HEMM**).

When mapping the memory to the EMM, the maximum range between space-adr and adr is 64K-byte, hence, the mapped memory are located within the same 64K-byte boundary.

When mapping the memory to the HEMM*, the maximum range between space-adr and adr is 256K-byte, hence, the mapped memory are located within the same 256K-byte boundary.

AL1 {E|G} is the option to map the whole memory space to **E** or **G** attribute.

---

* 4K-byte block for MICE-16 68000 with EPOD (F/W Versions 2.0 or later). Previous versions is 16k-byte block.

** The optional HEMM will be available with Firmware Version 3.0 to be released soon.

Up to 256K bytes of emulation memory can be mapped anywhere within 16M bytes and blocks within this designated as IR or I. Four independent 64K-byte memory banks are supported with 4K-byte block memory enable/disable and write protect capability. Any 4K-byte memory block can be disabled (i.e., mapped as external or guarded) or enabled depending on the amount of emulation memory needed. For EMM, each memory segment must be mapped to within the 64K-byte boundary (0, 10000, etc.) and all mapping addresses to the banks must be located within the same 64K-byte boundary. For HEMM*, it is within 256K-byte.

During execution of Instruction Step, Cycle or Go command the following conditions will cause MICE to display a warning message and force the emulation processor to stop:

1) When the guarded memory area is accessed (read, write, or fetch).
2) When writing on IR/ER read only area is attempted.

If an attempt is made to map a new address range or blocks as IR (read-only) or I (writable) to a bank on the EMM which is already set for another memory bank, the MICE will display an error message (see Example 3 below). The user should first map the address range or block to the target (external or guarded) and then map the desired block as IR or I.

All map contents can be saved to NOVRAM using SAve command (Section 5.3.13). However, NOVRAM can only accommodate no more than thirty sets of data (except for Guarded memory range, all are countable from MAp command display). Hence, any data saved beyond the thirty-set limit are "saved" in the MICE firmware as Guarded data.

Example 1: Map all 16M bytes memory to (G) Guarded memory attribute and display the result:

```
>MAp 0 FFFFFF G<CR>
>MAp<CR>
                        ADDRESS RANGE    ATTRIBUTES
 SUPERVISOR PROGRAM:  000000 FFFFFF    -
 USER PROGRAM       : 000000 FFFFFF    -
 SUPERVISOR DATA    : 000000 FFFFFF    -
 USER DATA          : 000000 FFFFFF    -

 EMULATION MEMORY ALLOCATION :
              BANK 1:  (UNUSED)
              BANK 2:  (UNUSED)
              BANK 3:  (UNUSED)
              BANK 4:  (UNUSED)
>
```

* The optional HEMM will be available with Firmware Version 3.0 to be released soon.

Example 2: Map the memory location 8000H TO BFFFH to Bank 1 of EMM, specifying IR (Read-only) as attribute and SP (supervisor program) as memory type, and location 10000H to 13FFFH to Bank 2 of EMM under SD (Supervisor Data) memory type.

```
>MAp SP 8000 BFFF IR 1<CR>
>MAp SD 10000 13FFF I 2<CR>
>
```

Example 3: Map memory location FF0000H to FFFFFFH to Bank 1 of EMM which is already mapped with other address range.

```
>MAp FF0000 FFFFFF I 1<CR>
 Error ! Bank already mapped by another address range
>
```

Example 4: Map the memory location FF0000H FFFFFFH to Bank 3 of EMM, specifying IR (read-only) as attribute and both SP (Supervisor Program) and SD (supervisor data) as memory types. Then map 30000H to 5FFFFF to External and display the result.

```
>MAp SP SD FF0000 FFFFFF IR 3<CR>
>MAp 30000 5FFFF E<CR>
>MAp<CR>
                          ADDRESS RANGE    ATTRIBUTES
SUPERVISOR PROGRAM:       000000 007FFF    -
                          008000 00BFFF    IR  (BANK 1)
                          00C000 02FFFF    -
                          030000 05FFFF    E
                          060000 FEFFFF    -
                          FF0000 FFFFFF    IR  (BANK 3)
USER PROGRAM         :    000000 02FFFF    -
                          030000 05FFFF    E
                          060000 FFFFFF    -
SUPERVISOR DATA      :    000000 00FFFF    -
                          010000 013FFF    I   (BANK 2)
                          014000 02FFFF    -
                          030000 05FFFF    E
                          060000 FEFFFF    -
                          FF0000 FFFFFF    IR  (BANK 3)
USER DATA            :    000000 02FFFF    -
                          030000 05FFFF    E
                          060000 FFFFFF    -

EMULATION MEMORY ALLOCATION :
              BANK 1:  000000 - 00FFFF
              BANK 2:  010000 - 01FFFF
              BANK 3:  FF0000 - FFFFFF
              BANK 4:  (UNUSED)
>
```

Example 5: Using Figure 4-5 below, demonstrate a mapping configuration where the memory types (SP, SD, UP and UD) are common to all 16M-bytes memory range.

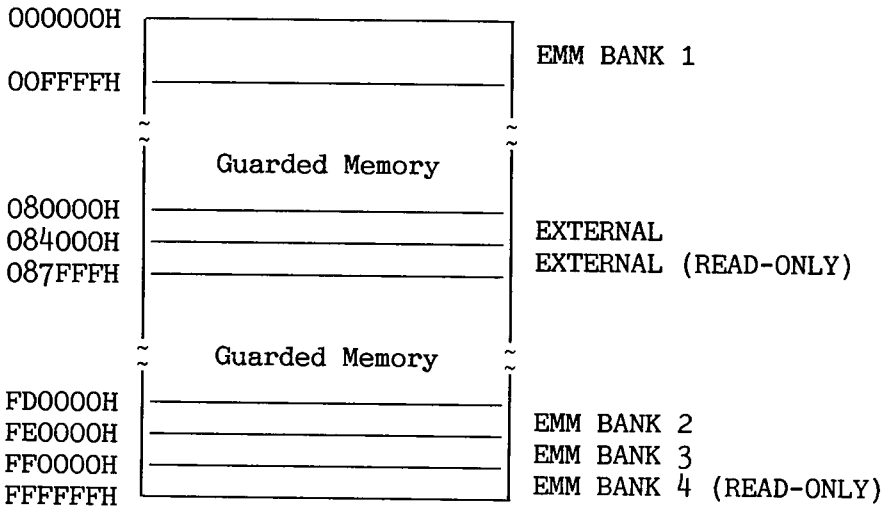


Figure 5-6 Memory Maps Structure With Common Memory Types

```
>MAp 0 FFFFFF G<CR>
>MAp 0 FFFF I 1<CR>
>MAp 80000 83FFF E<CR>
>MAp 84000 87FFF ER<CR>
>MAp FD0000 FDFFFF I 2<CR>
>MAp FE0000 FEFFFF I 3<CR>
>MAp FF0000 FFFFFF IR 4<CR>
>MAp<CR>
                        ADDRESS RANGE    ATTRIBUTES
 SUPERVISOR PROGRAM:    000000 00FFFF    I    (BANK 1)
                        010000 07FFFF    -
                        080000 083FFF    E
                        084000 087FFF    ER
                        084000 FCFFFF    -
                        FD0000 FDFFFF    I    (BANK 2)
                        FE0000 FEFFFF    I    (BANK 3)
                        FF0000 FFFFFF    IR   (BANK 4)
 USER PROGRAM       :   000000 00FFFF    I    (BANK 1)
                        010000 07FFFF    -
                        080000 083FFF    E
                        084000 087FFF    ER
                        084000 FCFFFF    -
                        FD0000 FDFFFF    I    (BANK 2)
                        FE0000 FEFFFF    I    (BANK 3)
                        FF0000 FFFFFF    IR   (BANK 4)
```

```
SUPERVISOR DATA    :  000000 00FFFF    I    (BANK 1)
                      010000 07FFFF    -
                      080000 083FFF    E
                      084000 087FFF    ER
                      084000 FCFFFF    -
                      FD0000 FDFFFF    I    (BANK 2)
                      FE0000 FEFFFF    I    (BANK 3)
                      FF0000 FFFFFF    IR   (BANK 4)
USER DATA          :  000000 00FFFF    I    (BANK 1)
                      010000 07FFFF    -
                      080000 083FFF    E
                      084000 087FFF    ER
                      084000 FCFFFF    -
                      FD0000 FDFFFF    I    (BANK 2)
                      FE0000 FEFFFF    I    (BANK 3)
                      FF0000 FFFFFF    IR   (BANK 4)

EMULATION MEMORY ALLOCATION :
             BANK 1:  000000 - 00FFFF
             BANK 2:  FD0000 - FDFFFF
             BANK 3:  FE0000 - FEFFFF
             BANK 4:  FF0000 - FFFFFF
>
```

Example 6: Using Figure 4-6 below, demonstrate a mapping configuration where the memory types (SP, SD, UP and UD) are distinctively defined to each address range of the 16M-bytes memory range.

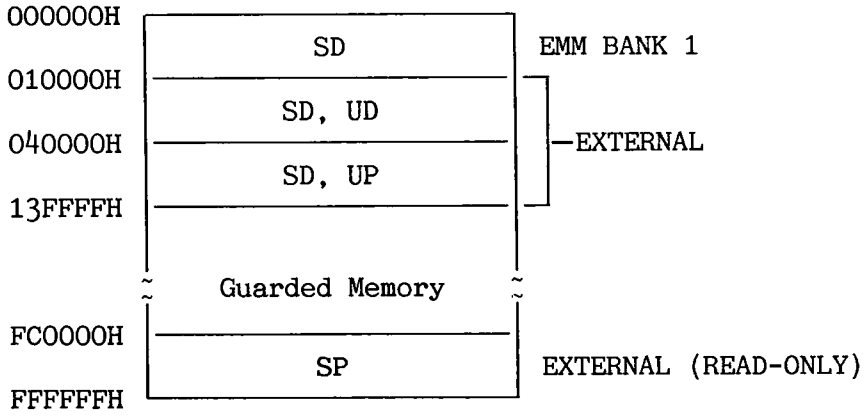


Figure 5-7 Memory Maps Structure With Distinctive Memory Types

```
>MAp 0 FFFFFF G<CR>
>MAp SD 0 FFFF I 1<CR>
>MAp SD UD 10000 3FFFF E<CR>
>MAp SD UP 40000 13FFFF E<CR>
>MAp SP FC0000 FFFFFF ER<CR>
>MAp<CR>
                          ADDRESS RANGE    ATTRIBUTES
 SUPERVISOR PROGRAM:      000000 FBFFFF    -
                          FC0000 FFFFFF    ER
 USER PROGRAM       :     000000 03FFFF    -
                          040000 13FFFF    E
                          140000 FFFFFF    -
 SUPERVISOR DATA    :     000000 00FFFF    I    (BANK 1)
                          010000 03FFFF    E
                          140000 FFFFFF    -
 USER DATA          :     000000 00FFFF    -
                          010000 03FFFF    E
                          040000 FFFFFF    -

 EMULATION MEMORY ALLOCATION :
              BANK 1:  000000 - 00FFFF
              BANK 2:  (UNUSED)
              BANK 3:  (UNUSED)
              BANK 4:  (UNUSED)
>
```

5.3.9 CHANGE COMMAND PROMPT - Prompt

Prompt [string]

Prompt is the command to change the the MICE command prompt.

string is a 1-8 character command prompt, input in ASCII. If no string is specified, prompt will revert to the basic prompt format.

Example: Change the current MICE command prompt to "68000>".

```
>Prompt 68000<CR>
68000>
```

5.3.10 EMULATION MEMORY READY SIGNAL SELECTION - REAdy

REAdy [Internal|External]

REAdy is the command for HEMM/EMM Ready Signal Selection.

Internal selects internally generated ($\overline{\text{DTACK}}$) ready signal.

External selects externally generated ($\overline{\text{DTACK}}$) ready signal.

Ready signal for HEMM/EMM memory ($\overline{\text{DTACK}}$ in the case of the 68000) can be generated internally (by the MICE) or externally (by the target).

Example: Display the current ready signal selection (with EMM or HEMM).

```
>REAdy<CR>
 Emulation memory Ready (DTACK*) signal generated internally
>
```

5.3.11 RECALL STATUS FROM NOVRAM - RECall

RECall

RECall is the command to recall the emulation memory mapping setting, timebase and other data currently saved in non-volatile RAM as the system defaults; and to check compatibility of the NOVRAM stored data with MICE Identification Code (ID). Note that this command is automatically executed on power-up and after a hardware reset.

NOTE

When the system operates in modes other than Terminal Mode (i.e., System or Debug Mode), all but the protocol data (Channel B baud rate, handshake code and Route /Device Leading Codes) are recalled from NOVRAM.

After the RECall execution, a list of the categories of the data (not the data itself) recalled from NOVRAM will be displayed. However, if the ID code does not match with the parameters stored in NOVRAM, there will be no further action from MICE except for displaying an error message (see example below). When this condition occurs, MICE setup is accomplished through the system default setup parameters.

Example: Recall status from NOVRAM.

```
>RECall<CR>
 Memory map, timebase, channel B baud rate, select code
 were recalled from NOVRAM
>
>RECall<CR>
 NOVRAM parameters inconsistent!          ;ID code does not match.
>
```

5.3.12 CHANGE OPERATIONAL MODE - ROute*

ROute [System]

ROute is the command for changing operational mode from Terminal to System Mode. "ROute" alone without parameter will display current status of serial Channel B.

System is the path of operational mode from Terminal to System Mode.

When MICE is setup in Multi-User Configuration described in Section 5.1, it can operate in Terminal, System and Debug Modes with USD as software tool. Changing from one mode of operation to another, requires a fixed cycle routing sequence. This fixed routing is illustrated in Figure 5-8 below.

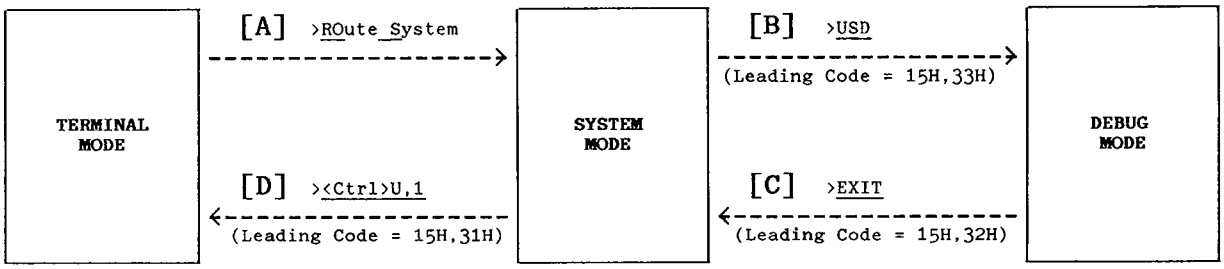


Figure 5-8 Operational Mode Change Cycle

Where:

[A] is the Terminal to System Mode sequence.

>ROute System<CR>

command is used for this operational mode change.

[B] is the System to Debug (USD) Mode sequence.
MICE must initially return to System Mode every time it changes operational mode from Terminal to Debug Mode and vise-versa. To switch into Debug Mode from System Mode, enter-

>USD<CR>

* ROute command is applicable only when a multi-user host computer is linked with Channel B of the MICE serial ports as explained in Section 5.1.

USD will then pass the required leading code and corresponding parameter (default is 15H,33H in HEX value) to Channel B of MICE to change into Debug Mode.

[C] is the Debug to System Mode sequence.
MICE must initially return to System Mode whenever it changes operational mode from Debug to Terminal Mode and vise-versa. To re-enter System Mode from Debug Mode, input-

>EXIT<CR>

USD will then pass the necessary leading code and corresponding parameter (default is 15H,32H in HEX value) to MICE through Channel B, to change into System Mode.

[D] is the System to Terminal Mode Sequence.
When changing from System Mode back to Terminal Mode, simply enter the Route leading code in ASCII-

><Ctrl>U,1

USD will then pass the required leading code and corresponding parameter (default is 15H,31H in HEX value or Control U,1 in ASCII) to Channel A of MICE to change into Terminal Mode. Note that the Route leading code is user programmable.

15H is the default Route leading code "introducer" preset in the factory and is user programmable.

33H is the built-in Route leading code parameter identifying System to Debug Mode change sequence.

32H is the built-in Route leading code parameter identifying Debug to System Mode change sequence.

31H is the built-in Route leading code parameter identifying System to Terminal Mode change sequence.

Example: Display current status of serial Channel B.

```
>ROute<CR>
 Route in terminal mode
 Serial channel B baud rate = 9600, 7, E
 Route leading code = 15
 Device leading code = 01
>
```

5.3.13 SAVE STATUS TO NOVRAM - SAve

SAve

SAve is the command to save current emulation memory mapping settings, timebase and other data to the non-volatile RAM; and also to subsequently check these parameters for compatibility with the MICE Identification Code (ID).

NOTE

Data saved in non-volatile RAM are not destroyed when power to the MICE-16 68000 is shut off. The data are automatically recalled and used as the system defaults when the MICE-16 68000 is subsequently powered up or when a hardware reset (warm or cold start) is executed.

After execution of SAve command, a list of the categories of the data (not the data itself) saved in the NOVRAM will be displayed. However, if the parameters saved in NOVRAM do not match with the Identification Code or error is detected with the saved data, an error message will display as illustrated in the following example.

Example: Save status to NOVRAM with matched and unmatched ID code.

```
>SAve<CR>
 Memory map, timebase, channel B baud rate, select code
 were saved to NOVRAM
>
>SAve<CR>
 CPM U5 NOVRAM failure!    ;ID code does not match or error detected.
>
```

5.3.14 SELECT LEADING CODE - SELect*

SELect [{Route|DEVice} [code]]

SELect is the command to use in selecting leading codes for either Route or Device. If the selected leading code already existed, a warning message will display. "SELect" alone without any parameter, will display leading codes status.

Route is the fixed path of operational mode change cycle, i.e. from System Mode, to Debug Mode, back to System Mode and then to Terminal Mode.

DEVice is the path of communication between MICE, the terminal and host computer devices.

code is a hexadecimal value representing each Route and Device path.

If the path of communication is between (from/to) **terminal and host computer** device, the data is followed by 01H 30H in Hex value when the Device leading code is 01H. If it is between **MICE and host computer** device, the data is followed by 01H 31H (also in Hex value with Device leading code of 01H).

The following diagrams illustrate communication between the above mentioned devices in Terminal and Debug (with Device leading code of 01H) Modes respectively with 'Register Display' as sample command being executed. Numbers enclosed in parenthesis indicate the sequence of communication.

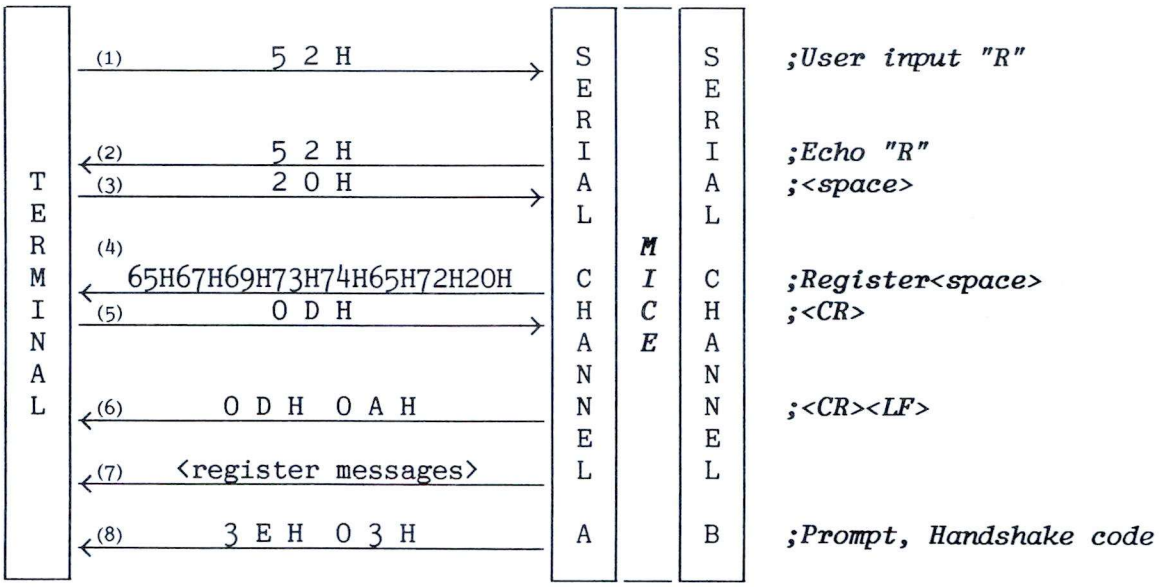


Figure 5-9 Path of Communication in Terminal Mode

* SELect command is applicable only when a multi-user host computer is linked with Channel B of the MICE serial ports as explained in Section 5.1.

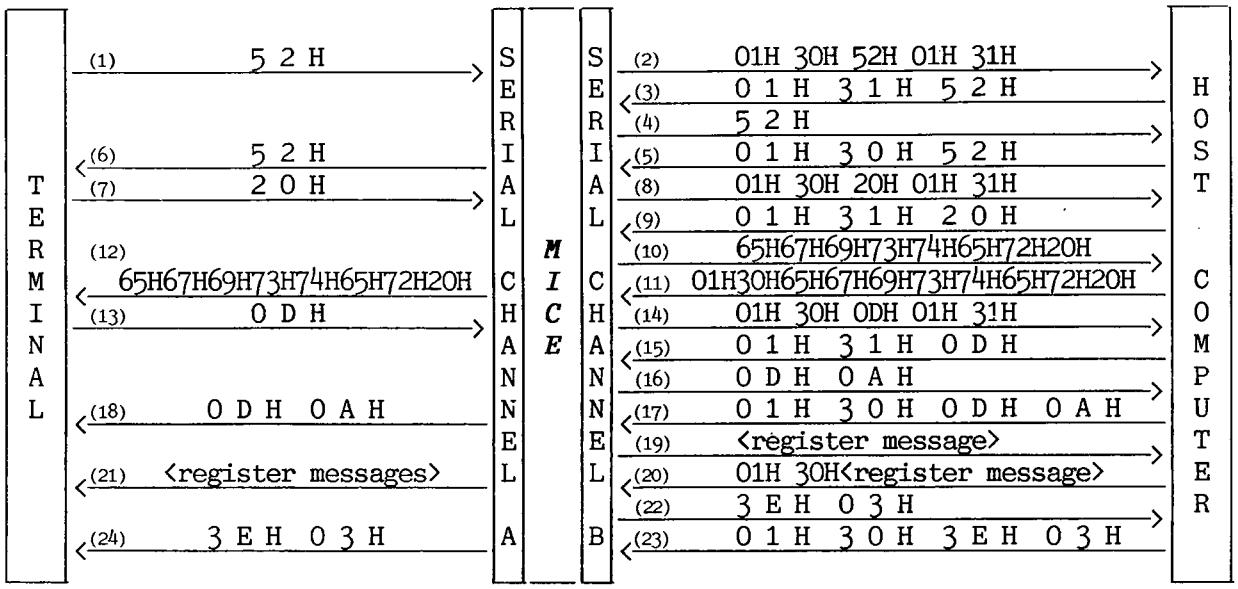


Figure 5-10 Path of Communication in Debug Mode

The user programmable Route and Device leading codes should be individually unique, not only from each other but also from the "Handshake" code (Section 5.3.4). If the leading code selected under "SELect" command, matches with the other two remaining codes, then a warning message (see Example 1 below) will display.

The factory set defaults for Route and Device leading codes which are recalled from NOVRAM on power on, are as shown in the Example 2 below.

Example 1: Select an existing leading code.

```
>SELect Route 1<CR>
 Warning: setting duplicated!
 Handshake code, route leading code and device leading code
 should be individually unique.
>
```

Example 2: Display leading code status.

```
>SELect Route 15<CR>
>SELect Device 1<CR>
>SELect<CR>
 Route leading code = 15
 Device leading code = 01
>
```

5.3.15 DISPLAY SETUP PARAMETERS - SETup

SETup

SETup is the command to display the MICE setup parameters as illustrated in the example below:

Example: Display the current MICE-16 68000 setup parameters.

```
>SETup<CR>
 Handshake code is 03
 BERR INTR (IPL0-2) BR disabled
 Internal clock
 Emulation Memory Ready (DTACK*) signal generated internally
 Time interval OFF in trace buffer timer display
 Memory access size is Word (Memory,COMpare,CHecksum,TEst,Input,Output,
 Fill, COpy, DOwnload, Upload)
 Memory verification ON (Assemble,Byte,Word,Long,Fill,Copy,Download)
 Wait OFF (0 Wait state insert)
 Route in terminal mode
 Serial channel B baud rate = 9600, 7, E
 Route leading code = 15
 Device leading code = 01
>
```

5.3.16 MEMORY ACCESS SIZE - SIze

SIze [Byte|Word]

SIze permits specification of data in byte or word format. This command determines data format for the Memory, COMpare, CHecksum, TEst, COpy, Input, Output DOwnload and Upload commands. "SIze<CR>" without any parameters displays the most recent data format selection for data format.

Byte|Word are the data format selection.

Example: Display the most recent selection for data format.

```
>SIze<CR>
 Memory access size is Word (Memory, COMpare, CHecksum, TEst, Input, Output,
 Fill, COpy, DOwnload, Upload)
>
```

5.3.17 MEMORY VERIFICATION SWITCH - Verify

Verify [On|OFf]

Verify is the command for Memory Verification. "Verify<CR>" without any parameters displays the current setting.

On|OFf switches verification for all memory write operations ON or OFF.

When Memory Verification is ON, the MICE verifies all memory write operations. If incorrect data are read back from any address, an error message will be displayed and the command will be terminated. If memory verification is turned OFF, the MICE will not verify any memory write operations.

Example: Turn memory verification off.

```
>Verify<CR>
 Memory verification ON (Assemble, Byte, Word, Long, Fill, Copy, Download)
>Verify OFf<CR>
>Verify<CR>
 Memory verification OFF (Assemble, Byte, Word, Long, Fill, Copy, Download)
>
```

5.3.18 INSERT WAIT STATE - WAit

WAit [On|OFf]

WAit is the command to display or change the current Insert Wait State setting.

On|OFf turns the Insert Wait State feature ON or OFF.

Targets with strict timing requirements may be highly sensitive to propagation delays caused when the emulation processor is used in place of the target CPU. To help improve system tolerance to such delays, the MICE-16 68000 can be set to insert a wait state after each machine cycle. Note that this setting has no effect if the target itself is designed to insert one or more wait states after each cycle. The Insert Wait State feature defaults to "OFF" when the MICE is powered up.

Example: Turn on insert wait state.

```
>WAit<CR>
 Wait OFF (0 Wait state insert)
>WAit On <CR>
>WAit<CR>
 Wait ON (1 Wait state insert)
>
```

5.4 MEMORY COMMANDS

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Line Assembly	Assemble	[space-adr]
Memory Modify	Byte	[space-adr [data]
Memory Checksum	CHecksum	space-adr {adr|Length length}
Memory Compare	COMpare	space-adr1 {adr|Length length} space-adr2
Memory Copy	COpy	space-adr1 {adr|Length length} [space-adr2]
Disassembly	Disassemble	[space-adr [adr|Length length]]
Memory Fill	Fill	space-adr {adr|Length length} data
Memory Modify	LOng	[space-adr [data]]
Memory Dump	Memory	[space-adr [adr|Length length]]
Memory Search	SEarch	space-adr {adr|Length length} data
Memory Test	TEst	space-adr {adr|Length length}
Memory Modify	Word	space-adr [data]]

Table 5-11 Memory Group Full Command Syntax Summary

The Memory Group commands provide access to the contents of designated memory locations. The MICE-16 68000 supports four memory types:

Supervisor Program (SP)
Supervisor Data (SD)
User Program (UP)
User Data (UD)

In executing all of the commands (Memory Group) described in this section, the memory type that user desired to access, should be specified. If memory type is not specified, the MICE will automatically treat the accessed memory as supervisor program (SP) space. The MICE-16 68000 also supports two basic data formats: Byte and Word. For the Memory Dump, CHecksum, TEst, and COpy commands, the desired format should be specified with the Memory Access Size command (Section 5.3.16). The total amount of addressable memory is 16M bytes.

Because the 68000 is word-oriented, even-numbered addresses must be assigned for the program counter (PC) in the Assemble and Disassemble commands (Sections 5.4.1 and 5.4.6 respectively) and for the user stack pointer (USP) and supervisor stack pointer (SSP). If an odd value is entered, the system will drop the least significant bit to modify it to a legal even value.

The MICE always makes sure that the memory and port addresses specified by the user are valid before performing an operation. Any reference to an invalid address will result in display of an error message and termination of the command. If Memory Verification (Section 5.3.17) is on, the MICE also verifies all memory write operations, and if data is not written correctly to the specified address, an error message will be displayed and the command terminated. The MICE, however, does not verify that there is memory or anything else connected to a port when performing a read operation.

Note that when the read/write functions are being performed, and if the external circuitry does not respond with the $\overline{\text{DTACK}}$ signal,

"No Target DTACK* at address xxxxxx"

will appear on the screen with the contents of memory remaining unchanged.

5.4.1 LINE ASSEMBLY - Assemble

Assemble [space-adr]

Assemble is the command for Line Assembly. "Assemble<CR>" alone, without any parameters, enables the line assembler beginning at the current PC.

space-adr specifies the memory location where the MICE is to begin storing the assembly language program input by the user. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP|UD] adr". If no memory type is specified, Supervisor Program (SP) is the memory type default for CPU in the supervisor state and User Program (UP) is the default for user state.

The MICE-16 68000 includes a resident assembler which will accept mnemonic input and convert it into machine code. Thus there is no need to enter programs or program changes in machine code using the Memory Modify command, nor to reassemble an entire program every time a change is made. The converted code is stored in the emulation processor's memory space starting at the specified starting address. The assembler does not recognize symbolic labels or any 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, the following three instructions are also supported during assembly:

DB - Define Byte (1-30 bytes in hex/ASCII/combined)
DW - Define Word (1-15 words in hex)
DS - Define Storage (0H-FFFFFFH in hex)

Hexadecimal, ASCII or a combination of both can be accepted for DB and DW, while DS are restricted to hexadecimal data. Remember that data in ASCII must be enclosed in single quotation marks (e.g. 'AR'. Note that the single quotation mark itself [''] may not be input in ASCII). If more than 30 bytes are keyed in for DB, or more than 15 words are input for DW, or a value greater than FFFFFFH is specified for DS, input is ignored and the MICE displays-

"Operand error!".

After the Assembly command is input, the following column headings appear:

```
>Assemble 400<CR>
LOCATION   OBJECT CODE      SOURCE CODE
000400                      *
   :
   :
```

The asterisk [*] indicates the current cursor position. The MICE-16 68000 then waits for assembly language line input from the user. Each line entered in the "SOURCE CODE" column consists of an op code and operand(s), and must be terminated with either <CR> or <Ctrl-J>. The MICE assembles the input line and stores the machine code beginning at the specified starting address. The code stored at the indicated program memory location is displayed in the "OBJECT CODE" column, and the MICE waits for input on the next line. If <ESC> is entered, the Line Assembly command ends.

When Memory Verification is ON, the MICE will verify any data written to memory. If an error is detected, it will display-

"Memory fill terminated -- write failure at xxxxxx"

If an invalid program instruction is entered, an error message-

"Operand error!" or "Mnemonic error!"

will display.

Example: Enter a simple program starting at location 8000H which get 10 words of data from 1000H to 101FH and add each word by 100H, then store these data to 2000H to 201FH

```
>MAp 0 FFFFFF G<CR>
>MAp 0 FFFF I 1<CR>
>Assemble 8000<CR>
LOC       OBJ              SOURCE CODE
008000    1A3C 0010        MOVE.B #10,D5
008004    207C 0000 1000   MOVEA.L #1000,A0
00800A    227C 0000 2000   MOVEA.L #2000,A1
008010    3018             MOVE.W (A0)+,D0
008012    6100 0FEC        BSR 9000
008016    32C1             MOVE.W D1,(A1)+
008018    5305             SUBQ.B #1,D5
00801A    66F4             BNE 8010
00801C    4E71             NOP
00801E    60FE             BRA $
008020                     <ESC>
>Assemble 9000<CR>
LOC       OBJ              SOURCE CODE
009000    0640 0100        ADDI.W #100,D0
009004    3200             MOVE.W D0,D1
009006    4E75             RTS
009008                     <ESC>
>
```

When making changes in an existing program, it is advisable to check the program code around the modified area using the Disassembly command (Section 5.4.6) before and after the change. This is important to ensure that any changes do not affect the surrounding program.

In MICE-16 68000 Assembler/Disassembler syntax instructions; Operand Format [value] represents the address content. If value is prefixed with a pound sign (#), it represents the given value only. Dollar sign ($) represents current PC.

Examples:

```
MOVE.W 100, D0<CR>      ;Move one word indicated by address 100, to D0
MOVE.W #100, D0<CR>     ;Move the value 100 to D0
BRA $<CR>               ;Branch itself (current PC)
```

5.4.2 MEMORY MODIFY - Byte|Word|LOng

```
Byte [space-adr [data]]
Word [space-adr [data]]
LOng [space-adr [data]]
```

Byte|Word|LOng indicates Memory Modify, and specifies the desired data format.

space-adr specifies the location where the Memory Modify operation is to begin. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP|UD] adr". If no memory type is specified, Supervisor Program (SP) will default.

data is a hexadecimal or ASCII string in 1 to 32-byte format. Data in ASCII must be enclosed in single quotation marks (e.g., 'AR'. Note that the single quotation marks itself ['] may not be entered in ASCII). The string may not be more than 32 bytes in length. Combined use of hex and ASCII is permitted. If the address range is smaller than the string, Memory Modify is still executed but the excess data are ignored.

The MICE displays the contents of the starting address and waits for further input. To change the value held in the location displayed, enter a new value in hexadecimal, then press <CR> to go to the next location in memory or <Ctrl-J> to go to the preceding location. Pressing <CR> or <Ctrl-J> alone, without first inputting a new value, leaves the contents of the location unchanged; pressing <ESC> terminates the function.

If Memory Verification is on, the MICE automatically checks data immediately after writing it to memory; if a mismatch occurs, it will display-

"Memory fill terminated -- write failure at xxxxxx"

Example 1: Examine and modify memory starting at 0H, then dump the results.

```
>Word 0<CR>
 000000 3131 0
 000002 1000 3800
 000004 0000
 000002 3800 1000
 000004 0000
 000006 0800 8000
 000008 F7F6 <ESC>
>Memory 0 F<CR>
          00   02   04   06   08   0A   0C   0E        ASCII-CODE
 000000  0000 1000 0000 8000 F7F6 F5F4 F3F2 F1F0      ................
>
```

Example 2: Modify the data at location 1000H to 100FH and dump the results.

```
>Word 1000 1 2 3 4 5 6 7 8<CR>
>Memory 1000 100F<CR>
         00   02   04   06   08   0A   0C   0E        ASCII-CODE
 001000  0001 0002 0003 0004 0005 0006 0007 0008      ................
>
```

Example 3: Map memory location 10000H to 1FFFFH to Bank 2 of EMM which can only be accessed by SP (supervisor program), and map the same location to the Bank 3 of emulation memory, but which can be accessed by SD (supervisor data) only. Then using Word command with CPU space to access the memory.

```
>MAp SP 10000 1FFFF I 2<CR>
>MAp SD 10000 1FFFF I 3<CR>
>Word SP 10000 1 2 3 4 5 6 7 8<CR>
>Memory 10000 1000F<CR>                    ;With SP as default memory type
         00   02   04   06   08   0A   0C   0E        ASCII-CODE
 010000  0001 0002 0003 0004 0005 0006 0007 0008      ................
>Word SD 10000 1111 2222 3333 4444 5555 6666 7777 8888<CR>
>Memory SD 10000 1000F<CR>
         00   02   04   06   08   0A   0C   0E        ASCII-CODE
 010000  1111 2222 3333 4444 5555 6666 7777 8888      .."""33DDUUffww..
>
```

Example 4: Dump the memory location 10000H to 1000FH with the default CPU space SP (supervisor program).

```
>Memory 10000 1000F<CR>
         00   02   04   06   08   0A   0C   0E        ASCII-CODE
 010000  0001 0002 0003 0004 0005 0006 0007 0008      ................
>
```

5.4.3 MEMORY CHECKSUM - CHecksum

CHecksum space-adr {adr|Length length}

CHecksum is the command for Memory Checksum.

space-adr specifies the memory location where the checksum operation is to begin. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP|UD] adr". If no memory type is specified, Supervisor Program (SP) will default.

adr is a hexadecimal address setting in the memory space of the emulation CPU specifying the last location in the range on which the checksum is to be carried out.

Length is the prefix to alert MICE that a "length" value (described below) is to be entered.

length is a hexadecimal value from 01H to 0FFFFFFH specifying the number of location to do a checksum on. Note that if a value not within this range is specified, or the "Length" parameter word is input without a valid following value, a

"Command syntax error"

message will result..

The checksum is calculated by taking the hexadecimal sum of the contents of the indicated range, with carry added, using module summation based on bus width (where byte = 0FFH, word = 0FFFFH).

Example 1: Fill the memory location 1000H to 17FFH with the data 55 and calculate the checksum with word size.

```
>Fill 1000 17FF 55<CR>
>CHecksum 1000 17FF<CR>
 Checksum is 5555
>
>CHecksum 1000 Length 7FF<CR>
 Checksum is 5555
>
```

Example 2: Change the size to byte and calculate the checksum, then change back the size to word.

```
>SIze Byte<CR>
>
>CHecksum 1000 17FF<CR>
 Checksum is AA
>SIze Word<CR>
>
```

5.4.4 MEMORY COMPARE - COMpare

COMpare space-adr1 {adr|Length length} space-adr2

COMpare is the command for Memory Compare.

space-adr1 specifies the starting address of block 1, the first of the two memory ranges to be compared. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP|UD] adr". If no memory type is specified, Supervisor Program (SP) will default.

adr is a hexadecimal address setting in the memory space of the emulation CPU specifying the last location in block 1.

Length is the prefix to alert MICE that a "length" value (described below) is to be entered.

length is a hexadecimal value from 01H to 0FFFFFFH specifying block length.

space-adr2 is a hexadecimal address in the memory space of the emulation processor specifying the starting point of block 2.

Block 1 is compared with block 2. If the blocks match exactly, the MICE will display-

"Memory comparison completed -- no difference found"

If the blocks do not match exactly, the MICE will display the first pair of non-matching addresses and their contents.

Example 1: Compare the block at 1000H-104FH with the block beginning at 2000H.

```
>COMpare 1000 Length 50 2000<CR>
 Memory comparison completed -- no difference found
>
```

Example 2: Compare the block at 0000H-004FH with the block beginning at 2000H. Note that only the first pair of non-matching addresses in the two ranges are listed.

```
>COMpare 0 Length 50 2000<CR>
 Memory comparison terminated -- difference found at:
 (000000) = 3131
 (002000) = 1111
>
```

5.4.5 MEMORY COPY - COpy

COpy space-adr1 {adr|Length length} [space-adr2]

COpy is the command to use when copying a content of a memory range to another within Emulation or Target Memory and also for copying a memory range from Target Memory to Emulation Memory.

space-adr1 specifies the first location in the memory range to be copied. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP|UD] adr". If no memory type is specified, Supervisor Program (SP) will default.

adr is a hexadecimal address setting specifying the last location in the memory range to be copied.

Length is the prefix to alert MICE that a "length" value (described below) is to be entered.

length is a hexadecimal value from 01H to OFFFFFFH specifying the number of locations to be copied.

space-adr2 is the starting address of the memory range to which the specified data are to be written. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP|UD] adr".

If "space-adr2" value is omitted, the contents of target memory in the specified range will be copied to the corresponding range in emulation memory.

1) Copying within emulation memory or Target Memory

The contents of the memory range beginning at space-adr1 are copied to the range beginning at space-adr2. If necessary, to prevent overwriting of as yet uncopied data in the source range (e.g., if the two ranges overlap), copying will begin at the end address of the destination range.

Example 1: Fill the memory location 1000H to 101FH with the data and display, then copy the 32 bytes data from 1000H to 1080H and display the results.

```
>Fill 1000 101F 11 22 33 44<CR>
>Memory 1000 101F<CR>
          00   02   04   06   08   0A   0C   0E        ASCII-CODE
 001000  1122 3344 1122 3344 1122 3344 1122 3344     ."3D."3D."3D."3D
 001010  1122 3344 1122 3344 1122 3344 1122 3344     ."3D."3D."3D."3D
>COpy 1000 101F 1080<CR>
>Memory 1080 10AF<CR>
          00   02   04   06   08   0A   0C   0E        ASCII-CODE
 001080  1122 3344 1122 3344 1122 3344 1122 3344     ."3D."3D."3D."3D
 001090  1122 3344 1122 3344 1122 3344 1122 3344     ."3D."3D."3D."3D
 0010A0  5555 5555 5555 5555 5555 5555 5555 5555     UUUUUUUUUUUUUUUU
>
```

Example 2: Copy the memory location 1000H to 101FH from supervisor space to 10000H at supervisor data space and display the copied result.

```
>COpy 1000 101F SD 10000<CR>
>Memory SD 10000 1003F<CR>
          00   02   04   06   08   0A   0C   0E        ASCII-CODE
 010000  1122 3344 1122 3344 1122 3344 1122 3344      ."3D."3D."3D."3D
 010010  1122 3344 1122 3344 1122 3344 1122 3344      ."3D."3D."3D."3D
 010020  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
 010030  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
>
```

2) Copying from Target Memory to Emulation Memory

This command transfers code from the specified portion of target memory to the corresponding location in MICE emulation memory. Note that if the emulation memory segment specified in the command is not enabled or it is write-protected, an error message will result.

Example: Copy all code at 8000H-8FFFH on the target to emulation memory.

```
>COpy 8000 8FFF<CR>
>
```

or:

```
>COpy 8000 Length 1000<CR>
>
```

5.4.6 DISASSEMBLY - Disassemble

Disassemble [space-adr [adr|Length length]]

Disassemble is the command for Disassembly. "Disassemble<CR>" alone, without any parameters, causes disassembly to start at the current PC.

space-adr specifies the memory location where the first instruction to be disassembled resides. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP|UD] adr". If no memory type is specified, Supervisor Program (SP) is the memory type default for CPU in the supervisor state and User Program (UP) is the default for user state.

adr is a hexadecimal address setting in the emulation program memory indicating the last location in the range to be disassembled and displayed.

Length is the prefix to alert MICE that a "length" value (described below) is to be entered.

length is a hexadecimal value from 01H to 0FFFFFFH specifying the number of location to be disassembled.

The contents of memory are disassembled and displayed, one page (16 statement lines) at a time, beginning at the starting address (if specified) or at the current PC. To continue disassembly, press <SP> or <CR>, (the screen will then scroll down one page at a time; to terminate, press <ESC>.

If an end address or length is specified, the contents of memory are disassembled continuously until the last location in the range is reached. The display can be halted or restarted at any time by pressing <Ctrl-S> or <Ctrl-Q>.

Example:

```
>Disassemble 8000 801E<CR>
LOC       OBJ              SOURCE CODE
008000    1A3C 0010        MOVE.B   #10,D5
008004    207C 0000 1000   MOVEA.L  #00001000,A0
00800A    227C 0000 2000   MOVEA.L  #00002000,A1
008010    3018             MOVE.W   (A0)+,D0
008012    6100 0FEC        BSR.W    009000
008016    32C1             MOVE.W   D1,(A1)+
008018    5305             SUBQ.B   #01,D5
00801A    66F4             BNE.B    008010
00801C    4E71             NOP
00801E    60FE             BRA.B    00801E
 Disassembly completed
>
```

If an illegal machine code is encountered during disassembly, it will be processed as a Define Word instruction and disassembly will continue. A question mark (?) will however appear on the screen.

Example:

```
>Disassemble 4000 400E<CR>
LOC        OBJ              SOURCE CODE
004000     327C 7E00        MOVEA.W  #7E00,A1
004004     FFFF             ?
004006     207C 0012 3456   MOVEA.L  #00123456,A0
00400C     4E71             NOP
00400E     4E71             NOP
 Disassembly completed
>
```

5.4.7 MEMORY FILL - Fill

Fill space-adr {adr|Length length} data

Fill is the command for Memory Fill.

space-adr specifies the memory location where the Fill operation is to begin. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP|UD] adr". If no memory type is specified, Supervisor Program (SP) will default.

adr is a hexadecimal address setting in the memory space of the emulation CPU specifying the last location to be filled.

Length is the prefix to alert MICE that a "length" value (described below) is to be entered.

length is a hexadecimal value from 01H to 0FFFFFFH specifying the number of locations to fill.

data is a hexadecimal or ASCII string in 1 to 32 bytes format. Data in ASCII must be enclosed in single quotation marks (e.g., 'AR') Note that the single quotation marks itself [''] may not be entered in ASCII. The string may not be more than 32 bytes in length. Combined use of hex and ASCII is permitted. If the address range is smaller than the string, Memory Fill is still executed but the excess or trailing data are ignored.

To fill a memory range with a particular value or string, input the starting address, the end address or length, and (in byte format) the desired value or string. If Memory Verification is on, the fill operation will be verified (I/O will also be verified if the target has memory-mapped I/O), and any mismatch will be reported with the message

"Memory fill terminated -- write failure at xxxxxx".

Example 1: Fill the memory location 1000H to 102F with the hexadecimal data and display the results.

```
>Fill 1000 102F 11 22 33 44<CR>
>Memory 1000 103F<CR>
        00   02   04   06   08   0A   0C   0E        ASCII-CODE
001000  1122 3344 1122 3344 1122 3344 1122 3344     ."3D."3D."3D."3D
001010  1122 3344 1122 3344 1122 3344 1122 3344     ."3D."3D."3D."3D
001020  1122 3344 1122 3344 1122 3344 1122 3344     ."3D."3D."3D."3D
001030  5555 5555 5555 5555 5555 5555 5555 5555     UUUUUUUUUUUUUUUU
>
```

Example 2: Fill the memory location 1000H to 102F with hexadecimal and ASCII data, then display the result.

```
>Fill 1000 Length 30 'AB' 12 43<CR>
>Memory 1000 Length 40<CR>
        00   02   04   06   08   0A   0C   0E        ASCII-CODE
001000  4142 1243 4142 1243 4142 1243 4142 1243      AB.CAB.CAB.CAB.C
001010  4142 1243 4142 1243 4142 1243 4142 1243      AB.CAB.CAB.CAB.C
001020  4142 1243 4142 1243 4142 1243 4142 1243      AB.CAB.CAB.CAB.C
001030  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
>
```

Example 3: Fill the memory location 10000H to 1003F with the string data in the SD (supervisor data) space only, then display the result.

```
>Fill SD 10000 1003F 'TEST'<CR>
>Memory SD 10000 1004F<CR>
        00   02   04   06   08   0A   0C   0E        ASCII-CODE
010000  5445 5354 5445 5354 5445 5354 5445 5354      TESTTESTTESTTEST
010010  5445 5354 5445 5354 5445 5354 5445 5354      TESTTESTTESTTEST
010020  5445 5354 5445 5354 5445 5354 5445 5354      TESTTESTTESTTEST
010030  5445 5354 5445 5354 5445 5354 5445 5354      TESTTESTTESTTEST
010040  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
>
```

5.4.8 MEMORY DUMP - Memory

Memory [space-adr [adr|Length length]]

Memory is the command for Memory Dump. "Memory<CR>" alone, without any parameters, displays the next 256 bytes, starting at the current PC.

space-adr specifies the memory location where the display operation is to begin. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP|UD] adr". If no memory type is specified, Supervisor Program (SP) will default.

adr is a hexadecimal address setting in the memory space of the emulation CPU specifying the last memory location to be filled.

Length is the prefix to alert MICE that a "length" value (described below) is to be entered.

length is a hexadecimal value from 01H to 0FFFFFFH specifying the number of locations to be filled.

To display the contents of memory in a specific range, input a starting address and either an end address or the desired length. If no end address is given, the contents of memory will be displayed one page (256 bytes) at a time, and the message "[MORE]" will appear after each page is displayed; press <CR> to view the next page, <ESC> to terminate the function.

When an end address is specified, the contents of memory are displayed continuously until the last location in the range is reached; the display can be halted or restarted at any time by pressing <Ctrl-S> or <Ctrl-Q>.

Memory contents are displayed in hexadecimal and ASCII; values without ASCII equivalents are indicated by periods. To terminate the function before all data in the specified range have been displayed, press <ESC>.

Example 1: Display memory content in the range 1000H to 10FFH with default SP (supervisor program) space.

```
>Memory 1000<CR>
        00   02   04   06   08   0A   0C   0E        ASCII-CODE
001000  4142 1243 4142 1243 4142 1243 4142 1243      AB.CAB.CAB.CAB.C
001010  4142 1243 4142 1243 4142 1243 4142 1243      AB.CAB.CAB.CAB.C
001020  4142 1243 4142 1243 4142 1243 4142 1243      AB.CAB.CAB.CAB.C
001030  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
001040  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
001050  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
001060  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
001070  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
001080  1122 3344 1122 3344 1122 3344 1122 3344      ."3D."3D."3D."3D
001090  1122 3344 1122 3344 1122 3344 1122 3344      ."3D."3D."3D."3D
0010A0  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
0010B0  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
0010C0  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
0010D0  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
0010E0  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
0010F0  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
[ More ]<ESC>
>
```

Example 2: Display memory content in the range 1000H to 101FH with default SP (supervisor program) space.

```
>Memory 1000 Length 20<CR>
        00   02   04   06   08   0A   0C   0E        ASCII-CODE
001000  4142 1243 4142 1243 4142 1243 4142 1243      AB.CAB.CAB.CAB.C
001010  4142 1243 4142 1243 4142 1243 4142 1243      AB.CAB.CAB.CAB.C
>
```

Example 3: Display memory content in the range 10000H to 1002FH with SD (supervisor data) space. Then display the same memory range with Supervisor program.

```
>Memory SD 10000 Length 30<CR>
        00   02   04   06   08   0A   0C   0E        ASCII-CODE
010000  5445 5354 5445 5354 5445 5354 5445 5354      TESTTESTTESTTEST
010010  5445 5354 5445 5354 5445 5354 5445 5354      TESTTESTTESTTEST
010020  5445 5354 5445 5354 5445 5354 5445 5354      TESTTESTTESTTEST
>Memory SP 10000 Length 30<CR>
        00   02   04   06   08   0A   0C   0E        ASCII-CODE
010000  1122 3344 1122 3344 1122 3344 1122 3344      ."3D."3D."3D."3D
010010  1122 3344 1122 3344 1122 3344 1122 3344      ."3D."3D."3D."3D
010020  5555 5555 5555 5555 5555 5555 5555 5555      UUUUUUUUUUUUUUUU
>
```

5.4.9 MEMORY SEARCH - SEarch

SEarch space-adr {adr|Length length} data

SEarch is the command for Memory Search.

space-adr is a hexadecimal address setting specifying the memory location where the Search operation is to begin. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP|UD] adr". If no memory type is specified, Supervisor Program (SP) will default.

adr is a hexadecimal address setting in the memory space of the emulation CPU specifying the last location to be filled.

Length is the prefix to alert MICE that a "length" value (described below) is to be entered.

length is a hexadecimal value from 01H to 0FFFFFFH specifying the number of locations to search through.

data is a 1 to 32 bytes of data in hexadecimal and/or ASCII value or string. Hex data must be input in byte format. Data in ASCII must be enclosed in single quotation marks (e.g., 'AR'). Note that the single quotation marks itself [''] may not be entered in ASCII. The string may not be more than 32 bytes in length. Combined use of hex and ASCII is permitted. If the address range is smaller than the string length, Memory Search is still executed but no resulting match will be accomplished. The message-

"Memory search completed -- Data not found"

will display.

Input the starting address, either the end address or the length of the range to be searched, and the value or string to search for. The MICE will search for the specified value or string in the designated range. To terminate the Search function before the specified data have been found, press <ESC>.

Example: Fill the memory content in the range 1000H to 1FFFH to 55H and modify the content in the range 1800H to 1803H. Then using SEarch command to find these data.

```
>Fill 1000 1FFF 55<CR>
>Word 1800 'TEST'<CR>
>SEarch 1000 1FFF 'TEST'<CR>
 Memory search completed -- Data not found
>SEarch 1000 1FFF 'TEST'<CR>
 Memory search terminated -- data matched at 001800
>
```

5.4.10 MEMORY TEST - TEst

TEst space-adr {adr|Length length}

TEst is the command for Memory Test.

space-adr specifies the memory location where the testing operation is to begin. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP|UD] adr". If no memory type is specified, Supervisor Program (SP) will default.

adr is a hexadecimal address setting in the memory space of the emulation processor specifying the last location in the range to be tested.

Length is the prefix to alert MICE that a "length" value (described below) is to be entered.

length is a hexadecimal value from 01H to 0FFFFFFH specifying the number of locations to be tested.

Each location in the specified range is tested as follows: the full 32-bit address of the location is taken as a three-byte value, and (assuming Memory Access Size is set to Byte) these three bytes are XORed with each other in left-to-right sequence to produce a one-byte value; this value is written to the location and read back for verification, and then its complement is written to the location and checked. Two passes are made through the entire range in this way. If an incorrect value is read back at any time, testing stops immediately and the failed address is displayed.

Remember that this test destroys the original contents of memory in the specified range.

Example 1: Test the memory range 1000H-104FH, then dump the contents of this range to see how they have been modified by the test operation.

```
>TEst 1000 Length 50<CR>
 Memory test completed -- no failure found
>Memory 1000 Length 50<CR>
          00   02   04   06   08   0A   0C   0E        ASCII
00001000  EFEE EDEC EBEA E9E8 E7E6 E5E4 E3E2 E1E0  ..............
00001010  FFFE FDFC FBFA F9F8 F7F6 F5F4 F3F2 F1F0  ..............
00001020  CFCE CDCC CBCA C9C8 C7C6 C5C4 C3C2 C1C0  ..............
00001030  DFDE DDDC DBDA D9C8 D7D6 D5D4 D3D2 D1D0  ..............
00001040  AFAE ADAC ABAA A9A8 A7A6 A5A4 A3A2 A1A0  ..............
>
```

Example 2: Test the memory range 10000H-40000H, where memory is enabled only through 1FFFFH.

```
>TEst 10000 40000<CR>
 Memory test terminated -- failure at 020000
>
```

5.5 PORT COMMANDS

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Port Input	Input	space-adr [Length in-length]
Port Output	Output	space-adr data

Table 5-12 Port Group Full Command Syntax Summary

These commands provide access to the contents of I/O ports. The MICE-16 68000 supports two basic data formats: Byte and Word. For the Port Input and Port Output commands, the desired data format must be specified with the Memory Access Size command (Section 5.3.16). I/O is memory-mapped, while the total amount of addressable memory is 16M bytes and the address range is user-defined.

The MICE always checks to make sure that the port addresses specified by the user are valid before an operation is performed. Any reference to an invalid address will result in an error message and termination of the command. The MICE, however, does not verify that there is memory or anything else connected to a port when executing a read or write operation.

NOTE

When the external ready signal is selected and read/-write functions are being performed, if external circuitry does not respond with the $\overline{DTACK}$ signal,

"No target DTACK* at address xxxxxxxx"

will appear on the console with the contents of memory remaining unchanged.

5.5.1 PORT INPUT - Input

Input space-adr [Length in-length]

Input is the command for Port Input.

space-adr is the first input port that is to be read and displayed. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP-|UD] adr". If no memory type is specified, Supervisor Program (SP) will default.

Length is the prefix to alert MICE that an "in-length" value (described below) is to be entered.

in-length is a hexadecimal value from 00H to 0FFH specifying the number of consecutive input ports (including the specified port address) to be used and displayed (note that 00H indicates 256).

Port contents are read and displayed beginning at the indicated address.

Example: Read and display input for 256 (0FFH) ports starting at port 1000H.

```
>Input 1000 Length FF<CR>
 PORT    00   02   04   06   08   0A   0C   0E        ASCII
 001000  EFEE EDEC EBEA E9E8 E7E6 E5E4 E3E2 E1E0  ................
 001010  FFFE FDFC FBFA F9F8 F7F6 F5F4 F3F2 F1F0  ................
 001020  CFCE CDCC CBCA C9C8 C7C6 C5C4 C3C2 C1C0  ................
 001030  DFDE DDDC DBDA D9D8 D7D6 D5D4 D3D2 D1D0  ................
 001040  AFAE ADAC ABAA A9A8 A7A6 A5A4 A3A2 A1A0  ................
 001050  BFBE BDBC BBBA B9B8 B7B6 B5B4 B3B2 B1B0  ................
 001060  8F8E 8D8C 8B8A 8988 8786 8584 8382 8180  ................
 001070  9F9E 9D9C 9B9A 9998 9796 9594 9392 9190  ................
 001080  6F6E 6D6C 6B6A 6968 6766 6564 6362 6160  onmlkjihgfedcba`
 001090  7F7E 7D7C 7B7A 7978 7776 7574 7372 7170  .~}|{zyxwvutsrqp
 0010A0  4F4E 4D4C 4B4A 4948 4746 4544 4342 4140  ONMLKJIHGFEDCBA@
 0010B0  5F5E 5D5C 5B5A 5958 5756 5554 5352 5150  _^]\[ZYXWVUTSRQP
 0010C0  2F2E 2D2C 2B2A 2928 2726 2524 2322 2120  /.-,+*)('&%$#"!
 0010D0  3F3E 3D3C 3B3A 3938 3736 3534 3332 3130  ?>=<;:9876543210
 0010E0  0F0E 0D0C 0B0A 0908 0706 0504 0302 0100  ................
 0010F0  1F1E 1D1C 1B1A 1918 1716 1514 1312 1110  ................
>
```

5.5.2 PORT OUTPUT - Output

Output space-adr data

Output is the command for Port Output.

space-adrt specifies one of the emulation processor's output ports. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: "[SP|SD|UP-|UD] adr". If no memory type is specified, Supervisor Program (SP) will default.

data is a hexadecimal or ASCII values in 1 to 32 bytes format, to be written to the specified output port in a sequential manner. Data in ASCII must be enclosed in apostrophe, (e.g., 'AR'). Note that the apostrophes itself [''] may not be input in ASCII. The data output may not be more than 32 bytes in length. Combined use of hex and ASCII is permitted.

If only one value is specified, it is written immediately to the indicated output port. If more than one value is specified, the first is written to the indicated port immediately and the following values are written at one-millisecond intervals.

Example 1: Attempt to write the value 123H to port 1000H when memory access size is set to Byte; correct the error and repeat the command.

```
>Output 1000 123<CR>
 Command syntax error
>SIze Word<CR>
>Output 1000 123<CR>
>
```

Example 2: Write the values 1H and 2H, and the ASCII code for the numeral "3," to port 1000H. The value 1H is written immediately, followed after one millisecond by the value 2H and after another millisecond by the code for "3".

```
>SIze Byte<CR>
>Output 1000 1 2 '3'<CR>
>
```

Example 3: Write the word value 41ED to port 1100H.

```
>SIze Word<CR>
>Output 1100 41ED<CR>
>
```

5.6 EMULATION COMMANDS

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Cycle Step	Cycle	[{Wait|count}]
Jump	Jump	adr
Register Display/Modify	Register	[register]
Reset Emulation Processor	RESet	[pc [ssp]]
Instruction Step	Step	[adr1 adr2 [CAL1]|count|CAL1]

Table 5-13 Emulation Group Full Command Syntax Summary

The total amount of addressable memory for the 68000 is 16M bytes. Because the 68000 is word-oriented, even-numbered addresses must be assigned for the:

- program counter (PC) in the Jump/RESet commands (Sections 5.6.2/5.6.4)
- user stack pointer (USP) in the Register command (Sections 5.6.3)
- supervisor stack pointer (SSP) in the Register/RESet commands

NOTE

1) *If an odd value is entered, the system will drop the least significant bit to convert it into an even value.*

   Before carrying out a command, the MICE always checks any memory addresses input by the user to make sure they are valid. A reference to an invalid address will result in display of an error message-

   "Command syntax error"

   and termination of the command.

2) *When executing the Cycle Step and Instruction Step commands (Section 5.6.1 and 5.6.5 respectively), refer to Table 5-2 for a description of the displayed status information. Note that breakpoints (Section 5.7.1) are not active during execution of these commands.*

3) *During execution of these commands if an attempt is made to refer to the memories defined as either non-existence or guided (G), or write protected (ER/IR),*

   "Write protected memory trespassed" or
   "Guarded memory trespassed"

   will display to warn user of such violation. Emulation processing however, will continue after <CR> is entered. the illegally entered data for external /internal write protected (ER/IR) or non-existence (G) memory block will be indecipherable.

4) *When performing read/write function with the external ready signal being selected and the external circuitry does not respond to the $\overline{DTACK}$ signal, the message-*

"Emulation processor can't step
- Due to no target DTACK at address xxxxxx"*

will display with the contents of memory remaining unchanged.

5) *Another warning message-*

"Emulation processor can't step -- check clock, BR, DTACK"*

will also display when there is no $\overline{DTACK}$ signal while MICE tries to gain control of the emulation processor.

6) *If $\overline{BR}$ (Bus Request) on the target is active for over 2.5 seconds during execution of cycle command, the MICE will display the message-*

"Bus Request"

5.6.1 CYCLE STEP - Cycle

Cycle [Wait|count]

Cycle is the command for Cycle Step.

Wait stops the CPU in a wait state (or $\overline{\text{DTACK}}$ signal is under inactive state).

count specifies the machine cycle interval between displayed messages. The range is 0-FFFFH.

Cycle Step stops the emulation processor, steps the program one cycle, and then halts the processor in Bus Acknowledge state ($\overline{\text{BGACK}}$). The MICE displays target status for the cycle just executed and awaits a <CR> to advance the emulation processor to the next machine cycle. The new status is then displayed and the MICE again waits. Enter <CR> to repeat the entire sequence, or <ESC> to terminate the single-cycle mode of emulation.

In the Cycle command, the displayed cycle status has already been executed. The Cycle Wait command is used to force the CPU to stop in a 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. For some targets which do not allow the CPU to stop in a wait state for very long (e.g., those which require dynamic RAM refresh), the Cycle Wait command will cause the target system to fail.

Note that if a word operand operation requires two cycles to execute, the MICE will display both cycles simultaneously with Cycle command. In the Cycle Wait command, however, only one cycle is displayed per step.

If, during execution of cycle or instruction step commands, an attempt is made to write into Read-only area (defined as 'IR'or 'ER' in the MAp command) this message will display-

"...Write protected memory trespassed!"

Likewise, if a nonexistent area (defined as 'G' in the MAp command) is accessed, the following message will display-

"...Guarded memory trespassed!"

There are two display modes for cycle stepping. The first is "continuous display," in which the machine cycles are automatically scrolled by pressing <SP> (for Go). The display may be halted or restarted at any time by pressing <SP> again.

The other display mode is "single step." In this mode, the MICE-16 68000 first stops the emulation processor, advances the program one cycle, halts the processor at the selected state (Bus Request or Wait state), and then displays target status information (see following example; also refer to Table 5-2 for a description of the displayed status information). After each step the MICE waits for a <CR> to advance the emulation processor to the next machine cycle. The new status is then displayed and the MICE again waits. Enter <CR> to repeat the entire sequence.

Note that if a Read-Modify-Write cycle requires two machine cycles to execute (e.g. TAS), the MICE will display both cycles simultaneously.

Example 1: Cycle-step the following program:

```
>Jump 8000<CR>
>Cycle<CR>
        ADDRESS         DATA        STATUS      TRACE BIT
        008000          1A3C          SP        11111111<CR>
        008002          0010          SP        11111111<CR>
        008004          207C          SP        11111111<CR>
        008006          0000          SP        11111111<CR>
        008008          1000          SP        11111111<CR>
        00800A          227C          SP        11111111<CR>
        00800C          0000          SP        11111111<CR>
        00800E          2000          SP        11111111<CR>
        008010          3018          SP        11111111<CR>
        008012          6100          SP        11111111<CR>
        001000          5555          SR        11111111<CR>
        008014          0FEC          SP        11111111<CR>
        000FFC          0000          SW        11111111<CR>
        000FFE          8016          SW        11111111<CR>
        009000          0640          SP        11111111<CR>
        009002          0100          SP        11111111<CR>
        009004          3200          SP        11111111<CR>
        009006          4E75          SP        11111111<ESC>
>
```

Example 2: Cycle the above program using a count of 3.

```
>Jump 8000<CR>
>Cycle 3<CR>
        ADDRESS         DATA        STATUS      TRACE BIT
        008004          207C          SP        11111111<CR>
        00800A          227C          SP        11111111<CR>
        008010          3018          SP        11111111<CR>
        008014          0FEC          SP        11111111<CR>
        009000          0640          SP        11111111<CR>
        009006          4E75          SP        11111111<CR>
        000FFE          8016          SR        11111111<CR>
        002000          5655          SW        11111111<CR>
        008010          3018          SP        11111111<ESC>
>
```

Example 3: If the program accesses a restricted area during cycle command, MICE-16 68000 will display a warning message.

```
>Word 8004<CR>
008004 207C
008006 0000 2
008008 1000 <ESC>
>Jump 8000<CR>
>Cycle<CR>
        ADDRESS          DATA          STATUS        TRACE BIT
        008000           1A3C            SP          11111111<CR>
        008002           0010            SP          11111111<CR>
        008004           207C            SP          11111111<CR>
        008006           0002            SP          11111111<CR>
        008008           1000            SP          11111111<CR>
        00800A           227C            SP          11111111<CR>
        00800C           0000            SP          11111111<CR>
        00800E           2000            SP          11111111<CR>
        008010           3018            SP          11111111<CR>
        008012           6100            SP          11111111<CR>
        021000           FFFF            SR          11111111<CR>
        ...Guarding memory trespassed!
        008014           0FEC            SP          11111111<ESC>
>Word 8006<CR>
008006 0002 0
008008 1000 <ESC>
>
```

5.6.2 JUMP - Jump

Jump adr

Jump is the command for Jump.

adr is a hexadecimal address in the emulation processor's program memory to which the program counter is to be set.

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

Note that Jump command does not support the setting of memory type. So if the user wants to change from supervision to user program, and vice-versa, user can use 'Register' command to modify the 'S' bit in 'SR' register.

Example: Change the program counter from the current address to 8000H.

```
>Register<CR>
      0        1        2        3        4        5        6        7
Dn  FFFF5655 00045655 FFFFFDFC 00000DDD 0000000D 0000000D 0000000D FFFFFFFF
An  00001002 00002002 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 00001000
SSP=00001000 USP=FFFFFFFE                SR   T S  III   XNZVC
PC=009000                                2700 0010011100000000
>Jump 8000<CR>
>Register<CR>
      0        1        2        3        4        5        6        7
Dn  FFFF5655 00045655 FFFFFDFC 00000DDD 0000000D 0000000D 0000000D FFFFFFFF
An  00001002 00002002 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 00001000
SSP=00001000 USP=FFFFFFFE                SR   T S  III   XNZVC
PC=008000                                2700 0010011100000000
>
```

5.6.3 REGISTER DISPLAY/MODIFY - Register

Register [register]

Register is the command to display or modify the contents of the emulation processor's registers.

register is the standard Motorola designation for the register, the contents of which are to be displayed or modified.

1) Display All Registers

The contents of all of the emulation processor's registers are displayed if no particular register is specified. Note that (1) the value displayed for the program counter (PC) is always the address of the next program instruction, and (2) the status register (SR) is displayed in both hexadecimal and binary.

Example: Display the contents of all registers.

```
>Register<CR>
      0        1        2        3        4        5        6        7
Dn  FFFF5655 00045655 FFFFFDFC 00000DDD 0000000D 0000000D 0000000D FFFFFFFF
An  00001002 00002002 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 00001000
SSP=00001000 USP=FFFFFFFE               SR   T S  III   XNZVC
PC=008000                               2700 0010011100000000
>
```

2) Modify Register(s)

If a register is specified, its contents are displayed and the MICE waits for input. To advance to the next register, enter <CR>; to go back to the previous register, enter <LF> or <Ctrl-J>. To terminate the command, press <ESC>. To change the contents of the currently displayed register, enter a new value in hexadecimal and press <CR> or <Ctrl-J>. When modifying flags, remember that input is limited to the binary values "0" and "1".

Example: Examine a particular register and change its contents.

```
>Register D0<CR>
D0    FFFF5655 11112222<CR>
D1    00045655 33334444<CR>
D2    FFFFFDFC 55556666<CR>
D3    00000DDD <ESC>
>Register<CR>
      0        1        2        3        4        5        6        7
Dn  11112222 33334444 55556666 00000DDD 0000000D 0000000D 0000000D FFFFFFFF
An  00001002 00002002 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 00001000
SSP=00001000 USP=FFFFFFFE               SR   T S  III   XNZVC
PC=008000                               2700 0010011100000000
```

```
>Register S<CR>
S     1  <CR>
I2    1 0<CR>
I1    1  <CR>
I0    1  <CR>
X     0 1<CR>
N     0 1<CR>
Z     0 1<CR>
V     0  <LF>
Z     1  <LF>
N     1  <LF>
X     1  <ESC>
>Register<CR>
      0        1        2        3        4        5        6        7
Dn  11112222 33334444 55556666 00000DDD 0000000D 0000000D 0000000D FFFFFFFF
An  00001002 00002002 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 00001000
SSP=00001000 USP=FFFFFFFE               SR   T S  III   XNZVC
PC=008000                               231C 0010001100011100
>
```

5.6.4 RESET/INITIALIZE EMULATION PROCESSOR - RESet

RESet [pc [ssp]]

RESet is the command to reset/initialize the emulation processor's program counter to the address where emulation is to begin. "RESet<CR>" alone, without any parameters, loads the emulation processor's program counter and supervisor stack pointer from the first four words of memory (bytes 000000-000007). Note, however, that if either of the addresses to be loaded is odd, the system will automatically set the PC to 400H and the SSP to 1FFF0H.

pc is a hexadecimal address to load in place of the current PC.

ssp is a hexadecimal address to load in place of the current ISP.

The $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ signal is asserted to the emulation processor, and then, the processor's program counter, supervisor stack pointer and status register are altered. Furthermore, bus request, bus error, and interrupt control signals to the emulation CPU that are selectively disabled and enabled with CONtrol command (Section 5.3.3), are disabled if no target system (or target has no power) is connected to the MICE. Addresses may optionally be specified to set the emulation processor's program counter and supervisor stack pointer to the desired locations in program memory.

NOTE

When the emulation processor is reset, the control signals previous to "RESet" command execution are retained if the MICE is connected to a power-applied target system, otherwise, these signals are disabled.

The 68000 provides a RESET output signal which eliminates the need to reset peripheral devices or slave processors on the target side which are connected to pin 18.

Example 1: Reset the emulation processor when locations 000000-000007 contain an odd address for either the PC or the SSP.

```
>Memory 0 7<CR>
        00   02   04   06   08   0A   0C   0E        ASCII-CODE
000000  0000 1000 0000 8000                         ........
>Word 6<CR>
000006 8000 8001<CR>
000008 F7F6 <ESC>
>RESet<CR>
        Odd PC/SSP values in memory locations 0-7 have
        been set by default to PC=400H and SSP=1FFF0H
>Word 6<CR>
000006 8001 8000<CR>
000008 F7F6 <ESC>
>RESet<CR>
>
```

Example 2: Display the contents of the emulation processor's flags and registers, reset the processor, then display the flags and registers again.

```
>Register<CR>
      0        1        2        3        4        5        6        7
Dn  11112222 33334444 55556666 00000DDD 0000000D 0000000D 0000000D FFFFFFFF
An  00001002 00002002 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 00001000
SSP=00001000 USP=FFFFFFFE                SR   T S  III   XNZVC
PC=008000                                271C 0010011100011100
>RESet 800 1FFF0<CR>
>Register<CR>
      0        1        2        3        4        5        6        7
Dn  11112222 33334444 55556666 00000DDD 0000000D 0000000D 0000000D FFFFFFFF
An  00001002 00002002 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 0001FFF0
SSP=0001FFF0 USP=FFFFFFFE                SR   T S  III   XNZVC
PC=000800                                271C 0010011100011100
>
```

5.6.5 INSTRUCTION STEP - Step

Step [adr1 adr2 [CALl]|count|CALl]

Step is the command for Instruction Step.

count is a hexadecimal value from OH to FFFFH specifying the step interval for status display. Note that OH indicates 65536 steps.

adr1,adr2 are hexadecimal addresses in the memory space of the emulation CPU specifying the beginning and the last memory locations where a program range status is to be skipped.

CALl is an option of instruction step to skip subroutines.

If count is not specified for instruction step, the default is one. In this mode, the MICE initially displays the current instruction to be executed and waits for the next input. Enter <CR> to make the emulation processor execute the instruction on display. After executing the displayed instruction, a new status will appear next line to the previous instruction, and the MICE again waits for subsequent input. Enter another <CR> to repeat the entire sequence, e.g. displaying new status after execution of each instruction on display.

Step with range (adr1 adr2) option will skip status display of the specified program area.

Step with "Call" option will skip all subroutine status in the program under real time execution. The program counter (PC) will point to the next instruction after CALL instruction "BSR" or "JSR" is executed.

If the instruction on display is "BSR" or "JSR", the new status next to "BSR" or "JSR" instruction will display after the subroutine is executed.

It is also possible to auto-step through program instructions; this is done by pressing <SP>. Press <SP> also to halt and to resume auto-stepping.

Enter <ESC> to terminate the single step mode of emulation.

Memory access messages (e.g. 01FFEC -SW- 8022) provide data on bus cycles (Table 5-3). The first six digits (01FFEC in this example) indicate the memory address, the uppercase (SW) represent the type of bus cycle and the last two to four digits (8022) are the data transferred.

Register contents associated with the current step in program execution can be observed at any time by pressing <R>, but remember that the contents displayed are those prior to execution of the current instruction. Also note that the current instruction is redisplayed after the register contents, and is only executed when a <CR> is subsequently entered.

Before user enters the Step with Call command, it is advisable to check and make sure that the subroutines for "BSR" or "JSR" instructions are bug-free.

Example 1: Instruction-step the previous program. Note that the processor does not begin emulation until the current status is displayed and <CR> is input.

```
>Jump 8000<CR>
>Step<CR>
LOC         OBJ              SOURCE CODE
008000      1A3C 0010        MOVE.B   #10,D5<CR>
008004      207C 0000 1000   MOVEA.L  #00001000,A0<CR>
00800A      227C 0000 2000   MOVEA.L  #00002000,A1<CR>
008010      3018             MOVE.W   (A0)+,D0<CR>
     001000 -SR- 5555
008012      6100 0FEC        BSR.W    009000<R>
       0         1         2         3         4         5         6         7
Dn  11115555 33334444 55556666 00000DDD 0000000D 00000010 0000000D FFFFFFFF
An  00001002 00002000 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 0001FFF0
SSP=0001FFF0 USP=FFFFFFFE                SR   T S  III   XNZVC
PC=008012                                2710 0010011100010000
008012      6100 0FEC        BSR.W    009000<CR>
     01FFEC -SW- 0000
     01FFEE -SW- 8016
009000      0640 0100        ADDI.W   #0100,D0<CR>
009004      3200             MOVE.W   D0,D1<CR>
009006      4E75             RTS<CR>
     01FFEC -SR- 0000
     01FFEE -SR- 8016
008016      32C1             MOVE.W   D1,(A1)+<R>
       0         1         2         3         4         5         6         7
Dn  11115655 33335655 55556666 00000DDD 0000000D 00000010 0000000D FFFFFFFF
An  00001002 00002000 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 0001FFF0
SSP=0001FFF0 USP=FFFFFFFE                SR   T S  III   XNZVC
PC=008016                                2700 0010011100000000
008016      32C1             MOVE.W   D1,(A1)+<CR>
     002000 -SW- 5655
008018      5305             SUBQ.B   #01,D5<CR>
00801A      66F4             BNE.B    008010<CR>
008010      3018             MOVE.W   (A0)+,D0<ESC>
>
```

If an instruction count other than one is entered, emulation begins immediately from the current program counter and continues until the count of the instruction to be executed equals the number specified. The current status is then displayed and the MICE waits for a <CR> before executing the next "count" instruction. Enter <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.

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

Example 2: Multi-step the program and compare results with the display in the first example:

```
>Jump 8000<CR>
>Step 3<CR>
LOC        OBJ              SOURCE CODE
00800A     227C 0000 2000   MOVEA.L  #00002000,A1<CR>
009000     0640 0100        ADDI.W   #0100,D0<CR>
008016     32C1             MOVE.W   D1,(A1)+<CR>
008010     3018             MOVE.W   (A0)+,D0<CR>
009004     3200             MOVE.W   D0,D1<CR>
008018     5305             SUBQ.B   #01,D5<CR>
008012     6100 0FEC        BSR.W    009000<CR>
009006     4E75             RTS<CR>
00801A     66F4             BNE.B    008010<CR>
009000     0640 0100        ADDI.W   #0100,D0<ESC>
>
```

Example 3: Step with Range option and then with Call option the following program and compare the results (shown in a and b below) with the given program.

```
>Disassemble 8000 801E<CR>
LOC        OBJ              SOURCE CODE
008000     1A3C 0010        MOVE.B   #10,D5
008004     207C 0000 1000   MOVEA.L  #00001000,A0
00800A     227C 0000 2000   MOVEA.L  #00002000,A1
008010     3018             MOVE.W   (A0)+,D0
008012     6100 0FEC        BSR.W    009000
008016     32C1             MOVE.W   D1,(A1)+
008018     5305             SUBQ.B   #01,D5
00801A     66F4             BNE.B    008010
00801C     4E71             NOP
00801E     60FE             BRA.B    00801E
 Disassembly completed
>
```

a)
```
>Step 8000 8010<CR>
>Register<CR>
        0        1        2        3        4        5        6        7
Dn  0003EFEE 0004F2F0 FFFFFDFC FFFFFFFF FFFFFF00 FFFFFF10 00000001 FFFFFFFF
An  00001002 00002000 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 00001000
SSP=00001000 USP=FFFFFFFF                SR   T S  III   XNZVC
PC=008012                                2708 0010011100001000
>Step 8012 8016
>Register<CR>
        0        1        2        3        4        5        6        7
Dn  0003EFEE 0004F2F0 FFFFFDFC FFFFFFFF FFFFFF00 FFFFFF10 00000001 FFFFFFFF
An  00001002 00002000 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 00000FFC
SSP=00000FFC USP=FFFFFFFF                SR   T S  III   XNZVC
PC=009000                                2708 0010011100001000
>
```

b) >Jump 8000<CR>
>Step CALL1<CR>

```
LOC          OBJ                 SOURCE CODE
008000       1A3C 0010           MOVE.B    #10,D5<CR>
008004       207C 0000 1000      MOVEA.L   #00001000,A0<CR>
00800A       227C 0000 2000      MOVEA.L   #00002000,A1<CR>
008010       3018                MOVE.W    (A0)+,D0<CR>
     001000 -SR- EFEE
008012       6100 0FEC           BSR.W     009000<CR>
008016       32C1                MOVE.W    D1,(A1)+<CR>
     002000 -SW- F0EE
008018       5305                SUBQ.B    #01,D5<CR>
00801A       66F4                BNE.B     008010<CR>
008010       3018                MOVE.W    (A0)+,D0<ESC>
>
```

5.7 TRACE COMMANDS

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Display Events	Ev[ent]	
Set Bus Event 1	Ev[ent]1	adx [datum] [status] [COunt count]
Set Bus Event 2	Ev[ent]2	adx [datum] [status]
Set External Trigger	Ev[ent]3	{High|Low}
Set Execution Event 1	Ev[ent]4	adr [SP|UP]*
Set Execution Event 1	Ev[ent]4	adr [SP|UP] [Vector-Base value]**
Set Execution Event 2	Ev[ent]5	adr [SP|UP]*
Set Execution Event 2	Ev[ent]5	adr [SP|UP] [Vector-Base value]**
Set Execution Event 3	Ev[ent]6	adr [SP|UP]*
Set Execution Event 3	Ev[ent]6	adr [SP|UP] [Vector-Base value]**
Clear Event	Ev[ent]	[[1|2|3|4|5|6] CLear]
Execute User Program	Go	[Run] [adr]
Halt Emulation Processor	HAlt	
List Trace Buffer	List	[frame [adr1 adr2] [status] [trace-bits]]
List with Source Code	List	{Source|Instruction} [frame]
List Trace Frame Total	List	Number
Display Trace Cycle Qualifier	Qualify	adx [status]
Display/Clear Qualifier	Qualify	[CLear]
Show Sync Setting	SYnc	
Set Sync In/Out	SYnc	[{Input|OUtput} {On|OFf}]
Timebase Selection	TImebase	[Go|Event] [1|10|100|1000|10000]
Display Trace Parameters	TRAce	
Set "Then" Trigger	Trigger	[Run] # [Then # [Then #]] [BACKward |CENter|FOrward|DELay count]
Set "And" Trigger	Trigger	[Run] # And # [And #] [BACKward|CENter |FOrward|DELay count]
Set "Or" Trigger	Trigger	[Run] # Or # [Or #] [BACKward|CENter |FOrward|DELay count]
Set "And-Then" Trigger	Trigger	[Run] # And # Then # [BACKward|CENter |FOrward|DELay count]
Set "Or-Then" Trigger	Trigger	[Run] # Or # Then # [BACKward|CENter |FOrward|DELay count]
Display/Clear Trigger	Trigger	[CLear]

* Applies to 68000 CPU only.
** Applies to 68010 CPU only.

Table 5-14 Trace Group Full Command Syntax Summary

These commands are used to trace program execution in realtime, with the emulation processor free running. Two bus breakpoints, three execution breakpoints and an external signal event can be set singly or in combination, using flexible trigger constructs, as conditions for termination of tracing and/or free-running emulation. An optional delay allows emulation/tracing to continue for up to FFFFH machine cycles after all other trigger conditions have been met.

System activity is recorded in a trace buffer with a capacity of up to 2048 machine cycles. A qualifier may be set to specify that only cycles meeting certain bus, external and/or processor state conditions be recorded. Timing information is recorded in the trace buffer as well, beginning either at the start of the trace or at the first event (i.e., breakpoint) encountered.

A synchronization command is provided to allow simultaneous start of emulation/tracing on any number of MICE-16 68000 units connected to a multiprocessor system. Operation is completely independent after activation by the sync signal.

The contents of the trace buffer may be listed after termination of the trace. The listing can be set to begin at any recorded cycle and to include either all subsequent cycles or only those cycles meeting certain bus, external and/or processor state conditions. The timing display may be set for cycle-to-cycle interval or cumulative execution time (see INTerval Command, Section 5.3.7). Machine code recorded from the data bus may be displayed in disassembled form with the use of Source option.

Note that if it is desired to use an external signal as a trigger condition, a signal I/O cable (three are provided with your MICE unit) must be connected between the signal source and the EXT TRIGGER INPUT port on the front of the MICE-16 68000. To have external signals recorded in the trace buffer, the 9-miniprobe trace cable supplied with the MICE-16 68000 must be attached to the source points and plugged into the TRACE BITS port on the MICE. Signals input at all of these ports must be TTL compatible.

5.7.1 DISPLAY/SET/CLEAR BREAKPOINTS - Event

Display Breakpoint Settings	Event
Set Bus Breakpoints 1	Ev[ent]1 adx [datum] [status] [COunt count]
Clear Event 1	Ev[ent]1 [CLear]
Set Bus Breakpoint 2	Ev[ent]2 adx [datum] [status]
Clear Event 2	Ev[ent]2 [CLear]
Set External Hardware Breakpoint	Ev[ent]3 {High|Low}
Clear Event 3	Ev[ent]3 [CLear]
Set Execution Breakpoint 1	Ev[ent]4 adr [SP|UP]*
Set Execution Breakpoint 1	Ev[ent]4 adr [SP|UP] [Vector-Base value]**
Clear Event 4	Ev[ent]4 [CLear]
Set Execution Breakpoint 2	Ev[ent]5 adr [SP|UP]*
Set Execution Breakpoint 2	Ev[ent]5 adr [SP|UP] [Vector-Base value]**
Clear Event 5	Ev[ent]5 [CLear]
Set Execution Breakpoint 3	Ev[ent]6 adr [SP|UP]*
Set Execution Breakpoint 3	Ev[ent]6 adr [SP|UP] [Vector-Base value]**
Clear Event 6	Ev[ent]6 [CLear]

Event is the command to display, set or clear breakpoints.

1,2,3,4,5,6 specifies the breakpoint to be set or cleared.

adx is a hexadecimal address setting, with option to include wildcard nibbles/bits, e.g. 1234, 1XX4, 1(X1X0)X4, 1(X1X0X0XX)4.

datum is a byte, word or long-word in hexadecimal with wildcard nibbles/bits option (e.g. 1234, 1XX4, 1(X1X0)X4, 1(X1X0X0XX)4), to be matched on the data bus; may include wildcard nibbles and /or bits, or is specified as "X" (don't care) if only 'status' and/-or 'count' must be matched along with the breakpoint address.

status is 1 to 7 processor status codes (SP SR SW UP UR UW AK) specifying processor state to be matched as part of the breakpoint condition.

COunt is the prefix to alert MICE that a "count" value is to be entered.

count is a hexadecimal value from 0H to 0FFFFH specifying the number of times the breakpoint condition must be matched to complete the specific event (Event 1). Default is for 01H.

CLear is the parameter to clear a particular breakpoint setting or to clear all of them at once.

High|Low specifies the signal level to be matched at the EXT TRIGGER INPUT port for Event 3. Default is for a TTL "low" signal.

adr is a hexadecimal address setting in the program memory space of the emulation processor.

* Applies to 68000 CPU only.
** Applies to 68010 CPU only.

SP|UP specifies the only two processor states (out of seven shown in Table 5-3) recognized by Events 4, 5 and 6.

Vector-Base value is a hexadecimal value of vector-base register for processing "exception" vector address on 68010 CPU only. New value is only assigned when user's program required such value to change. If omitted, MICE will then use the current vector-base register contents (set during Event command setting) as default value. If vector-base value is changed during program execution and the new value does not match with the default or previously set value, Events 4, 5 and 6 will not break at the set address. Also the resulting program execution will be indecipherable.

Breakpoints only affect realtime emulation/tracing, not operation, during the Cycle Step or Instruction Step commands. Any breakpoints that are set, therefore, become active only after input of the Go command (Section 5.7.2). In addition, Events 1 to 3 affect emulation/tracing only as specified in the Trigger setting (Section 5.7.9). The power-on default setting of the MICE-16 68000 is: Events 1, 2, 4, 5, and 6 clear, Event 3 set for a "low" signal, Trigger set for Event 1 OR Event 2. Events 4, 5 and 6, if set, are automatically combined with each other and the Trigger setting in a logical OR construct; the Trigger setting can be cleared, however, in order to deactivate Events 1 to 3. Events 4, 5 and/or 6 can be cleared with Event Clear command.

5.7.1.1 DISPLAY BREAKPOINT SETTINGS - Event

Event

Inputting "Event" alone, without any parameters, causes the MICE-16 68000 to display all current breakpoint settings.

Example: Display the current settings for Events 1 to 6.

```
>Event<CR>
EV1 (bus) clear
EV2 (bus) clear
EV3 (external) Low
EV4 (execution) clear
EV5 (execution) clear
EV6 (execution) clear
>
```

5.7.1.2 SET BUS BREAKPOINTS - Event 1, Event 2

```
Ev[ent ]1 adx [datum] [status] [COunt count]
Ev[ent ]2 adx [datum] [status]
```

Events 1 and 2 are realtime bus breakpoints. They consist of up to three (Event 2) or four (Event 1) elements: address, data, status, and count (Event 1 only).

An address must be specified; all other parameters are optional. When specified, data-bus value and processor status must be matched in the same machine cycle as the breakpoint address. If more than one status is included, any one of those specified will satisfy the breakpoint condition. If none is given, the breakpoint may be encountered during any type of processor activity.

For Event 1, an optional count may be specified to indicate the number of times the breakpoint condition must be matched in order to complete the event. If no count is specified, the default is 01H and the event will be encountered at the first match.

Whenever Event 1 or 2 is matched, an external sync signal is output. A negative signal pulse of 200 ns duration is generated at the SYNC 1 OUTPUT port for Event 1 and at the SYNC 2 OUTPUT port for Event 2. These signals can be sent to an external device such as an oscilloscope or a logic analyzer. For further details refer to Figure C-3, Output Signal Timing diagram. For instructions on generating a continuous SYNC 2 output, refer to the description of the Set Trigger command (Section 5.7.9).

If the Trigger setting is matched at a bus breakpoint, the MICE will stop tracing at the end of the current cycle or the next cycle, depending on CPU speed.

NOTE

Bus breakpoints match bus activity, not the program counter. If the desired breakpoint address immediately follows any jump or branch instruction, set the actual breakpoint 2 or 3 words beyond the program counter. This is not necessary for execution breakpoints (Events 4, 5 and 6) because they match execution activity in the microprocessor, breaking emulation only when the instruction at the specified address is actually to be executed and not when it is just prefetched.

Example 1: Set Event 1 for address 8050H and a count of 5; set Event 2 for any address in the range of 10H to 1FH, data of 41H and status of SW; then display the current breakpoint settings. (Remember that Events 1 to 3 are not active unless specified in the Trigger setting, the default for which is Event 1 OR Event 2.)

```
>Event 1 8050 COunt 5<CR>
>Event 2 1X 41,SW<CR>
>Event<CR>
 EV1 (bus) 008050 XXXX Count 0005
 EV2 (bus) 00001X 0041 SW
 EV3 (external) Low
 EV4 (execution) clear
 EV5 (execution) clear
 EV6 (execution) clear
>
```

Example 2: Set Event 1 for any location X1200H in any 1M-byte memory block in the 68000's memory range, and Event 2 for any location X1200H within the first 1M-byte block. Also set a count of 20 and a data value with multiple wildcards for Event 1.

```
>Event 1 X1200 12(X01X)2X(XXX0) COunt 20<CR>
>Event 2 0X1200<CR>
>Event<CR>
 EV1 (bus) XX1200 0012(X01X)2X(XXX0) Count 20
 EV2 (bus) 0X1200 XXXX
 EV3 (external) Low
 EV4 (execution) clear
 EV5 (execution) clear
 EV6 (execution) clear
>
```

5.7.1.3 SET EXTERNAL HARDWARE BREAKPOINT - Event 3

Ev[ent]3 {High|Low}

The external hardware breakpoint can be set to match either high/floating or low signal input. Note, however, that a setting of high/floating will be matched immediately if the EXT TRIGGER INPUT port is not connected to an appropriate signal source. The system default for Event 3 after power-on or a hardware reset is low. Event 3 logic is TTL compatible; a match occurs when the specified logic state is detected. For further details refer to Appendix C, Breakpoint Timing.

NOTE

The input signal must meet the specifications for a 74LS14 Schmitt trigger IC.

Example: Set Event 3 for a high signal level at the EXT TRIGGER INPUT port (remember that the setting does not become effective until Event 3 is specified in the Trigger setting).

```
>Event 3 High<CR>
>
```

5.7.1.4 SET EXECUTION BREAKPOINTS - Event 4, Event 5, Event 6

Set Execution Breakpoint 1	Ev[ent]4 adr [SP|UP]*
Set Execution Breakpoint 1	Ev[ent]4 adr [SP|UP] [Vector-Base value]**
Set Execution Breakpoint 2	Ev[ent]5 adr [SP|UP]*
Set Execution Breakpoint 2	Ev[ent]5 adr [SP|UP] [Vector-Base value]**
Set Execution Breakpoint 3	Ev[ent]6 adr [SP|UP]*
Set Execution Breakpoint 3	Ev[ent]6 adr [SP|UP] [Vector-Base value]**

The MICE-16 68000 provides three realtime execution breakpoints consisting of a required address. Tracing is always terminated and the emulation processor halted when an execution breakpoint is encountered.

When either Event 4, 5, or Event 6 is matched, the MICE halts the emulation processor at the cycle specified by the break condition. All machine cycles up to the breakpoint are recorded, including prefetch cycles.

Execution breakpoints remain effective. Unlike bus breakpoints, these breakpoints end the trace at a point in the program where the specified address is actually going to be executed.

The following program segment illustrates the difference between execution breakpoints and bus breakpoints:

```
>Disassemble 8000 801E<CR>
LOC        OBJ              SOURCE CODE
008000     1A3C 0010        MOVE.B   #10,D5
008004     207C 0000 1000   MOVEA.L  #00001000,A0
00800A     227C 0000 2000   MOVEA.L  #00002000,A1
008010     3018             MOVE.W   (A0)+,D0
008012     6100 0FEC        BSR.W    009000
008016     32C1             MOVE.W   D1,(A1)+
008018     5305             SUBQ.B   #01,D5
00801A     66F4             BNE.B    008010
00801C     4E71             NOP
00801E     60FE             BRA.B    00801E
 Disassembly completed
>
```

Setting a bus breakpoint at instruction NOP (801C) in the above example would cause the program to break at a bus prefetch cycle, even though program execution has not reached the breakpoint. If the same breakpoint address is set using an execution breakpoint, the program will break only if execution reaches the breakpoint address.

NOTE

1) *When the program stops at the breakpoint address, the user's program instruction at that location has not yet been executed. The same address cannot be specified again. Setting any breakpoint where the PC equals the specified address will halt tracing immediately.*

* Applies to 68000 CPU only.
** Applies to 68010 CPU only.

2) Execution breakpoints must be set at the first word of an instruction.

3) For Events 4, 5 and 6, emulation is controlled with 68000 line emulator code 0A0H.

4) Stack contents from SP-1 thru SP-6 (SP-8 for 68010) will be modified if the breakpoint is executed; but the USP and SSP registers will not be affected.

5) The code for breakpoint addresses is modified to 0A0H 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.

6) 0A0H instructions defined in user's program are executed correctly without any conflict with internal 0A0H codes generated by Event 4, 5 or 6.

7) If any supervisor data memory read cycle occurs at 028H (or a line emulator code 0A0H in the user's program is executed), a few non-real time cycles will be inserted to process this code.

8) The execution breakpoint can be set to RAM/ROM area.

9) If the execution breakpoint is set and the bus breakpoint is matched, the CPU will finish executing the current instruction cycle and stop at the opcode fetch of the next instruction.

Example: Set the bus breakpoint 1 at address 801C and display the register, then clear the bus breakpoint. Set the execution breakpoint to the same address and display register again.

```
>EVent CLear<CR>
>Event 1 801C SP<CR>
>Go 8000<CR>
>Trigger EV1<CR>
 EV1 (bus) Encountered
 Emulation processor stopped
 Last frame = 001A,Timer = 00.015 619
>Register<CR>
       0        1        2        3        4        5        6        7
Dn  0003EDEC 0004F0EE FFFFFDFC FFFFFFFF FFFFFF00 FFFFFF0F 00000001 FFFFFFFF
An  00001004 00002002 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 00001000
SSP=00001000 USP=FFFFFFFF                SR   T S  III   XNZVC
PC=008012                                2708 0010011100001000
>
```

```
>Trigger CLear<CR>
>Event 4 801C<CR>
>Go 8000<CR>
 Trace in progress...
 EV4 (execution) Encountered
 Emulation processor stopped
 Last frame = 0128,Timer = 00.000 000
>Register<CR>
      0        1        2        3        4        5        6        7
Dn  0003F2F0 0004F2F0 FFFFFDFC FFFFFFFF FFFFFF00 FFFFFF00 00000001 FFFFFFFF
An  00001020 00002020 0003FFFF 7B7B7A7A 00800110 00000000 FFFFFFFF 00001000
SSP=00001000 USP=FFFFFFFF                SR   T S  III   XNZVC
PC=00801C                                2704 0010011100000100
>
```

5.7.1.5 CLEAR BREAKPOINTS - Event CLear

```
Ev[ent ]1 [CLear]
Ev[ent ]2 [CLear]
Ev[ent ]3 [CLear]
Ev[ent ]4 [CLear]
Ev[ent ]5 [CLear]
Ev[ent ]6 [CLear]
```

Inputting "Event Clear" alone, without any other parameters, clears the settings for Events 1, 2, 4, 5, and 6, and sets Event 3 to low. To clear Event 'n' only, input "Event 'n' Clear."

Example 1: Display all breakpoint settings, then clear execution breakpoint Event 4.

```
>Event<CR>
 EV1 (bus) clear
 EV2 (bus) clear
 EV3 (external) low
 EV4 (execution) 00801C
 EV5 (execution) clear
 EV6 (execution) clear
>Event 4 CLear<CR>
>Event<CR>
 EV1 clear
 EV2 clear
 EV3 (external) low
 EV4 (execution) clear
 EV5 (execution) clear
 EV6 (execution) clear
>
```

Example 2: Clear all breakpoints, then display the breakpoint setting.

```
>Event CLear<CR>
>Event<CR>
 EV1 (bus) clear
 EV2 (bus) clear
 EV3 (external) low
 EV4 (execution) clear
 EV5 (execution) clear
 EV6 (execution) clear
>
```

5.7.2 GO/EXECUTION - Go

Go [Run] [adr]

Go is the command for Go/Execution.

Run specifies free run. During free run all breakpoint settings are ignored, but target information is still recorded in the trace buffer. (Any command other than Halt, Memory, Port and emulation commands may be input without stopping the emulation processor.)

adr is the hexadecimal address setting in the emulation CPU's program memory where emulation is to begin. Default is to begin from current PC.

If an address is specified, the emulation processor's program counter is first set accordingly. The MICE-16 68000 then starts realtime emulation of the emulation processor, and immediately begins recording system status information as specified in the Trace Cycle Qualify setting. If no address is specified, emulation starts at the current program counter. Note that the green "EP RUN" indicator on the front panel is lit whenever the emulation CPU is active. The recorded information can be viewed with List Trace Buffer commands.

During emulation,

"Trace in progress..."

is displayed as long as data is being recorded in the trace buffer. When the Trigger condition is matched, the message-

"EV# encountered"

is displayed. If an attempt is made to write into Read-only area (defined as "IR" or "ER" in the MAp command (Section 5.3.8), the message-

"...Write protected memory trespassed!"

will display. Likewise, if a non-existent area (defined as "G" in the MAp command is accessed, the following message will display-

...Guarded memory trespassed!"

If the Run option was not specified for either the Trigger setting or Event 4, the additional message-

"Emulation processor stopped"

will also display. If no breakpoint is encountered (i.e., "Trace in progress..." remains on the screen), the user may terminate emulation at any time by pressing <ESC>; the message-

"Emulation processor stopped by user"

will then be displayed.

Note however, that if several breakpoints are closely matched simultaneously, the order in which they are listed in the "EV# encountered" message may not reflect the actual order in which they were encountered during emulation. This is especially true when both execution and bus/signal breakpoints are matched almost simultaneously: because of the higher priority of execution breakpoints, Events 4 and 5 will be displayed before Events 1 to 3 even if they were encountered slightly later.

Example 1: Set Event 1 to address 8010H and Trigger to Event 1 with center trace. Set timebase to 1*us* and the timer at the beginning of emulation.

```
>Event 1 8010<CR>
>Trigger EV1 CENter<CR>
>TImebase<CR>
 Timebase=1 us, timer starts from first event
>TImebase GO 1<CR>
>Go 8000<CR>
 EV1 (bus) Encountered
 Trace in progress...
 Emulation processor stopped
 Last frame = 0408,Timer = 00.000 408
>
```

The time that elapses before the trace buffer is filled depends on command specifications, and may be quite lengthy. If the trigger condition has not yet been matched, you may enter an <ESC> to terminate the trace, retaining the information already recorded in the trace buffer.

Example 2: Set the trigger for a low signal at the external trigger input port, and when this is not matched enter an <ESC> to terminate tracing.

```
>Event 3 Low<CR>
>TRigger EV3<CR>
>TRigger<CR>
 Trigger EV3 delay 0000
>Go 8000<CR>
 Trace in progress...<ESC>
 Emulation processor stopped by user
 Last frame = 07FF, timer = 02.709 591
>
```

Example 3: Specify an "And" trigger construct for the 5th supervisor data memory write into address range 0H-FFH, and a supervisor data memory read of 8020H; then qualify the cycles to be traced as SP.

```
>Event CLear<CR>
>Event 1 XX SW COunt 5<CR>
>Event 2 8020 SP<CR>
>TRigger EV1 And EV2<CR>
>Qualify X SP<CR>
>Go 8000<CR>
 EV1 (bus) Encountered
 Trace in progress....
 EV2 (bus) Encountered
 Emulation processor stopped
 Last frame = 00CA,Timer = 00.000 102
>
```

Example 4: The emulation processor can be free run by using the "Run" option, which masks all breakpoint settings. (Target information is still recorded in the trace buffer during free run.) In this state, emulation may be stopped by entering a Halt, Register, Cycle Step or Instruction Step command. Entering a List command will stop the trace but leave the emulation processor running.

```
>Go Run 8000<CR>
 Emulation processor free running, all breakpoints masked
Run>HALt<CR>
 Emulation processor stopped by user
>
```

Example 5: By clearing the Trigger setting, the emulation processor can also be free run without using the "Run" option. (Target information is still traced.) In this state, emulation can only be stopped by entering <ESC>.

```
>TRigger CLear<CR>
>Go 8000<CR>
 Trace in progress...<ESC>
 Emulation processor stopped by user
 Last frame = 07FF, timer = 02.051 072
>
```

5.7.3 HALT - HAlt

HAlt

HAlt is the command to halt the emulation processor.

This command may be used during free-run emulation to stop program execution at any time. Inputting such commands as Register, Cycle, Step, Memory, Input and Output will also halt emulation. Note that the green "EP RUN" indicator on the front panel is lit whenever the emulation processor is active.

The Halt command may also be used after tracing has been stopped by the specified trigger condition, and the emulation processor is in free run state due to a "Run" qualifier in the trigger setting or in the breakpoint setting for Event 4.

Example: Input the Halt command during emulation to stop the emulation processor.

```
>RESet<CR>
>Event CLear<CR>
>Event 1 1020 SW<CR>
>Trigger Run EV1
>Go 8000<CR>
 Trace in progress....<ESC>
 Emulation processor stopped by user
 Last frame = 07FF,Timer = 03.300 627
Run>HAlt<CR>
 Emulation processor stopped by user
>
```

5.7.4 LIST TRACE BUFFER - List

List Trace Results	List [frame [adr1 adr2] [status] [trace-bits]]
List Trace Results with Source/Instruction Code	List {Source|Instruction} [frame]
List Total Frames and Time	List Number

List is the command to display information recorded in the trace buffer. This information is organized into frames, each representing one execution cycle, and displayed one page (screenful) at a time. Address and data bus activity, processor status, trace bit levels and timing information are displayed for each frame. "List" alone, or "List Source" with no other parameters, displays all frames in the last page of the buffer.

frame is a hexadecimal value (from 0H to 07FFH) indicating the frame at which the listing is to begin. The default is for the first frame in the last page of the buffer.

adr1,adr2 are hexadecimal values specifying a particular range of memory, and indicating that only those cycles with address bus activity within that range are to be listed. "adr1" indicates the starting address, "adr2" the end address. Default is 0H to FFFFFFH.

status is 1 to 7 processor status codes (SP SR SW UP UR UW AK) representing specific type of processor activity (see Table 5-3) and indicating that only cycles in which the indicated type of activity take place are to be listed. The default is for all cycles regardless of processor status.

trace-bits is a one-byte hexadecimal setting indicating signal levels for trace bits input from the external port specifying that only cycles in which the indicated levels are matched are to be listed. This setting includes option to specify wildcard nibble /bit e.g., "XX" "FX" "F(10XX)" etc., where "X" is a nibble or bit indicating 'don't care'. The default is for all cycles regardless of trace bit levels.

Source is the parameter to list the trace buffer containing frame, address, data, status, trace-bits, timer and assembly source-code. Note that the source code listing must start at an instruction fetch cycle, otherwise incorrect code will be displayed for the first few machine cycle.

Instruction is same with "Source" parameter above. Only the trace buffer contents to be listed will not include status, trace-bits, timer and read/write cycle.

Number displays the number of the last frame recorded in the trace buffer and the overall elapsed time since timer start.

Trace information recorded during emulation is displayed using the List command. If a frame number is specified, the MICE lists the trace buffer starting at the specified frame. If no frame number is specified, the MICE lists all frames in the last page of the buffer. Qualifiers for address range, processor activity and/or external signal levels can optionally be entered as described above so that only cycles in which the specified condition is matched are listed.

One page (i.e., one screenful) of trace information is displayed at a time; press <CR> to display the next page and <SP> for continuous scrolling. To terminate the List operation press <ESC>. Note that the command also ends if the trace buffer is empty or a frame number higher than that of the last recorded frame is specified.

Example: Run a trace up to address 8020H and list the last page in the trace buffer.

```
>Event CLear<CR>
>Qualify CLear<CR>
>Event 1 801E<CR>
>Trigger EV1<CR>
>Go 8000<CR>
 Trace in progress...
 EV1 (bus) Encountered
 Emulation processor stopped
 Last frame = 0129,Timer = 00.000 102
>List<CR>
FRAME ADDRESS     DATA     STATUS   TRACE BIT  TIMER
011A  000FFC      0000       SW     11111111   00.000 097
011B  000FFE      8016       SW     11111111   00.000 097
011C  009000      0640       SP     11111111   00.000 097
011D  009002      0100       SP     11111111   00.000 098
011E  009004      3200       SP     11111111   00.000 098
011F  009006      4E75       SP     11111111   00.000 098
0120  009008      6766       SP     11111111   00.000 099
0121  000FFC      0000       SR     11111111   00.000 099
0122  000FFE      8016       SR     11111111   00.000 099
0123  008016      32C1       SP     11111111   00.000 100
0124  008018      5305       SP     11111111   00.000 100
0125  00201E      F2F0       SW     11111111   00.000 100
0126  00801A      66F4       SP     11111111   00.000 101
0127  00801C      4E71       SP     11111111   00.000 101
0128  00801E      60FE       SP     11111111   00.000 102
0129  008020      5F5E       SP     11111111   00.000 102
>
```

5.7.4.1 LIST TRACE RESULTS - List

List [frame [adr1 adr2] [status] [trace-bits]]

Example 1: After the trace operation in the previous example, list all frames in the first page of the trace buffer.

```
>List 0<CR>
FRAME ADDRESS     DATA     STATUS   TRACE BIT   TIMER
0000  008000      1A3C       SP     11111111    00.000 000
0001  008002      0010       SP     11111111    00.000 000
0002  008004      207C       SP     11111111    00.000 000
0003  008006      0000       SP     11111111    00.000 000
0004  008008      1000       SP     11111111    00.000 000
0005  00800A      227C       SP     11111111    00.000 000
0006  00800C      0000       SP     11111111    00.000 000
0007  00800E      2000       SP     11111111    00.000 000
0008  008010      3018       SP     11111111    00.000 000
0009  008012      6100       SP     11111111    00.000 001
000A  001000      EFEE       SR     11111111    00.000 001
000B  008014      0FEC       SP     11111111    00.000 001
000C  000FFC      0000       SW     11111111    00.000 002
000D  000FFE      8016       SW     11111111    00.000 002
000E  009000      0640       SP     11111111    00.000 002
000F  009002      0100       SP     11111111    00.000 003
[ More ]<ESC>
>
```

Example 2: List the trace buffer starting at frame 0, and limit the listing to supervisor program fetch cycles in the memory range of 8000H to 10000H.

```
>List 0 8000 10000 SP<CR>
FRAME ADDRESS     DATA     STATUS   TRACE BIT   TIMER
0000  008000      1A3C       SP     11111111    00.000 000
0001  008002      0010       SP     11111111    00.000 000
0002  008004      207C       SP     11111111    00.000 000
0003  008006      0000       SP     11111111    00.000 000
0004  008008      1000       SP     11111111    00.000 000
0005  00800A      227C       SP     11111111    00.000 000
0006  00800C      0000       SP     11111111    00.000 000
0007  00800E      2000       SP     11111111    00.000 000
0008  008010      3018       SP     11111111    00.000 000
0009  008012      6100       SP     11111111    00.000 001
000B  008014      0FEC       SP     11111111    00.000 001
000E  009000      0640       SP     11111111    00.000 002
000F  009002      0100       SP     11111111    00.000 003
0010  009004      3200       SP     11111111    00.000 003
0011  009006      4E75       SP     11111111    00.000 003
0012  009008      6766       SP     11111111    00.000 004
[ More ]<ESC>
>
```

5.7.4.2 LIST TRACE RESULTS WITH SOURCE/INSTRUCTION CODE - List

List {Source|Instruction} [frame]

Example 1: Starting at frame 0H, list the traced data, displaying mnemonic code together with machine activity.

```
>List Source 0<CR>
FRAME ADDRESS     DATA      STATUS   TRACE BIT  TIMER
0000  008000      1A3C        SP     11111111   00.000 000
          MOVE.B   #10,D5
0001  008002      0010        SP     11111111   00.000 000
0002  008004      207C        SP     11111111   00.000 000
          MOVEA.L  #00001000,A0
0003  008006      0000        SP     11111111   00.000 000
0004  008008      1000        SP     11111111   00.000 000
0005  00800A      227C        SP     11111111   00.000 000
          MOVEA.L  #00002000,A1
0006  00800C      0000        SP     11111111   00.000 000
0007  00800E      2000        SP     11111111   00.000 000
0008  008010      3018        SP     11111111   00.000 000
          MOVE.W   (A0)+,D0
0009  008012      6100        SP     11111111   00.000 001
          BSR.W    009000
000B  008014      0FEC        SP     11111111   00.000 001
[ More ]<ESC>
>
```

Example 2: Starting at frame 00H, list the address, traced data and assembly source code, displaying mnemonic code together with machine activity.

```
>List Instruction 0<CR>
FRAME  ADDRESS     DATA      SOURCE CODE
 0000  008000      1A3C      MOVE.B   #10,D5
 0002  008004      207C      MOVEA.L  #00001000,A0
 0005  00800A      227C      MOVEA.L  #00002000,A1
 0008  008010      3018      MOVE.W   (A0)+,D0
 0009  008012      6100      BSR.W    009000
 000E  009000      0640      ADDI.W   #0100,D0
 0010  009004      3200      MOVE.W   D0,D1
 0011  009006      4E75      RTS
 0015  008016      32C1      MOVE.W   D1,(A1)+
 0016  008018      5305      SUBQ.B   #01,D5
 0018  00801A      66F4      BNE.B    008010
 001A  008010      3018      MOVE.W   (A0)+,D0
 001B  008012      6100      BSR.W    009000
 0020  009000      0640      ADDI.W   #0100,D0
 0022  009004      3200      MOVE.W   D0,D1
 0023  009006      4E75      RTS
[ More ]<ESC>
>
```

WARNING

When displaying mnemonic code with the List Source command, the following conditions may cause incorrect information to be listed for the initial frame:

1) The machine cycle recorded in the initial frame is not a fetch cycle for the beginning of an instruction.

2) The specified starting frame contains prefetch instructions that were not executed due to a program branch.

5.7.4.3 LIST TOTAL FRAMES AND TIME - List Number

List Number

The MICE-16 68000 displays "Last frame = " followed by the number of the last frame recorded in the trace buffer, and "timer = " followed by the total elapsed time since timer start. (Note that the frame count begins at zero, so the total number of frames is one more than the value displayed. If

"Last frame = 07FF"

is displayed, the trace buffer is full.)

Example: Display the number of the last frame in the trace buffer and the total elapsed time since timer started.

```
>List_Number<CR>
 Last frame = 0129,timer = 00.000 102
>
```

5.7.5 CYCLE QUALIFY - Qualify

Display Cycle Qualifiers	Qualify
Set Trace Cycle Qualifiers	Qualify adx [status]
Clear Cycle Qualifier	Qualify [CLear]

Qualify displays, sets or clears the cycle qualifier, which specifies which machine cycles are to be recorded in the trace buffer. "Qualify" alone, without any parameters, displays the current Cycle Qualify setting.

adx is a hexadecimal address setting with option to include wildcard nibbles/bits, e.g. 1234, 1XX4, 1(X1X0)X4, 1(X1X0X0XX)4.

status is 1 to 7 processor status codes (SP SR SW UP UR UW AK) indicating the type of processor activity to be recorded.

CLear is the parameter to clear the Cycle Qualify setting.

The power-on default setting of the MICE-16 68000 is to record all machine cycles during a trace operation. Cycle Qualify is used to limit recording to cycles in which certain areas of program memory are accessed and/or certain types of processor activity take place. This makes it possible to record more useful information in the trace buffer with fewer trace commands.

Example 1: Set Cycle Qualify for all SP and SR cycles between 82000H and 82070H, then display the current setting.

```
>Qualify 820(0XXX)0 SP SR<CR>
>Qualify<CR>
 Cycle qualifier: 0820(0XXX)0 SP,SR
>
```

Example 2: Clear the current Cycle Qualify setting, and thereby enable the recording of all machine cycles in subsequent trace commands.

```
>Qualify CLear<CR>
>Qualify<CR>
 Cycle qualifier: all addresses and status
>
```

5.7.6 SET SYNCHRONIZATION - SYnc

SYnc [{Input|Output} {On|OFf}]

SYnc displays or sets synchronization. "SYnc" alone, without any parameters, displays the current sync setting.

Input|Output specifies synchronization of emulation in a multiprocessor environment.

On|OFf is the parameter to set or clear synchronization.

This setting only specifies synchronization conditions. It is not effective until the Go command is input to start the emulation processor. Synchronization is not active during the Cycle Step and Instruction Step commands.

To use this multi-MICE synchronization feature, connect the START SLAVE OUT signal on the master unit to SYNC START IN on the slave unit, then input "SYnc Output On" at the master unit and "SYnc Input On" at the slave. When the master unit begins emulation it will transmit a sync signal to the slave.

After entering synchronization commands (and any required trace control commands) on all linked units, input the Go command, first on all slave units and then on the controlling unit (i.e., the one with sync input off). When a Go command is input at the controlling unit, emulation will begin simultaneously on all linked units. Each unit will then run independently until any specified Trigger condition is encountered or emulation is terminated by the user (i.e. <ESC> is input).

Example 1: Configure the emulator as an intermediate member in a daisy chain by activating both sync in and sync out signals, then display the current setting.

```
>SYnc<CR>
 Sync Start In OFF
 Start Slave Out OFF
>SYnc Input On<CR>
>SYnc Output On<CR>
>SYnc<CR>
 Sync Start In ON
 Start Slave Out ON
>
```

Example 2: Clear the sync-out signal.

```
>SYnc Output OFf<CR>
>SYnc<CR>
 Sync Start In ON
 Start Slave Out OFF
>
```

5.7.7 TIMEBASE SELECTION - TImebase

TImebase [Go|Event] [1|10|100|1000|10000]

TImebase is the command for Timebase Selection, and to specify timer start at the beginning of emulation or at the first breakpoint encountered. "TImebase" without any parameters displays the current Timebase setting.

Go|Event specifies timer start when the emulation processor begins running or when the first event is encountered.

1-10000 specifies the Timebase selection as indicated in Table 4-11 below.

A timer is provided for emulation/trace mode. Execution time for instruction cycles is displayed at the List Trace Buffer command (Section 5.7.4), with the unit of measurement determined by the Timebase selection. The default is for a timebase of 1 us, or the most recent user setting. The timer is activated when emulation starts, unless the Event option is specified to start it when any one of the events specified in the Trigger setting (EV1-EV3) is matched.

Timebase Number	1	2	3	4	5
Unit of Measurement	1 *us*	10 *us*	100 *us*	1000 *us* (1 ms)	10000 *us* (10 ms)
Maximum Timer Length	16 sec	2'40 sec	26'40 sec	4 hrs	44 hrs

Table 5-15 Timebase Selections

Example: Set the timebase to 1 us and initiate the timer when any breakpoint specified in the Trigger setting is matched; then display the current Timebase setting to verify input.

```
>TImebase Event 1<CR>
>TImebase<CR>
 Timebase = 1 us, timer starts from first event
>
```

The format for timing data displayed in the TIMER column at the List Trace Buffer, Cycle Step and Instruction commands is-

"hh:mm:ss.mmm uuu"

where: **hh** = hours
mm = minutes
ss = seconds
mmm = milliseconds
uuu = microseconds

Note that if the maximum timer length is exceeded, "------" (broken lines) will be displayed in the TIMER field of the trace buffer listing.

5.7.8 DISPLAY CURRENT TRACE PARAMETERS - TRAce

TRAce

TRAce is the command to display the current MICE trace control settings as illustrated in the example below.

Example: Display the current MICE-16 68000 trace control settings.

```
>TRAce<CR>
 EV1 (bus) 00801E XXXX Count 00001
 EV2 (bus) clear
 EV3 (external) Low
 EV4 (execution) clear
 EV5 (execution) clear
 EV6 (execution) clear
 Trigger EV1
 Cycle qualifier: all addresses and statuses
 Timebase = 1 us, timer starts from Go
 Sync start in OFF
 Start slave out OFF
>
```

5.7.9 DISPLAY/SET/CLEAR TRIGGER - Trigger

Display Trigger	Trigger
Set "Then" Trigger	Trigger [Run] # [Then # [Then #]] [BACKward|CENter|FOrward|DELay count]
Set "And" Trigger	Trigger [Run] # And # [And #] [BACKward|CENter|FOrward|DELay count]
Set "Or" Trigger	Trigger [Run] # Or # [Or #] [BACKward|CENter|FOrward|DELay count]
Set "And-Then Trigger	Trigger [Run] # And # Then #] [BACKward|CENter|FOrward|DELay count]
Set "Or-Then" Trigger	Trigger [Run] # Or # Then #] [BACKward|CENter|FOrward|DELay count]
Clear Trigger	Trigger [CLear]

Trigger specifies Trigger display/set/clear. "Trigger" alone, without any parameters, displays the current Trigger setting.

Run is an optional parameter which allows the emulation processor to free run after the trace ends.

is EV1, EV2 or EV3. Note that no event may be specified more than once in the Trigger setting.

DELay is the prefix to alert MICE that a "count" value (described below) is to be entered.

count is a hexadecimal value (0H to FFFFH) which specifies a cycle count. After all other Trigger conditions have been matched, the trace will continue until the specified number of cycles has elapsed.

BACKward is the delay count = 0.

CENter is the delay count = 03FFH.

FOrward is the delay count = 07FFH.

Trigger CLear is the command to clear the current Trigger setting.

The Trigger condition may be a single event or a logical construct of up to three breakpoints with "And, Or and Then" connectives. Only Events 1 to 3 can be used in the Trigger definition; execution breakpoints are automatically joined to the specified construct by a logical "OR". The default at power-on and after a hardware reset is the construct "Event 1 OR Event 2."

The Trigger setting only specifies the break condition. It is not effective until the Go command is input to start the emulation processor. Note also that the Trigger setting has no effect during execution of the Cycle Step and Instruction Step commands.

When emulation begins, the MICE-16 68000 immediately starts recording target system and emulation processor status (in accordance with the Cycle Qua-

lify setting) in real-time. Data are recorded until the Trigger condition, including any cycle-count delay, is matched.

Up to 2048 machine cycles can be recorded. If more than 2048 cycles elapse before tracing ends, only the last 2048 cycles will be recorded.

When the trace ends, the event(s) matched are displayed and (unless the Run option is used) the MICE stops the emulation processor one cycle beyond the location where the Trigger condition is matched. The recorded information can be examined with the List Trace Buffer command.

Note that whenever the processor is running, external hardware sync signals are generated as follows:

- at the SYNC 1 OUTPUT port the first time Event 1 is matched
- at the SYNC 2 OUTPUT port each time Event 2 is matched.

1) Trigger Display

Example: Display the current Trigger setting.

```
>Trigger<CR>
 Trigger EV1 OR EV2
>
```

2) Trigger Setting

Only Events 1 to 3 may be specified in the Trigger setting. The following rules apply for Trigger constructs:

a) A specific breakpoint may not be repeated within a construct.

b) There are no special syntax requirements for settings with no more than one breakpoint.

c) Input breakpoints as indicated in the following table. Use the connective "Then" to indicate a required sequence, "And" to indicate a multiple breakpoint requirement without regard to sequence, and "Or" to indicate that either of two events, or any one of three, will satisfy the Trigger requirement. All legal Trigger constructs are shown below. "EV#" indicates Event 1, 2 or 3.

Single	EV#
Arm (Then)	EV# Then EV# EV# Then EV# Then EV#
And	EV# And EV# EV# And EV# And EV#
Or	EV# Or EV# EV# Or EV# Or EV#
Combination	EV# And EV# Then EV# EV# Or EV# Then EV#

Table 5-16 Trigger Constructs

Example 1: Specify Event 1 as the trigger, use a delay of FFH cycles and continue emulation after the trace stops.

```
>TRigger Run EV1 Delay FF<CR>
>TRigger<CR>
 Trigger Run EV1 Delay 00FF
>
```

Example 2: Specify a combination Trigger construct using a delay of 10FFH.

```
>TRigger EV1 Or EV2 THen EV3 Delay 10FF<CR>
>TRigger<CR>
 Trigger EV1 OR EV2 Then EV3 Delay 10FF
>
```

3) Trigger Clear

Example: Clear the current Trigger setting.

```
>TRigger CLear<CR>
>TRigger<CR>
 Trigger clear
>
```

Chapter 6

APPLICATION NOTES

6.1 TRACE AND EMULATION

1) To control emulation of the processor, the MICE-16 shares the use of $\overline{\text{HALT}}$ and $\overline{\text{RESET}}$. When a double bus fault occurs, the MICE will stop the processor and display the message-

"Emulation processor halted"

2) If $\overline{\text{DTACK}}$ on the target is inactive or $\overline{\text{BR}}$ on the target is active for over 2.5 seconds during trace or emulation processor free run, inputting a Register or Instruction Step command will generate the message-

"Emulation processor can't step -- check clock, BR, DTACK*"

Use the Reset command to recover.

3) The contents of the stack pointers USP and SSP must be even, or inputting a Register or Instruction Step command will result in the message-

"Emulation processor can't step -- check clock, BR, DTACK*"

Use the Reset command to recover.

4) The trace buffer latch time for the address and status bus is at the end of a processor $\overline{\text{STROBE}}$ signal, such as $\overline{\text{LDS}}$, or $\overline{\text{UDS}}$. Trace bits are only read into the buffer at this moment in each cycle. The trace buffer latch time for the data bus is at the end of a delayed $\overline{\text{STROBE}}$ signal.

5) When emulation is stopped at the specified address by an execution breakpoint, the user's program instruction at that location has not yet been executed. The breakpoint defining this location should be cleared before continuing emulation, i.e., the same trigger address should not be used again. If a breakpoint is set where the program address equals the trigger address, tracing will stop immediately. "Cycle" or "Step" command can be used to forward the processor away from breakpoint location.

6) Execution breakpoints must be set at the first byte of the instruction.

7) Execution Breakpoint on target program memory can be RAM/ROM.

8) If emulation breaks in the middle of a multi-cycle instruction, executing the Register command will finish executing the current instruction and then display register contents. Executing the Cycle Step command, however, will display cycle status starting at the current break location.

9) For Event 4, Event 5 and Event 6, emulation is controlled through the use of 68000 line emulator code 0A0H. If any supervisor data memory read cycle occurs at 28H (or the line emulator code 0A0H in the user's program is executed), a few non-real time cycles will be inserted to process this code.

10) The CPU will not halt during single Step command execution with the "STOP" instruction, even when there is no interrupt (either INT or "NMI") execution requested. Under this circumstances, other commands, such as "Go" or "Cycle" commands has to be executed to stop the CPU.

11) The MICE-16 68000 uses level seven (non-maskable) interrupt requests for internal control during execution of the Instruction Step command. We recommend that the user avoid using level seven interrupts during Instruction Step.

6.2 HIDDEN COMMANDS

1) **"NMI"** - is used to force NMI signal into CPU and at the same time, auto-execute cycle step.

2) **"SIGNALS"** - is used to read status of the more significant target signals, i.e. $\overline{HALT}$, $\overline{RESET}$, $\overline{BR}$, $\overline{DTACK}$, $\overline{BERR}$, $\overline{BGACK}$ and $\overline{VPA}$.

3) **"WRITE [On|OFf]** - is to enable MICE internal hardware to block (change to "read cycles") or un-block "write cycles" before MICE inserts the NMI internally while executing the instruction Step command. The default is "WRITE OFf" (blocked or change to "read cycles").

EXAMPLE: Use Instruction Step to execute a user program to program the peripheral Z8536 CIO (Count/Timer and Parallel I/O), which has an internal state machine. The state diagram is described below:

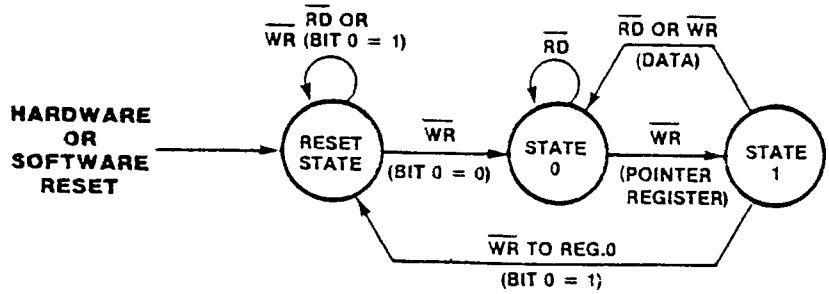


Figure 6-1 State Machine Operation

If "WRITE OFf" is entered, the accessed internal register can NOT be written correctly but can be read correctly because a read to this internal register appears at State 1 and then return to State 0 before the real "write cycle" is implemented in the internal register.

If "WRITE On" is entered, the internal register can be written correctly but the stack contents SP-1 and SP-2 will be modified. However, the SSP and USP register will not be affected.

APPENDIX A

WARNING AND ERROR MESSAGES

1. **"<board> <device> (#) failure <location>"**

   A component failure was detected during the power-on self-test routine. The module board and device where the failure occurred are displayed, along with device number and memory location for some malfunctions.

   This message may indicate that the MICE-16 68000 requires service. If the problem cannot be resolved, contact your nearest Microtek distributor for assistance.

2. **"Command syntax error"**

   Illegal parameter input follows the command keyword. This may mean a misspelling, incorrect order, or an illegal argument (for instance, a word value specified for port output when memory access size is set to byte). Check Command Help or the appropriate section of this user's manual for the correct syntax.

3. **"Command word error"**

   The MICE-16 68000 does not recognize the first word in the command line just input. Check Command Help or Section 5.3.5 of this manual for a list of legal command keywords.

4. **"Emulation processor is halted"**

   A double bus fault occurred and MICE-16 68000 halts the processor.

5. **"Frame number exceeds trace buffer frame count"**

   The starting frame number specified in the List Trace Buffer command is greater than the number of the last frame in the trace buffer.

6. **"...Guarded memory trespassed!"**

   An attempt is made to write a non-existence or guarded memory block .

7. **"Memory comparison terminated -- difference found at:**
   (xxxxxx) = xxxx
   (xxxxxx) = xxxx"

   Compared memory blocks do not match at the addresses shown.

8. **"Memory fill terminated -- write failure at xxxxxx"**

   Mismatch occurred while MICE-16 68000 automatically checked data immediately after they have been written to memory during Memory Verification.

9. **"Memory write failure at xxxxxx"**

   Data cannot be written to the specified memory location. Verify that the address is valid and that the location is write-enabled.

10. **"No target VCC; BERR, BR, INTR (IPLO-2) disabled"**

    Target is not connected to MICE or power is not supplied to the target at the time of the self-test. User should perform warm reset (Section 3.4) after target is installed and powered-on.

11. **"NOVRAM parameters inconsistent**
    Proper default parameters have been set"

    The data stored in NOVRAM do not match with MICE-16 68000 Identification Code during execution of RECall command. When this condition occurs, MICE will setup itself with the system default setup parameters.

12. **"Operand error"**

    Specified value for either instructions DB (Define Byte), DW (Define Word) or DS (Define Storage) exceeds maximum limit.

13. **"No Target DTACK* at address xxxxxx"**

    Target memory or input/output ready signal is inactive for over 2.5 seconds. Use a valid physical location in the command specification.

14. **"...Write protected memory trespassed"**

    An attempt is made to write a write protected memory block .

15. **"Error! Bank already mapped by another address range"**

    Two different memory address ranges are mapped to the same memory bank.

APPENDIX B

MNEMONICS AND INSTRUCTIONS

B.1 68000/68010 Mnemonics

ABCD	Bcc	CMP	EXT	MOVEC	NOP	RTD	SUBQ
ADD	BCHG	CMPA	ILLEGAL	MOVEM	NOT	RTE	SUBX
ADDA	BCLR	CMPI	JMP	MOVEP	OR	RTR	SWAP
ADDI	BKPT	CMPM	JSR	MOVEQ	ORI	RTS	TAS
ADDQ	BRA	DBcc	LEA	MOVES	PEA	SBCD	TRAP
ADDX	BSET	DIVS	LINK	MULS	RESET	Scc	TRAPV
AND	BSR	DIVU	LSL	MULU	ROL	STOP	TST
ANDI	BTST	EOR	LSR	NBCD	ROR	SUB	UNLK
ASL	CHK	EORI	MOVE	NEG	ROXL	SUBA	
ASR	CLR	EXG	MOVEA	NEGX	ROXR	SUBI	

The symbol "cc" in a 68000 mnemonic may represent the following conditions:

Mnemonic	Condition	Mnemonic	Condition
CC	carry clear	LS	low or same
CS	carry set	LT	less than
EQ	equal	MI	minus
GE	greater or equal	NE	not equal
GT	greater than	PL	plus
HI	high	VC	overflow clear
LE	less or equal	VS	overflow set

B.2 68000 Instruction Codes in Hexadecimal with Disassembly Examples:

LOCATION	OBJECT	SOURCE CODE	
000400	C509	ABCD.B	-(A1),-(A2)
000402	C501	ABCD.B	D1,D2
000404	D602	ADD.B	D2,D3
000406	D713	ADD.B	D3,(A3)
000408	D763	ADD.W	D3,-(A3)
00040A	D75B	ADD.W	D3,(A3)+
00040C	D7AB 2000	ADD.L	D3,(2000,A3)
000410	D7B2 284E	ADD.L	D3,(4E,A2,D2.L)
000414	D732 B810	ADD.B	D3,(10,A2,A3.L)
000418	D732 20F0	ADD.B	D3,(-10,A2,D2.W)
00041C	D772 B04E	ADD.W	D3,(4E,A2,A3.W)
000420	D778 3000	ADD.W	D3,003000
000424	D7B9 0003 0000	ADD.L	D3,030000
00042A	D893	ADD.L	(A3),D4
00042C	D823	ADD.B	-(A3),D4
00042E	D81B	ADD.B	(A3)+,D4
000430	D86B 2000	ADD.W	(2000,A3),D4
000434	D873 2056	ADD.W	(56,A3,D2.W),D4
000438	D8B8 2000	ADD.L	002000,D4
00043C	D8B9 0002 0000	ADD.L	020000,D4
000442	D83C 0020	ADD.B	#20,D4
000446	D83A 0052	ADD.B	(0052,PC),D4
00044A	D83B 90AE	ADD.B	(-52,PC,A1.W),D4

LOCATION	OBJECT	SOURCE CODE	
00044E	D87B 1080	ADD.W	(-80,PC,D1.W),D4
000452	D87B 9052	ADD.W	(52,PC,A1.W),D4
000456	D9D3	ADDA.L	(A3),A4
000458	D9E3	ADDA.L	-(A3),A4
00045A	D9DB	ADDA.L	(A3)+,A4
00045C	D8EB 2000	ADDA.W	(2000,A3),A4
000460	D8F3 2066	ADDA.W	(66,A3,D2.W),A4
000464	D9F8 2000	ADDA.L	002000,A4
000468	D9F9 0002 0000	ADDA.L	020000,A4
00046E	D8FC 2000	ADDA.W	#2000,A4
000472	D8FA 0150	ADDA.W	(0150,PC),A4
000476	D8FB 90EE	ADDA.W	(-12,PC,A1.W),A4
00047A	D8FB 10AE	ADDA.W	(-52,PC,D1.W),A4
00047E	D8FB 1052	ADDA.W	(52,PC,D1.W),A4
000482	0638 0010 1000	ADDI.B	#10,001000
000488	0645 2000	ADDI.W	#2000,D5
00048C	06A9 3000 0000 0023	ADDI.L	#30000000,(0023,A1)
000494	5003	ADDQ.B	#08,D3
000496	5E78 2500	ADDQ.W	#07,002500
00049A	5CA2	ADDQ.L	#06,-(A2)
00049C	5A2A 0020	ADDQ.B	#05,(0020,A2)
0004A0	5872 0026	ADDQ.W	#04,(26,A2,D0.W)
0004A4	56B2 80FA	ADDQ.L	#03,(-06,A2,A0.W)
0004A8	D509	ADDX.B	-(A1),-(A2)
0004AA	DF4B	ADDX.W	-(A3),-(A7)
0004AC	D581	ADDX.L	D1,D2
0004AE	C602	AND.B	D2,D3
0004B0	C139 0004 0000	AND.B	D0,040000
0004B6	C4BB C010	AND.L	(10,PC,A4.W),D2
0004BA	C4BB 40F0	AND.L	(-10,PC,D4.W),D2
0004BE	C732 0860	AND.B	D3,(60,A2,D0.L)
0004C2	0202 0030	ANDI.B	#30,D2
0004C6	0278 0080 2000	ANDI.W	#0080,002000
0004CC	02AB 0000 3000 1000	ANDI.L	#00003000,(1000,A3)
0004D4	02B3 0010 0128 3844	ANDI.L	#00100128,(44,A3,D3.L)
0004DC	023C 0020	ANDI	#20,CCR
0004E0	023C 0020	ANDI	#20,CCR
0004E4	027C 0020	ANDI	#0020,SR
0004E8	027C 0020	ANDI	#0020,SR
0004EC	027C 0172	ANDI	#0172,SR
0004F0	027C 0172	ANDI	#0172,SR
0004F4	E105	ASL.B	#8,D5
0004F6	E564	ASL.W	D2,D4
0004F8	E0EA 1200	ASR.W	(1200,A2)
0004FC	E0F2 80EE	ASR.W	(-12,A2,A0.W)
000500	E1F2 8812	ASL.W	(12,A2,A0.L)
000504	E1F2 08EE	ASL.W	(-12,A2,D0.L)
000508	E0F9 0006 8000	ASR.W	068000
00050E	E0F8 0680	ASR.W	000680
000512	6900 0AEC	BVS.W	001000
000516	6400 0D1C	BCC.W	001234

LOCATION	OBJECT	SOURCE CODE	
00051A	65FE	BCS.B	00051A
00051C	670E	BEQ.B	00052C
00051E	6CDE	BGE.B	0004FE
000520	6E00 2ADE	BGT.W	003000
000524	6200 1ADA	BHI.W	002000
000528	6F00 06FE	BLE.W	000C28
00052C	6300 FEFE	BLS.W	00042C
000530	6DFE	BLT.B	000530
000532	6B00 FBCC	BMI.W	000100
000536	6600 FCC8	BNE.W	000200
00053A	6A00 FDC4	BPL.W	000300
00053E	6000 FEC0	BRA.W	000400
000542	61BC	BSR.B	000500
000544	6800 00BA	BVC.W	000600
000548	0744	BCHG.L	D3,D4
00054A	0878 0003 2000	BCHG.B	#03,002000
000550	0784	BCLR.L	D3,D4
000552	08AB 0003 1000	BCLR.B	#03,(1000,A3)
000558	08C3 0017	BSET.L	#17,D3
00055C	0FEA 1234	BSET.B	D7,(1234,A2)
000560	08F3 0006 0010	BSET.B	#06,(10,A3,D0.W)
000566	0704	BTST.L	D3,D4
000568	0833 0007 00F0	BTST.B	#07,(-10,A3,D0.W)
00056E	45BC 3000	CHK.W	#3000,D2
000572	43B9 0003 0000	CHK.W	030000,D1
000578	43B2 4810	CHK.W	(10,A2,D4.L),D1
00057C	4200	CLR.B	D0
00057E	4212	CLR.B	(A2)
000580	425A	CLR.W	(A2)+
000582	4262	CLR.W	-(A2)
000584	42B8 1000	CLR.L	001000
000588	42AA 3456	CLR.L	(3456,A2)
00058C	4232 08AA	CLR.B	(-56,A2,D0.L)
000590	42B5 4810	CLR.L	(10,A5,D4.L)
000594	B200	CMP.B	D0,D1
000596	B212	CMP.B	(A2),D1
000598	B25A	CMP.W	(A2)+,D1
00059A	B262	CMP.W	-(A2),D1
00059C	B2AA 4321	CMP.L	(4321,A2),D1
0005A0	B232 0814	CMP.B	(14,A2,D0.L),D1
0005A4	B2B5 4810	CMP.L	(10,A5,D4.L),D1
0005A8	B2BB 4810	CMP.L	(10,PC,D4.L),D1
0005AC	B2B8 1000	CMP.L	001000,D1
0005B0	B2C0	CMPA.W	D0,A1
0005B2	B2FB 0814	CMPA.W	(14,PC,D0.L),A1
0005B6	B3E2	CMPA.L	-(A2),A1
0005B8	B3F8 1000	CMPA.L	001000,A1
0005BC	B3EA 1234	CMPA.L	(1234,A2),A1
0005C0	0C39 0020 0001 2000	CMPI.B	#20,012000
0005C8	0C75 1003 4810	CMPI.W	#1003,(10,A5,D4.L)
0005CE	B94B	CMPM.W	(A3)+,(A4)+
0005D0	54C9 FB2E	DBCC.W	D1,000100
0005D4	55C9 FC2A	DBCS.W	D1,000200

LOCATION	OBJECT	SOURCE CODE	
0005D8	57CA FD26	DBEQ.W	D2,000300
0005DC	51CA FE22	DBF.W	D2,000400
0005E0	5CCB FF1E	DBGE.W	D3,000500
0005E4	5ECB 001A	DBGT.W	D3,000600
0005E8	52CC 0116	DBHI.W	D4,000700
0005EC	5FCC 0212	DBLE.W	D4,000800
0005F0	53CD 030E	DBLS.W	D5,000900
0005F4	5DCD 0A0A	DBLT.W	D5,001000
0005F8	5BCE 0C3A	DBMI.W	D6,001234
0005FC	56CE 507A	DBNE.W	D6,005678
000600	5ACF 0B20	DBPL.W	D7,001122
000604	50CF 2D3E	DBT.W	D7,003344
000608	58C8 4F5C	DBVC.W	D0,005566
00060C	59C8 717A	DBVS.W	D0,007788
000610	85C0	DIVS.W	D0,D2
000612	85E9 1234	DIVS.W	(1234,A1),D2
000616	84FA 1234	DIVU.W	(1234,PC),D2
00061A	B503	EOR.B	D2,D3
00061C	BB21	EOR.B	D5,-(A1)
00061E	B372 3828	EOR.W	D1,(28,A2,D3.L)
000622	B771 0834	EOR.W	D3,(34,A1,D0.L)
000626	B5B8 4000	EOR.L	D2,004000
00062A	0A29 0020 3000	EORI.B	#20,(3000,A1)
000630	0A03 0045	EORI.B	#45,D3
000634	0A78 8463 3400	EORI.W	#8463,003400
00063A	0A59 0024	EORI.W	#0024,(A1)+
00063E	0AB9 0000 0274	EORI.L	#00000274,012800
	0001 2800		
000648	0AB1 0002 4645	EORI.L	#00024645,(11,A1,D1.L)
	1811		
000650	0A3C 0024	EORI	#24,CCR
000654	0A7C 1024	EORI	#1024,SR
000658	C34A	EXG.L	A1,A2
00065A	C745	EXG.L	D3,D5
00065C	CB8A	EXG.L	D5,A2
00065E	4882	EXT.W	D2
000660	48C4	EXT.L	D4
000662	4AFC	ILLEGAL	
000664	4ED1	JMP	(A1)
000666	4EF1 0823	JMP	(23,A1,D0.L)
00066A	4EF1 0823	JMP	(23,A1,D0.L)
00066E	4EAA 1234	JSR	(1234,A2)
000672	4EB1 088E	JSR	(-72,A1,D0.L)
000676	4EB8 0704	JSR	000704
00067A	43D2	LEA.L	(A2),A1
00067C	41F3 4056	LEA.L	(56,A3,D4.W),A0
000680	43F9 0012 3456	LEA.L	123456,A1
000686	4E52 7000	LINK	A2,#7000
00068A	E72C	LSL.B	D3,D4
00068C	E94D	LSL.W	#4,D5
00068E	E3F8 4C80	LSL.W	004C80
000692	E3E9 1234	LSL.W	(1234,A1)
000696	EEA8	LSR.L	D7,D0
000698	E609	LSR.B	#3,D1

LOCATION	OBJECT	SOURCE CODE	
00069A	E2D8	LSR.W	(A0)+
00069C	E2F1 0812	LSR.W	(12,A1,D0.L)
0006A0	1982 0870	MOVE.B	D2,(70,A4,D0.L)
0006A4	14B4 0890	MOVE.B	(-70,A4,D0.L),(A2)
0006A8	359A 0822	MOVE.W	(A2)+,(22,A2,D0.L)
0006AC	353B A8DD	MOVE.W	(-23,PC,A2.L),-(A2)
0006B0	257C 0000 2000 0035	MOVE.L	#00002000,(0035,A2)
0006B8	23F8 1000 2000 1000	MOVE.L	001000,001000
0006C0	15BB 0868 2034	MOVE.B	(68,PC,D0.L),(34,A2,D2.W)
0006C6	3C45	MOVEA.W	D5,A6
0006C8	347B A823	MOVEA.W	(23,PC,A2.L),A2
0006CC	247C 0000 2000	MOVEA.L	#00002000,A2
0006D2	2478 1000	MOVEA.L	001000,A2
0006D6	42F8 2800	MOVE.W	CCR,002800
0006DA	44EA 0023	MOVE.W	(0023,A2),CCR
0006DE	46DA	MOVE.W	(A2)+,SR
0006E0	40C7	MOVE.W	SR,D7
0006E2	4E63	MOVE.L	A3,USP
0006E4	4E6C	MOVE.L	USP,A4
0006E6	4C9C FC38	MOVEM.W	(A4)+,D3-D5/A2-A7
0006EA	48E2 1C3F	MOVEM.L	A7-A2/D5-D3,-(A2)
0006EE	48E2 1C3F	MOVEM.L	A7-A2/D5-D3,-(A2)
0006F2	4CB8 0D10 3500	MOVEM.W	003500,D4/A0/A2-A3
0006F8	48AA 0D10 1234	MOVEM.W	D4/A0/A2-A3,(1234,A2)
0006FE	48AA 0D10 BBCD	MOVEM.W	D4/A0/A2-A3,(-4433,A2)
000704	4CF2 FE38 0811	MOVEM.L	(11,A2,D0.L),D3-D5/A1-A7
00070A	070B 2080	MOVEP.W	2080(A3),D3
00070E	03CB 0080	MOVEP.L	D1,0080(A3)
000712	767F	MOVEQ.L	#0000007F,D3
000714	C7FC 1234	MULS.W	#1234,D3
000718	C2F2 08CC	MULU.W	(-34,A2,D0.L),D1
00071C	4803	NBCD.B	D3
00071E	4838 6400	NBCD.B	006400
000722	482A 1234	NBCD.B	(1234,A2)
000726	4402	NEG.B	D2
000728	4453	NEG.W	(A3)
00072A	449A	NEG.L	(A2)+
00072C	44B2 C812	NEG.L	(12,A2,A4.L)
000730	4038 2300	NEGX.B	002300
000734	4062	NEGX.W	-(A2)
000736	40B2 083C	NEGX.L	(3C,A2,D0.L)
00073A	4E71	NOP	
00073C	4602	NOT.B	D2
00073E	469C	NOT.L	(A4)+
000740	4639 0000 8000	NOT.B	008000
000746	4672 3820	NOT.W	(20,A2,D3.L)
00074A	466B 1234	NOT.W	(1234,A3)
00074E	46B9 0005 0000	NOT.L	050000
000754	8803	OR.B	D3,D4
000756	8E78 2000	OR.W	002000,D7
00075A	879A	OR.L	D3,(A2)+
00075C	8C2A 0012	OR.B	(0012,A2),D6

LOCATION	OBJECT	SOURCE CODE	
000760	8772 0812	OR.W	D3,(12,A2,D0.L)
000764	0020 0028	ORI.B	#28,-(A0)
000768	0052 1006	ORI.W	#1006,(A2)
00076C	00B2 1000 2876 0834	ORI.L	#10002876,(34,A2,D0.L)
000774	0039 0038 0001 1100	ORI.B	#38,011100
00077C	003C 0003	ORI	#03,CCR
000780	007C 1003	ORI	#1003,SR
000784	487B A820	PEA.L	(20,PC,A2.L)
000788	4E70	RESET	
00078A	E73C	ROL.B	D3,D4
00078C	E09C	ROR.L	#8,D4
00078E	E7F8 0800	ROL.W	000800
000792	E6F3 0822	ROR.W	(22,A3,D0.L)
000796	E533	ROXL.B	D2,D3
000798	E852	ROXR.W	#4,D2
00079A	E094	ROXR.L	#8,D4
00079C	E5DB	ROXL.W	(A3)+
00079E	E4F3 0806	ROXR.W	(06,A3,D0.L)
0007A2	4E73	RTE	
0007A4	4E77	RTR	
0007A6	4E75	RTS	
0007A8	8B03	SBCD.B	D3,D5
0007AA	870A	SBCD.B	-(A2),-(A3)
0007AC	54C0	SCC.B	D0
0007AE	55D3	SCS.B	(A3)
0007B0	57DB	SEQ.B	(A3)+
0007B2	51E3	SF.B	-(A3)
0007B4	5CEB 3000	SGE.B	(3000,A3)
0007B8	5EF3 0811	SGT.B	(11,A3,D0.L)
0007BC	52F3 0822	SHI.B	(22,A3,D0.L)
0007C0	5FF3 0811	SLE.B	(11,A3,D0.L)
0007C4	53F3 88EF	SLS.B	(-11,A3,A0.L)
0007C8	5DF9 0001 0000	SLT.B	010000
0007CE	5BC4	SMI.B	D4
0007D0	56D4	SNE.B	(A4)
0007D2	5AEB 0012	SPL.B	(0012,A3)
0007D6	50DB	ST.B	(A3)+
0007D8	58E3	SVC.B	-(A3)
0007DA	59EA 1111	SVS.B	(1111,A2)
0007DE	4E72 3300	STOP	#3300
0007E2	9602	SUB.B	D2,D3
0007E4	9713	SUB.B	D3,(A3)
0007E6	9763	SUB.W	D3,-(A3)
0007E8	975B	SUB.W	D3,(A3)+
0007EA	97AB 7000	SUB.L	D3,(7000,A3)
0007EE	97B2 284E	SUB.L	D3,(4E,A2,D2.L)
0007F2	9732 B8B2	SUB.B	D3,(-4E,A2,A3.L)
0007F6	9732 204E	SUB.B	D3,(4E,A2,D2.W)
0007FA	9772 B0B2	SUB.W	D3,(-4E,A2,A3.W)
0007FE	9778 3000	SUB.W	D3,003000
000802	97B9 0003 0000	SUB.L	D3,030000
000808	9893	SUB.L	(A3),D4

LOCATION	OBJECT	SOURCE CODE	
00080A	9823	SUB.B	-(A3),D4
00080C	981B	SUB.B	(A3)+,D4
00080E	986B 6000	SUB.W	(6000,A3),D4
000812	9873 2056	SUB.W	(56,A3,D2.W),D4
000816	98B8 2000	SUB.L	002000,D4
00081A	98B9 0002 0000	SUB.L	020000,D4
000820	983C 0020	SUB.B	#20,D4
000824	983A 0052	SUB.B	(0052,PC),D4
000828	983B 90AE	SUB.B	(-52,PC,A1.W),D4
00082C	987B 9052	SUB.W	(52,PC,A1.W),D4
000830	987B 2052	SUB.W	(52,PC,D2.W),D4
000834	99D3	SUBA.L	(A3),A4
000836	99E3	SUBA.L	-(A3),A4
000838	99DB	SUBA.L	(A3)+,A4
00083A	98EB 2000	SUBA.W	(2000,A3),A4
00083E	98F3 2052	SUBA.W	(52,A3,D2.W),A4
000842	99F8 2000	SUBA.L	002000,A4
000846	99F9 0002 0000	SUBA.L	020000,A4
00084C	98FC 2000	SUBA.W	#2000,A4
000850	98FA 1166	SUBA.W	(1166,PC),A4
000854	99FB 9008	SUBA.L	(08,PC,A1.W),A4
000858	98FB 9052	SUBA.W	(52,PC,A1.W),A4
00085C	0438 0010 1000	SUBI.B	#10,001000
000862	0445 2000	SUBI.W	#2000,D5
000866	04A9 3000 0000	SUBI.L	#30000000,(0023,A1)
	0023		
00086E	5103	SUBQ.B	#08,D3
000870	5F78 2500	SUBQ.W	#07,002500
000874	5DA2	SUBQ.L	#06,-(A2)
000876	5B2A 0020	SUBQ.B	#05,(0020,A2)
00087A	5972 0076	SUBQ.W	#04,(76,A2,D0.W)
00087E	57B2 008A	SUBQ.L	#03,(-76,A2,D0.W)
000882	9509	SUBX.B	-(A1),-(A2)
000884	9581	SUBX.L	D1,D2
000886	4845	SWAP.W	D5
000888	4AC4	TAS.B	D4
00088A	4AF8 3600	TAS.B	003600
00088E	4AF4 0012	TAS.B	(12,A4,D0.W)
000892	4A03	TST.B	D3
000894	4A2B 2000	TST.B	(2000,A3)
000898	4A78 3000	TST.W	003000
00089C	4A5B	TST.W	(A3)+
00089E	4AB3 0855	TST.L	(55,A3,D0.L)
0008A2	4E59	UNLK	A1
0008A4	5352	SUBQ.W	#01,(A2)
0008A6	5150	SUBQ.W	#08,(A0)
0008A8	5F5E	SUBQ.W	#07,(A6)+
0008AA	5D5C	SUBQ.W	#06,(A4)+
0008AC	5B5A	SUBQ.W	#05,(A2)+
0008AE	5958	SUBQ.W	#04,(A0)+

APPENDIX C

BREAKPOINT TIMING

C.1 EMULATION STATUS

BUS REQUEST (BR) state stepping is used in the MICE-16 68000. The emulation processor bus is at high impedance whenever it stops. In this state all signals on the bus are floating. Signal changes can only be tested by using the Cycle Wait command. The following sections describe break operation for Events 1 to 6.

C.2 Timing for Events 1 and 2

Emulation stops at the end of the bus cycle* following the cycle in which the break condition was matched.

When the Cycle Step command is input, the MICE-16 68000 displays the address of the next cycle (assuming no branch instruction is executed). Entering a Register command instead of a Cycle Step command at this time may show that the PC does not equal the same value as displayed in the Cycle Step command, though. If the current instruction has not yet been completed, the Register command will finish it. Therefore, the PC displayed in the Register command always points to the starting address of the the next unexecuted instruction, as illustrated below.

```
LOCATION  OBJECT          STATUS  SOURCE CODE
   :        :               :        :
 008002    4E71            SP      NOP
 008004    4E71            SP      NOP          ;breakpoint**
 008006    23C0 0001 0000  SP      MOVE.L D0,010000
 00800C    4E71            SP      NOP
 00800E    4E71            SP      NOP
 008010    4E71            SP      NOP
   :        :               :        :
```

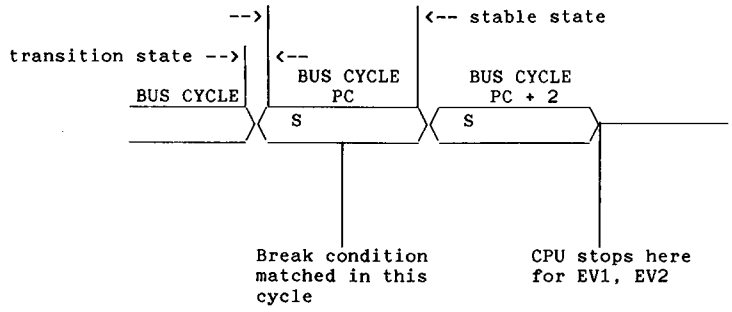
In this example, the break condition is matched and emulation stops at the following bus cycle (location 008006H) as shown in Figure C-1. The Cycle Step command displays location 008008H, but a Register command displays 00800CH instead. Because the first word (23C0H) of the instruction (MOVE.L D0,010000) was already fetched when emulation stopped, entering a Register command completes execution of this instruction and displays PC=00800CH.

---

* **bus cycle**: Complete operation of bus activity such as opcode/operand fetch or read/write data from/to memory.

** The break condition (Event 1 or 2) is matched and emulation/tracing stops at the next cycle.

The following diagram shows the timing of break condition match and emulation halt during bus cycle operation.



S is the fetch state. (No R/W operation, or branch instruction is assumed in this example.)
PC is the program counter (the increment per bus cycle is PC=PC+2).
stable state is a full bus cycle.
transition state is the end of a bus cycle and the start of the next one.

Figure C-1 Timing Diagram for Break/Halt During Bus Cycle Operation

Once emulation stops, prefetch instructions already in the queue are executed. Note that prefetched instructions requiring external bus cycles* are not executed, though.

Matching Event 1 or 2, generates output signal $\overline{\text{SYNC.1}}$ or $\overline{\text{SYNC.2}}$ respectively as described in Section 5.7.1.2.

Address Strobe Negate to SYNC Output Assert	SYNC Output Pulse Width (typical)
t_{SYDLY} (max) = 160ns	t_{SPW} = 200ns

Table C-2 Timing for Events 1/2

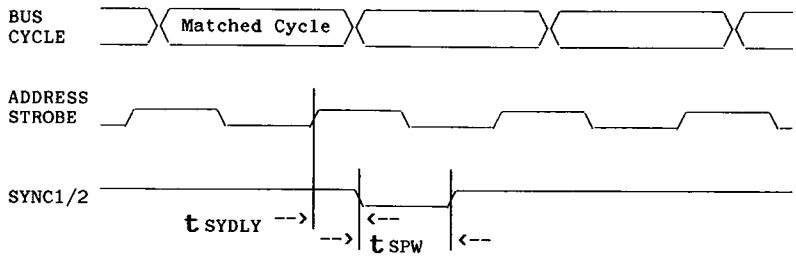


Figure C-3 $\overline{\text{SYNC.1}}$/$\overline{\text{SYNC.2}}$ Output Timing

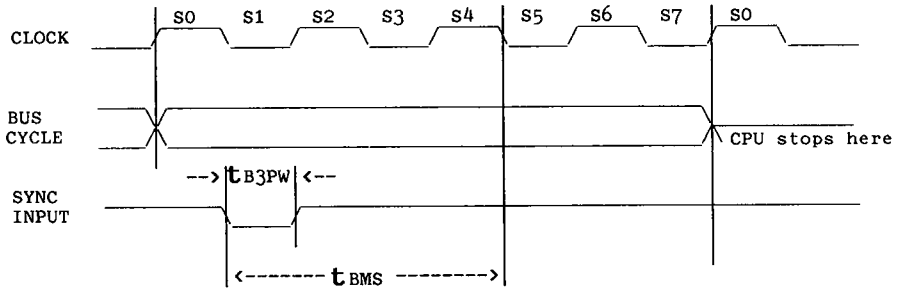
* An external bus cycle accesses external peripherals (e.g. memory, I/O port, etc.).

C.3 Timing for Event 3

When external trigger input is used to halt emulation, the break position is determined by the timing of the SYNC INPUT signal. Regardless of when the MICE receives this signal, it will process it at the falling edge of the next S4 clock cycle.

Break Match Timing Minimum Value (t_{BMS})	Stop CPU Setup Time Minimum Value	MHz
130ns + t_{ASI}	t_{ASI} = 20ns	12

Table C-4 Timing for Event 3



t_{B3PW}: Event 3 minimum pulse width is 10ns.

Figure C-5 Timing for Event 3

C.4 Timing for Events 4, 5 and 6

Events 4, 5 and 6 are real time execution breakpoints. All the hardware timing are automatically processed within MICE , thus freeing user from worrying about them. When setting the execution breakpoint, user must set it at the first word of the instruction. When the breakpoint is matched, the user's program instruction at that location is not executed yet. Further details are explained in Section 5.7.1.4.

APPENDIX D

TARGET INTERFACE DIAGRAM

MICROTEK INTERNATIONAL INC.

D.1 Interface Diagram from Target to Emulation Processor

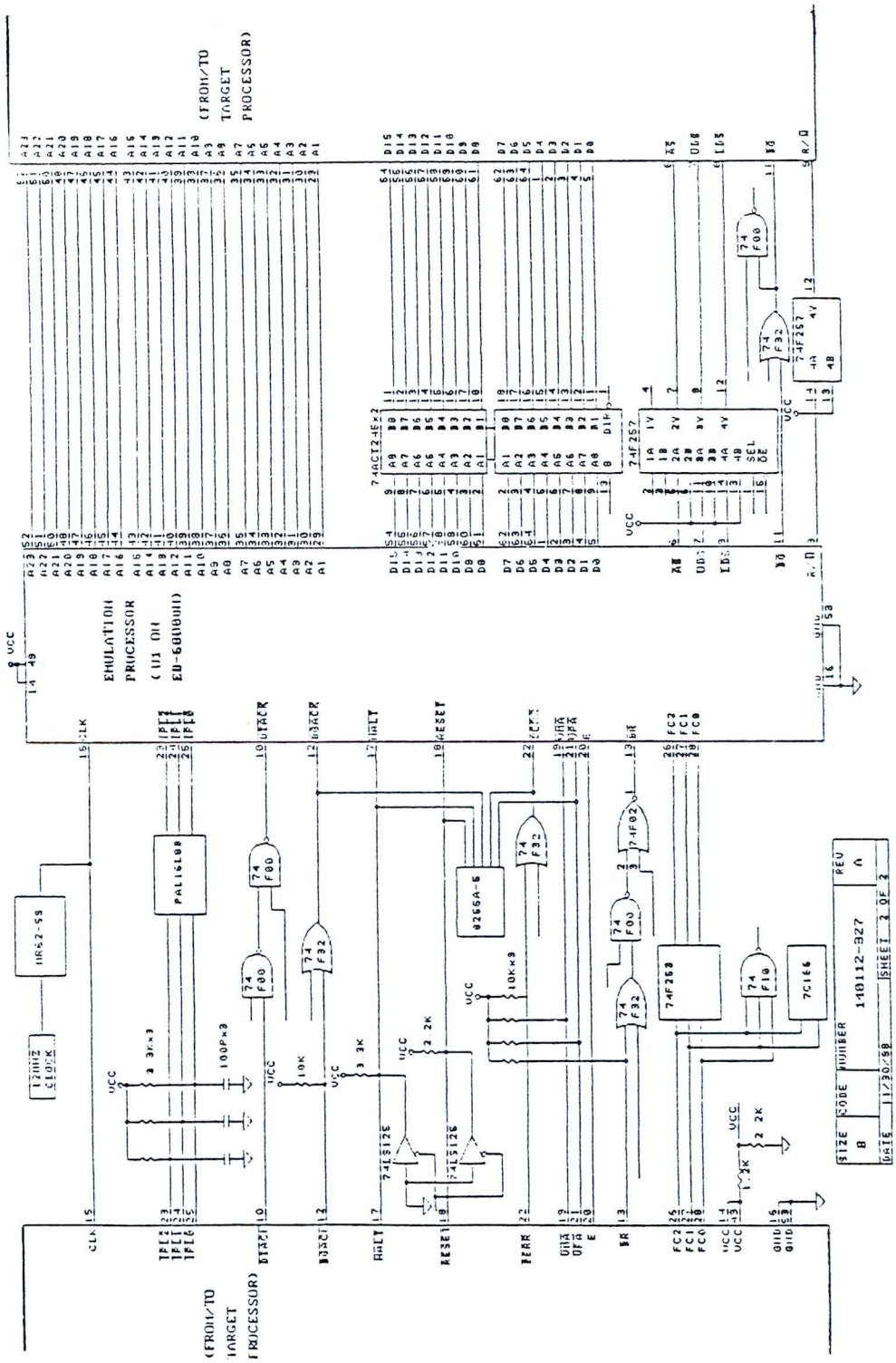


Figure D-1 Interface Diagram from Target to Emulation Processor

APPENDIX E

WRITITNG A DRIVER PROGRAM

E.1 AVAILABLE MICE DRIVER PROGRAMS

The following table (Table E-1) lists the currently available driver programs (based on Universal Symbolic Debugger) for MICE-16 68000.

HOST COMPUTER	O.S.	DRIVER	STANDARD MEDIUM
IBM PC/XT/AT	MS-DOS	USD/PC	5 1/4" DSDD floppy diskette
IBM PS/2	MS-DOS	USD/PC	3 1/2" 2S/HD floppy diskette
NEC PC-9801 family	MS-DOS	USD/9801	5 1/4" DSDD floppy diskette
VAX	VMS	USD/VMS	TU-80 Magtape, Files-11 format
MicroVAX	VMS	USD/VMS	TK-50 Cartridge tape, Files-11 format
MicroVAX	ULTRIX-32	USD/ULTRIX	TK-50 Cartridge tape, Tar format
Sun Microsystems	UNIX	USD/SUN	1/4" Cartridge tape, Tar format
Apollo Computer	AEGIS	USD/APOLLO	1/4" Cartridge tape, Tar format
			1/4" Cartridge tape, RBAK/WBAK format

Table E-1 MICE-16 68000 Software Programs

E.2 GUIDES FOR WRITING A CUSTOM USER DRIVER PROGRAM

If for some reasons user wishes to write his own MICE-16 68000 driver program, the following guides is provided and should be taken into considerations in writing such program:

E.2.1 System Setup

Setup MICE with the computer system taking into consideration the RS-232C communication port requirement as described in detail in Sections 2.3.2.

E.2.2 MICE Command Transparency

STEP	D R I V E R A C T I O N	R E M A R K S
1	Sends 'CR' to try to initiate communication with MICE. If it receives no response, send the character which has just been input from keyboard. Receive all responses from MICE and display them until a prompt (>) followed by handshaking code appear on the screen.	;User has to enter "M<CR>" on power on. ;MICE responds with self-diagnostic results.

MICE Command Transparency (con't)

STEP	D R I V E R A C T I O N	R E M A R K S
2	Pick up one key (keyboard input) say "char-X", and send it to MICE. Receive response from MICE and display it until MICE echo with message- "char-Y". where: If "char-X" is BS, "char-Y" is handshaking code. If "char-X" is ESC, "char-Y" is handshaking code, else "char-Y" is the same as "char-X".	;BS (Back Space = 08H) ;ESC = 1BH
3	If "char-X" is 'CR', Receive response from MICE and display received messages until a prompt (>) followed by handshaking code appear on the screen. At the same time, scan keyboard input during display and send any received data to MICE.	;CR = 0DH
4	Goto Step 2.	

E.2.3 Upload

STEP	D R I V E R A C T I O N	R E M A R K S
1	Send upload command to MICE. Receive echo messages until 'CR' is detected.	;Command syntax: "U start_adr end_adr CR"
2	Receive upload data until handshaking code is detected.	
3	Store record to file.	
4	Send ACK (CTRL-F) to MICE.	
5	If not end-record, goto Step 2. Otherwise close the file.	

E.2.3.1 MICE Upload Command

Upload [space-adr [adr]]

Upload is the MICE command for uploading information (from MICE) by formatted hex record.

space-adr specifies the memory location where the checksum operation is to begin. It is a hexadecimal address in the memory space of the emulation CPU, optionally preceded by specification for memory type: [SP|SD|UP|UD] adr. If no memory type is specified, the Supervisor Program (SP) will default.

adr is a hexadecimal address setting in the memory space of the emulation CPU specifying the last location in the range on which the checksum is to be carried out.

This command is only applicable when MICE-16 68000 is connected to a host computer. Information can be uploaded from the target to the host.

Memory contents defined by the memory range start-address through end-address are transferred to the host system using Motorola format. 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-16 68000 reads data from memory and sends the data to the host system using the Motorola format. Data transmission continues in this manner until an end address is sent. MICE-16 68000 then sends an end-of-record file and prompts for the next command.

If anything other than an <ACK> is received when MICE-16 68000 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-16 68000 prompts for the next command. The command can also be aborted by the host system when MICE-16 68000 receives an <ESC> character from the console.

Example: Upload from MICE-16 68000 using Motorola format. Note that load records are shown on separate lines for clarity. No <CR> or <LF> characters are generated by MICE-16 68000, and anything received other than an <ACK>, is treated the same as a <NAK>.

```
>Upload 0 7<CR>
S20C0000002300A8917046E8F801<ACK>
S804000000FB<ACK>
>
```

E.2.4 Download

STEP	D R I V E R A C T I O N	R E M A R K S
1	Send '%' to MICE and receive responses until handshaking code* is detected.	
2	Get one record from file.	
3	Send one record to MICE and receive one character (say char-Z) from MICE.	
4	If "char-Z" is ACK (Ctrl-F), receive until handshaking code is detected. If it is the end-record, close the file, else goto Step 2. If "char-Z" is NACK (Ctrl-U), receive until handshaking code is detected. If the same record is sent 5 times, abort download, else goto step 3. Else display "char-Z", receive and display until handshake code appears on screen. Abort download.	 ;For next record ;Try again. ;Error message from MICE.

*Handshake code is user programmable, default is 03H.

E.2.4.1 MICE Download Command

```
DOwnload                %
<hex record>    or      <hex record>
    :                       :
    :                       :
```

DOwnload or % is the download command keyword.

<hex record> are the records to be transfered.

This command is only applicable when MICE-16 68000 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.

High speed binary (Microtek) format download is supported by this command under Universal Symbolic Debugger (version 2.3 or later) with 8 data bit setting.

Motorola loading format is recognized by MICE-16 68000 and is described in detail in the following sections.

E.2.4.2 Download Protocol for Motorola Format - S

S load-record

S is the keyword to use to signal start of record.

load-record is a Motorola load record containing up to 255 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 255 bytes of program data. The general format of a record, shown with spaces separating each field, is:

RECORD MARK	LOAD STATUS	RECORD LENGTH	LOAD ADDRESS	PROGRAM DATA	CHECK-SUM
S	**a**	**##**	**aaaaaaaa**	**dd...dd**	**cc**

where:

S is the keyword to use to signal start of record.

a indicates load-record status of subsequent data.

0 Starts output of the first record.
1 The object data that follows will be at a two-byte memory address (S1 format).
2 The object data that follows will be at a three-byte memory address (S2 format).
3 The object data that follow will be at a four-byte memory address (S3 format).
7 Last record (S3 format).
8 Last record (S2 format).
9 Last record (S1 format).

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

aaaaaaaa is the hexadecimal memory address where the data that follow, are to be loaded. S1 format uses 4 digits, S2 format uses 6 digits and S3 format uses 8 digits.

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

cc is a two ASCII hexadecimal one's complement of the sum of the record.

When MICE receives an "S", 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 also reprompts but the alternate negative acknowledgement response <NAK sequence> is sent:

NAK LF CR NUL NUL NUL NUL NUL NUL

Example: Download our program with an end-record into a target system using Motorola S2 format.

```
                0 1 2 3 4 5 6 7 8 9 A B
>S20C0000002300A8917046E8F801<ACK sequence>
>S804000000FB<ACK SEQUENCE>
>
```

In the above example, the load record is

```
##      = 0CH
aaaaaa  = 000000H
dd...dd = 23H, 00H, A8H, 91H, 70H, 46H, E8H, F8H
cc      = 01H
```

where: 0CH+00H+00H+00H+23H+00H+A8H+91H+70H+46H+E8H+F8H+01H = FFH

For the end-record,

```
##      = 04H
aaaa    = 000000H
cc      = FBH
```

where: 04H+00H+00H+00H+FBH = FFH

APPENDIX F

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

Table F-1 Hexadecimal-Decimal Conversion Chart

APPENDIX G

ASCII LISTS AND DEFINITIONS

G.1 ASCII 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
44	68	D	62	98	b
45	69	E	63	99	c
46	70	F	64	100	d

ASCII List (con't)

HEX	DEC	CHAR	HEX	DEC	CHAR
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

Table G-1 ASCII List

G.2 ASCII Definitions

ABBR	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 Acknowldgment	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
!		Exclamation	21
"		Quotation mark	22
#		Number sign	23
$		Dollar sign	24
%		Percent Sign	25
&		Ampersand	26
'		Apostrophe	27

ASCII Definition (con't)

ABBR	CTRL	MEANING	HEX
(		Opening (left) parenthesis	28
)		Closing (right) parenthesis	29
*		Asterisk	2A
+		Plus	2B
,		Comma	2C
-		Hyphen (minus)	2D
.		Period (decimal)	2E
/		Slant	2F
0		Zero	30
1		One	31
2		Two	32
3		Three	33
4		Four	34
5		Five	35
6		Six	36
7		Seven	37
8		Eight	38
9		Nine	39
:		Colon	3A
;		Semicolon	3B
<		Less Than	3C
=		Equals	3D
>		Greater Than	3E
?		Question Mark	3F
@		Commercial at	40
A		Uppercase A	41
B		Uppercase B	42
C		Uppercase C	43
D		Uppercase D	44
E		Uppercase E	45
F		Uppercase F	46
G		Uppercase G	47
H		Uppercase H	48
I		Uppercase I	49
J		Uppercase J	4A
K		Uppercase K	4B
L		Uppercase L	4C
M		Uppercase M	4D
N		Uppercase N	4E
O		Uppercase O	4F

ASCII Definition (con't)

ABBR	CTRL	MEANING	HEX
P		Uppercase P	50
Q		Uppercase Q	51
R		Uppercase R	52
S		Uppercase S	53
T		Uppercase T	54
U		Uppercase U	55
V		Uppercase V	56
W		Uppercase W	57
X		Uppercase X	58
Y		Uppercase Y	59
Z		Uppercase Z	5A
[		Opening (left) bracket	5B
\		Reverse slant	5C
]		Closing (right) bracket	5D
^		Circumflex	5E
-		Underscore	5F
`		Grave accent	60
a		Lowercase a	61
b		Lowercase b	62
c		Lowercase c	63
d		Lowercase d	64
e		Lowercase e	65
f		Lowercase f	66
g		Lowercase g	67
h		Lowercase h	68
i		Lowercase i	69
j		Lowercase j	6A
k		Lowercase k	6B
l		Lowercase l	6C
m		Lowercase m	6D
n		Lowercase n	6E
o		Lowercase o	6F
p		Lowercase p	70
q		Lowercase q	71
r		Lowercase r	72
s		Lowercase s	73
t		Lowercase t	74
u		Lowercase u	75
v		Lowercase v	76
w		Lowercase w	77

ASCII Definition (con't)

ABBR	CTRL	MEANING	HEX
x		Lowercase x	78
y		Lowercase y	79
z		Lowercase z	7A
{		Opening (left) brace	7B
|		Vertical line	7C
}		Closing (right) brace	7D
~		Tilde	7E
DEL		Delete	7F

Table G-2 ASCII Definitions

INDEX

MICROTEK INTERNATIONAL INC.

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

MICE-IIIS 68000 with LAM-IIS

User's Manual Addendum

MICROTEK

USER'S MANUAL ADDENDUM

MICE-IIIS 68000 (with LAM-IIS)

Doc No. 149-000688
First Edition
October 1992

READ ME FIRST!!

This addendum contains unique information that applies to MICE equipped with LAM-IIS only. For other MICE information not mentioned in this addendum, please refer to the main manual or *"MICE-16 68000 User's Manual"*. MICE equipped with LAM-IIS is referred to as "MICE-IIIS 68000.

Trademarks Acknowledgement

IBM,PC,XT, AT and **PS/2** are trademarks of International Business Machines.
MS-DOS and **MS-WINDOWS** are trademarks of Microsoft Corporation
Sun Microsystems is a trademark of Sun Microsystems, Inc.
UNIX is a trademark of AT&T Bell Laboratories.
MRI is a trademark of Microtec Research Inc.
NEC is a trademark of NEC Corporation.

MICE-IIIS 68000 (WITH LAM-IIS) AND THIS ADDENDUM ARE COVERED BY THE LIMITED WARRANTY SUPPLIED WITH MICE-IIIS 68000.

Printed in Taiwan, ROC 10/92

CONTENTS

Addendum 1
LAM-IIS INTRODUCTION

When MICE-16 68000 is equipped with the upgraded Logic Analyzer Module (LAM-IIS), it is designated as "**MICE-IIIS 68000**". It offers several new and improved features designed to make the MICE a more efficient emulation machine. The following table summarizes the areas where improvements has been implemented:

Features	With LAM	With LAM-IIS
Event	Two Bus Breakpoints with: - Bit wildcard address - Bit wildcard data - Multiple bus status - Up to 64K event count (EV1 only)	Four Bus Breakpoints with: - Range or bit wildcard address - Bit wildcard data - Multiple bus status - Up to 64K event count - 8 external trace bits
Trigger	And/Or/Then DELay/BACKward/FOrward/CENter	Provides up to 8 of the following Trigger level: - And/Or/Not - Trace {On|Off}/Timer {On/Off} The following Trigger settings are available: - If...Else/Then/REset/DELay - Trace {On/Off} Timer {On/Off}
Qualify Trace	One Qualify trace with: - Bit wildcard address - Multiple bus status	Two Qualify trace with: - Range or bit wildcard address - Multiple bus status - 8 external trace bits
Break on Read before Write	None	1Mbytes (4 independent banks at 256Kbytes each) range address.
Code Coverage	None	1Mbytes (4 independent banks at 256Kbytes each) range address.

Trace Buffer	2K frames with: - 32 channels to address - 32 channels to data - 8 channels to status - 8 channels to external trace bits - 24 bits timer	32K frames with: - 32 channels to address - 32 channels to data - 16 channels to status - 8 channels to external trace bits - 8 channels to state and event - 32 bits timer
External Trigger Inputs	Buffer interface	Latch interface may be specified to high or low level trigger. Signal valid during low state $\overline{\text{STROBE}}$.
Timebase	Unit of measurement: - from 1μs to 1000μs Timer length: - max: 44 hrs. - min : 16 sec.	Unit of measurement: - from 0.1μs to 1000μs Timer length: - max: 49 days 17 hrs. - min : 7 min. 9 sec.

Table-1 LAM and LAM-IIS Features Comparison

All features of LAM version are supported by LAM-IIS. Hence only features unique to LAM-IIS are discussed in this addendum.

1.1 KEY FEATURES OF MICE-IIIS 68000

1) Real-Time Trace:

The trace buffer is 32K frames deep and 128 bits wide. Signals monitored by the trace feature are:

EP address bus (32 bits)
EP data bus (32 bits)
EP status signals (16 bits)
External trace bits (8 bits)
Trigger state (8 bits)
Timer stamp (32 bits)

2) Trace with Cycle Qualifiers:

Qualifiers can be set so that only cycles with the specified range or bit wildcard address and/or external trace bits are recorded in the trace buffer. The address may include range address, wildcard bits or a wildcard nibble; any combination of processor states may be specified.

3) **Multiple (Seven) Real-Time Breakpoints:**

The MICE-IIIS 68000 provides three execution breakpoints; four bus breakpoints (with options for range/wildcard address, data, status, external trace bits and count qualify); and one external hardware breakpoint (with conditional qualify). All execution breakpoints are hardware breakpoints for programs located in either RAM or ROM, but bus breakpoints may be located in either program or data memory in RAM or ROM.

4) **Powerful Trigger Constructs:**

The trigger conditions is composed of up to 8 "trigger-level" where each level specifies an event combination. Each level may be a single event or a logic combination of EV1~ EV5 with **ANd**, **OR** and **Not** as operators. A cycle delay feature is also included allowing trace/emulation to continue 0-65535 cycles after all trigger conditions have been met.

5) **Initialize and Code Coverage**

LAM-IIS supports Initialize and Code Coverage RAM of 1Mbytes range address. The range consisted of 4 independent banks at 256Kbytes each. When using Initialize, program will break on Read-Before-Write at the specified address. When using Code Coverage, user will be able to monitor the performance program execution.

1.2 HOW TO UPGRADE FROM LAM TO LAM-IIS

LAM-IIS module when shipped to update a LAM equipped MICE-16 68000, is delivered consisting of the following items:

One LAM-IIS board
Four EPROMs U11, U12, U14 and U15 with LAM-IIS Firmware (Version 5.0 or above)
One MICE-16 68000 User's Manual with Addendum for LAM-IIS

Follow these steps to upgrade to LAM-IIS equipped MICE:

1. Replace existing LAM with the LAM-IIS board
2. Replace EPROMs on CPM board with the new EPROMs provided.

NOTE

The CPM board should be of Revision-G or above in order to be compatible with LAM-IIS firmware.

Use USD-III Version 2.0 to fully utilize LAM-IIS features under USD.

Addendum 2
INITIALIZATION

This addendum explains the unique power up screen display and default setup of MICE-IIIS 68000 with LAM-IIS.

2.1 POWER UP SELF-TEST

MICE-IIIS 68000 power up self-test arrangement is the same for both LAM and LAM-IIS. Use >S<CR> in lieu of >M<CR> to skip self-test.

2.2 POWER UP SCREEN DISPLAY

With LAM-IIS, the resulting power up self-test screen display when proper communications are established after keying >M<CR>, are as illustrated in the following Figures 2-1a/1b. Figure 2-2 shows power up screen display with self-test skipped:

```
*** MICE-16 68000/68010 (W/LAM2S), Firmware V5.0 ***
        MICROTEK International Inc., 1992

The emulation processor is 68000 ( 16-bit )

Self-test in progress...
CPM ROM    OK
CPM RAM    OK
EPM RAM    OK
LAM2S      OK
EMM RAM    OK
Self-test completed
Emulation memory is EMM
Four independent 64K-byte memory banks
Mapping resolution is 16K-byte

No Target VCC; BERR, BR, INTR (IPL0-2) disabled
>
```

Figure 2-1a LAM-IIS Self-Test Power Up Status Screen with EMM

```
*** MICE-16 68000/68010 (W/LAM2S), Firmware V5.0 ***
          MICROTEK International Inc., 1992

The emulation processor is 68000 ( 16-bit )

Self-test in progress...
CPM ROM    OK
CPM RAM    OK
EPM RAM    OK
LAM2S      OK
HEMM RAM BANK1  OK
HEMM RAM BANK2  OK
HEMM RAM BANK3  OK
HEMM RAM BANK4  OK
Self-test completed

Emulation memory is HEMM
Four independent 256K-byte memory banks
Mapping resolution is 4K-byte

No Target VCC; BERR, BR, INTR (IPL0-2) disabled
>
```

Figure 2-1b LAM-IIS Self-Test Power Up Status Screen with HEMM

```
*** MICE-16 68000/68010 (W/LAM2S), Firmware V5.0 ***
          MICROTEK International Inc., 1992

The emulation processor is 68000 ( 16-bit )

LAM2S Initializing   ...

No Target VCC; BERR, BR, INTR (IPL0-2) disabled

>
```

Figure 2-2 LAM-IIS Power Up Status Screen with Self-Test Skipped

Refer to Section 3.2 of the main manual for other status of power up screen displays.

2.3 MICE-IIIS 68000 DEFAULT SETUP PARAMETERS

The LAM-IIS setup defaults are practically the same as those stated in Chapter 3, Section 3.3 for LAM except for the following items:

1) Events 1 to 8 - Events 1, 2, 3, 4, 6, 7 and 8 Clear; Event 5 Low, no conditional

2) Trigger Condition - Trigger A
Trigger Timer On Trace On
Trigger Level A EV1 OR EV2 OR EV3 OR EV4 Trace On
TImer On
Trigger Level B to H, Clear

3) Timebase Selection - Recall from NOVRAM
(factory NOVRAM setting:
Timebase is 0.1 μs)

4) Cycle Qualify - All machine cycles recorded

Addendum 3
COMMAND SUMMARY

3.1 COMMAND FUNCTIONAL GROUPS OVERVIEW

The following is an overview of LAM-IIS equipped MICE-IIIS 68000 commands, grouped according to their functions.

1) Setup Commands

BAud	**ID**entify	**REC**all	**SET**up
CLock	**INT**erval	**RO**ute	**SI**ze
CONtrol	**MA**p	**SA**ve	Verify
HANdshake	Prompt	**SEL**ect	**WA**it
Help	**REA**dy		

2) Memory Commands

Assemble	**COM**pare	Fill	**SE**arch
Byte	**CO**py	**LO**ng	**TE**st
CHecksum	Disassemble	Memory	**W**ord

3) Port Commands

Input	Output

4) Emulation Commands

Cycle	Register	**RES**et	Step
Jump			

5) Trace Commands

COVerage	**HA**lt	Qualify	**TRA**ce
Event	**IN**Itialize	**SY**nc	Trigger
Go	List	**TI**mebase	

3.2 COMMAND FUNCTIONAL GROUPS SYNTAX LISTING

3.2.1 Setup Commands

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Port B Baud Rate Setting	BAud	[baud-rate [parity] [data-bits]]
Display Clock Source	CLock	
Control Signal Enable/Disable	CONtrol	[[Berr] [Intr] [BR] {Enable|Disable}]
Set Handshaking Code	HANdshake	[code]
Command Help	Help	[GRoup {Emulation|Memory|Port|Setup| Trace}|command|ALl]
Identify Emulator	IDentify	
Timer Interval Switch	INTerval	[On|OFf]
Memory Mapping	MAp	[space-adr adr {{I|IR} {1|2|3|4} |E|ER|G}|ALl {E|G}]
Change Command Prompt	Prompt	[string]
Ready Signal Selection	REAdy	[Internal|External]
Recall Status from NOVRAM	RECall	
Change Operation Mode	ROute	[System]
Save Status to NOVRAM	SAve	
Select Leading Code	SELect	[{Route|DEVice} [code]]
Display Setup Parameters	SETup	
Memory Access Size	SIze	[Byte|Word]
Memory Verification Switch	Verify	[On|OFf]
Insert Wait State	WAit	[On|OFf]

The Setup group of commands are used to configure MICE for the target before emulation or debugging could start.

3.2.2 Memory Commands

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Line Assembly	Assemble	[space-adr]
Memory Modify	Byte	[space-adr [data]]
Memory Checksum	CHecksum	space-adr {adr|Length length}
Memory Compare	COMpare	space-adr1 {adr|Length length} space-adr2
Memory Copy	COpy	space-adr1 {adr|Length length} space-adr2
Disassembly	Disassemble	[space-adr [adr|Length length]]
Memory Fill	Fill	space-adr {adr|Length length} data
Memory Modify	LOng	[space-adr [data]]
Memory Dump	Memory	[space-adr [adr|Length length]]
Memory Search	SEarch	space-adr {adr|Length length} data
Memory Test	TEst	space-adr {adr|Length length}
Memory Modify	Word	[space-adr [data]]

The Memory Group commands provide access to the contents of designated memory locations. The MICE-IIIS 68000 supports four memory types: Supervisor Program (SP), Supervisor Data (SD), User Program (UP), and User Data (UD). For all of the commands described in this section, the memory type that it desired to access, should be specified. If memory type is not specified, the MICE will automatically treat the accessed memory as supervisor program space). The MICE-IIIS 68000 also supports three basic data formats: Byte, Word and Long Word. For the Memory Dump, CHecksum, Test, and Copy commands, the desired format should be specified using the Memory Access Size command (Section 5.3.16 of the main manual). The total amount of addressable memory is 16M bytes.

Because the 68000 is word-oriented, even-numbered addresses must be assigned for the program counter (PC) in the Assemble and Disassemble commands (Sections 5.4.1 and 5.4.6 of the main manual respectively) and for the supervisor stack pointer (SSP) and user stack pointer (USP). If an odd value is entered, the system will drop the least significant bit to modify it to a legal even value.

The MICE always makes sure that the memory and port addresses specified by the user are valid before performing an operation. Any reference to an invalid address will result in display of an error message and termination of the command. If Memory Verification (Section 5.3.17 of the main manual) is on, the MICE also verifies all memory write operations, and if data is not written correctly to the specified address, an error message will be displayed and the command terminated. The MICE, however, does not verify that there is memory or anything else connected to a port when performing a read operation.

Note that when the external ready signal is selected and read/write functions are being performed, and if the external circuitry does not respond with the $\overline{\text{DTACK}}$ signal,

```
"Address xxxxxxxx is not ready"
```

will appear on the screen with the contents of memory remaining unchanged.

3.2.3 Port Commands

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Port Input	Input	space-adr [Length in-length]
Port Output	Output	space-adr data

These commands provide access to the contents of I/O ports. The MICE-IIIS 68000 supports three basic data formats: Byte, Word and Long Word. For the Port Input and Port Output commands, the desired data format must be specified with the Memory Access Size command (Section 5.3.16 of the main manual). I/O is memory mapped, while the total amount of addressable memory is 16M bytes and the address range is user-defined.

The MICE always checks to make sure that the port addresses specified by the user are valid before an operation is performed. Any reference to an invalid address will result in an error message and termination of the command. If the Memory Verification Switch is on, the MICE will also verify all memory write operations (Section 5.3.17 of the main manual), and if data has not been written correctly to the specified port address, an error message will be displayed and the command will be terminated. The MICE, however, does not verify that there is memory or anything else connected to a port when executing a read operation.

NOTE

When the external ready signal is selected and read/write functions are being performed, if external circuitry does not respond with the $\overline{DTACK}$ signal,

"Address xxxxxxxx is not ready"

will appear on the console with the contents of memory remaining unchanged.

3.2.4 Emulation Commands

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Cycle Step	Cycle	[Wait|count]
Jump	Jump	adr
Register Display/Modify	Register	[register]
Reset Emulation Processor	RESet	[pc [ssp]]
Instruction Step	Step	[adr1 adr2 [CALI]|count|CALI]

The total amount of addressable memory for the 68000 is 16M bytes. Because the 68000 is word-oriented, even-numbered addresses must be assigned for the program counter (PC) in the Jump and RESet commands (Sections 5.6.2. and 5.6.4 of the main manual respectively) as well as for the supervisor stack pointer (SSP) and user stack pointer (USP) in the Register command (Section 5.6.3 of the main manual). If an odd value is entered, the system will drop the least significant bit to convert it into an even value.

Before carrying out a command, the MICE always checks any memory and/or port addresses input by the user to make sure they are valid. A reference to an invalid address will result in display of an error message-

"Command syntax error"

and termination of the command.

When executing the Cycle Step and Instruction Step commands (Section 5.6.1 and 5.6.5 of the main manual respectively), refer to Table 5-2 for a description of the displayed status information. Note that breakpoints (Section 5.7.1 of the main manual) are not active during execution of these commands.

During execution of these commands, if an attempt is made to refer to the memories which are defined as either guided (G) or write protected (ER/IR),

"Write protected memory trespassed"
or
"Guarded memory trespassed"

will display to warn user of such violation. Emulation processing however, will continue after <CR> is entered. the illegally entered data for external /internal write protected (ER/IR) or non-existence (G) memory block will be indecipherable.

When performing read/write function with the external ready signal being selected and the external circuitry does not respond to the $\overline{\text{DTACK}}$ signal, the message-

"Address xxxxxxxx is not ready"

will display with the contents of memory remaining unchanged.

3.2.5 Trace Commands

FUNCTION	COMMAND SYNTAX	
	KEYWORD	PARAMETER(S)
Set Coverage	COVerage	[adr1 adr2[CLear]|[ENable|DIsable| CLear|Reset]]
	COVerage	List [adr]
	COVerage	SP|UP|SD|UD
Display Events	Event	
Set Bus Event 1	Ev[ent]	1 {Range adr1 adr2|adx} [datum] [status] [External trace-bits] [COunt count]
Set Bus Event 2	Ev[ent]	2 {Range adr1 adr2|adx} [datum] [status] [EXternal trace-bits] [COunt count]
Set Bus Event 3	Ev[ent]	3 {Range adr1 adr2|adx} [datum] [status] [EXternal trace-bits] [COunt count]

Set Bus Event 4	Ev[ent]	4 {Range adr1 adr2 adx} [datum] [status] [EXternal trace-bits] [COunt count]
Set External hardware breakpoint	Ev[ent]	5 [COnditional] {High|Low}
Set Execution Event 1	Ev[ent]	6 adr [fetch-state]
Set Execution Event 2	Ev[ent]	7 adr [fetch-state]
Set Execution Event 3	Ev[ent]	8 adr [fetch-state]
Clear Events	Ev[ent]	[1|2|3|4|5|6|7|8] [CLear]
Execute User Program	Go	[Run] [adr]
Halt Emulation Processor	HAlt	
Set Initialize	INItialize	[adr1 adr2[CLear]|[ENable|DIsable| CLear|Reset]]
	INItialize	List [adr]
	INItialize	SP|UP|SD|UD
List Trace Buffer	List	[frame [adr1 adr2] [status] [EXternal trace-bits]]
List with Source Code	List	{Source|Instruction} [frame]
List Trace Frame Total	List	Number
Set Trace Cycle Qualifier	Qualify	{1|2} [{Range adr1 adr2|adx} [status] [EXternal trace-bits]]
Display/Clear/En/Disable Qualifier	Qualify	{1|2} [CLear|DIsable|ENable]
Set Sync Input/Output	SYnc	[{Input|OUtput} {On|OFf}]
Timebase Selection	TImebase	[0.1|1|10|100|1000]
Display Trace Condition	TRAce	
Set Trigger Level	Trigger	Level level# [Not] ev# [{OR|ANd} [Not] ev# [{OR|ANd} [Not] ev# [{OR|ANd} [Not] ev# [{OR|ANd} [Not] ev#]]]] [Trace {On|OFf}] [TImer {On|OFf}]
Display/Clear Trigger Level	Trigger	Level level# [CLear]
Set Trigger	Trigger	[RUn] {level# [<THen level#>] [Reset | If state# <ELse state#>] | If state# <ELse state#>} [FOrward|BAckward| CEnter|DElay count]
Set Initial Timer & Trace	Trigger	[TImer {On|OFf}] [Trace {On|OFf}]
Display/Clear Trigger	Trigger	[CLear]

These commands are used to trace program execution in real-time, with the emulation processor free running. Four bus breakpoints, three execution breakpoints and an external signal event can be set singly or in combination, using flexible trigger constructs, as conditions for termination of tracing and/or free-running emulation. An optional delay allows emulation/tracing to continue for up to FFFFH machine cycles after all other trigger conditions have been met. Timer On/Off or Trace On/Off may also be specified under any trigger conditions.

System activity is recorded in a 32K trace buffer with a capacity of up to 32768 machine cycles. A qualifier may be set to specify that only cycles meeting certain address bus, processor state conditions and/or external trace-bits be recorded. State and events are also recorded in the trace buffer.

A synchronization command is provided to allow simultaneous start of emulation/tracing on any number of MICE-IIIS 68000 units connected to a multiprocessor system. Operation is completely independent after activation by the sync signal.

The contents of the trace buffer may be listed after termination of the trace. The listing can be set to begin at any recorded cycle and to include either all subsequent cycles or only those cycles meeting certain bus, external and/or processor state conditions. Machine code recorded from the data bus may be displayed in disassembled form with the use of Source option.

If it is desired to use an external signal as a trigger condition, a signal I/O cable (provided with your MICE unit) must be connected between the signal source and the EXT TRIGGER INPUT port on the MICE-IIIS 68000 front panel. To record external signals in the trace buffer, the 9-miniprobe trace cable supplied with the MICE must be attached to the source points and plugged into the TRACE BITS port on the MICE. External trace-bits may also be used as a breakpoint condition. Signals input at all of these ports must be TTL compatible.

Tracing and free-run emulation are in real-time. Information monitored during tracing and emulation are displayed in the trace buffer listing and at the Cycle and Step commands. These are defined under the "COLUMN HEADING" in Table 3-1 below.

Table 3-1 shows the types of processor activities which are to be specified as conditions for trace-buffer recording/listing and bus breakpoints 1 to 4. The codes shown at the left column of the table are used to specify that only machine cycles with certain types of processor activity can be recorded in the trace buffer and displayed in the trace buffer listing. The codes are also used in Event 1 to Event 4 commands where a breakpoint is encountered at the specified address only when the indicated type of processor activity is matched.

Note that there is no limitation on the number of status types that may be specified. If no status is specified, all machine cycles will default.

COLUMN HEADING	DEFINITION
FRAME	is the sequential position of the displayed execution cycle; the range is 0-7FFF (0-32767).
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.
TRACE BITS	indicates signal level input from miniprobes connected to the external TRACE-BITS port.
STATE	indicates trigger states.
EVENT	indicates the "Event encounter-flag" of EV1～EV5.

Table 3-1 Information Recorded in the Trace Buffer

Addendum 4
LAM-IIS COMMAND DESCRIPTION AND EXAMPLES

The following pages will describe the command syntax which are unique to the LAM-IIS equipped MICE-IIIS 68000 under system operating configurations described in Chapter 1, Section 1.3 of the main manual. These commands are arranged in alphabetical order and each command is provided with one or more typical examples. Details and examples of other commands which are common to both LAM and LAM-IIS are provided in Chapter 5 of the main manual.

For greater convenience in interpreting the symbols and notations used in each command syntax and in the given examples, user should first become familiar with the Conventions of Notation explained in Chapter 4, Section 4.1 of the main manual.

Using the Assemble command (Section 5.4.1 of the main manual), the following programs were derived and utilized to obtain most of the command execution examples given in this chapter. User should use these programs to duplicate the examples.

1. Writing data 55AA to memory 1000-1FFFH.

```
LOC       OBJ            SOURCE CODE
000400    327C 1000      MOVEA.W  #1000,A1
000404    32FC 55AA      MOVE.W   #55AA,(A1)+
000408    B2FC 2000      CMPA.W   #2000,A1
00040C    66F6           BNE.B    000404
00040E    4EF8 040E      JMP      00040E
000412    4E71           NOP
000414    4E71           NOP
000416    4E71           NOP
```

2. Checking memory 1000-1FFFH. If it does not equal to 55AA, then go to 518.

```
LOC         OBJ              SOURCE CODE
000500      327C 1000        MOVEA.W   #1000,A1
000504      3019             MOVE.W    (A1)+,D0
000506      B07C 55AA        CMP.W     #55AA,D0
00050A      660C             BNE.B     000518
00050C      B2FC 2000        CMPA.W    #2000,A1
000510      66F2             BNE.B     000504
000512      4EF8 0512        JMP       000512
000516      4E71             NOP
000518      4EF8 0518        JMP       000518
00051C      4E71             NOP
00051E      4E71             NOP
000520      4E71             NOP
```

4.1 CODE-COVERAGE TEST - COVerage

```
COVerage [adr1 adr2[ CLear]][ENable|DIsable|CLear|Reset]]
COVerage List [adr]
COVerage SP|UP|SD|UD
```

COVerage is the command to set or clear coverage settings. COVerage alone without any parameter will display the current coverage memory type and address ranges settings.

adr1 is the hexadecimal address specifying the starting point of a memory block.

adr2 is the hexadecimal address indicating the ending point of a memory block.

CLear is used to clear a particular range of address or to clear all existing address ranges.

ENable is a global control to enable all of the address settings.

NOTE

If the INItialize command (Break on Read-Before-Write function, Section 4.6 of this Addendum) is active at the time when COVerage command is enabled, the INItialize command will automatically switch to inactive condition. Likewise, COVerage function is disabled when INItialize command is enabled while COVerage is active. Hence only one command (COVerage or INItialize) is active at a time.

DIsable is a global control to disable the settings.

Reset re-initializes the code-coverage test without clearing the address ranges setting.

List lists the address ranges and the percentage of memory which were accessed for a coverage test.

adr is the hexadecimal address specifying the starting point of the address ranges to be listed.

SP|UP|SD|UD are the four codes specifying memory type to be executed with the code coverage test.

Where: SP is the supervisor program
SD is the supervisor data
UP is the user program
UD is the user data
If none is specified, all four memory types will default.

Code-coverage test checks the execution efficiency of a users program and displays the total rate of accessed memory ranges (actual accessed ranges / user specified ranges). The address ranges actually covered or accessed are also displayed.

A maximum of 1M-bytes in four independent 256K-byte ranges of memory (0~3FFFFH, 40000H~7FFFFH, etc.) can be set for code-coverage test. Up to 20 coverage address ranges can be set within these four independent 256K-byte memory banks of the 16M memory map. The locations of each of the coverage address must be kept within the boundaries of these four independent 256K-byte ranges. No coverage address range should straddle across any other range other than the specified four banks. If a non-specified range is included in a coverage address settings, the following error message will display:

```
"At most 20 address sets."
```

or

```
"At most 4 bank sets."
```

Code-coverage test is effective only in real-time emulation. It will not work under Cycle Step or Instruction Step commands. Note that the prefetched cycle during free-running is also in effect when performing code-coverage test.

If COVerage is invoked while INItialize is active, the former is disabled and the following message will display:

```
"Coverage test is globally disabled"
```

User should therefore enable the COVerage command first, before invoking COVerage. INItialize is automatically disabled whenever COVerage is enabled if the former is active. Both commands are disabled by default on MICE power up.

Examples:

1. Using the program provided at the beginning of this chapter, enable code-coverage test, set memory ranges to be covered, run emulation and then list the program execution efficiency and the address ranges actually covered:

```
>RESet<CR>
>COVerage ENable<CR>
>COVerage 400 430<CR>
>COVerage 1000 2000<CR>
>COVerage 3000 4000<CR>
>COVerage<CR>
 The ranges for performing coverage test :
 Memory type : SP,SD,UP,UD
  000400 000430
  001000 002000
  003000 004000
>Go 400<CR>
 Trace in progress...
 Emulation processor stopped by user
 Last frame = 7FFF,timer = 02.048 093 7
>COVerage List<CR>
 Percentage of memory accessed in the program ranges :
 49.90%
 The list of address ranges are covered :
 000400 000411
 001000 001FFF
>
```

2. Using the program provided at the beginning of this chapter, clear one of the set ranges and list the program execution efficiency and the address ranges actually covered.

```
>COVerage<CR>
 The ranges for performing coverage test :
 Memory type : SP,SD,UP,UD
  000400 000430
  001000 002000
  003000 004000
>COVerage List<CR>
 Percentage of memory accessed in the program ranges :
 49.90%
 The list of address ranges are covered :
 000400 000411
 001000 001FFF
>COVerage 400 430 CLear<CR>
>COVerage List<CR>
 Percentage of memory accessed in the program ranges :
 49.98%
 The list of address ranges are covered :
 001000 001FFF
>
```

3. Using the program provided at the beginning of this chapter, reset the coverage, qualify the memory type "UP", run emulation and list the program execution efficiency and the address ranges actually covered.

```
>COVerage RESet<CR>
>Register S<CR>
S      1 0
I2     1 <ESC>
>COVerage UP<CR>
>COVerage 400 430<CR>
>COVerage<CR>
 The ranges for performing coverage test :
 Memory type : UP
  000400 000430
  001000 002000
  003000 004000
>COVerage List<CR>
 Percentage of memory accessed in the program ranges :
 00.00%
>Go 400<CR>
 Trace in progress...
 Emulation processor stopped by user
```

```
 Last frame = 7FFF,timer = 01.406 338 9
>
>COVerage_List<CR>
 Percentage of memory accessed in the program ranges :
 00.21%
 The list of address ranges are covered :
 000400 000411
>
```

4. Using the program provided at the beginning of this chapter, reset and disable the coverage test, then run emulation and list the program execution efficiency and the address ranges actually covered.

```
>COVerage_REset<CR>
>COVerage_DIsable<CR>
>COVerage<CR>
 Coverage test globally disabled.
>COVerage_List<CR>
 Coverage test globally disabled.
>Go_400<CR>
 Trace in progress...
 Emulation processor stopped by user
 Last frame = 7FFF,timer = 01.223 123 1
>COVerage_List<CR>
 Coverage test globally disabled.
>COVerage_Enable<CR>
>COVerage_List<CR>
 Percentage of memory accessed in the program ranges :
 00.00%
>
```

5. Clear all the coverage settings.

```
>COVerage_CLear<CR>
>COVerage<CR>
 No address range setting for performing coverage test.
>
```

4.2 DISPLAY/SET/CLEAR BREAKPOINTS - Event

Display Breakpoint Settings	**Event**
Set Bus Breakpoint 1	**Ev[ent]1 {Range adr1 adr2|adx} [datum] [status] [EXternaltrace-bits] [COunt count]**
Set Bus Breakpoint 2	**Ev[ent]2 {Range adr1 adr2|adx} [datum] [status] [EXternal trace-bits] [COunt count]**
Set Bus Breakpoint 3	**Ev[ent]3 {Range adr1 adr2|adx} [datum] [status] [EXternal trace-bits] [COunt count]**
Set Bus Breakpoint 4	**Ev[ent]4 {Range adr1 adr2|adx} [datum] [status] [EXternal trace-bits] [COunt count]**
Set External Hardware Breakpoint	**Ev[ent]5 [COnditional] {High|Low}**
Set Execution Breakpoint 1	**Ev[ent]6 adr [fetch-state]**
Set Execution Breakpoint 1	**Ev[ent]6 adr [fetch-state] [Vector-Base value]**[1]
Set Execution Breakpoint 2	**Ev[ent]7 adr [fetch-state]**
Set Execution Breakpoint 2	**Ev[ent]7 adr [fetch-state] [Vector-Base value]**
Set Execution Breakpoint 3	**Ev[ent]8 adr [fetch-state]**
Set Execution Breakpoint 3	**Ev[ent]8 adr [fetch-state] [Vector-Base value]**
Clear Events	**Ev[ent] [[1|2|3|4|5|6|7|8] CLear]**

Event is the command to display, set or clear breakpoints.

1,2,3,4,5,6,7 specifies the breakpoint to be set or cleared.

Range is the required prefix when specifying an address range.

adr1 is a hexadecimal starting address for range breakpoint.

adr2 is a hexadecimal ending address for range breakpoint.

adx is a hexadecimal address setting, with option to include wildcard nibbles/bits, e.g. 1234, 1XX4, 1(X1X0)X4, 1(X1X0X0XX)4.

---

[1] [Vector-Based value] option is applicable to 68010 CPU only. It does not work on 68000 CPU.

datum is a byte or word[2] in hexadecimal with wildcard nibbles/bits option (e.g. 1234, 1XX4, 1(X1X0)X4, 1(X1X0X0XX)4), to be matched on the data bus; may include wildcard nibbles and/or bits, or is specified as "X" (don't care) if only 'status' and/or 'count' must be matched along with the breakpoint address.

status is 1 to 7 processor status codes (SP SR SW UP UR UW AK) specifying processor state to be matched as part of the breakpoint condition.

EXternal is the required prefix when specifying trace-bits.

trace-bits is the 8 bits external signals from the external TRACE BITS port.

COunt is the required prefix when specifying a "count" (see next paragraph).

count is a hexadecimal value from 0H to 0FFFFH specifying the number of times the breakpoint condition must be matched to complete the specific event (Event 1 or 4). Default is for O1H.

COnditional specifies the signal at the EXT TRIGGER INPUT port for Event 5. This signal is only valid during low state $\overline{\text{STROBE}}$ ($\overline{\text{LDS}}$ or $\overline{\text{UDS}}$) condition.

High|Low specifies the signal level to be matched at the EXT TRIGGER INPUT port for Event 5. Default is for a TTL "low" signal.

adr is a hexadecimal address setting in the program memory space of the emulation processor.

fetch-state is 1 to 2 processor status codes (SP UP).

Vector-Base value is a hexadecimal value of vector-base register for processing "exception" vector address on 68010 CPU only. New value is only assigned when user's program required such value to change. If omitted, MICE will then use the current vector-base register contents (set during Event command setting) as default value.

---

[2] "datum" can not be a "word" when MICE is emulating 68000 (8-bit).

If vector-base value is changed during program execution and the new value does not match with the default or previously set value, Events 4, 5 and 6 will not break at the set address. Also the resulting program execution will be indecipherable.

CLear is the parameter to clear a particular breakpoint setting or to clear all of them at once.

Breakpoints affect only real-time emulation/tracing, not operation during the Cycle Step or Instruction Step commands. Any breakpoints that have been set, therefore, become active only after input of the Go command (Section 4.3 of this Addendum). In addition, Events 1 to 5 affect emulation/tracing only as specified in the Trigger setting (Section 4.11 of this Addendum). The power-on default setting of the MICE-IIIS 68000 is:

- Events 1, 2, 3, 4, 6, 7 and 8 Clear
- Event 5 set for a "low" signal, not qualified by $\overline{\text{LDS}}$ or $\overline{\text{UDS}}$
- Trigger set for EV1 OR EV2 OR EV3 OR EV4

Events 6, 7 and 8, if set, are automatically combined with each other and the Trigger setting in a logical OR construct; The Trigger setting may be cleared in order to deactivate Events 1 to 5 and control emulation/tracing with Events 6, 7 and/or 8 alone.

4.2.1 DISPLAY BREAKPOINT SETTINGS - Event

Event

"Event" alone without any parameters, will display all current breakpoint settings

Example: Display the current settings for Event 1 to 8.

```
>Event<CR>
 EV1 (Bus) Range 112233 556677 9XAX SP External XD Count EEFF
 EV2 (Bus) 1234XX XX78 SR,SW External FF Count 9ABC
 EV3 (Bus) 123456 90AB  External XX Count 0001
 EV4 (Bus) XXXXXX XXXX  External XX Count 0001
 EV5 (External) Conditional High
 EV6 (Execution) 000450 SP
 EV7 (Execution) 000560 SP,UP
 EV8 (Execution) 000780 SP
>
```

4.2.2 SET BUS BREAKPOINTS - Event 1, Event 2, Event 3, Event 4

Ev[ent]1 {Range adr1 adr2|adx} [datum] [status] [EXternal trace-bits] [COunt count]
Ev[ent]2 {Range adr1 adr2|adx} [datum] [status] [EXternal trace-bits] [COunt count]
Ev[ent]3 {Range adr1 adr2|adx} [datum] [status] [EXternal trace-bits] [COunt count]
Ev[ent]4 {Range adr1 adr2|adx} [datum] [status] [EXternal trace-bits] [COunt count]

Events 1 to 4 are real-time bus breakpoints. They consist of up to five elements: address, data, status, trace-bits and count.

An address must be specified; all other parameters are optional. When specified, data-bus value, processor status and trace-bits value must be matched in the same bus cycle as the breakpoint address. If more than one status is included, any one of those specified will satisfy the breakpoint condition. If none is given, the break-point may be encountered during any type of processor activity.

Address setting can be specified in range or wildcard. When using the prefix "Range", "adr1" should be the start address and "adr2" the end address of the range breakpoint. When wildcard breakpoint is used, only one address is neces-sary. The prefix "External" is used only when specifying trace-bits signals of TRACE BITS port. The user may connect up to 8 target signals to LAM-IIS board through the TRACE BITS port. These signals are not only recorded in the trace buffer but may be also specified in each event setting by the user.

An optional count may be specified to indicate the number of times the breakpoint condition must be matched in order to complete the event. If no count is specified, the default is 01H and the event will be encountered at the first match.

Whenever Event 1 or 2 is matched, an external sync signal is output. A negative signal pulse of 200 ns duration is generated at the SYNC 1 OUTPUT port for Event 1 and at the SYNC 2 OUTPUT port for Event 2.

These signals can be sent to an external device such as an oscilloscope or a logic analyzer. For further details refer to Section 5.3, LAM-IIS TIMING NOTES. For in-structions on generating a continuous SYNC 2 output, refer to the description of the Set Trigger command (Section 4.11 of this Addendum).

If the Trigger setting is matched at a bus breakpoint, the MICE will stop tracing at the end of the current cycle or the next cycle, depending on CPU speed and its wait cycle.

NOTE

1) ***When setting a bus breakpoint at a program fetch cycle, the breakpoint may be set at any byte address for an 8-bit-wide data bus, but must be set at a word boundary for a 16-bit-wide data bus.***

2) ***Bus breakpoints match bus activity, not the program counter. If the desired breakpoint address immediately follows any jump or branch instruction, set the actual breakpoint 2 or 3 words beyond the program counter. This is not necessary for execution breakpoints (Events 6, 7 and 8) because they match execution activity in the microprocessor, breaking emulation only when the instruction at the specified address is actually to be executed and not when it is just prefetched.***

Examples:

1. Set Event 1 for address 8050H and a count of 5; set Event 2 for any address in the range of 10H to 1FH, using wildcard option, data of 41H and status of SW; Set Event 3 for any address in the range of 1083H to 1097H, External trace-bits 23H and a count of 94FDH; Set Event 4 for status of SR, UR; then display the current breakpoint settings. (Remember that Events 1 to 5 are not active unless specified in the Trigger setting, the default for which is EV1, EV2, EV3 or EV4.)

```
>Event 1 8050 COunt 5<CR>
>Event 2 1X 41 SW<CR>
>Event 3 Range 1083 1097 External 23 COunt 94FD<CR>
>Event 4 X SR UR<CR>
>Event<CR>
 EV1 (Bus) 008050 XXXX  External XX Count 0005
 EV2 (Bus) 00001X 0041 SW External XX Count 0001
 EV3 (Bus) Range 001083 001097 XXXX  External 23 Count 94FD
 EV4 (Bus) XXXXXX XXXX SR,UR External XX Count 0001
 EV5 (External) Low
 EV6 (Execution) Clear
 EV7 (Execution) Clear
 EV8 (Execution) Clear
>
```

2. Set Event 1 for range address 1000H to 2000H, data of 1234H and status of UW; Set Event 2 of range address 10H to 1FH (using "Range" prefix), data of 1200H to 12FFH, status of SP, external trace-bits 3XH and a count of 10H; Set Event 3 for any location X1200H in any 1M-byte memory block in the 68000 memory range; Set Event 4 for any location X1200H within the first 1M-byte block. Also set a count of 20 and a data value with multiple wildcard for Event 4.

```
>Event 1 Range 1000 2000 1234 UW<CR>
>Event 2 Range 10 1F 12XX SP EXternal 3X COunt 10<CR>
>Event 3 X1200<CR>
>Event 4 0X1200 (X01X)2X(XXX0) COunt 20<CR>
>Event<CR>
 EV1 (Bus) Range 001000 002000 1234 UW External XX Count 0001
 EV2 (Bus) Range 000010 00001F 12XX SP External 3X Count 0010
 EV3 (Bus) XX1200 XXXX  External XX Count 0001
 EV4 (Bus) 0X1200 (X01X)2X(XXX0)  External XX Count 0020
 EV5 (External) Low
 EV6 (Execution) Clear
 EV7 (Execution) Clear
 EV8 (Execution) Clear
>
```

4.2.3 SET EXTERNAL HARDWARE BREAKPOINTS - Event 5

Ev[ent]5 [COnditional] {High|Low}

The external hardware breakpoint can be set to match either high/floating or low signal input. Note, however, that a setting of high/floating will be matched immediately if the EXT TRIGGER INPUT port is not connected to an appropriate signal source. The system default for Event 5 after power-on or a hardware reset is low. Event 5 logic is TTL compatible; a match occurs when the specified logic state is detected. For further details refer to Appendix C of the main manual for Breakpoint Timing.

The optional parameter "COnditional" allows signal from EXT TRIGGER INPUT port to remain valid only when the bus cycle is valid ($\overline{\text{STROBE}}$ is low whenever the bus cycle is valid. For MICE-IIIS 68000, $\overline{\text{STROBE}}$ is generated from CPU $\overline{\text{LDS}}$ or $\overline{\text{UDS}}$ signal). The following timing diagram illustrates how "COnditional" option works with Event 5 is set for a high signal level at the EXT TRIGGER INPUT port.

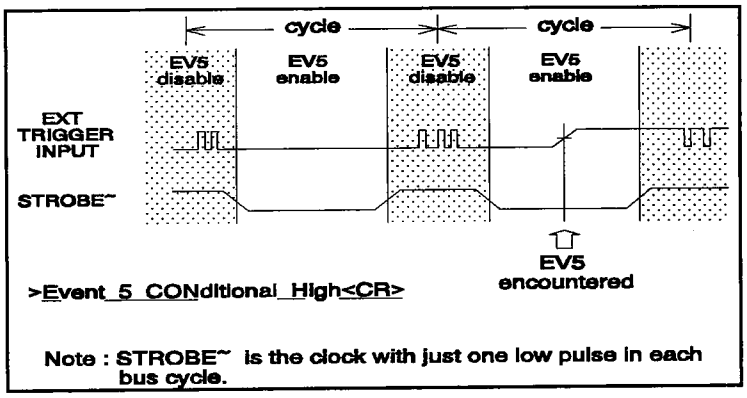


Figure 5-1 Event 5 COnditional High

NOTE

The input signal must meet the specifications for a 74LS14 Schmitt trigger IC.

Example: Set Event 5 for a low signal level at the EXT TRIGGER INPUT port and $\overline{LDS}$ or $\overline{UDS}$ low (remember that the setting does not become effective until Event 5 is specified in the Trigger setting).

```
>Event 5 COnditional Low<CR>
>
```

4.2.4 SET EXECUTION BREAKPOINTS - Event 6, Event 7, Event 8

Ev[ent]6 adr [fetch-state]
Ev[ent]6 adr [fetch-state] [Vector-Base value][3]
Ev[ent]7 adr [fetch-state]
Ev[ent]7 adr [fetch-state] [Vector-Base value]
Ev[ent]8 adr [fetch-state]
Ev[ent]8 adr [fetch-state] [Vector-Base value]

The LAM-IIS equipped MICE-IIIS 68000 provides three realtime execution breakpoints consisting of a required address. Tracing is always terminated and the emulation pro-cessor halted when an execution breakpoint is encountered.

When either Event 6, 7, or Event 8 is matched, the MICE halts the emulation processor at the cycle specified by the break condition. All machine cycles up to the breakpoint are recorded, including prefetch cycles.

Execution breakpoints remain effective. Unlike bus breakpoints, these breakpoints end the trace at a point in the program where the specified address is actually going to be executed.

The following program segment illustrates the difference between execution breakpoints and bus breakpoints:

```
>Disassemble 8000 801E<CR>
LOC         OBJ                 SOURCE CODE
008000      1A3C 0010           MOVE.B    #10,D5
008004      207C 0000 1000      MOVEA.L   #00001000,A0
00800A      227C 0000 2000      MOVEA.L   #00002000,A1
008010      3018                MOVE.W    (A0)+,D0
008012      6100 0FEC           BSR.W     009000
008016      32C1                MOVE.W    D1,(A1)+
008018      5305                SUBQ.B    #01,D5
00801A      66F4                BNE.B     008010
00801C      4E71                NOP
00801E      60FE                BRA.B     00801E
 Disassembly completed
>
```

[3] [Vector-Based value] option is applicable to 68010 CPU only. It does not work on 68000 CPU.

Setting a bus breakpoint at instruction NOP (801C) in the above example would cause the program to break at a bus prefetch cycle, even though program execu-tion has not reached the breakpoint. If the same breakpoint address is set using an execution breakpoint, the program will break only if execution reaches the breakpoint address.

NOTE

1) ***When the program stops at the breakpoint address, the user's program instruction at that location has not yet been executed. The same address cannot be specified again. Setting any breakpoint where the PC equals the specified address will halt tracing immediately.***

2) ***Execution breakpoints must be set at the first word of an instruction.***

3) ***For Events 6, 7 and 8, emulation is controlled with 68000 line emulator code 0A0H.***

4) ***Stack contents from SP-1 thru SP-6 (SP-8 for 68010) will be modified if the breakpoint is executed; but the USP and SSP registers will not be affected.***

5) ***The code for breakpoint addresses is modified to 0A0H 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.***

6) ***0A0H instructions defined in user's program are executed correctly without any conflict with internal 0A0H codes generated by Event 6, 7 or 8.***

7) ***If any supervisor data memory read cycle occurs at 028H (or a line emulator code 0A0H in the user's program is executed), a few non-real time cycles will be inserted to process this code.***

8) ***The execution breakpoint can be set to RAM/ROM area.***

9) ***If the execution breakpoint is set and the bus breakpoint is matched, the CPU will finish executing the current instruction cycle and stop at the opcode fetch of the next instruction.***

Example: Using the program provided at the beginning of this chapter, set the bus breakpoint 1 at address 40E and display the register, then clear the bus breakpoint. Set the execution breakpoint to the same address and display register again.

```
>EVent CLear<CR>
>Disassemble 400 414<CR>
LOC         OBJ              SOURCE CODE
000400      327C 1000        MOVEA.W  #1000,A1
000404      32FC 55AA        MOVE.W   #55AA,(A1)+
000408      B2FC 2000        CMPA.W   #2000,A1
00040C      66F6             BNE.B    000404
00040E      4EF8 040E        JMP      00040E
000412      4E71             NOP
000414      4E71             NOP
 Disassembly completed
>Event 1 40E<CR>
>Go 400<CR>
 EV1 (Bus) encountered
 Emulation processor stopped
 Last frame = 0008,timer = 00.000 002 7
>Register<CR>
      0        1        2        3        4        5        6        7
Dn  FFFFFFF6 FFFFFFFF 7FFDDFF7 FFFFBFF6 FFFFFFFF DFFFFFFF FFFFFFFF FFF7FFFF
An  FFFF7FFF 00001002 7FFEFFFF EFFFFFDF DFF7FFFD F7FBFFFF FFFFF2DF 0001FFF0
SSP=0001FFF0 USP=FFFFFFFE                SR  T S  III   XNZVC
PC=000404                               2719 0010011100011001
>Event 1 CLear<CR>
>Event 6 40E<CR>
>Go 400<CR>
 Trace in progress...
 EV6 (Execution) encountered
 Emulation processor stopped
 Last frame = 1FFB,timer = 00.005 463 8
>Register<CR>
      0        1        2        3        4        5        6        7
Dn  FFFFFFF6 FFFFFFFF 7FFDDFF7 FFFFBFF6 FFFFFFFF DFFFFFFF FFFFFFFF FFF7FFFF
An  FFFF7FFF 00002000 7FFEFFFF EFFFFFDF DFF7FFFD F7FBFFFF FFFFF2DF 0001FFF0
SSP=0001FFF0 USP=FFFFFFFE                SR  T S  III   XNZVC
PC=00040E                               2714 0010011100010100
>
```

4.2.5 CLEAR BREAKPOINTS - Event CLear

Ev[ent]1 CLear
Ev[ent]2 CLear
Ev[ent]3 CLear
Ev[ent]4 CLear
Ev[ent]5 CLear
Ev[ent]6 CLear
Ev[ent]7 CLear
Ev[ent]8 CLear

Keying "Event Clear" alone, without any other parameters, clears the settings for Events 1, 2, 3, 4, 6, 7 and 8, and sets Event 5 to low, not qualify $\overline{LDS}$ or $\overline{UDS}$. To clear Event n only, input "Event n Clear."

Examples:

1. Display all breakpoint settings, then clear bus breakpoint Event 4.

```
>Event<CR>
 EV1 (Bus) Range 001000 002000 1234 UW External XX Count 0001
 EV2 (Bus) Range 000010 00001F 12XX SP External 3X Count 0010
 EV3 (Bus) XX1200 XXXX  External XX Count 0001
 EV4 (Bus) 0X1200 (X01X)2X(XXX0)  External XX Count 0020
 EV5 (External) Conditional Low
 EV6 (Execution) 000500 UP
 EV7 (Execution) 008000 SP,UP
 EV8 (Execution) Clear
>
>Event 4 CLear<CR>
>Event<CR>
 EV1 (Bus) Range 001000 002000 1234 UW External XX Count0001
 EV2 (Bus) Range 000010 00001F 12XX SP External 3X Count 0010
 EV3 (Bus) XX1200 XXXX  External XX Count 0001
 EV4 (Bus) Clear
 EV5 (External) Conditional Low
 EV6 (Execution) 000500 UP  Count 0001
 EV7 (Execution) 008000 SP,UP
 EV8 (Execution) Clear
>
```

2. Display all breakpoint settings, then clear.

```
>Event<CR>
 EV1 (Bus) Range 001000 002000 1234 UW External XX Count 0001
 EV2 (Bus) Range 000010 00001F 12XX SP External 3X Count 0010
 EV3 (Bus) XX1200 XXXX  External XX Count 0001
 EV4 (Bus) 0X1200 (X01X)2X(XXX0)  External XX Count 0020
 EV5 (External) Conditional Low
 EV6 (Execution) 000500 UP
 EV7 (Execution) 008000 SP,UP
 EV8 (Execution) Clear
>
>Event CLear<CR>
>Event<CR>
 EV1 (Bus) Clear
 EV2 (Bus) Clear
 EV3 (Bus) Clear
 EV4 (Bus) Clear
 EV5 (External) Low
 EV6 (Execution) Clear
 EV7 (Execution) Clear
 EV8 (Execution) Clear
>
```

4.3 GO/EXECUTION - Go

Go [Run] [adr]

Go is the command for Go/Execution.

Run specifies free run. During free run all breakpoint settings are ignored, but target information is still recorded in the trace buffer. (Any command other than Halt, Clock, Memory, Port, COVerage, INItialize commands and emulation group of commands may be input without stopping the emulation processor.)

adr is the hexadecimal (logical) address setting in the emulation CPU's program memory where emulation is to begin. Default is to begin from current PC.

If an address is specified, the emulation processor's program counter is first set accordingly. The MICE-IIIS 68000 then starts real-time emulation of the emulation processor, and immediately begins recording system status information as specified in the Trace Cycle Qualify setting. If no address is specified, emulation starts at the current program counter. Note that the green **"EP RUN"** indicator on the front panel is lit whenever the emulation CPU is active. The recorded information can be viewed with List Trace Buffer commands.

During emulation,

```
"Trace in progress..."
```

is displayed as long as data is being recorded in the trace buffer. When the Trigger condition is matched, the message-

```
"EV# encountered"
```

is displayed. If an attempt is made to write into Read-only area (defined as "IR" or "ER" in the MAp command (Section 5.3.8 of the main manual), the message-

```
"...Write protected memory trespassed!"
```

will display. Likewise, if a non-existent area (defined as "G" in the MAp command is accessed, the following message will display-

```
...Guarded memory trespassed!"
```

If the Run option was not specified for either the Trigger setting or Event 6, the additional message-

```
"Emulation processor stopped"
```

will also display. If no breakpoint is encountered (i.e., `"Trace in progress..."` remains on the screen), the user may terminate emulation at any time by pressing <ESC>; the message-

```
"Emulation processor stopped by user"
```

will then be displayed.

Note, however, that if several breakpoints are matched nearly simultaneously, the order in which they are listed in the `"EV# encountered"` message may not reflect the actual order in which they were encountered during emulation. This is especial-ly true when both execution and bus/signal breakpoints are matched almost simultaneously: because of the higher priority of execution breakpoints, Events 6, 7 and 8 will be displayed before Events 1 to 5 even if they were encountered slightly later.

Examples:

1. Using the program provided at the beginning of this chapter, display register contents and then start tracing all bus activity until trigger address 40C is accessed. Use a timebase of 1 μs, with the timer set to start when emulation begins.

```
>Register<CR>
      0        1        2        3        4        5        6        7
 Dn  FFFFFFF6 FFFFFFFF 7FFDDFF7 FFFFBFF6 FFFFFFFF DFFFFFFF FFFFFFFF FFF7FFFF
 An  FFFF7FFF 00002000 7FFEFFFF EFFFFFDF DFF7FFFD F7FBFFFF FFFFF2DF 0001FFF0
 SSP=0001FFF0 USP=FFFFFFFE                SR  T S  III   XNZVC
 PC=000400                                 2714 0010011100010100
>Event 1 40C<CR>
>TRigger Level A EV1<CR>
>TRigger A<CR>
>TImebase 1<CR>
>Go<CR>
 EV1 (Bus)  encountered
 Emulation processor stopped
 Last frame = 0008,timer = 00.000 002 0
>
```

The time that elapses before the trace buffer is filled, depends on command specifications, and may be quite lengthy. If the trigger condition has not yet been matched, you may enter an <ESC> to terminate the trace, retaining the information already recorded in the trace buffer.

2. Using the program provided at the beginning of this chapter, set the trigger for a low signal at the external trigger input port, and when this is not matched enter an <ESC> to terminate tracing.

```
>Event 5 Low<CR>
>Trigger Level A EV5<CR>
>Trigger A<CR>
>Go 400<CR>
 Trace in progress...<ESC>
 Emulation processor stopped by user
 Last frame = 7FFF,timer = 00.528 890 0
>
```

3. Using the program provided at the beginning of this chapter, set bus breakpoint EV1, address 40C and Trigger Level A EV1. Then free run by using "Run" option. Clear the Trigger, then execute "Go" command again.

```
>Event 1 40C<CR>
>TRigger<CR>
 TRigger A  Delay 0000
 Trigger Timer On Trace On
 Level A  EV5 Trace On Timer On
 Level B  Clear
 Level C  Clear
 Level D  Clear
 Level E  Clear
 Level F  Clear
 Level G  Clear
 Level H  Clear
>TRigger LEvel A EV1<CR>
>Go 400<CR>
 EV1 (Bus) encountered
 Emulation processor stopped
 Last frame = 0008,timer = 00.000 002 0
>Go Run 400<CR>
 Emulation processor free running, all breakpoints masked
Run>HAlt<CR>
 Emulation processor stopped by user
```

```
>TRigger CLear<CR>
>TRigger<CR>
 Trigger Clear
 Trigger Timer On Trace On
 Level A  EV1 Trace On Timer On
 Level B  Clear
 Level C  Clear
 Level D  Clear
 Level E  Clear
 Level F  Clear
 Level G  Clear
 Level H  Clear
>Go 400<CR>
 EV1 (Bus) encountered
 Emulation processor stopped by user
 Last frame = 7FFF,timer = 01.420 365 0
>
```

4.4 HALT - HAlt

HAlt

HAlt is the command to halt the emulation processor.

This command may be used during free-run emulation to stop program execution at any time. Executing such commands as Register, Cycle, Step, Memory, Input, Output, COVerage and INItialize will also halt emulation. Note that the green **"EP RUN"** indicator on the front panel is lit whenever the emulation processor is active.

The Halt command may also be used after tracing has been stopped by the specified trigger condition, and the emulation processor is in free run state due to a "Run" qualifier in the trigger setting or in the breakpoint setting for Event 6.

Example: Using the program provided at the beginning of this chapter, input the Halt command during emulation to stop the emulation processor.

```
>Event 1 40C<CR>
>Go Run<CR>
 Emulation processor free running, all breakpoints masked
Run>HAlt<CR>
 Emulation processor stopped by user
>
```

4.5 COMMAND HELP - Help

Help [GRoup {Emulation|Memory|Port|Setup|Trace}|command|ALl]

Help is the keyword for Command Help. "Help" alone, without any para-meter, causes the MICE to display a list of command keywords arranged in functional groups.

GRoup is the prefix to use to display a specific functional group of commands. When followed by the desired group's name, the syntax for all the commands in that particular group will display. Definitions of typical parameters used in such group, will also appear on screen. MICE commands are divided into the following groups:

Setup Commands	(see Section 3.2.1 of this Addendum)
Memory Commands	(see Section 3.2.2 of this Addendum)
Port Commands	(see Section 3.2.3 of this Addendum)
Emulation Commands	(see Section 3.2.4 of this Addendum)
Trace Commands	(see Section 3.2.5 of this Addendum)

command is any of the specific MICE command keywords. Keying "Help" followed by a specific command keyword, will display the full syntax for that particular command and the definitions for the typical parameters used by such command.

ALl is the parameter to display full syntax and typical parameter definitions for all available MICE-IIIS 68000 commands.

In order to make efficient use of screen space, special conventions of notation are used in Command Help displays. These conventions are explained in Section 4.1 of the main manual.

Examples:

1. Display a command overview.

```
>Help<CR>
  Setup Group:
   BAud              IDentify          RECall            SETup
   CLock             INTerval          ROute             SIze
   CONtrol           MAp               SAve              Verify
   HANdshake         Prompt            SELect            WAit
   Help              REAdy
  Memory Group:
   Assemble          COMpare           Fill              SEarch
   Byte              COpy              LOng              TEst
   CHecksum          Disassemble       Memory            Word
  Port Group:
   Input             Output
  Emulation Group:
   Cycle             Register          RESet             Step
   Jump
  Trace Group:
   COVerage          HAlt              Qualify           TRAce
   Event             INItialize        SYnc              Trigger
   Go                List              TImebase
>
```

2. Display full syntax for all MICE-IIIS 68000 commands. (When the message "**[More]**" appears, press any key to scroll for the next screen.)

```
>Help ALl<CR>
 Line Assembly              Assemble [space-adr]
 Port B Baud rate Setting   BAud [baud-rate [parity] [data-bits]]
 Memory Modify              Byte [space-adr [data]]
 Memory Checksum            CHecksum space-adr {adr|Length length}
 Display Clock Source       CLock
 Memory Compare             COMpare space-adr1 {adr|Length length} space-adr2
 Control Signal En/Disable  CONtrol [[Berr] [Intr] [BR] {Enable|Disable}]
 Memory Copy                COpy space-adr1 {adr|Length length} space-adr2
 Set Coverage               COVerage [adr1 adr2[ CLear]|[ENable|DIsable|CLear|
                                  Reset]]
                            COVerage List [adr]
                            COVerage SP|UP|SD|UD
 Cycle Step                 Cycle [Wait|count]
 Disassembly                Disassemble [space-adr [adr|Length length]]
```

```
 Set Bus Event 1            Ev[ent ]1 {Range adr1 adr2|adx} [datum] [status]
                                      [EXternal trace-bits] [COunt count]
[ More ]
 Display/Clear Event 1      Ev[ent ]1 [CLear]
 Set Bus Event 2            Ev[ent ]2 {Range adr1 adr2|adx} [datum] [status]
                                      [EXternal trace-bits] [COunt count]
 Display/Clear Event 2      Ev[ent ]2 [CLear]
 Set Bus Event 3            Ev[ent ]3 {Range adr1 adr2|adx} [datum] [status]
                                      [EXternal trace-bits] [COunt count]
 Display/Clear Event 3      Ev[ent ]3 [CLear]
 Set Bus Event 4            Ev[ent ]4 {Range adr1 adr2|adx} [datum] [status]
                                      [EXternal trace-bits] [COunt count]
 Display/Clear Event 4      Ev[ent ]4 [CLear]
 Set External Trigger       Ev[ent ]5 [COnditional] {High|Low}
 Display/Clear Ext-Trigger  Ev[ent ]5 [CLear]
 Set Execution Event 1      Ev[ent ]6 adr [fetch-state]
 Display/Clear Exe-Event 1  Ev[ent ]6 [CLear]
 Set Execution Event 2      Ev[ent ]7 adr [fetch-state]
 Display/Clear Exe-Event 2  Ev[ent ]7 [CLear]
[ More ]
 Set Execution Event 3      Ev[ent ]8 adr [fetch-state]
 Display/Clear Exe-Event 3  Ev[ent ]8 [CLear]
 Memory Fill                Fill space-adr {adr|Length length} data
 Execution                  Go [Run] [adr]
 Halt Emulator              HAlt
 Set Handshaking Code       HANdshake [code]
 Command Help               Help [GRoup {Emulation|Memory|Port|Setup|Trace}
                                 |command|ALl]
 Identify Emulator          IDentify
 Set Initialize             INItialize [adr1 adr2[ CLear]|[ENable|DIsable|
                                       CLear|Reset]]
                            INItialize List [adr]
                            INItialize SP|UP|SD|UD
 Port Input                 Input space-adr [Length in-length]
 Timer Interval Switch      INTerval [On|OFf]
 Jump                       Jump adr
[ More ]
 List Trace                 List [frame [adr1 adr2] [status]
                                 [EXternal trace-bits]]
 List Source Code           List {Source|Instruction} [frame]
 List Frame Total           List Number
 Memory Modify              LOng [space-adr [data]]
 Memory Map                 MAp [space-adr adr {{I|IR} {1|2|3|4}|E|ER|G}
                                | ALl{E|G}]
 Memory Dump                Memory [space-adr [adr|Length length]]
 Port Output                Output space-adr data
```

```
 Change Prompt              Prompt [string]
 Set Trace Cycle Qualifier  Qualify {1|2} [{Range adr1 adr2|adx}
                                          [status [EXternal trace-bits]]
 Display/Clear/Disable      Qualify [1|2] [CLear|DIsable|ENable]
 Enable Qualifier
 Ready Signal Selection     REAdy [Internal|External]
 Recall Status from NOVRAM  RECall
[ More ]
 Register Display/Modify    Register [register]
 Reset Emulation Processor  RESet [pc [ssp]]
 Change Operational Mode    ROute [System]
 Save Status to NOVRAM      SAve
 Memory Search              SEarch space-adr {adr|Length length} data
 Select leading code        SELect [{Route|DEVice} [code]]
 Display Setup Parameters   SETup
 Memory Access Size         SIze [Byte|Word]
 Instruction Step           Step [adr1 adr2 [CALl]|count|CALl]
 Synchronization            SYnc [{Input|OUtput} {On|OFf}]
 Memory Test                TEst space-adr {adr|Length length}
 Timebase Selection         TImebase [0.1|1|10|100|1000]
 Display Trace Parameters   TRAce
 Set Trigger Level          Trigger Level level# [Not] ev# [{OR|ANd} [Not] ev#
                                    [{OR|ANd} [Not] ev# [{OR|ANd} [Not] ev#
                                    [{OR|ANd} [Not] ev# ]]]]
[ More ]
                                    [Trace {On|OFf}] [TImer {On|OFf}]
 Display Trigger Level/     Trigger Level level# [CLear]
 Clear Trigger Level
 Set Trigger                Trigger [RUn] {level# [<THen level#>] [Reset |
                            If state# <ELse state#>]|If state# <ELse state#>}
                            [FOreward|BACKward|CENter|DELay count]
 Set Initial Timer & Trace  Trigger [TImer {On|OFf}] [Trace {On|OFf}]
 Display/Clear Trigger      Trigger [CLear]
 Memory Verification        Verify [On|OFf]
 Insert Wait State          WAit [On|OFf]
 Memory Modify              Word [space-adr [data]]

 ---Definition----------------------------------------------------------------
 adr          is a hexadecimal address setting.
 adr1         is a hexadecimal address indicating the starting point of the
              memory block.
[ More ]
 adr2         is a hexadecimal address indicating the end point of the memory
              block.
 adx          is a hexadecimal address setting, with option to include
              wildcard nibbles/bits, e.g. 1234, 1XX4, 1(X1X0)X4, 1(X1X0X0XX)4.
```

ALl	is the parameter to Display full syntax for all available commands.
BACKward	is the delay count = 0.
baud-rate	is baud rate of serial channel B (110 150 300 600 1200 2400 4800 9600 19200).
CALl	is the keyword for skipping subroutines.
CENter	is the delay count = FFFH/3FFFH for 8K/32K trace buffer.
CLear	is the parameter to clear a particular setting or clear all of them at once.
code	is a one-byte hexadecimal Value.
command	is a specific MICE command.
COnditional	the signal is only valid during the condition of low state

[More]

	STROBE.
count	is a hexadecimal Value (from 0H to 0FFFFH).
data	is 1 to 32 bytes of data in hex and/or ASCII.
data-bits	is either 7 or 8 data bits of serial channel B.
datum	is a byte or word in hexadecimal, with wildcard nibbles/bits option, e.g. 1234, 1XX4, 1(X1X0)X4, 1(X1X0X0XX)4.
DIsable	is the parameter to disable the settings.
ENable	is the parameter to enable the settings.
ev#	is EV1, EV2, EV3, EV4 or EV5 (do not specify same event twice in one command).
fetch-state	is 1 to 2 processor status codes (SP UP).
FOrward	is the delay count = 1FFFH/7FFFH for 8K/32K trace buffer.
frame	is a hexadecimal Value (from 0H to 1FFFH/7FFFH for 8K/32K) indicating the frameat which the listing is to begin.
Instruction	is the parameter to list the trace buffer containing frame, address, data and assembly source-code, but do not include

[More]

	status, trace-bits, timer and read/write cycle.
in-length	is a hexadecimal Value (from 01H to 0FFFFFFH).
length	is a hexadecimal Value (from 01H to 0FFFFFFH) specifying the number of location to access.
level#	is one of level identifiers -- A, B, C, D, E, F, G or H.(do not specify same level twice in one command.
List	is the parameter to list the test result.
parity	is E (even), O (odd) or N (none) parity of serial channel B.
pc	is a hexadecimal address to load in place of the current PC.
Range	is the required prefix when specifying a address range.
register	is the standard Motorola designation for the register, the contents of which are to be displayed or modified.
Reset	initializes the code-coverage test or Read-Before-Write flag, but the range setting will not be cleared.
Source	is the parameter to list the trace buffer containing frame, address, data, status, trace-bits, timer and assembly

```
[ More ]
                source-code.
 space-adr      is a hexadecimal address setting, with option to specify
                memory type: "[UP|UD|SP|SD] adr".
 SP|SD|UP|UD    is one to four two-letter codes specifying memory type.
                Where : SP is the supervisor program
                        SD is the supervisor data
                        UP is the user program
                        UD is the user data
                If none of the memory type is specified in this command,all four
                memory types will default.
 ssp            is a hexadecimal address to load in place of the current SSP.
 state#         level# [RESet] | (level# {<Then level#> [ RESet |
                If state# <ELse state#> ] | If state# <ELse state#> } )
 <    >         Recursive (Minimum 1 time).
 (    )         Reserved Symbols.
 status         is 1 to 7 processor status codes (SP SR SW UP UR UW AK).
[ More ]
 string         is a 1-8 character command prompt, input in ASCII.
 System         is the path of operational mode from Terminal to System mode.
 trace-bits     is a one-byte hexadecimal setting indicating signal levels for
                trace bits input from the external port. Includes option to
                specify wildcard nibbles/bits.
 Wait           stops the CPU in a wait state.

 Note : Uppercase character(s) in the command keyword and command parameters
        represent the minimum abbreviated (shorthand) input that would remain
        transparent to MICE. Lowercase character(s) are optional. A space
        should be entered between a command keyword and a subsequent parameter,
        as well as between parameters.
>
```

3. Display a single command syntax.

```
>Help BAud<CR>
 Port B Baud rate Setting   BAud [baud-rate [parity] [data-bits]]

 --- Definition ----------------------------------------------------
 baud-rate    is baud rate of serial channel B (110 150 300 600 1200
              2400 4800 9600 19200).
 data-bits    is either 7 or 8 data bits of serial channel B.
 parity       is E (even), O (odd) or N (none) parity of serial
              channel B.
>
```

4. Display the "Setup" group command syntax.

```
>Help GRoup Setup<CR>
 Port B Baud rate Setting   BAud [baud-rate [parity] [data-bits]]
 Display Clock Source       CLock
 Control Signal En/Disable  CONtrol [[Berr] [Intr] [BR] {Enable
                                |Disable}]
 Set Handshaking Code       HANdshake [code]
 Command Help               Help [GRoup {Emulation|Memory|Port
                            |Setup|Trace}|command|ALl]
 Identify Emulator          IDentify
 Timer Interval Switch      INTerval [On|OFf]
 Memory Map                 MAp [space-adr adr {{I|IR} {1|2|3|4|}
                                |E|ER|G}|ALl {E|G}]
 Change Prompt              Prompt string
 Ready Signal Selection     REAdy [Internal|External]
 Recall Status from NOVRAM  RECall
 Change Operational Mode    ROute [System]
 Save Status to NOVRAM      SAve
 Select leading code        SELect [{Route|DEVice} [code]]
[ More ]
 Display Setup Parameters   SETup
 Memory Access Size         SIze [Byte|Word]
 Memory Verification        Verify [On|OFf]
 Insert Wait State          WAit [On|OFf]

 --- Definition ----------------------------------------------------
 ALl          is the parameter to display full syntax for all
              available commands.
 baud-rate    is baud rate of serial channel B (110 150 300 600 1200
              2400 4800 9600 19200).
 code         is a one-byte hexadecimal value.
```

```
 command      is a specific MICE command.
[ More ]
 data-bits    is either 7 or 8 data bits of serial channel B.
 parity       is E (even), O (odd) or N (none) parity of serial
              channel B.
 string       is a 1-8 character command prompt, input in ASCII.
 System       is the path of operational mode from Terminal to
              System mode.
>
```

5. Display the "Register" command syntax.

```
>Help Register<CR>
 Register Display/Modify    Register [register]

 --- Definition ----------------------------------------------------
 register     is the standard Motorola designation for the
              register, the contents of which are to be displayed
              or modified.
>
```

4.6 BREAK ON READ BEFORE WRITE - INItialize

INItialize [adr1 adr2[CLear]|[ENable|DIsable|CLear|Reset]]
INItialize List [adr]
INItialize SP|UP|SD|UD

INItialize is the command to set, display or clear "INItialize" or "Read-Before-Write" function.

adr1 is the hexadecimal address specifying the starting point of a memory block.

adr2 is the hexadecimal address indicating the last point of a memory block.

CLear is used to clear a particular range of address or to clear all existing address ranges.

ENable is a global control to enable all of the address settings.

NOTE

If the COVerage (Section 4.1 of this Addendum) command is active at the time when the INItialize command is enabled, the COVerage command is automatically switch to inactive condition. Likewise, INItialize function is disabled when COVerage command is enabled while INItialize is active. Hence only one command (INItialize or COVerage) is active at a time.

DIsable is a global control to disable, but without clearing the settings.

Reset re-initializes the Read-Before-Write function without clearing the address ranges setting.

List lists the address ranges which were actually written after running emulation.

Adr is the hexadecimal address specifying the starting point of the address ranges to be listed.

SP|UP|SD|UD are the four codes specifying memory type to be executed with the Read-Before-Write function.

Where: SP is the supervisor program
SD is the supervisor data
UP is the user program
UD is the user data

If none is specified, all four memory types will default.

INItialize command is used to ensure that only memory range which were previously written, can be read during emulation, otherwise the "Read-Before-Write" break will be encountered. This function is most helpful when emulating such areas as vector space, input/output, RAM, etc. The written areas may be displayed with the INItialized List command. Whenever processor tries to read an unwritten area, emulation is aborted and the following message will display-

```
"Read-Before-Write Encountered"
```

If INItialize is invoked while COVerage is active, INItialize is ignored and the following message will display:

```
"Read-Before-Write is globally disabled"
```

User should therefore enable the INItialize command first, before invoking INItialize. COVerage is automatically disabled whenever INItialize is enabled if the former is active. Both commands are disabled by default on MICE power up.

Up to 20 initialized ranges of memory may be set in four independent 256K-byte of memory segments for this function.

NOTE

The command is effective only in real-time emulation. It will not work under Cycle Step or Instruction Step commands.

Examples:

1. Using the program provided at the beginning of this chapter, enable "INItialize" function. Set ranges and try to read an unwritten address. List the written areas after running.

```
>INItialize Enable<CR>
>INItialize 1000 1200<CR>
>INItialize 2000 2100<CR>
>INItialize 3000 3100<CR>
>INItialize<CR>
 The ranges for Read-Before-Write breakpoint :
 Memory type : SP,SD,UP,UD
  001000 001200
  002000 002100
  003000 003100
>Go 500<CR>
 Read-Before-Write encountered
 Emulation processor stopped
 Last frame = 0005,timer = 00.000 000 7
```

```
>List<CR>
FRAME ADDRESS DATA STATUS TRACE BITS STATE   EVENT TIMER
 0000 000500  327C   SP     00000000   111    11111 00.000 000 0
 0001 000502  1000   SP     00000000   000    00000 00.000 000 0
 0002 000504  3019   SP     00000000   000    00000 00.000 000 2
 0003 000506  B07C   SP     00000000   000    00000 00.000 000 5
 0004 001000  EFEE   SR     00000000   000    00000 00.000 000 7
 0005 000508  55AA   SP     00000000   000    00000 00.000 000 7
>INItialize List<CR>
 The memory ranges of Read-Before-Write have been written :
 None
>
```

2. Using the program provided at the beginning of this chapter, try to write an address before reading an "INItialized" ranges. List the written areas after running.

```
>INItialize<CR>
 The ranges for Read-Before-Write breakpoint :
 Memory type : SP,SD,UP,UD
  001000 0010FF
  002000 0020FF
  003000 003100
>Go 400<CR>
 Trace in progress...
 Emulation processor stopped by user
 Last frame = 7FFF,timer = 05.018 640 2
>Event 6 512
>Go 500<CR>
 Trace in progress...
 EV6 (Execution) encountered
 Emulation processor stopped
 Last frame = 4802,timer = 00.005 633 5
>INItialize List<CR>
 The memory ranges of Read-Before-Write have been written :
 001000 0010FF
>
```

3. Using the program provided at the beginning of this chapter, clear a range of the "INItialized" settings and list the written area.

```
>INItialize<CR>
 The ranges for Read-Before-Write breakpoint :
 Memory type : SP,SD,UP,UD
  001000 0010FF
  002000 0020FF
  003000 003100
>INItialize_List<CR>
 The memory ranges of Read-Before-Write have been written :
 001000 0010FF
>INItialize_1000_10FF_CLear<CR>
>INItialize_List<CR>
 The memory ranges of Read-Before-Write have been written :
 None
>
```

4. Using the program provided at the beginning of this chapter, reset the "INItialized" sets, qualify the space SD and list the written areas before and after emulation.

```
>INItialize_Reset<CR>
>INItialize<CR>
 The ranges for Read-Before-Write breakpoint :
 Memory type : SP,SD,UP,UD
  001000 001200
  002000 0020FF
  003000 003100
>INItialize_SD<CR>
>INItialize_List<CR>
 The memory ranges of Read-Before-Write have been written :
 None
>Go_400<CR>
 Trace in progress...
 Emulation processor stopped by user
 Last frame = 7FFF,timer = 04.055 596 4
>Go_500<CR>
 Trace in progress...
 EV6 (Execution) encountered
 Emulation processor stopped
 Last frame = 4802,timer = 00.005 633 6
>INItialize_List<CR>
 The memory ranges of Read-Before-Write have been written :
 001000 001200
>
```

5. Using the program provided at the beginning of this chapter, reset and disable the "INItialize" function. Then list the written areas after emulation.

```
>INItialize Reset<CR>
>INItialize DIsable<CR>
>Go 400<CR>
 Trace in progress...
 Emulation processor stopped by user
 Last frame = 7FFF,timer = 07.024 674 8
>Go 500<CR>
 Trace in progress...
 EV6 (Execution) encountered
 Emulation processor stopped
 Last frame = 4802,timer = 00.005 633 5
>INItialize List<CR>
 Read-Before-Write breakpoint globally disabled.
>INItialize ENable<CR>
>INItialize List<CR>
 The memory ranges of Read-Before-Write have been written :
 None
>
```

6. Clear the "INItialize" function.

```
>INItialize CLear<CR>
>
```

4.7 LIST TRACE BUFFER - List

List Trace Results	**List [frame [adr1 adr2] [status] [External trace-bits]]**
List Trace Results with Source/Instruction Code	**List {Source|Instruction} [frame]**
List Total Frames and Time	**List Number**

List is the command to display information recorded in the trace buffer. This information is organized into frames, each representing one execution bus cycle, and displayed one page (screenful) at a time. Address and data bus activity, processor status, trace bit levels and trigger information (STATE and EVENT column) are displayed for each frame. "List" alone, or "List Source" with no other parameters, displays all frames in the last page of the buffer.

frame is a hexadecimal value (from 0H to 7FFFH with 32K trace buffer) indicating the frame at which the listing is to begin. The default is to begin at the first frame in the last page of the buffer.

adr1, adr2 are hexadecimal values specifying a particular range of memory, and indicating that only those cycles with address bus activity within that range, are to be listed. "adr1" indicates the starting address, "adr2" the end address. Default is 0H to FFFFFFH.

status is 1 to 7 processor status codes (SP SR SW UP UR UW AK) representing specific type of processor activity (see Table 5-3 of the main manual) and indicating that only cycles in which the indicated type of activity take place are to be listed. The default is for all cycles regardless of processor status.

EXternal is the required prefix when specifying trace-bits.

trace-bits is a one-byte hexadecimal setting indicating signal levels for trace bits input from the external TRACE BITS port specifying that only cycles in which the indicated levels are matched, are to be listed. This setting includes option to specify wildcard nibbles/bits e.g., "XX" "FX" "F(10XX)" etc., where "X" is a nibble or bit indicating 'don't care'. The default setting is for all cycles regardless of trace bit levels.

Source is the parameter to list the trace buffer containing frame, address, data, status, trace-bits, state, event and assembly source code. Note that the source code listing must start at an instruction fetch cycle, otherwise incorrect code will be displayed for the first few machine cycle.

Instruction is same with "Source" parameter above. Only the trace buffer contents to be listed will not include status, trace-bits, state, event and read/write cycle.

Number displays the number of the last frame recorded in the trace buffer and the overall elapsed time since timer start.

Trace information recorded during emulation is displayed using the List command. If a frame number is specified, the MICE lists the trace buffer starting at the specified frame. If no frame number is specified, the MICE lists all frames in the last page of the buffer. Qualifiers for address range, processor activity and/or external signal levels can optionally be entered as described above so that only cycles in which the specified condition is matched are listed.

One page (i.e., one screenful) of trace information is displayed at a time; press <CR> to display the next page and <SP> for continues scrolling. To terminate the List operation press <ESC>. Note that the command also ends if the trace buffer is empty or a frame number higher than that of the last recorded frame is specified.

Example: Using the program provided at the beginning of this chapter, run a trace up to address 40E and list the last page in the trace buffer.

```
>Disassemble 400 40E<CR>
LOC         OBJ              SOURCE CODE
000400      327C 1000        MOVEA.W  #1000,A1
000404      32FC 55AA        MOVE.W   #55AA,(A1)+
000408      B2FC 2000        CMPA.W   #2000,A1
00040C      66F6             BNE.B    000404
00040E      4EF8 040E        JMP      00040E
 Disassembly completed
>Event 6 40E<CR>
>Go 400<CR>
 Trace in progress...
 EV6 (Execution) encountered
 Emulation processor stopped
 Last frame = 3802,timer = 00.004 097 5
>List<CR>
FRAME ADDRESS DATA STATUS TRACE BITS STATE   EVENT TIMER
 37F3 00040E  A0F8   SP    00000000   000    00000 00.004 091 7
 37F4 000404  32FC   SP    00000000   000    00000 00.004 091 9
 37F5 000406  55AA   SP    00000000   000    00000 00.004 092 4
 37F6 000408  B2FC   SP    00000000   000    00000 00.004 092 7
 37F7 001FFC  55AA   SW    00000000   000    00000 00.004 092 9
 37F8 00040A  2000   SP    00000000   000    00000 00.004 093 2
 37F9 00040C  66F6   SP    00000000   000    00000 00.004 093 4
 37FA 00040E  A0F8   SP    00000000   000    00000 00.004 093 7
 37FB 000404  32FC   SP    00000000   000    00000 00.004 093 9
 37FC 000406  55AA   SP    00000000   000    00000 00.004 094 4
 37FD 000408  B2FC   SP    00000000   000    00000 00.004 094 7
 37FE 001FFE  55AA   SW    00000000   000    00000 00.004 094 9
 37FF 00040A  2000   SP    00000000   000    00000 00.004 095 2
 3800 00040C  66F6   SP    00000000   000    00000 00.004 095 4
 3801 00040E  A0F8   SP    00000000   000    00000 00.004 095 7
 3802 000410  040E   SP    00000000   000    00000 00.004 095 9
>
```

4.7.1 List Trace Results - List

List [frame [adr1 adr2] [status] [EXternal trace-bits]]

Examples:

1. After the trace operation in the previous example, list all frames in the first page of the trace buffer.

```
>List 0<CR>
FRAME ADDRESS DATA STATUS TRACE BITS STATE   EVENT TIMER
 0000 000400  3204   SP     00000000   111    11111 00.000 000 0
 0001 000402  1204   SP     00000010   000    00010 00.000 000 0
 0002 000404  3604   SP     00000100   000    00100 00.000 000 2
 0003 000406  570E   SP     00000110   000    00110 00.000 000 4
 0004 000408  3A04   SP     00001000   000    01000 00.000 000 7
 0005 001000  551A   SW     00000000   000    00000 00.000 000 9
 0006 00040A  2A04   SP     00001010   000    01010 00.000 001 2
 0007 00040C  6E06   SP     00001100   000    01100 00.000 001 4
 0008 00040E  AE04   SP     00001110   000    01110 00.000 001 7
 0009 000404  3604   SP     00000100   000    00100 00.000 001 9
 000A 000406  570E   SP     00000110   000    00110 00.000 002 4
 000B 000408  B2FC   SP     00000000   000    00000 00.000 002 7
 000C 001002  55AA   SW     00000000   000    00000 00.000 002 9
 000D 00040A  2000   SP     00000000   000    00000 00.000 003 2
 000E 00040C  66F6   SP     00000000   000    00000 00.000 003 4
 000F 00040E  A0F8   SP     00000000   000    00000 00.000 003 7
[ More ]<ESC>
>
```

2. List the trace buffer starting at frame 0, and limit the listing to supervisor data memory write (SW) cycles in the memory range of 1200H to 120FH.

```
>List 0 1200 120F SW<CR>
FRAME ADDRESS DATA STATUS TRACE BITS STATE   EVENT TIMER
 0705 001200  55AA   SW     00000000   000    00000 00.000 512 9
 070C 001202  55AA   SW     00000000   000    00000 00.000 514 9
 0713 001204  55AA   SW     00000000   000    00000 00.000 516 9
 071A 001206  55AA   SW     00000000   000    00000 00.000 518 9
 0721 001208  55AA   SW     00000000   000    00000 00.000 520 9
 0728 00120A  55AA   SW     00000000   000    00000 00.000 522 9
 072F 00120C  55AA   SW     00000000   000    00000 00.000 524 9
 0736 00120E  55AA   SW     00000000   000    00000 00.000 526 9
>
```

3. List all frames in the trace buffer after the execution operation in Go command (Section 4.3, Example 7 in this Addendum).

```
>Disassemble 400 418
LOC         OBJ              SOURCE CODE
000400      303C 0002        MOVE.W    #0002,D0
000404      907C 0001        SUB.W     #0001,D0
000408      0C40 0000        CMPI.W    #0000,D0
00040C      6706             BEQ.B     000414
00040E      4EB8 0510        JSR       000510
000412      60F0             BRA.B     000404
000414      4EB8 0520        JSR       000520
000418      4E71             NOP
 Disassembly completed
>Fill 500 5FF 4E 75
>Event CLear
>Event 1 404
>Event 2 510
>Event 3 520
>Event 4 418
>Trigger TImer On
>Trigger Level A EV1 TImer OFf
>Trigger Level B EV2 TImer On
>Trigger Level C EV3 TImer On
>Trigger Level D EV4 TImer On
>Trigger A If B Reset ELse ( C THen D )
>Go 400
 EV4 (Bus) encountered
 Emulation processor stopped
 Last frame = 0020,timer = 00.000 004 0
>List 0<CR>
FRAME ADDRESS DATA STATUS TRACE BITS STATE   EVENT TIMER
 0000 000400  303C   SP    00000000   011    10000 00.000 000 0
 0001 000402  0002   SP    00000000   000    00000 00.000 000 0
 0002 000404  907C   SP    00000000   000    00000 00.000 000 2
 0003 000406  0001   SP    00000000   000    00001 00.000 000 4
 0004 000408  0C40   SP    00000000   001    00000 00.000 000 4
 0005 00040A  0000   SP    00000000   001    00000 00.000 000 4
 0006 00040C  6706   SP    00000000   001    00000 00.000 000 4
 0007 00040E  4EB8   SP    00000000   001    00000 00.000 000 4
 0008 000410  0510   SP    00000000   001    00000 00.000 000 4
 0009 000510  4E75   SP    00000000   001    00000 00.000 000 4
 000A 01FFEC  0000   SW    00000000   001    00010 00.000 000 4
 000B 01FFEE  0412   SW    00000000   000    00000 00.000 000 7
 000C 000512  4E75   SP    00000000   000    00000 00.000 000 9
 000D 01FFEC  0000   SR    00000000   000    00000 00.000 001 2
 000E 01FFEE  0412   SR    00000000   000    00000 00.000 001 4
 000F 000412  60F0   SP    00000000   000    00000 00.000 001 7
[ More ]
```

```
 0010 000414  4EB8   SP    00000000   000    00000 00.000 001 9
 0011 000404  907C   SP    00000000   000    00000 00.000 002 2
 0012 000406  0001   SP    00000000   000    00001 00.000 002 5
 0013 000408  0C40   SP    00000000   001    00000 00.000 002 5
 0014 00040A  0000   SP    00000000   001    00000 00.000 002 5
 0015 00040C  6706   SP    00000000   001    00000 00.000 002 5
 0016 00040E  4EB8   SP    00000000   001    00000 00.000 002 5
 0017 000414  4EB8   SP    00000000   001    00000 00.000 002 5
 0018 000416  0520   SP    00000000   001    00000 00.000 002 5
 0019 000520  4E75   SP    00000000   001    00000 00.000 002 5
 001A 01FFEC  0000   SW    00000000   001    00100 00.000 002 5
 001B 01FFEE  0418   SW    00000000   010    00000 00.000 002 8
 001C 000522  4E75   SP    00000000   010    00000 00.000 003 0
 001D 01FFEC  0000   SR    00000000   010    00000 00.000 003 3
 001E 01FFEE  0418   SR    00000000   010    00000 00.000 003 5
 001F 000418  4E71   SP    00000000   010    00000 00.000 003 8
[ More ]
 0020 00041A  4E71   SP    00000000   010    01000 00.000 004 0
>
```

The columns "STATE" and "EVENT" denotes the trigger state and event information respectively. Each bit is defined as follows:

STATE Bits 2 - 0 is the trigger state 0 - 7

EVENT Bit 4 is active when Event 5 is matched ("Event encounter-flag" of EV5)
Bit 3 is active when Event 4 is matched ("Event encounter-flag" of EV4).
Bit 2 is active when Event 3 is matched ("Event encounter-flag" of EV3).
Bit 1 is active when Event 2 is matched ("Event encounter-flag" of EV2).
Bit 0 is active when Event 1 is matched ("Event encounter-flag" of EV1).

Note that bits 4-0 remain in effect until the state changes and the state value of frame 0 is in random. In the above example, the sequence occurs- as follows:

Event 1 (address 404H) is matched, it changes to State 1 and timer turned OFF.
Event 2 (address 510H) is matched, it changes to State 0 and timer turned ON.
Event 1 (address 404H) is matched, it changes to State 1 and timer turned OFF.
Event 3 (address 520H) is matched, it changes to State 2 and timer turned ON.
Event 4 (address 418H) is matched, emulation will then finally stop.

4.7.2 LIST TRACE RESULTS WITH SOURCE/INSTRUCTION CODE - List

List {Source|Instruction} [frame]

Examples:

1. Starting at frame 00H, list the address, traced data, CPU status, trace bits, state and event, displaying mnemonic code together with machine activity.

```
>List Source 0<CR>
FRAME ADDRESS DATA STATUS TRACE BITS STATE   EVENT TIMER
 0000 000400  303C   SP    00000000   011    10000 00.000 000 0
          MOVE.W   #0002,D0
 0001 000402  0002   SP    00000000   000    00000 00.000 000 0
 0002 000404  907C   SP    00000000   000    00000 00.000 000 2
          SUB.W    #0001,D0
 0003 000406  0001   SP    00000000   000    00001 00.000 000 4
 0004 000408  0C40   SP    00000000   001    00000 00.000 000 4
          CMPI.W   #0000,D0
 0005 00040A  0000   SP    00000000   001    00000 00.000 000 4
 0006 00040C  6706   SP    00000000   001    00000 00.000 000 4
          BEQ.B    000414
 0007 00040E  4EB8   SP    00000000   001    00000 00.000 000 4
          JSR      000510
 0008 000410  0510   SP    00000000   001    00000 00.000 000 4
 0009 000510  4E75   SP    00000000   001    00000 00.000 000 4
          RTS
[ More ]<ESC>
>
```

2. Starting at frame 00H, list the address, traced data and assembly source code, displaying mnemonic code together with machine activity.

```
>List Instruction 0<CR>
FRAME   ADDRESS     DATA      SOURCE-CODE
 0000   00000B00     2E7C     MOVEA.L   #00006000,A7
 0002   00000B06     207C     MOVEA.L   0002FFF0,AO
 0006   00000B0C     4E60     MOVE.L    00,USP
 0007   00000B0E     4EF9     JMP       00020412
 000A   00000410     4E71     NOP
 000B   00000412     207C     MOVEA.L   #00020700,AO
 000E   00000418     227C     MOVEA.L   #00020600,A1
 0010   0000041E     2018     MOVE.L    (AO)+,DO
 0012   00000420     2280     MOVE.L    DO,(A1)
 0013   00000422     2219     MOVE.L    (A1)+D1
 0016   00000424     B280     CMP.L     D0,D1
 0017   00000426     6100     BSR,WL    00000C00
 001E   00000C00     670A     BEQ,B     00000C0C
 0024   00000C0C     4E75     RTS
 002A   00000428     07D8     BSET.B    D3,(AO)+
[ More ]<ESC>
>
```

NOTE

1. ***When an instruction is fulfilled while executing a series of program control instructions, only the first instruction (NOT necessarily the fulfilled instruction) will display.***

2. ***When displaying mnemonic code with the List Source command, the following conditions may cause incorrect information to be listed for the initial frame:***

   a) ***The machine cycle recorded in the initial frame is not a fetch cycle for the beginning of an instruction.***

   b) ***The specified starting frame contains prefetch instructions that were not executed due to a program branch.***

4.7.3 LIST TOTAL FRAMES AND TIME - List Number

List Number

The MICE-IIIS 68000 displays **"Last frame = "** followed by the number of the last frame recorded in the trace buffer, and **"Timer = "** followed by the total elapsed time since timer started. (Note that the frame count begins at zero, so the total number of frames is one more than the value displayed. If

"Last frame = 7FFFH" (for 32K trace buffer)

is displayed, the trace buffer is full.)

Example:

1. Display the number of the last frame in the trace buffer and the total elapsed time since timer start.

```
>List Number<CR>
 Last frame = 0018, Timer = 00.000 006 7
>
```

4.8 CYCLE QUALIFY - Qualify

Qualify {1|2} [{Range adr1 adr2|adx} [status] [EXternal trace-bits]]
Qualify [1|2] [CLear|DIsable|ENable]

Quality displays, sets, enables, disables or clears the cycle qualifier, which specifies which machine cycles are to be recorded in the trace buffer. "Qualify" alone, without any parameters, displays the current Cycle Qualify setting.

1|2 are the two available Qualifiers under LAM-IIS which perform "OR" function.

Range is the required prefix when specifying an address range.

adr1 is a hexadecimal address indicating the starting address.

adr2 is a hexadecimal address indicating the end address.

adx is a hexadecimal address setting with option to include wildcard nibbles/bits, e.g. 1234, 1XX4, 1(X1X0)X4, 1(X1X0X0XX)4.

status is 1 to 7 processor status codes (SP SR SW UP UR UW AK AW) indicating the type of processor activity to be recorded.

EXternal is the required prefix when specifying trace-bits.

trace-bits is a 1-byte hexadecimal setting for trace bits.

CLear is the parameter to clear the Cycle Qualify setting.

The power-on default setting of the MICE-IIIS 68000 is to record all machine cycles during a trace operation for either Qualify 1 or Qualify 2. Cycle Qualify is used to limit recording to cycles in which certain areas of program memory are accessed and/or certain types of processor activity take place. This makes it possible to record more useful information in the trace buffer with fewer trace commands.

Examples:

1. Set Cycle Qualify 1 for all SP and SR cycles at address 82000, 82010...82070; and cycle Qualify 2 for SW cycles between 92000...92700 and trace bits with wildcard; then display the current setting.

```
>Qualify 1 820(0XXX)0 SP SR<CR>
>Qualify 2 Range 92000 92700 SW EXternal 4(XX10)<CR>
>Qualify<CR>
Cycle qualifier 1 : 0820(0XXX)0 SP,SR External XX  Enable
Cycle qualifier 2 : Range 092000 092700 SW External 4(XX10) Enable
>
```

2. Disable cycle Qualify 2, set cycle Qualify 1 for UW cycle at all address, then display the setting.

```
>Qualify ENable<CR>
>Qualify 2 Disable<CR>
>Qualify 1 X UW<CR>
>Qualify<CR>
 Cycle qualifier 1 : XXXXXX UW External XX Enable
 Cycle qualifier 2 : Range 092000 092700 SW EXternal 4(XX10) Disable
>
```

3. Clear the current Cycle Qualify setting.

```
>Qualify CLear<CR>
>Qualify<CR>
 Cycle qualifier 1 : All address and status Enable
 Cycle qualifier 2 : All address and status Disable
>
```

4.9 TIMEBASE SELECTION - TImebase

TImebase [0.1|1|10|100|1000]

TImebase is the command for timebase selection.

0.1...1000 specifies the Timebase selection as indicated in Table 4-1 below.

A timer is provided for trace mode. Execution time is displayed in the Go command (Section 4.3 of this Addendum), with the unit of measurement determined by the Timebase selection. The default is for a timebase of 0.1 μs, or the most recent user setting. The timer is activated when emulation starts, unless the timer OFf option is specified in the Trigger command.

Timebase Number	1	2	3	4	5
Unit of Measurement	0.1μs	1μs	10μs	100μs	1000μs
Maximum Timer Length	7min 9sec	1hr 11min	11hrs 55min	4days 23hrs	49days 17hrs

Table 4-1 LAM-IIS Timebase Selections

Example: Set the timebase to 0.1 μs when emulation starts; then display the current Timebase setting to verify input.

```
>TImebase 0.1<CR>
>TImebase<CR>
 Timebase = 0.1 us
>
```

The format for timing data displayed at the last (`"Timer ="`) line of Go command execution (see Example 1 of Section 4.3 of this Addendum) is-

"dd hh:mm:ss.mmm uuu n"

where:
- **dd** = days
- **hh** = hours
- **mm** = minutes
- **ss** = seconds
- **mmm** = milliseconds
- **uuu** = microseconds
- **n** = 100 nano seconds

4.10 DISPLAY CURRENT TRACE PARAMETERS - TRAce

TRAce

TRAce is the command to display the following settings:

Breakpoints EV1 to EV7
Trigger condition (Trigger setting and Trigger Level settings)
Cycle qualifier
Timebase
Synchronization setting

Example: Display the current MICE-IIIS 68000 trace control settings.

```
>TRAce<CR>
 EV1 (Bus) 000400 XXXX SR,SW External XX Count 0020
 EV2 (Bus) 000480 4E71  External XX Count 0001
 EV3 (Bus) Range 000500 0005FF XXXX  External 34 Count 0001
 EV4 (Bus) XX34XX X6X7 UP External 9X Count AABB
 EV5 (External) High
 EV6 (Execution) 000500 SP,UP
 EV7 (Execution) 000520 SP,UP
 EV8 (Execution) clear
 Trigger A Delay 0000
 Trigger Timer On Trace On
 Level A  EV1 Or EV2 Or EV3 Or EV4 Trace On Timer On
 Level B  Clear
 Level C  Clear
 Level D  Clear
 Level E  Clear
 Level F  Clear
 Level G  Clear
 Level H  Clear
 Cycle qualifier 1 : XXXXXX UW XX  Enable
 Cycle qualifier 2 : 092000 092700  SW 4(XX10)  Disable
 Timebase = 1 ms
 Sync start in Off
 Start slave out Off
>
```

4.11 DISPLAY/SET/CLEAR TRIGGER - Trigger

Set Trigger Level	**Trigger Level level# [Not] ev# [{OR|ANd} [Not] ev# [{OR|ANd} [Not] ev# [{OR|ANd} [Not] ev# [{OR|ANd} [Not] ev#]]]] [Trace {On|OFf}] [TImer {On|OFf}]**
Display/Clear Trigger Level	**Trigger Level level# [CLear]**
Set Trigger	**Trigger [RUn] {level# [<THen level#>] [Reset | If state# <ELse state#>] | If state# <ELse state#>} [FOrward|BAckward |CEnter|DElay count]**
Set Initial Timer & Trace	**Trigger [TImer {On|OFf}] [Trace {On|OFf}]**
Display/Clear Trigger	**Trigger [CLear]**

Trigger specifies Trigger display/set/clear. "Trigger" alone, without any para-meters, displays the current Trigger condition.

Level is the required prefix when specifying a level identifier.

level# is one of level identifiers- A, B, C, D, E, F, G or H.

Ev# is one of the Events 1, 2, 3, 4 or 5.

NOTE

Do not specify the same level and Event twice in one command.

Trace {On|OFf} Enable/disables trace program execution in the trace buffer.

Timer {ON|OFf} Enable/disables time measurement of user's program execution.

Run is an optional parameter which allows the emulation processor to free run after the trace ends.

state# level# [RESet] | (level# {<Then level#> [RESet | If state# <ELse state#>] | If state# <ELse state#> })

< **>** Recursive (Minimum 1 time).

(**)** Reserved Symbols.

FOrward is the delay count = 7FFFH for 32K trace buffer.

BAckward is the delay count = 0.

CEnter is the delay count = 3FFFH for 32K trace buffer.

count is a hexadecimal value (from 0H to 0FFFFH).

CLear is the command to clear the current Trigger setting.

The Trigger setting is composed of up to 8 Trigger Levels. Each level may be a single event or a logical construct of up to five breakpoints with **OR**, **ANd**, and **Not** (i.e., the Event never occured). Only Events 1 to 5 can be used in the Trigger Level definition; execution breakpoints are automatically joined to the specified construct by a logical **OR**. EV1~EV5 can only be used once in a Trigger Level setting. "Trace On|OFf" or "Timer On|OFf" may also be specified at any Trigger Level or at the beginning of free-running emulation. The default at power-on and after a hardware reset is the construct "Level A EV1 OR EV2 OR EV3 OR EV4 Trace On Timer On". Refer to Figure 4-2a (next page) for the "Trigger Level" setting flow chart.

LAM-IIS supports a very powerful Trigger condition by setting "Trigger Level" and "Trigger". **THen** connectives and **If...ELse...** could be flexibly used in the Trigger setting. In order to be familiar with the complex command structure, it is recommended that user establishes a tree structure first as shown in Examples II-4 and II-5 below before writing a Trigger setting syntax. The "Trigger" setting syntax flow chart is shown in Figure 4-2b.

As illustrated in the Trigger command flow chart (Figure 4-2a/b), the Trigger setting only specifies the break condition. It is not effective until the Go command is input to start the emulation. Note also that the Trigger setting has no effect during the Cycle Step and Instruction Step commands.

When emulation begins, the MICE-IIIS 68000 immediately starts recording target system and emulation processor status (in accordance with the Cycle Qualify setting) in real-time. Data are recorded until the Trigger condition, including any cycle-count delay, is matched.

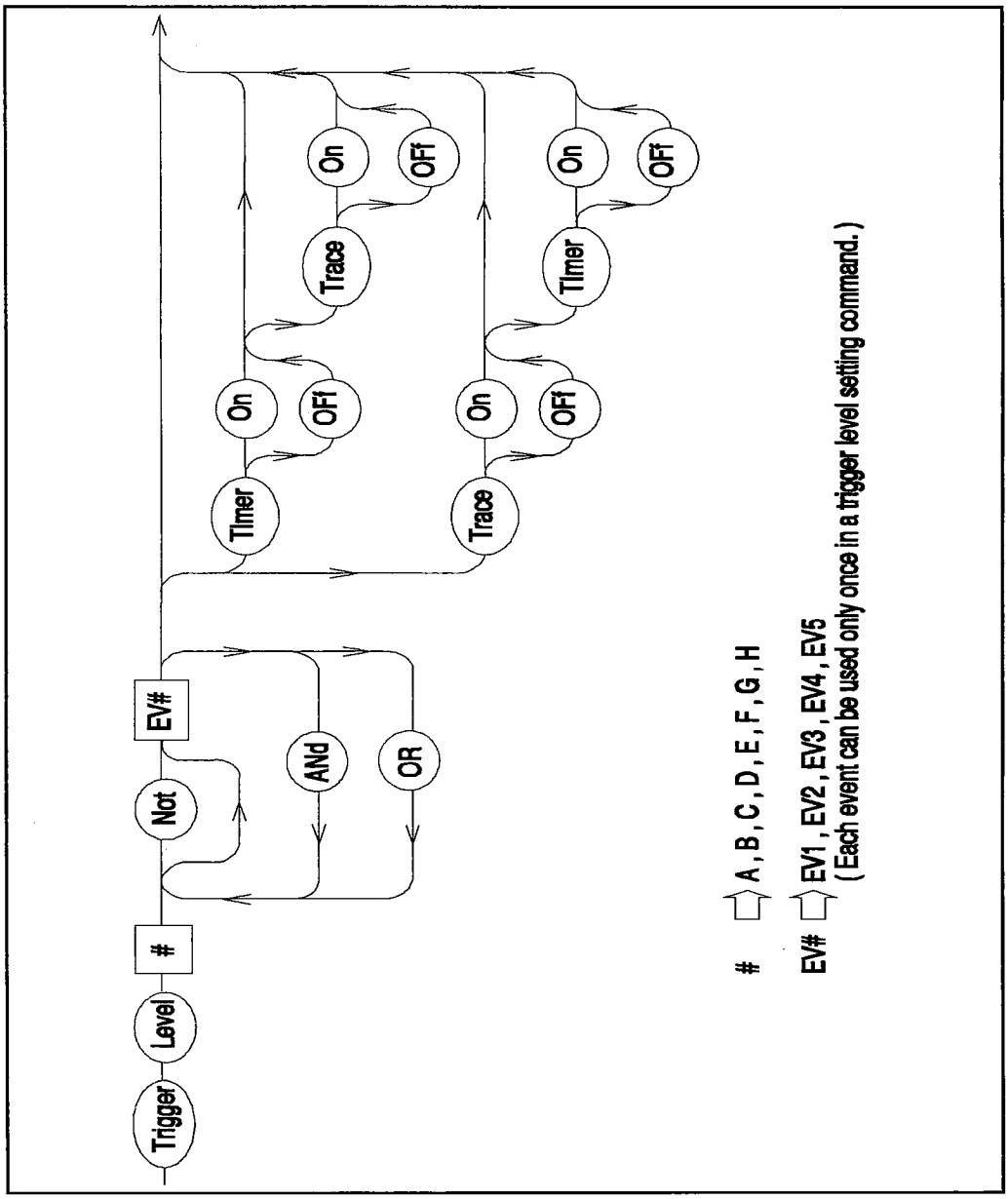


Figure 4-2a LAM-IIS Trigger Level Setting Syntax Rules and Flow Chart

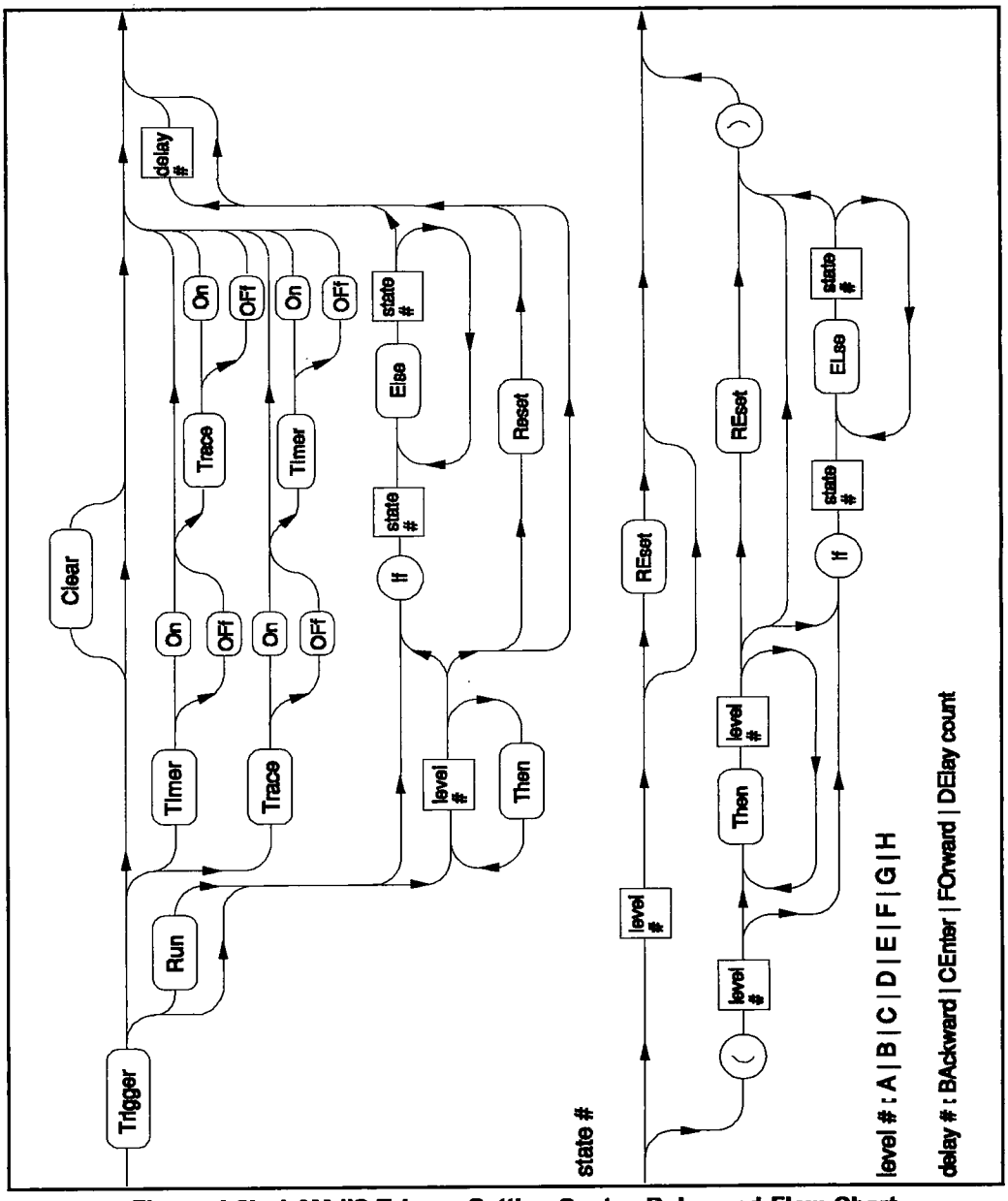


Figure 4-2b LAM-IIS Trigger Setting Syntax Rules and Flow Chart

Up to 32768 machine cycles can be recorded. If more than 32768 cycles had elapsed before tracing ends, only the last 32768 cycles will be recorded.

When the trace ends, the event(s) matched are displayed and (unless the Run option was used) the MICE stops the emulation processor one cycle (or two cycles, depending on the CPU speed and its wait cycle) beyond the location where the Trigger condition was matched. The recorded information can be examined with the List Trace Buffer command. The "STATE" column in the List command execution denotes the change of state. Watch "Event encounter flag" of EV1~EV5 in the "EVENT" column resets to 00000 whenever the "STATE" changes.

Note that whenever the processor is running, external hardware sync signals are generated, at the SYNC 1 OUTPUT port the first time Event 1 is matched and at the SYNC 2 OUTPUT port each time Event 2 is matched.

Examples:

I. Trigger Display

1) Display the current Trigger and Triger Level settings.

```
>Trigger<CR>
 Trigger A Delay 0000
 Trigger Timer On Trace On
 Level A  EV1 Or EV2 Or EV3 Or EV4 Trace On Timer On
 Level B  Clear
 Level C  Clear
 Level D  Clear
 Level E  Clear
 Level F  Clear
 Level G  Clear
 Level H  Clear
>
```

2) Display the current Trigger Level setting.

```
>Trigger Level<CR>
 Level A  EV1 Or EV2 Or EV3 Or EV4 Trace On Timer On
 Level B  Clear
 Level C  Clear
 Level D  Clear
 Level E  Clear
 Level F  Clear
 Level G  Clear
 Level H  Clear
>
```

II. Setting Trigger Condition

1) Specify some combinations of Trigger constructs in Trigger Levels.

```
>Trigger Level A EV1 OR EV2 OR EV3 OR EV4<CR>
>Trigger Level B EV1 ANd EV3 ANd EV5<CR>
>Trigger Level C Not EV2<CR>
>Trigger Level D EV1 OR EV4 OR EV5<CR>
>Trigger Level E EV2 ANd EV3 OR Not EV4<CR>
>Trigger Level F EV1 OR EV2 OR EV3 ANd EV4 TImer OFf<CR>
>Trigger Level G EV1 ANd EV5 TRace OFf<CR>
>Trigger Level H EV3 Trace OFf TImer OFf<CR>
>Trigger<CR>
 Trigger If (A If B ELse C ELse D Reset) ELse (E If F ELse G ELse H Reset)
 Delay 0000
 Trigger Timer On Trace On
 Level A  EV1 Or EV2 Or EV3 Or EV4 Trace On Timer On
 Level B  EV1 And EV3 And EV5 Trace On Timer On
 Level C  Not EV2 Trace On Timer On
 Level D  EV1 Or EV4 Or EV5 Trace On Timer On
 Level E  EV2 And EV3 Or Not EV4 Trace On Timer On
 Level F  EV1 Or EV2 Or EV3 And EV4 Trace On Timer Off
 Level G  EV1 And EV5 Trace Off Timer On
 Level H  EV3 Trace Off Timer Off
>
```

Note that the precedence of the logical factors is **NOT > AND > OR** and each level may specify Trace On|OFf and Timer On|OFf while the trigger level is matched.

2) Using the Trigger Level setting in the preceding example, set a single level trigger with 10FFH delay and then display result.

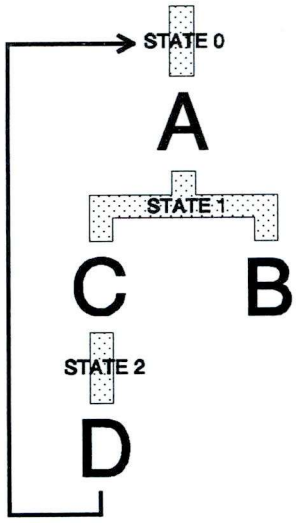
```
>Trigger B DElay 10FF<CR>
>Trigger<CR>
 Trigger B DElay 10FF
 Trigger Timer On Trace On
 Level A  EV1 Or EV2 Or EV3 Or EV4 Trace On Timer On
 Level B  EV1 And EV3 And EV5 Trace On Timer On
 Level C  Not EV2 Trace On Timer On
 Level D  EV1 Or EV4 Or EV5 Trace On Timer On
 Level E  EV2 And EV3 Or Not EV4 Trace On Timer On
 Level F  EV1 Or EV2 Or EV3 And EV4 Trace On Timer Off
 Level G  EV1 And EV5 Trace Off Timer On
 Level H  EV3 Trace Off Timer Off
>
```

3) Using the Trigger Level setting in Example 1, set a "**Then**" trigger connectives and continue emulation after the trace stops.

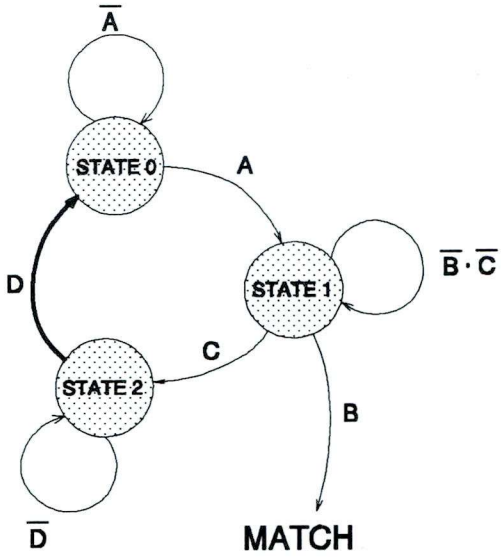
```
>Trigger RUn A THen D THen F<CR>
>Trigger<CR>
 Trigger RUn A THen D THen F Delay 0000
 Trigger Timer On Trace On
 Level A  EV1 Or EV2 Or EV3 Or EV4 Trace On Timer On
 Level B  EV1 And EV3 And EV5 Trace On Timer On
 Level C  Not EV2 Trace On Timer On
 Level D  EV1 Or EV4 Or EV5 Trace On Timer On
 Level E  EV2 And EV3 Or Not EV4 Trace On Timer On
 Level F  EV1 Or EV2 Or EV3 And EV4 Trace On Timer Off
 Level G  EV1 And EV5 Trace Off Timer On
 Level H  EV3 Trace Off Timer Off
>
```

4) Construct a tree structure of the following Trigger condition:

```
Trigger A IF ( C THen D Reset ) ELse B
Trigger Level A EV1 OR EV4
Trigger Level B EV3
Trigger Level C EV2
Trigger Level D EV5
Event 1 8008
Event 2 8010
Event 5 High
Event 4 8018
Event 3 8020
```

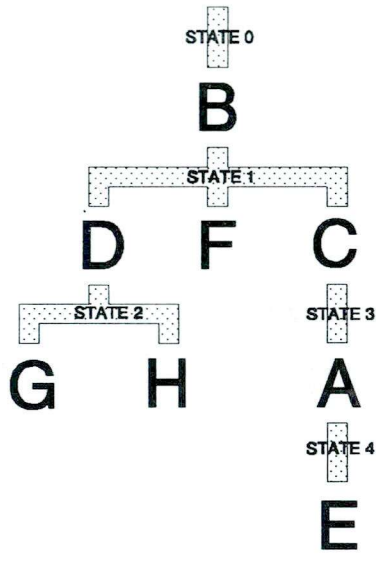


Tree

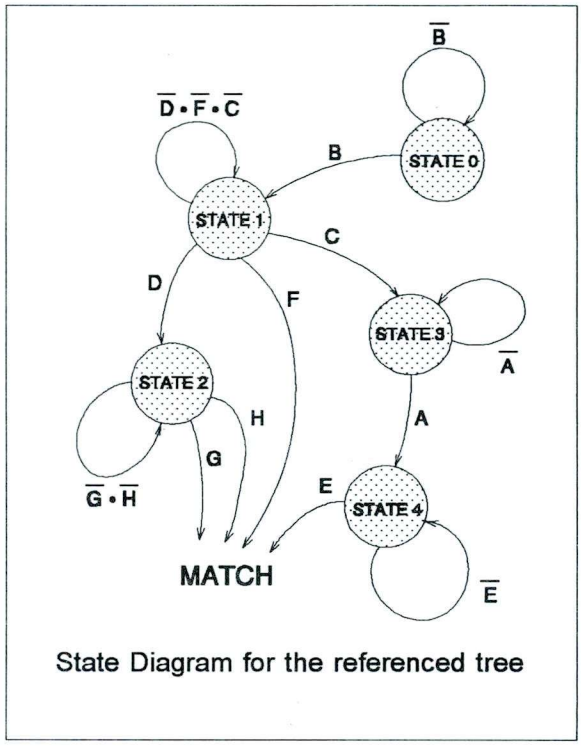


State Diagram for the constructed tree

5) Specify a Trigger setting in reference to the following tree structure.



Tree



State Diagram for the referenced tree

```
>Trigger B If ( D If G ELse H ) ELse F ELse ( C THen A THen E )<CR>
>Trigger<CR>
 Trigger B If ( D If G ELse H ) ELse F ELse ( C THen A THen E) Delay 0000
 Trigger Timer On Trace On
 Level A  EV1 Or EV2 Or EV3 Or EV4 Trace On Timer On
 Level B  EV1 And EV3 And EV5 Trace On Timer On
 Level C  Not EV2 Trace On Timer On
 Level D  EV1 Or EV4 Or EV5 Trace On Timer On
 Level E  EV2 And EV3 Or Not EV4 Trace On Timer On
 Level F  EV1 Or EV2 Or EV3 And EV4 Trace On Timer Off
 Level G  EV1 And EV5 Trace Off Timer On
 Level H  EV3 Trace Off Timer Off
>
```

6) Set Timer On and Trace OFf when emulation begins to run in Trigger condition.

```
>Trigger TImer On Trace OFf<CR>
>Trigger<CR>
 Trigger A Delay 0000
 Trigger Timer On Trace Off
 Level A  EV1 Or EV2 Or EV3 Or EV4 Trace On Timer On
 Level B  Clear
 Level C  Clear
 Level D  Clear
 Level E  Clear
 Level F  Clear
 Level G  Clear
 Level H  Clear
>
```

III. Trigger Clear

1) Clear the current Trigger Level C setting.

```
>TRigger Level C CLear <CR>
>TRigger<CR>
 Trigger B If ( D If G ELse H ) ELse F ELse ( C THen A THen E ) Delay 0000
 Trigger Timer On Trace On
 Level A  EV1 Or EV2 Or EV3 Or EV4 Trace On Timer On
 Level B  EV1 And EV3 And EV5 Trace On Timer On
 Level C  Clear
 Level D  EV1 Or EV4 Or EV5 Trace On Timer On
 Level E  EV2 And EV3 Or Not EV4 Trace On Timer On
 Level F  EV1 Or EV2 Or EV3 And EV4 Trace On Timer Off
 Level G  EV1 And EV5 Trace Off Timer On
 Level H  EV3 Trace Off Timer Off
>
```

2) Clear the current Trigger setting.

```
>Trigger CLear<CR>
>Trigger<CR>
 Trigger Clear
 Trigger Timer On Trace On
 Level A  EV1 Or EV2 Or EV3 Or EV4 Trace On Timer On
 Level B  EV1 And EV3 And EV5 Trace On Timer On
 Level C  Clear
 Level D  EV1 Or EV4 Or EV5 Trace On Timer On
 Level E  EV2 And EV3 Or Not EV4 Trace On Timer On
 Level F  EV1 Or EV2 Or EV3 And EV4 Trace On Timer Off
 Level G  EV1 And EV5 Trace Off Timer On
 Level H  EV3 Trace Off Timer Off
>
```

Addendum 5
APPLICATION NOTES

5.1 LAM-IIS APPLICATION NOTES

1) Under LAM version, EV1 & EV2 were bus breakpoints; EV3 was an external hardware breakpoint; EV4, EV5 & EV6 were execution breakpoints. Under LAM-IIS version, EV1 to EV4 are bus breakpoints; EV5 is an external hardware breakpoint; EV6, EV7 & EV8 are execution breakpoints.

2) Whenever EV1 and EV2 are matched, a low pulse will occur at SYNC 1 OUTPUT and SYNC 2 OUTPUT ports respectively. For more information on timing, see Section 5.2 below.

3) If the trace buffer is not full (i.e., last frame not equal to 7FFFH for 32K trace buffer) when it stops recording, the data under STATE and EVENT fields of Frame 0 are inaccurate and should be ignored.

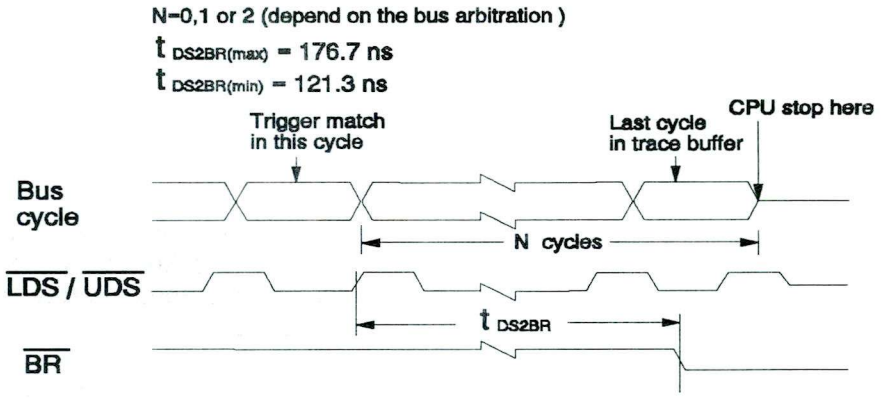
4) When upgrading MICE-16 68000 (with LAM) to MICE-IIIS 68000 (with LAM-IIS version), the CPM board on which the firmware (EPROM U11, U12, U14, U15) are to be replaced, must be of REV-G or later.

5.2 LAM-IIS TIMING NOTES

The following timing diagrams only applies to MICE-IIIS 68000 equipped with LAM-IIS:

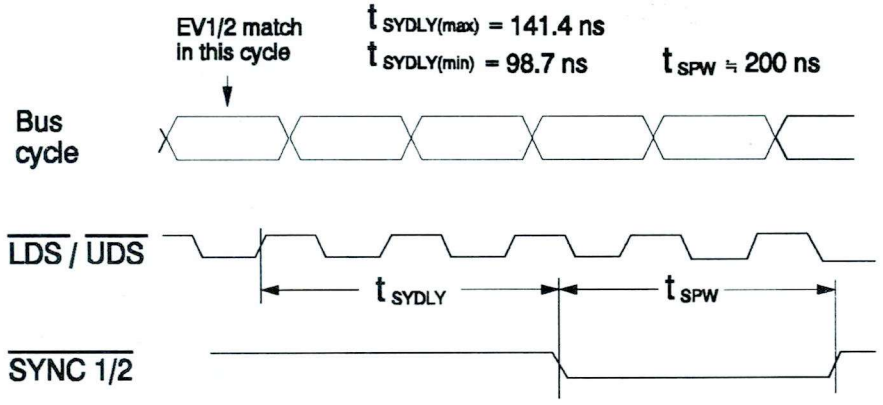
1) **Timing of the Emulation Processor Breaking at the Trigger Command Setting with "DELay 0".**

   Literally, whenever the emulation stops, MICE asserts $\overline{BR}$ signal to stop running the program. From this point, Emulation Processor ceases to be the bus master. The time between the point $\overline{LDS}/\overline{UDS}$ is negated during a "Trigger matched bus cycle", to the point when BR is asserted, is equal to t_{DS2BR}.



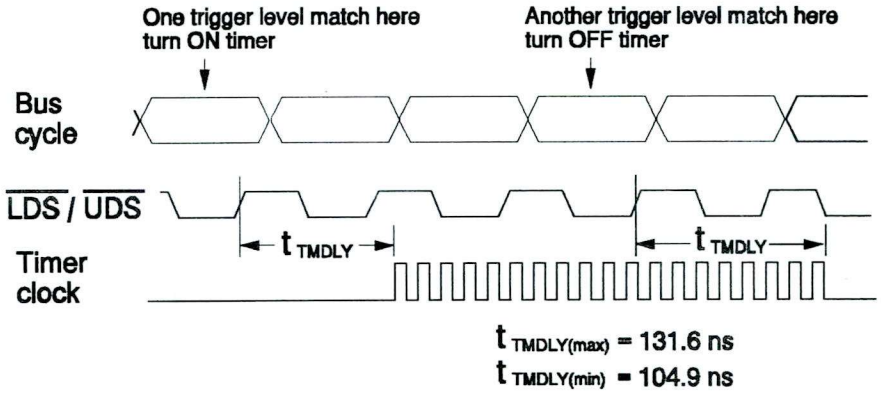
2) SYNC 1/2 OUTPUT Signals Timing

Although the "Event encounter-flag" of EV1 to EV5 is cleared when the STATE changes during free running, the $\overline{SYNC1}$ or $\overline{SYNC2}$ output signal will be asserted for about 200ns, whenever the EV1 or EV2 is matched and sets the "Event encounter- flag". The time between the point when $\overline{LDS}/\overline{UDS}$ is negated during the bus cycle where EV1 or EV2 is matched, to the point when $\overline{SYNC1}$ or $\overline{SYNC2}$ is asserted, is equal to t_{SYDLY}.



3) Timer On/OFf Timing

The timer can be turned On or turned Off at any bus cycle by specifying timer On/Off locations at the end of each "Trigger Level". The timer is then turned On/Off accor-dingly whenever the "Trigger Level" is matched. But in reality, there is a time delay between the time "Trigger Level" is actually matched and the time the timer clock is actually On (or Off). The time between the point when $\overline{LDS}/\overline{UDS}$ is negated during the bus cycle in which "Trigger Level" is matched, to the point when timer clock is put On (or Off), is equal to t_{TMDLY}.



4) Trace On/OFf Timing

The trace buffer can also be enabled/disabled at any bus cycle by specifying trace On/Off locations at the end of each "Trigger Level". The trace buffer is then turned On/Off whenever the "Trigger Level" is matched. But in reality, there is a time delay between the actual time "Trigger Level" is matched and the actual time the trace buffer is enabled or disabled. Hence, when one "Trigger Level" is matched in the current cycle and the trace buffer is turned Off, tracing will actually stop in the next and subsequent cycles (not in the current cycle). Tracing is resumed when another "Trigger Level" is matched and trace buffer is turned On.