

ICD-178
FOR
8048

USER'S MANUAL

**ICD-178
FOR
8048**

USER'S MANUAL

Preliminary

Copyright© 1984, U.S. ZAX CORPORATION. All Rights reserved under International and Pan-American Copyright Conventions. No part of this publication may be reproduced in any manner whatsoever without written permission from ZAX CORPORATION.

Prepared by:

U.S. ZAX CORPORATION
2572 White Road, Irvine, California 92714

USER'S MANUAL: Part No. 20-108-00, Rev. A
Printed: October 1984

PRINTED IN U.S.A.

DISCLAIMER

Although every effort has been made to make this USER'S MANUAL technically accurate, ZAX Corporation assumes no responsibility for any errors, omissions, inconsistencies or misprints within this document. ZAX Corporation also assumes no responsibility for any incidental or consequential damages arising out of the use or inability to use this equipment. Further, ZAX Corporation reserves the right to revise this document at any time, by means of additions or deletions or any other reconstruction, without obligation to notify any person or organization of such revisions or changes, unless otherwise stated.

COPYRIGHT

This manual and the software described in it are copyrighted with all rights reserved. No part of this manual or the programs may be copied, in whole or part, without written permission from ZAX Corporation, except in the normal use of the software or to make a backup copy for use with the same system. This exception does not allow copies to be made for other persons.

PRECAUTIONS

Do not connect this equipment to any external device or attach the target processor probe before first reading the information in Section 1.7. (See "How To Operate Your ICD-178.")

Failure to correctly hookup this equipment may result in severe damage to the ICD-178 and the target system.

USERS MANUAL

INTRODUCTION

This USER'S MANUAL contains information on the ZAX ICD-178 for 8048, IN-CIRCUIT EMULATOR. This manual is designed to allow you to install, configure and run the ICD-178 in a very short time. This manual will instruct you in; identifying components and controls of the emulator, connecting the various accessories (probes, power cable, cooling fan), connecting the system to a host computer and performing emulation by using the debugger commands.

HOW TO USE THIS MANUAL

This USER'S MANUAL is divided into three main sections. SECTION I is entitled ICD DESCRIPTION and OPERATION. This section begins with a description of the ICD-178 and includes both general and emulation specifications of the unit. This section also describes the controls and components of the ICD-178 and shows how to connect the system to a host computer and run the emulator. If you are using the ZAX ICD-178 emulator for the first time, read this section first to become familiar with the various functions and to learn the correct procedure for installation and operation.

SECTION II contains information on the debugger commands and is entitled COMMAND REFERENCE GUIDE. It is the section that you will be using the most, and therefore is presented in the greatest detail. This section will show you the command names, functions, formats and input parameters, a directory of debugger operations, and a detailed analysis of each command with accompanying examples. In addition to this section, a separate COMMAND REFERENCE GUIDE may be found at the front of this manual. This compact, fold-out guide can be used when you become familiar with the command input parameters.

SECTION III contains specific technical information on the emulator and is entitled TECHNICAL REFERENCES. The information in this section will give you a detailed analysis on important emulator features and functions.

TABLE OF CONTENTS

SECTION I - ICD DESCRIPTION and OPERATION

1.1	Parts List	1-1
1.2	Description	1-1
1.3	Specifications (General)	1-2
1.4	Specifications (Emulation)	1-3
1.5	Controls and Component Functions	1-7
1.6	System Configurations.....	1-19
1.7	How to Operate Your ICD-178	1-21

SECTION II - COMMAND REFERENCE GUIDE

2.1	Command Functions	2-1
2.2	Debugging Operations	2-3
2.3	How to Enter Commands	2-7
2.4	Command Format Example	2-8
2.5	Notes on Command Formats	2-9
2.6	Command Interruption and Correction	2-10
2.7	Error Messages	2-11
2.8	Command Directory	2-13
2.9	Master Command Guide	2-15
2.10	Quick-Command Guide	2-99

SECTION III - TECHNICAL REFERENCES

3.1	Emulation Select Switch	3-3
3.2	Serial Interface	3-5
3.2.1	Introduction	3-5
3.2.2	Serial Interface Specifications	3-5
3.2.3	SIO S-791 Module Components	3-6
3.2.4	Transmission Format Switches	3-10
3.2.5	RS-232 Interface	3-11
3.2.6	Current Loop Interface	3-14
3.2.7	TTL Interface	3-16
3.2.8	Serial Interface Control Signals	3-19
3.3	Memory Emulation	3-21
3.3.1	ICD Emulation Memory	3-21
3.3.2	Memory Emulation	3-22

3.4	CPU Emulation	3-24
3.4.1	ICD/Target System Interface	3-25
3.4.2	MCU/ICD Deviations	3-26
3.4.3	Clock Selection	3-30
3.4.4	RESET Signal	3-33
3.4.5	Monitor (Interrupt Signal)	3-34
3.5	ROM (EPROM) Programmer	3-35
3.5.1	Specifications	3-35
3.5.2	ROM Programming	3-36
3.6	Real-time Trace	3-37
3.6.1	Real-time Trace Specifications	3-37
3.6.2	Real-time Trace Acquisition	3-38
3.6.3	Real-time Trace: Theory of Operation	3-39
3.7	Probes	
3.7.1	Event Trigger Probe	3-53
3.7.2	Map Control Probe	3-54
3.7.3	Emulation Qualify Probe	3-55

SECTION I

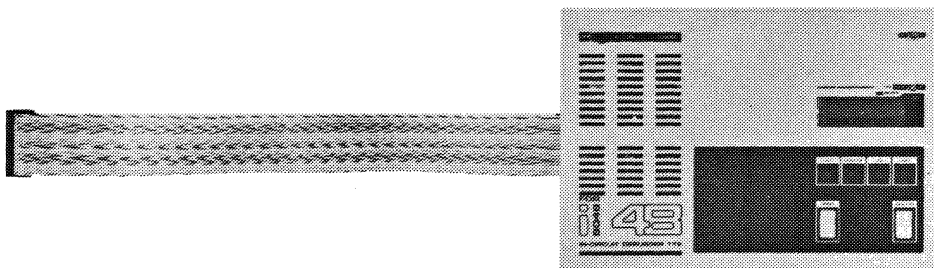
ICD DESCRIPTION and OPERATION

1.1 PARTS LIST

These items are included with your ICD-178;

ITEM	QTY
External Break Cable	1
Event Trigger Cable	1
Power Cord	1
Fan Unit	1
SCI Dip Plug	1 set
ROM Select Module [i80xx/i80xxH] [i87xx/i87xxH]	4
In-circuit Target Processor Probe	1
User's Manual	1

NOTE: See "Controls and Component Functions" (Section 1.5) to identify these items and their functions.



1.2 ICD DESCRIPTION

The ZAX ICD-178 IN-CIRCUIT EMULATOR functions as a stand-alone device for developing and maintaining hardware and software of 8048 microprocessors. The ICD-178 is designed for simple operation to allow you to begin debugging tasks quickly and effectively. The ICD-178 comes standard with 4K bytes of emulation memory, a 2K deep x 32 bits wide real-time trace buffer, a built-in EPROM programmer and 32 powerful debugger commands.

1.3 SPECIFICATIONS (General)

Dimensions	300mm (11.8in) wide 210mm (8.2in) deep 80mm (4.2in) high
Probe Length	510mm (20in) long
Weight	3.3kg (7.3lb)
Power Requirements	115VAC/230VAC, 50/60Hz
Processors	i8048 i8039 i8049 i8040 i8050 i8748 i8035 i8749
Clock speed (maximum)	11MHz
Memory Size	4K Bytes static RAM
Memory Mapping	256 Bytes
Real Time Trace Buffer	2K x 32 bits
System Commands	32
Breakpoints	3(hardware), 8(software) 1(external probe)
Communication Ports	2 channels RS 232C
Baud Rates	14 usable - to 19,200bps (factory set at 9,600bps)
Operating Temperature	0°C to 45°C
Storage Temperature	-10°C to 55°C
Humidity	30% to 85%

1.4 SPECIFICATIONS (Emulation)

Memory Area

Program memory	The entire area of the program memory (4K-bytes) is available. This memory is composed of high-speed static RAM.
User memory	The entire area of the 4K Byte memory space is available to the target system.
Mapping	Both the program and user memory can be mapped in 256 Byte units. Four types of mapping modes are available; User Memory Emulation Read/Write Memory Emulation Read Only Memory No Memory (Illegal Access)

I/O Port Ports 1 through 7 and the BUS port are all available to the target system.

Breakpoints 3 Hardware, 8 Software and External trigger break.

Hardware Break A,B,C and Event Trigger (enable/disable)
A,B,C Breaks Address 12 bit. Each bit may be specified 0,1 or "don't care."

Status:

- Opcode fetch
- External Data Memory access
- External Data Memory read
- External Data Memory write
- Port access
- Port read
- Port write

Arm:

Break occurs only at the breakpoint after an event trigger is generated.

1.4 SPECIFICATIONS (Emulation)

Event Trigger break

Address 12 bit. Each bit may be specified 0,1 or "don't care."

Status:

- Opcode fetch
- Memory access
- Memory read
- Memory write
- Port access
- Port read
- Port write

Data:

8 bit. Each bit may be specified 0,1 or "don't care."

Event Trigger output

One channel of the event trigger output can output to a logic analyzer, oscilloscope, etc. If a specified event condition occurs during the emulation, only the pulse output and the emulation is continued. The trigger can also be used for a hardware break.

Function:

Enable/disable. Address is 12 bit and may be specified as 0, 1 or "don't care" designation.

Status:

- Opcode fetch
- Memory access
- Memory read
- Memory write
- Port access
- Port write

Data:

8 bit name. Each bit can be specified as 0, 1 or "don't care."

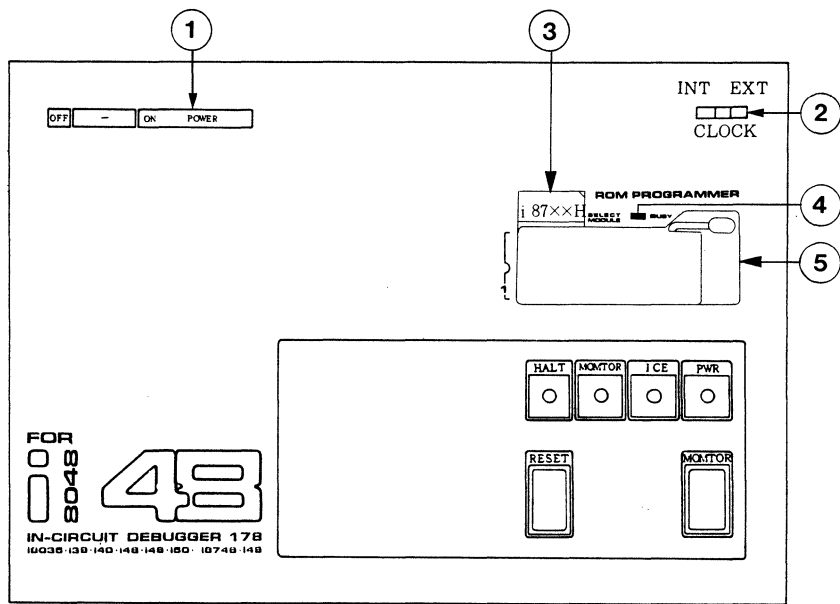
1.4 SPECIFICATIONS (Emulation)

External Trigger break	Breakpoint; 1 channel - TTL level Functional specification; HIGH edge or LOW edge.
Software Break	8 points; 0 - 7
Specification	Specify enable/disable of all software points.
Real-time trace	The addresses, data and status during emulation may be stored in storage mem- ory in real-time.
Trace capacity	2K x 32 bits.
Trace data	A 0-11, D 0-7 Program memory Data memory RD WR PORT
Trigger function	End monitor Begin monitor End event Begin event Center event Multiple event

1.5 CONTROLS and COMPONENT FUNCTIONS

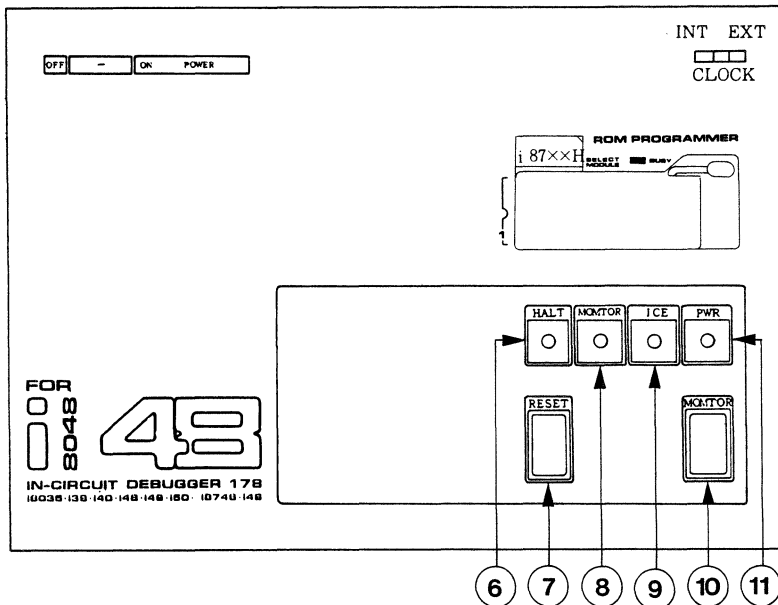
Section 1.5 explains and identifies various controls and components of the ICD-178 Emulator.

- 1 POWER switch. This switch is used to supply power to the ICD-178.
- 2 CLOCK select switch. This switch is used to select between the external target(EXT) and the internal(INT) ICD CPU clock.
- 3 PERSONALITY MODULE. This module allows the user to select various supporting hardware devices.
- 4 BUSY lamp. This LED comes on when the ROM programmer is running. DO NOT REMOVE THE ROM PROCESSOR WHEN THE BUSY LAMP IS ON.
- 5 ROM socket. This is the 8048 series ROM processor programming socket. To insert a ROM processor, lift up the small lever located in the recessed slot, insert the ROM processor and push the lever back into the recessed slot. Note the correct placement of the ROM processor in the socket.

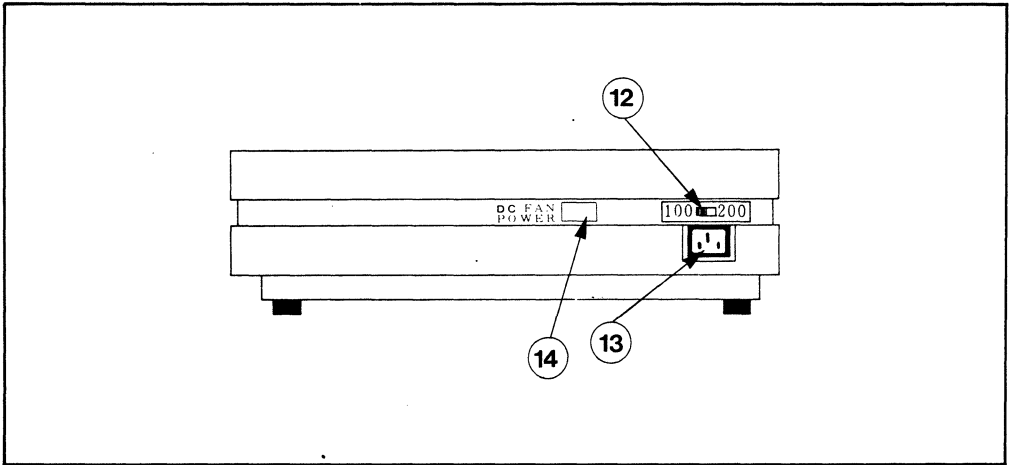


ICD Operator Panel Controls and Components

- 6 HALT lamp. This LED comes on when the ICD CPU has stopped after executing a HELP instruction or when BUS ACKNOWLEDGE (BUSAK) is in progress.
- 7 RESET switch. This momentary switch is used to reset the ICD-178 monitor. It is activated when the MONITOR lamp is on.
- 8 MONITOR lamp. This LED comes on to indicate that control is currently in the ICD monitor. It is off during emulation.
- 9 ICE (In-circuit Enable) lamp. This LED comes on when the ICD is operating in the incircuit mode (I1 or I2) with the target system.
- 10 MONITOR break switch. This momentary switch is used to return control to the ICD monitor during emulation.
- 11 POWER lamp. This LED comes on when the ICD is energized.

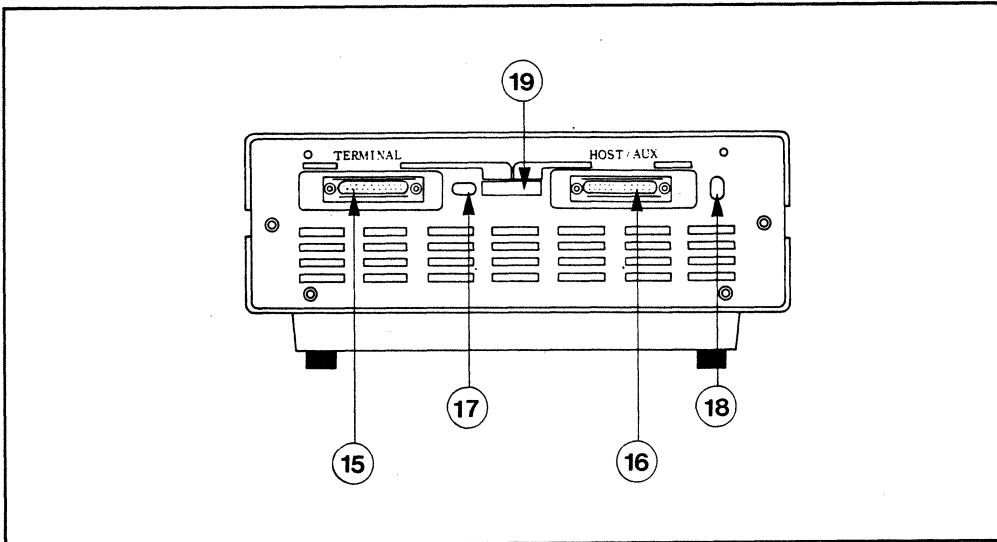


ICD Operator Panel Controls and Components



ICD Side Panel Controls and Components

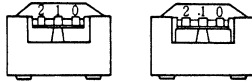
- 12 POWER select switch. This switch is used to select the proper power requirements for the ICD. Set the switch to 110/117V to run on a power supply of 110-120VAC or select 200/240V to run on a power supply of 200-240VAC.
NOTE: Set this switch BEFORE connecting the power plug.
- 13 Power connector. This connector is for the AC power plug.
- 14 DC Fan connector. This connector is for the 24V DC fan.



ICD End Panel Controls and Components

- 15 TERMINAL port connector. This connector attaches a console to the ICD-178. In a standalone configuration (LOCAL), it is used to input or output ICD operator commands. In the REMOTE mode it is used for auxiliary I/O.
- 16 HOST/AUX (Host/Auxiliary) port connector. This interface connector is used to connect a Host Computer system or auxiliary I/O to the ICD-178. In a standalone configuration (LOCAL), this port dumps objects, registers or memory to a host computer, printer, teletypewriter, etc. In REMOTE mode it can be used to input or output ICD operator commands under control of the host computer.
- 17 LOCAL/REM (Local/Remote) select switch. This switch is used to select the ports through which the ICD operator commands are entered.
- 18 DCE/DTE select switch. This switch is used to set the HOST/AUX port to RS232C data terminal equipment(DTE) or data circuit-terminating equipment(DCE). The factory setting is DTE.

TERMINAL HOST/AUX

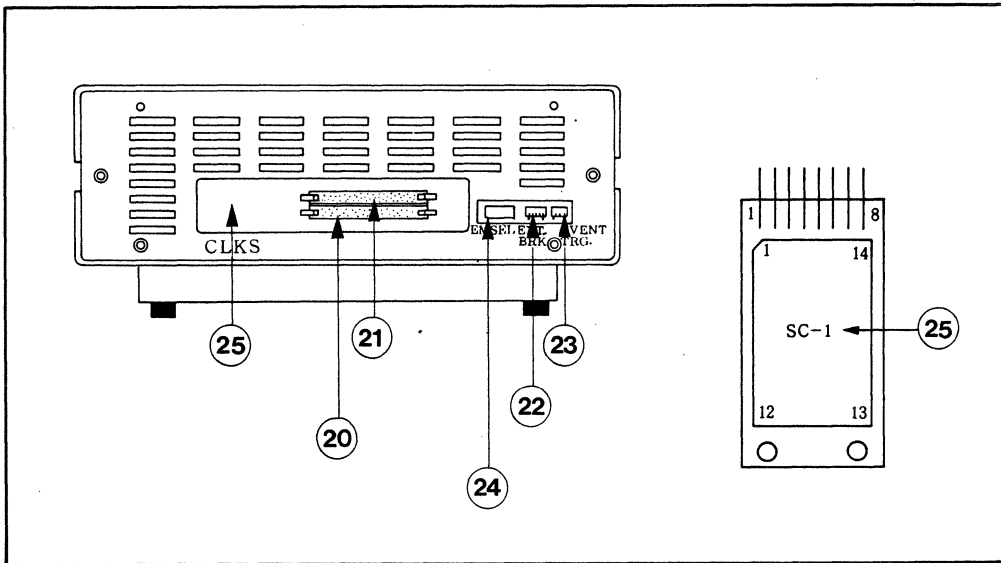


BAUD RATE SET SWITCH NO.	BAUD RATE(BPS)
0	19,200
1	9,600
2	4,800
3	2,400
4	1,200
5	600
6	300
7	150
8	75
9	110
A	134.5
B	200
C	1,800
D	2,000
E	—
F	—

ICD Baud Rates

- 19 BAUDRATE switches. These switches are used to set the baud rates for the TERMINAL and the HOST/AUX port. There are 14 usable baud rates available to the user. Baud rate switch letters E and F should not be used. Factory setting is 1 (9600bps) for TERMINAL and HOST/AUX ports.

NOTE: To set baud rate, turn dial using flat-head screwdriver or equivalent.



ICD End Panel Controls and Components

- 20 CPA In-circuit probe plug connector.
- 21 CPB In-circuit probe plug connector.

CAUTION: DO NOT REVERSE CONNECTIONS WHEN ATTACHING THE PROBES. CPA/CPB PROBE MISMATCH WILL CAUSE DAMAGE TO THE ICD-178.

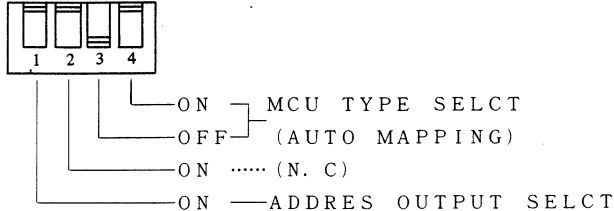
- 22 EXT. BRK. (External Break) connector. This is used to connect an external break cable to the ICD.
- 23 EVENT TRG. (Event Trigger) connector. This is used to connect an event trigger cable to the ICD.
- 24 E.M.SEL (Emulation Select) switch. This switch is used to set the machine cycle operation of the target system.
- 25 CLKS (Clock) Simulator. When the external clock from the target system is used, the clock circuit must be simulated on the SC1 dip plug in order to use the internal 8048 clock circuit.

Note: To remove clock simulator board, carefully pull the board straight out with a pair of needle-nose pliers. To replace, slide the board in the recessed slot on the left side of the holder and push it firmly in the socket.

24 EMULATION SELECT SWITCH

The Emulation Select switch implements specialized emulation conditions for the ICD-178 by modifying the machine cycle operation to the target system. Set this switch before performing emulation.

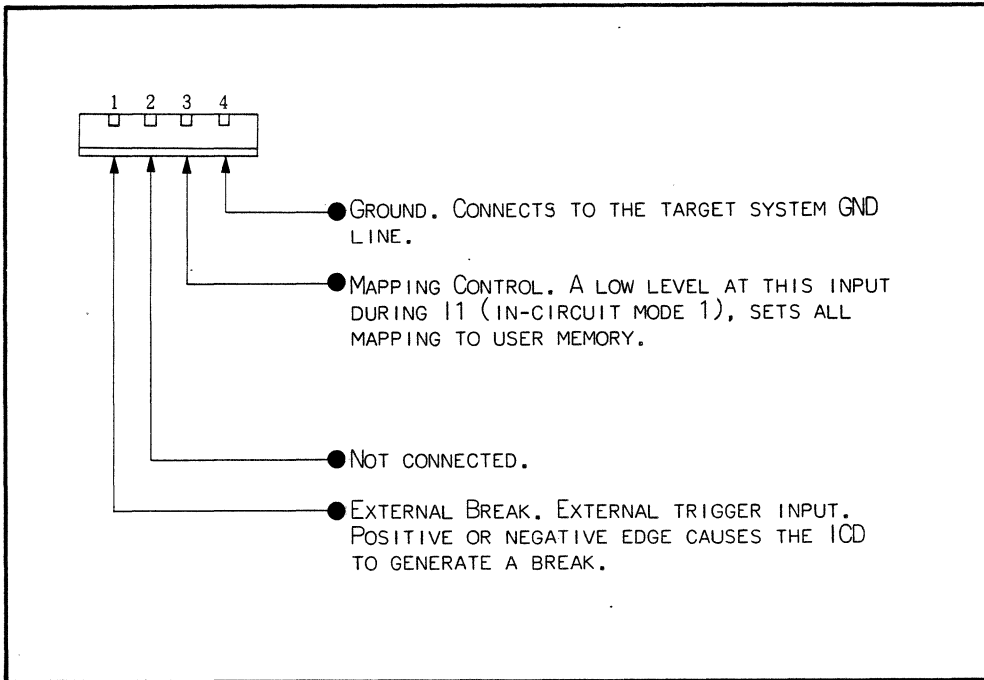
- Bit 3, 4 setting these bits (see below) effects the selection of the Object MCU and corresponding ICD internal program memory for the various processor families.
- Bit 2 not connected.
- Bit 1 OFF. This interfaces the ICD to memory devices that have slow access times.
- Bit 1 ... ON. This interfaces the ICD to memory devices that have normal access times.



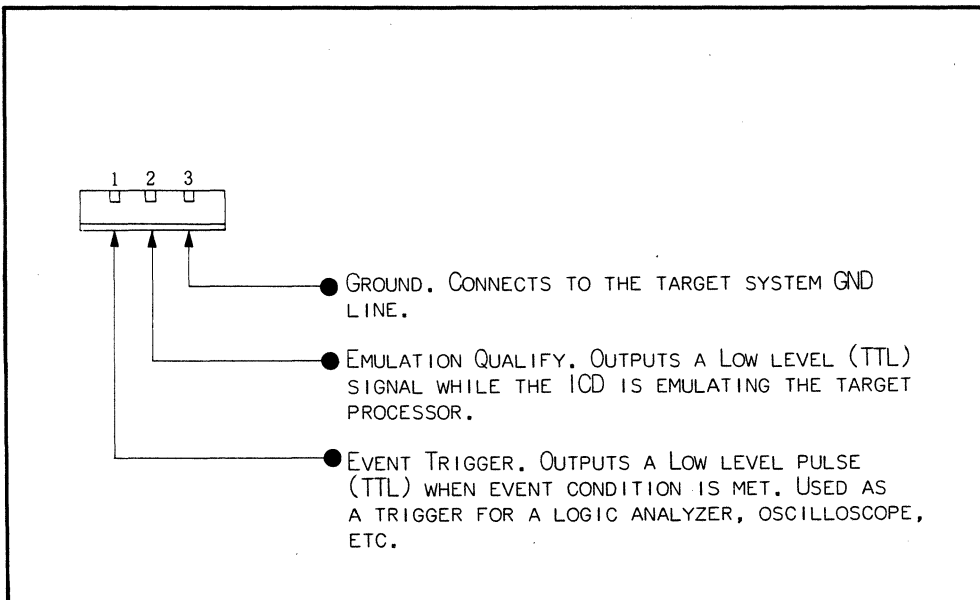
BIT 3	BIT 4	GROUP	OBJECT MCU	INTERNAL PROGRAM MEMORY
OFF	OFF	0	8535,8039,8040	0KBYTE
OFF*	ON*	1	8048,8748	1KBYTE
ON	OFF	2	8049,8749	2KBYTE
ON	ON	3	8050	4KBYTE

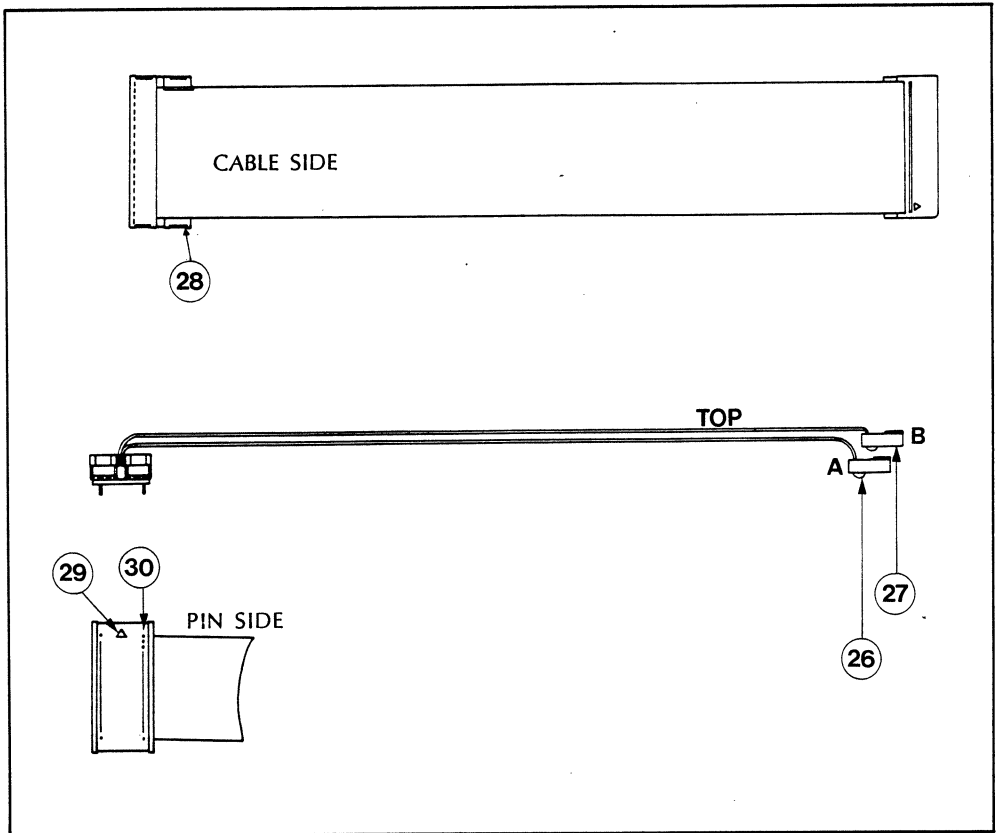
* FACTORY SETTING

External Break Cable



Event Trigger Cable





CPU Plug Probe

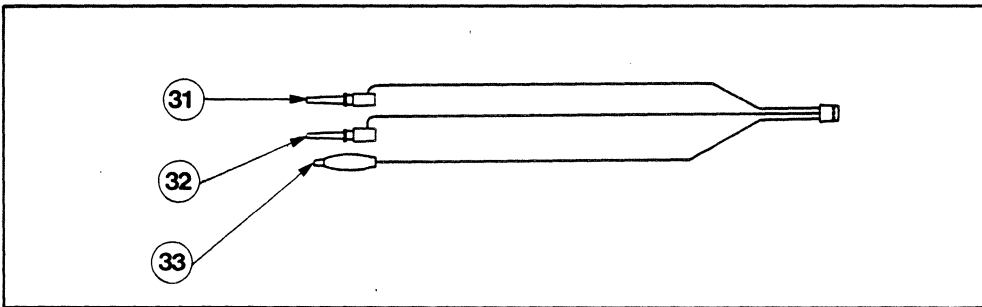
CSA/CSB In-circuit Probe Connector. Connects CPA to CSA and CPB to CSB.

IMPORTANT: WHEN CONNECTING THE IN-CIRCUIT PROBE, NOTE THAT THE CABLE WITH THE LONGEST LENGTH (CSB) MUST BE CONNECTED TO THE TOP (CPB) PROBE CONNECTOR OF THE ICD.

- 26 CSA In-circuit Probe Connector A (BOTTOM).
- 27 CSB In-circuit Probe Connector B (TOP).
- 28 Pin #1. Plugs into the CPU target processor socket. Note the proper polarity mark.
- 29 Polarity mark.
- 30 Pin #1.

External Break Cable. Connects to the EXT. BRK. socket of the ICD-178.

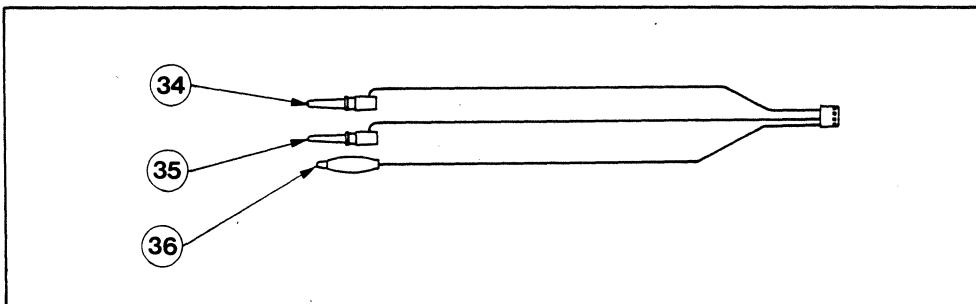
- 31 External Break Probe (red).
- 32 Map Control Probe (yellow).
- 33 Ground clip (black).



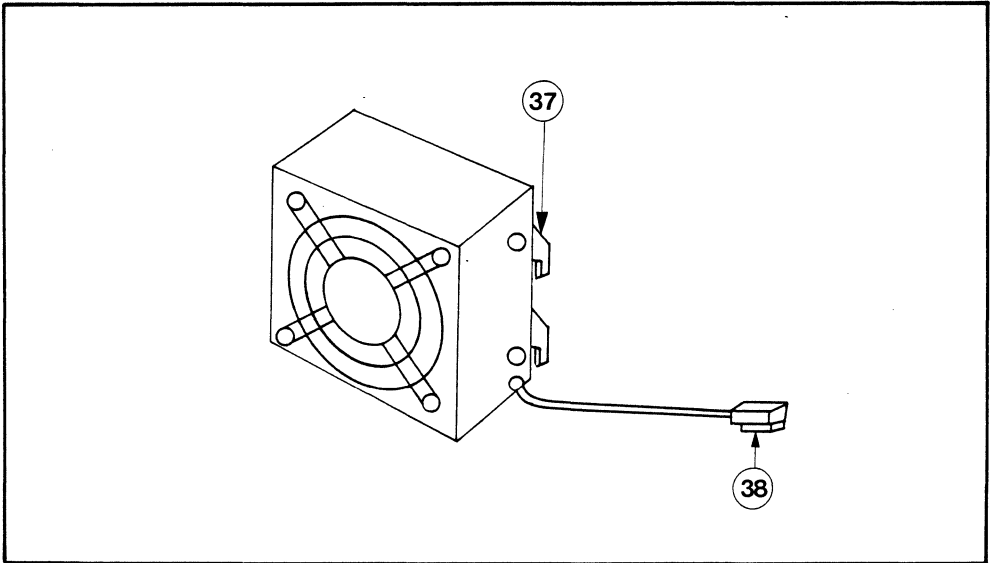
External Break Connector Diagram

Event Trigger Cable. Connects to the Event TRG. socket of the ICD-178.

- 34 Event Trigger Probe (green).
- 35 Emulation Qualifier Probe (white).
- 36 Ground clip (black).



Event Trigger Connector Diagram



DC Fan Unit

DC Fan Unit. Attaches to the rear panel of the ICD-178.

- 37 Fan Hook. Insert the fan hooks into the grooves located on the rear panel of the ICD-178.
- 38 Fan Plug. Insert this plug into the DC Fan Power Connector (24V DC) located on the rear panel.



1.6 SYSTEM CONFIGURATIONS

Section 1.6 explains the various ICD system configurations. Refer to this information when interfacing your ICD-178 to other communication devices, such as a host computer, printer, terminal console, and when connecting the emulator to the target system. Also included is a diagrammatic representation of the system configurations and start-up procedure for the ICD-178.

The ICD-178 may be set up to run in one of three different modes;

1) LOCAL mode (standalone configuration)

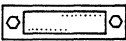
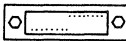
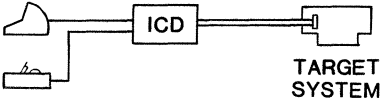


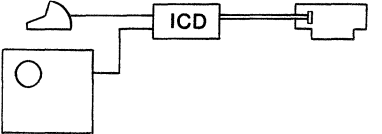


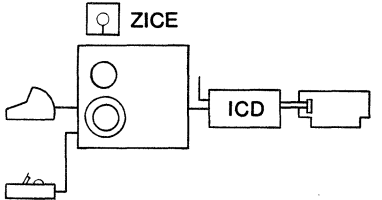
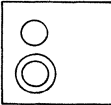
In this mode, the ICD-178 is connected to a console terminal through which commands are entered. The ICD may be used with or without a target system in this mode. The HOST/AUX port may be used to connect a printer to the ICD. Hardcopy results may then be produced by issuing a "PRINT" command.

2) LOCAL mode (standalone with host computer)

In this mode, the ICD-178 performs object file input/output with a host computer. ICD operator commands are entered from the console terminal connected to the host computer. A "USER" command allows direct communication between the ICD console terminal and the host computer.

3) REMOTE mode

In this mode, input/output from the host computer is performed by the utility software program ZICE. ICD operator commands are entered using the host computer's keyboard. The ICD-178 can then use the "Z" commands found in the COMMAND REFERENCE GUIDE.

SYSTEM CONFIGURATION ▼ ICD▶	LOCAL <input type="checkbox"/> REM ▲	DCE <input type="checkbox"/> DTE	TERMINAL 	HOST/AUX 
LOCAL (standalone) 	LOCAL <input checked="" type="checkbox"/> REM ▲	DCE <input checked="" type="checkbox"/> DTE	 CONSOLE	 PRINTER
LOCAL (standalone with Host Computer) 	LOCAL <input checked="" type="checkbox"/> REM ▲	DCE <input checked="" type="checkbox"/> DTE	 CONSOLE	 HOST SYSTEM
REMOTE 	LOCAL <input checked="" type="checkbox"/> REM ▲	DCE <input checked="" type="checkbox"/> DTE		 HOST SYSTEM

System Configuration

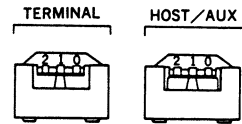
1.7 HOW TO OPERATE YOUR ICD-178

Section 1.7 tells you how to get your ICD-178 running and ready to accept commands. Before connecting the ICD-178 to any external device, make sure that the power to each component is OFF. Refer to Section 1.5, CONTROLS and COMPONENT FUNCTIONS to identify the controls and external components of the ICD-178.

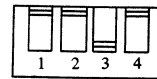
- 1) Set the LOCAL/REM and DCE/DTE switches of the ICD-178 to the positions specified in the System Configuration diagram (Section 1.6).



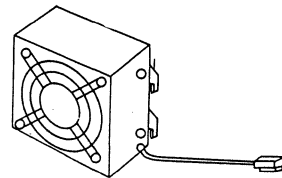
- 2) Set the ICD baud rates for the TERMINAL and HOST/AUX ports (normal setting is 9600 bps), then set your system to that same rate.



- 3) Set the ICD Emulation Select switch for the proper machine cycle operation desired, otherwise leave at the factory setting (see Section 1.5).



- 4) Attach the cooling fan to the ICD-178 and connect the power plug to the socket on the side of the unit.



- 5) Connect the RS-232 cable(s) to your monitor, printer or host computer and then to the ICD's TERMINAL or HOST/AUX port connectors.

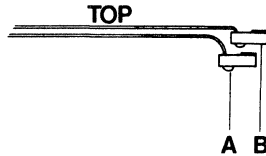


- 6) Remove the existing CPU (8048) from the target system and insert the CPU In-circuit probe (40-pin end) into the target system's CPU socket.



NOTE: Do not place the cables close to the target system's power supply or interference may result.

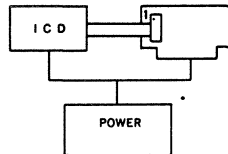
- 7) Connect the CSA/CSB In-circuit probe connectors to the ICD CPA/CPB connectors. The cable with the longest length (CSB) must be connected to the TOP (CPB) In-circuit plug probe connector on the ICD's end panel.



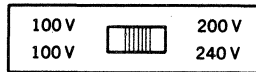
IMPORTANT: THE CABLE WITH THE LONGEST LENGTH (CSB) MUST BE CONNECTED TO THE TOP IN-CIRCUIT PLUG PROBE CONNECTOR.

CAUTION: DO NOT REVERSE PLUG PROBE CONNECTIONS. CSA/CSB MISMATCH WILL CAUSE SEVERE DAMAGE TO THE ICD-178.

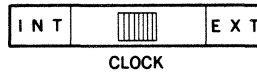
- 8) Connect the ICD and target system to the same primary power supply. Be sure all power is OFF before connecting components.



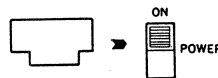
- 9) Select the proper voltage by setting the POWER select switch located on the ICD's mainframe.



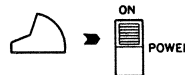
- 10) Set the CLOCK switch located on the ICD's Operator Panel to either EXT (external-target clock) or INT (internal-ICD clock). INT is the normal setting.



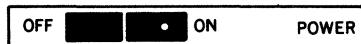
- 11) Turn the target system's power ON.



- 12) Turn the console's power ON.



- 13) Turn the ICD's power ON. A response message will then be displayed on the console, after which the emulator will wait for a command.



SECTION II

COMMAND REFERENCE GUIDE

2.1 COMMAND FUNCTIONS

The ICD-178 for 8048 is capable of executing 32 major debugger commands. A portion of these commands are divided into various sub-commands for a total of 63 different debugger command formats.

The following is a listing of the 32 major debugger commands with a brief explanation of each command's main function. To see a detailed explanation of each command, turn to Section 2.9 - MASTER COMMAND GUIDE. If you are using the ZAX ICD-178 emulator for the first time, read Sections 2.3 - 2.6 to become acquainted with the input format, syntax explanation and address/data parameters.

EMULATION MODE	Page
Incircuit (Sets or displays incircuit mode).....	2-57
Map (Sets or displays event point conditions)..	2-61
Pin (Sets or displays CPU pin status).....	2-69

DEBUGGING & BREAK/EVENT

Break (Sets or displays breakpoint conditions)...	2-17
Event (Sets or displays event point conditions)..	2-35
Go (Begins program execution).....	2-43
Next (Begins program execution trace mode).....	2-65
Trace (Sets or displays trace mode conditions)...	2-89

MEMORY:I/O:REGISTER

Assemble (Converts assembly code to machine code)...	2-15
Compare (Compares data between user and program memory).....	2-29
Disassemble (Displays processor assembly code).....	2-31
Dump (Displays hexadecimal memory contents).....	2-33
Examine (Displays memory contents in either ASCII or hex data).....	2-39
Fill (Fills memory with ASCII or hex data).....	2-41
Move (Moves data between user and program memory).....	2-63
Port (Examines and changes I/O port contents)...	2-71
Register (Changes or displays processor registers)..	2-77
Search (Searches memory contents for ASCII or hex data).....	2-83

PROGRAM INPUT/OUTPUT		Page
Load	(Loads object program from host system file or specified port).....	2-59
Save	(Saves memory contents to a disk file on the host system).....	2-81
Verify	(Compares an Intel hex format file to the ICD memory).....	2-95
REAL-TIME TRACE		
History	(Displays real-time storage contents).....	2-45
ROM MANIPULATION		
Rom	(ROM writer).....	2-79
OTHERS		
Calculation	(Displays calculation of hexadecimal and decimal data).....	2-27
HOST	(Sets LOCAL HOST-ON mode for standalone operation with host computer system).....	2-53
Identify	(Displays current ICD device name and firmware version).....	2-55
Offset	(Sets value in offset register for relative addressing).....	2-67
Print	(Controls output of ICD commands to an external printer).....	2-73
Quit	(Reboots to host system).....	2-75
Supervisor	(Sets a breakpoint as a supervisor call)...	2-85
User	(Allows terminal console to be used for host computer console).....	2-93
Zice	(Expands operation commands).....	2-97

2.2 DEBUGGING OPERATIONS

DEBUGGING OPERATIONS deals with using specific emulator commands to perform basic debugging operations. These programs are grouped under the following categories;

- A) PROGRAM DEBUGGING
- B) DEBUGGING THE TARGET SYSTEM
- C) USING THE REAL-TIME TRACE
- D) REGISTER/MEMORY MANIPULATION
- E) COMMUNICATION WITH A HOST COMPUTER
- F) ROM MANIPULATION.

The following is a listing of specific debugging instructions which are listed under the six major categories. The information in () shows the command used to execute the operation.

To locate a specific example when using DEBUGGING OPERATIONS, locate the two separate numbers that follow the operation instruction. The first number refers to the section/page number, and the second number indicates the example number.

A) PROGRAM DEBUGGING	Page/No.
Set an instruction execution break (Break)	2-22/2
Set a break and specify the address range (Break) ..	2-22/3
Set a breakpoint so that a break occurs when specific data is written in (Break)	2-22/4
Set a software break (Break)	2-23/5
Display the current breakpoint status (Break)	2-23/6
Cancel breakpoints (Break)	2-23/7
Display the passing status of an event (Break)	2-24/8
Perform tracing of all instructions (Trace)	2-90/57
Trace the branch instructions only (Trace)	2-90/58
Perform a step trace (Trace)	2-91/59
Display the current trace status (Trace)	2-91/60
Cancel trace (Trace)	2-91/61
Trace a 5-step program (Next)	2-65/36

A) PROGRAM DEBUGGING (continued)

Generate an event trigger when a specific address or data message is written in (Event)	2-36/15
Set an event trigger as a break (Event)	2-36/16
Display the current event status (Event)	2-36/17
Cancel the event points (Event)	2-37/18
Set a breakpoint so that a break occurs only after an event is generated (Break)	2-24/9
Restart the user program (Go)	2-43/21

B) DEBUGGING the TARGET SYSTEM

Transfer the contents of the target memory to the program memory (Move)	2-63/35
Substitute the ICD memory for a part of the target system memory (Map)	2-62/32
Map the memory (Map)	2-62/33
Display the current mapping status (Map)	2-62/34
Make the Input Interrupt signal invalid (Pin)	2-69/40
Display the current interrupt signal status (Pin) ..	2-69/41

C) USING THE REAL-TIME TRACE

Count the number of execution machine cycles (History)	2-50/22
Record the memory contents 2K cycles before a break in the user program (History)	2-50/23
Record the memory contents 2K cycles after the start of the user program (History)	2-51/24
Record the memory contents 2K cycles before and after the generation of an event trigger (History)	2-51/25
Record the contents of the user program for a duration of 256 machine cycles each time an event trigger is generated (History)	2-52/26
Perform a real-time trace memory search for the I/O port write cycle (History)	2-52/27

D) REGISTER/MEMORY MANIPULATION

Change the contents of a register (Register)	2-77/46
Display the contents of a register (Register)	2-77/47
Fill memory (Fill)	2-41/20
Display and disassemble the memory contents (Disassemble)	2-31/13
Replace the memory contents with assemble codes (Assemble)	2-15/1
Display the memory contents in byte units (Dump) ...	2-33/14
Change the memory contents in byte units (Examine) .	2-40/19
Transfer the memory contents between user and ICD memories (Move)	2-63/35
Display the address where specific data is located (Search)	2-83/55
Compare memory data and display the unmatching data and the location address (Compare)	2-29/12
Display the memory contents in Intel Hex format (Save)	2-82/54
Set offset registers for relative addressing (Offset)	2-67/37
Disassemble and display the memory contents specified by relative addressing (Offset)	2-68/38
Display the current offset register status (Offset)	2-68/39
Write data into the BUS port (Port)	2-72/42
Change the port contents in byte units (Port)	2-72/43

E) COMMUNICATION WITH A HOST COMPUTER

Read a file from the host system (Load)	2-60/30
Read a file from ZICE in the host system (Load)	2-60/31
Save a file (Save)	2-82/52
Save a file from ZICE in host system (Save)	2-82/53
Compare and verify the memory contents with a file in the host system (Verify)	2-96/63
Compare and verify the memory contents with a file read from ZICE in the host system (Verify)	2-96/64
Reboot the host system (Quit)	2-75/45
Use the monitor connected to the TERMINAL port as the console of the host system (User)	2-93/62
Output the contents displayed on the console to a list device (ZICE)	2-97/65
Connect the ICD to the XICE of the host computer system (Host).....	2-53/28

F) ROM MANIPULATION

Write the contents of ROM into the emulator memory (ROM)	2-80/48
Write the memory contents into ROM (ROM)	2-80/49
Write the Intel Hex file of the host system into ROM (ROM)	2-80/50
Display the current Group and ROM type (ROM)	2-80/51

2.3 HOW TO ENTER COMMANDS

Use the MASTER COMMAND GUIDE to show you the required input format, address/data parameters, and syntax explanations for the 32 debugger commands. To locate a command for a particular application, see either COMMAND FUNCTIONS or DEBUGGING OPERATIONS (Section 2.1, 2.2). Once you have found the command, see SYNTAX for the proper input format.

All commands begin with a derivative of the command name. This is usually the first or first two letters of the command name (such as CO for COMPARE). After the command name is entered, command parameters specify the operation of each command.

Items which are designated as KEYWORDS are displayed by BOLD characters (such as M, ON, OFF, HI, UP). They represent input choices that you must enter. Lowercase letters show items which you must supply (USER-SUPPLIED ITEMS). USER-SUPPLIED ITEMS include input information such as an address or data parameter. This information may be shown by an abbreviated format such as "int_point" for "initiation pointer."

Once you are familiar with the USER-SUPPLIED ITEMS, you may wish to dispense with the MASTER COMMAND GUIDE and use the COMMAND REFERENCE GUIDE. This guide shows you all available input parameters for the 63 different command formats.

To see an example of a command with the format and parameters explained, see the COMMAND FORMAT EXAMPLE (Section 2.4).

2.4 COMMAND FORMAT EXAMPLE

COMMAND: History format display

FUNCTION: Displays in disassembled or machine cycle format;

SYNTAX: `>H M|D [,int_point] [,term_point]<cr>`

TERMS: M :display in machine cycle format.
D :display in machine cycle and disassembled format.

int_point :initiation - point to begin display at [decimal 1 to 2047].

term_point :termination - display termination pointer [decimal 1 (default) to 2047].

EXAMPLE:

NOTES:

In this example;

COMMAND: - *the command name is found here.*

FUNCTION: - *the command description is found here.*

SYNTAX: - *the command input format is found here. The syntax for this command shows a vertical line ("|") between M and D which indicates that you must select either M or D. The brackets ("[]") around int_point and term_point indicate that these two parameters are OPTIONAL.*

TERMS: - *TERMS explains the parameters of the syntax. In this example, int_point and term_point represent address or data information which you must supply. Explanations for this information are shown on the right.*
- *the "_" between int and point (int_point) and term and point (term_point) indicate that these words go together to form a statement.*

EXAMPLE: - *an example for the command would follow the "EXAMPLE" title.*

NOTES: -*any important points about the command would follow the "NOTES" title.*

2.5 NOTES ON COMMAND FORMATS

*Keywords such as; M, ON, OFF, ME, CLR etc, are displayed by BOLD characters. They represent single letters or words which you must enter. Any combination of upper/lower case letters may be used.

*Lowercase letters show items which you must supply, such as "filename" in which you would enter the name of a selected file.

*A vertical line "|" is shown to indicate a choice between items which must be selected, such as;

ON|OFF

ON or OFF may be entered, but not both.

*Items shown in brackets, "[]" indicate parameters which are OPTIONAL.

*Include all punctuation such as commas, equal signs, colons, slashes, etc.

*The address/data parameters are given at the end of the explanations and grouped as "[phy12/hexdata/L byte/ASCII 32, etc]."

These address/data parameters include;

[phy10/12]	:hexadecimal physical memory address.
[hexdata]	:word or byte data (8/16 bit). Valid range is 0-F and X.
[L byte]	:number of bytes. Hexadecimal 16-bit, 0-F.
[ASCII 32]	:ASCII codes. All ASCII A thru Z characters(32) can be used.
decimal	:decimal values are specified within the brackets.

*Commands and examples are numerically identified and may be located by referencing either Section 2.1 - COMMAND FUNCTIONS, 2.2 - DEBUGGING OPERATIONS or 2.8 - COMMAND DIRECTORY.

Squares **4** identify command formats.

Circles **7** identify example types.

2.6 COMMAND INTERRUPTION and CORRECTION

INTERRUPTION Process

- Emulation stop
after GO command :press the MONITOR switch on Operator Panel.
- Stop :pressing the <ESC> key on console keyboard
stops the corresponding command.
- Interrupt <SP> :pressing the (space) bar on the console
keyboard causes an interrupt after one
line is displayed on screen. To restart
display, press (space) bar again.
- CTL-S :pressing the CTR-S key on the console
keyboard temporarily freezes the display.
- CTL-Q :pressing the (CTL-Q) key on the console
keyboard restarts the display.

INPUT STATEMENT CORRECTION

- Correction of
one character :(delete) Rubs out the specified
character(s).
- Backspace :<BS>(backspace) Backspace rubs out the
the character as it backspaces over it.
- Cancellation of
all data entry :<NAK> Pressing (CTL-U) cancels all
entry data.

2.7 ERROR MESSAGES

ERROR MESSAGE -----	COMMAND OCCURRENCE -----	DISPLAYED WHEN; -----
UNABLE BREAK ADDRESS	Break Go	a software break is specified in the non-RAM area
MULTI BREAK ADDRESS	Break Go	a software break is duplicated at the same address
WARNING UNABLE SOFT BREAK	Break	a software break is set at the address presently not mapped in RAM
*** FILE NOT FOUND	All	no file exists
*** DISK READ ERROR	All	a disk read error occurs
*** DISK WRITE ERROR	All	a sum check error occurs
*** NO DIRECTORY SPACE	All	no blank directory area available
C?>	All	a command code error occurs
P?>	All	a parameter code error occurs
/?>	All	a modifier code error occurs
MEMORY WRITE ERROR AT XXX	Assemble,Fill Load,Move, Examine	there is a memory modification error (Verify failed).
MEMORY TIMEOUT ERROR AT XXX	Assemble,Dump, Examine,Load, Fill, Move	memory in the target system does not respond to ICD access.
XXXX INPUT ERROR	Examine, Assemble, Port	an input error occurs - re-enter the command when this message is displayed.
BREAK BUSY	Break,Go	the break specification exceeds the specified limit.

2.8 COMMAND DIRECTORY

ICD-178 for 8048/49 MASTER COMMAND GUIDE
Major Commands: 32
Total Command Formats: 63

No.	COMMAND	Page
1	Assemble specification	2-15
2	Break status	2-17
3	Hardware Break specification	2-17
4	Hardware Break designation	2-17
5	Hardware arm/individual designation	2-19
6	Software Break specification	2-19
7	Software Break op code specification	2-19
8	Software Break designation	2-19
9	External Break HI/LOW designation	2-21
10	External Break ON/OFF designation	2-21
11	Event Break designation	2-21
12	Write protect break designation	2-21
13	Break initialize	2-21
14	Calculation	2-27
15	Compare	2-29
16	Disassemble	2-31
17	Dump	2-33
18	Event status	2-35
19	Event specification	2-35
20	Event designation	2-35
21	Examine memory only	2-39
22	Examine and change memory	2-39
23	Fill	2-41
24	Go	2-43
25	History - real-time trace status	2-45
26	History - real-time trace counter reset	2-45
27	History - monitor trigger storage	2-45
28	History - event trigger	2-47
29	History - format display	2-47
30	History - search (machine cycle)	2-49

No.	COMMAND	Page
31	HOST	2-53
32	Identification	2-55
33	Incircuit status	2-57
34	Incircuit specification	2-57
35	Load	2-59
36	Mapping status	2-61
37	Mapping specification	2-61
38	Mapping - external data mapping	2-61
39	Move	2-63
40	Next	2-65
41	Offset status	2-67
42	Offset specification	2-67
43	Pin status	2-69
44	Pin specification	2-69
45	Port examination	2-71
46	Port examination and change	2-71
47	Print	2-73
48	Quit	2-75
49	Register status	2-77
50	Register reset	2-77
51	Register change	2-77
52	ROM status	2-79
53	ROM specification	2-79
54	Save	2-81
55	Search	2-83
56	Supervisor status	2-85
57	Supervisor specification	2-85
58	Trace status	2-89
59	Trace designation	2-89
60	Trace specification	2-89
61	User	2-93
62	Verify	2-95
63	ZICE	2-97

1 COMMAND: ASSEMBLE specification

FUNCTION: Converts 8048 assembly code to 8048 machine code and changes the memory contents. This command is used for writing software patches using the emulators internal memory.

SYNTAX: `>A mem_addr <cr>`
`xxx (8048 assemble code)`
`xxx <cr>`

TERMS: mem_addr :memory address where assembled code is stored [phy10].
 xxx :next storage location.
 (8048 a. c.) :instruction to be assembled and stored.
 (cr) :exit assemble mode.

1 EXAMPLE: REPLACE THE CONTENTS OF THE MEMORY WITH ASSEMBLE CODES.

The following command can replace the contents of the memory with Assemble codes when a part of the memory contents needs to be changed. In this example, "MOV @RO, A" and "INC R0" are written into 100(H) and 101(H) respectively;

```
>A 100 <CR>
100 MOV @RO,A<CR>
101 INC R0<CR>
102 <CR>
>_
```

NOTES: 1) Values for the operand must be given in a hexadecimal or decimal number beginning with 0-9. The hexadecimal number should end with "H." Addition/subtraction on the same line is permitted.

2) This command cannot be used when the program memory is mapped in the target system.

2 COMMAND: BREAK status

FUNCTION: Displays the current break status.

SYNTAX: `>B <cr>`

3 COMMAND: Hardware BREAK specification

FUNCTION: Sets a hardware break.

SYNTAX: `>B [/A] [/B] [/C] stat,addr|port [,passcount] <cr>`

TERMS: A|B|C :hardware break name.

stat >>> break status;

M	:memory access
MR	:memory read
MW	:memory write
P	:port access
PR	:port read
PW	:port write
OF	:operation code fetch

addr|port >>> specifies a break address or break port;

addr	:[phy12].
port	:[specifies port; BUS, P1 thru P7].
passcount	:break pass count [decimal 1 (default) to 65535].

4 COMMAND: Hardware BREAK designation

FUNCTION: Turns ON, OFF or clears (CLR) a specified hardware breakpoint.

SYNTAX: `>B [/A] [/B] [/C] ON|OFF|CLR <cr>`

TERMS: A|B|C :hardware breakpoint name.

5 COMMAND: Hardware arm/individual BREAK specification

FUNCTION: ARM enables a break to occur after an event trigger takes place. INDividual allows a break to occur without regard to an event trigger.

SYNTAX: `>B [/A] [/B] [/C] ARM|IND <cr>`

TERMS: A|B|C :hardware break name.

6 COMMAND: Software BREAK specification

FUNCTION: Sets a software break.

SYNTAX: `>B [/0] [/1] ... [/7] addr [,passcount] <cr>`

TERMS: 0-7 :software break name [0 thru 7].

addr :break address [phy12].

passcount :specifies break passing count [decimal 1 (default) to 65535].

7 COMMAND: Software BREAK op code specification

FUNCTION: Specifies an operation code or enables/disables a software break.

SYNTAX: `>B S= code|EN|DI <cr>`

TERMS: code :operation code.
EN :enables a software break.
DI :disables a software break.

8 COMMAND: Software BREAK designation

FUNCTION: Turns ON, OFF or clears (CLR) a specified software break.

SYNTAX: `>B [/0] [/1] ... [/7] ON|OFF|CLR <cr>`

TERMS: 0-7 :software break name.

9 COMMAND: External BREAK HI/LOW designation

FUNCTION: Specifies an external break to occur at the HI going or LOw going edge.

SYNTAX: `>B[/X]HI|LO <cr>`

10 COMMAND: External BREAK designation

FUNCTION: Turns ON, OFF or clears (CLR) an external break.

SYNTAX: `>B/X ON|OFF|CLR <cr>`

11 COMMAND: Event BREAK designation

FUNCTION: Turns an event break ON or OFF.

SYNTAX: `>B/E ON|OFF <cr>`

12 COMMAND: Write protect BREAK designation

FUNCTION: Turns a write protect break ON or OFF.

SYNTAX: `>B/W ON|OFF <cr>`

13 COMMAND: BREAK initialize

FUNCTION: Clears the event pass condition and resets the ARM specification.

SYNTAX: `>B INI <cr>`

BREAK

2 EXAMPLE: SET AN INSTRUCTION EXECUTION BREAK.

This command is used when a program break is desired after the execution of the instruction at a specified address. In this example, a break is generated after the instruction at 112(H) of the program memory has been executed when the address is fetched;

```
>B OF,112 <CR>
>G 100 <CR>
PC MC OP PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP
112 90 MOVX @RO,A 08 00 00 00 00 00 00 00 00 00
<Break Hardware A>
>_
```

3 EXAMPLE: SET A BREAK WITH THE ADDRESS RANGE SPECIFIED.

The following command can set a program memory 1 fetch break if a break is to occur after execution of the instruction stored in the fetched address. In this example, a break is generated after executing the instruction at the fetched address, 100(H) - 1FF(H);

```
>B OF,1XX <CR>
>G 10 <CR>
PC MC OP PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP
120 00 NOP 08 00 00 00 00 00 00 00 00 00
<Break Hardware B>
>_
```

4 EXAMPLE: SET A BREAK TO OCCUR WHEN DATA IS WRITTEN INTO A SPECIFIC ADDRESS.

A memory write break can be generated by the following example if a program break is required for memory writing. In this example, a break in the program occurs when data is written into the memory, 30(H) - 3F(H);

```
>B MW,03X <CR>
>G 30 <CR>
PC MC OP PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP
112 90 MOVX @RO,A 08 00 30 00 44 00 00 00 00 00
<Break Hardware C>
>_
```

5 EXAMPLE: SET A SOFTWARE BREAK.

A software break can be generated by the following command when the user program resides in RAM and a break near the execution of the instruction is desired. In this example a break is generated after execution of the instruction stored at 300(H);

```

>B 300 <CR>
>B S=EN <CR>
>G 300 <CR>
PC MC OP          PSW  A R0 R1 R2 R3 R4 R5 R6 R7 SP
300 00 NOP          08 34 45 22 00 00 00 00.00 00 00
<Break Software 0>
>_

```

6 EXAMPLE: DISPLAY BREAK POINTS.

Displays the current hardware, software, EXT. BRK., write protect, and event trigger break status. In this example, hardware breaks A, B and C are currently ON, with addresses at 112, 1XX and 03X respectively. Software 0 is currently ON at address 300. Software (S) break is enabled. Code used for software break is 00(H). Event (E) break is ON. Write protect (W) is ON;

```

>B <CR>
A (ON)  OF  112  1  0 IND (0001_0001_0010)
B (ON)  OF  1XX  1  0 IND (0001_XXXX_XXXX)
C (ON)  MW  03X  1  0 IND (0000_0011_XXXX)
O (ON)  300  1  0
S (EN)  00
E (OFF)  1  0
W (ON)  1  0
>_

```

7 EXAMPLE: REMOVE BREAKPOINTS.

Removes the current breakpoints. In this example, Hardware breakpoints A and B are removed;

```

>B/A/B CLR <CR>
>B <CR>
C (ON)  MW  03X  1  0 IND (0000_0011_XXXX)
I (ON)  300  1  0
S (EN)  00
E (OFF)  1  0
W (ON)
>_

```

BREAK

8 EXAMPLE: DISPLAY THE EVENT PASS STATUS.

This command shows if an event has been passed and if so, issues a message. In this example, "Event Done" shows that an event has been passed. If this message is not shown, the event has not been passed. Note that the >BN (cr) command returns to the pre-event pass state;

```
>B <CR>
Event done
C (ON) MW 03X 1 0 IND (0000_0011_XXXX)
I (ON) 300 1 0
S (EN) 00
E (OFF) 1 0
W (ON)
>_
```

9 EXAMPLE: GENERATE A BREAK AFTER AN EVENT POINT IS PASSED.

This command causes breakpoints to be effective only after passing event points. In this example, a break occurs if address 100(H) of the program memory is read after data named 55(H) has been read from 8F(H) of the external data memory;

```
>EV CLR <CR>
>EV ST=MW A=8F D=55 <CR>
>B INI <CR>
>B/A OP,100 <CR>
>B/A ARM <CR>
>G 400 <CR>
PC MC OP PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP
100 05 EN I 08 00 00 00 00 00 00 00 00 00
<Break Hardware A>
>_
```

- NOTES:
- 1) A maximum of 12 breakpoints is permitted; 3 hardware, 8 software, and 1 external trigger break. If this limit is exceeded, the following message will be displayed;

 ** Break Busy
 - 2) If a software break is set at a non-RAM address, the following address will appear;

 ** Unable Soft Break
 - 3) A break is automatically set to the ON status after clearing.
 - 4) A breakpoint specification automatically displays the latest event (A, B, C or 0 - 7), if it has been previously set.
 - 5) A break cannot be effected by accessing the internal data memory.
 - 6) An event break is not effected in the port write cycle of the instructions; [ANL BUS, #Data], [ANL Pp, #Data], [ORL BUS, #Data] and [ORL Pp, #Data].

14 COMMAND: CALCULATION

FUNCTION: Performs subtraction and addition of hex and decimal numbers. Both subtraction and addition operations can be performed on the same line. This command also performs hex/decimal data conversion.

SYNTAX: `>C operand [\pm operand] [\pm operand]... <cr>`

TERMS: operand :a decimal or hexadecimal constant in the range of -8388608 to 8388607, or 0 to FFFFFFF(H).

10 EXAMPLE: PERFORM A CALCULATION OF HEX AND DECIMAL DATA.

In these examples, decimal data is subtracted from decimal data and hex data is added to hex data with the answers displayed in both hex and decimal format;

```
>C 1234+1234 <CR>
H: 0009A4
D: 2468
```

```
>C 1000H-FFFFH <CR>
H: 000001
D: 1
```

11 EXAMPLE: CONVERT A HEXADECIMAL NUMBER TO A DECIMAL EQUIVALENT.

In this example, the hex address 105(H) is converted to a decimal equivalent;

```
>C 105H <CR>
H: 000105
D: 261
```


15 COMMAND: COMPARE

FUNCTION: Compares the contents of a specified memory and then displays the non-matching data.

SYNTAX: `>CO[memdata]beg_addr,end_addr,[memdata]comp_addr [,UP|,PU] <cr`

TERMS: memdata >>> specifies type of data memory;

R: :internal (default)
 D: :external
 P: :program

beg_addr :beginning address for comparison [phy12].
 end_addr :ending address for comparison [phy12,L byte].
 comp_addr :beginning memory address to be compared [phy12].

UP :use beg_addr for the user memory and comp_addr for the ICD program memory.

PU :use beg_addr for the ICD program memory and comp_addr for the user memory.

12 EXAMPLE: COMPARE THE CONTENTS OF DIFFERENT MEMORIES AND DISPLAY THE UNMATCHING DATA AND THEIR LOCATION ADDRESSES.

The target memory and the program memory are compared and the unmatched data is displayed along with their location addresses. In this example, 0 -- FFF(H) of the target memory is compared with 0 -- FFF(H) and the unmatched data and their location addresses are displayed ("Adrs" and "M" in the heading mean the address and memory content respectively);

```
>CO P:0,FFF,P:0,UP <CR>
Adrs  M Adrs  M
000  00  000 B8
001  01  001 FF
002  02  002 00
003  03  003 00
004  04  004 00
005  05  005 00
006  06  006 00
007  07  007 00
>_
```


16 COMMAND: DISASSEMBLE

FUNCTION: Disassembles and displays the memory contents.

SYNTAX: >DI [beg_addr][,end_addr] <cr>

TERMS: beg_addr :beginning memory address [phy12].
 end_addr :ending memory address [phy12/L byte].
 no addr :start at the current instruction pointer,
 current CS.

13 EXAMPLE: DISASSEMBLE AND DISPLAY THE CONTENTS OF THE MEMORY.

Displays the memory contents in instruction words after they have been converted to assemble codes from hex format. In this example, beginning with address 400, the 20 byte contents are converted to assembled codes and then displayed. If disassembling is not possible, xxx will appear in the instruction portion;

```
>DI 400,L20 <CR>
400 B800 MOV RO,#0H
402 B910 MOV R1,#10H
404 BA20 MOV R2,#20H
406 BB30 MOV R3,#30H
408 BC40 MOV R4,#40H
40A BD50 MOV R5,#50H
40C BE60 MOV R6,#60H
40E BF70 MOV R7,#70H
410 23FF MOV A,#0FFH
412 A0 MOV @RO,A
413 0400 JMP OH
415 00 NOP
416 B80F MOV RO,#0FH
418 B901 MOV R1,#1H
41A BA02 MOV R2,#2H
41C BB03 MOV R3,#3H
41E BC04 MOV R4,#4H
>_
```


17 COMMAND: DUMP

FUNCTION: Displays the memory contents in either byte or word units. The contents are represented by a hex number or ASCII code.

SYNTAX: `>D [/W][memdata] beg_addr [,end_addr] <cr>`

TERMS: /W :dump memory contents on Word basis (default is byte basis).

memdata >>> specifies type of data memory;

R: :internal (default)

D: :external

P: :program

beg_addr :beginning memory address for display [phy12].

end_addr :ending memory address for display [phy12/L byte].

14 EXAMPLE: DISPLAY THE MEMORY CONTENTS IN BYTE UNITS.

This command example displays the contents of the program memory in hexadecimal representation;

```
>D P:100,L30 <CR>
  0 1 2 3 4 5 6 7 8 9 A B C D E F  ASCII CODE
100 A0 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
>
```


18 COMMAND: EVENT status

FUNCTION: Displays the current event status.

SYNTAX: >EV <cr>

19 COMMAND: EVENT specification

FUNCTION: Sets or releases an event trigger.

SYNTAX: >EV [ST=stat] [A=addr|P=port] [D=data] <cr>

TERMS: stat >>> event trigger status;

M	:memory access
MR	:memory read
MW	:memory write
P	:port access
PR	:port read
PW	:port write
OP	:op code fetch
ANY	:don't care

addr :directs event trigger at a specified break address [phy12]

port :[specifies bus port; BUS, P1 thru P7].

data :specifies event data [phy8].

20 COMMAND: EVENT designation

FUNCTION: Turns ON, OFF or clears (CLR) an event trigger.

SYNTAX: >EV ON|OFF|CLR <cr>

EVENT

- 15 EXAMPLE: GENERATE AN EVENT TRIGGER WHEN SPECIFIC DATA IS WRITTEN IN A SPECIFIC ADDRESS.

This command is used when it is necessary to externally take out a passing point in a program. In this command, an event trigger is effected when the user program writes 22(H) in location 33(H) of the external data memory;

```
>EV ST=MW A=33 D=22 <CR>  
>_____
```

- 16 EXAMPLE: SET AN EVENT TRIGGER AS A BREAK.

After setting a breakpoint by specifying status, address, data, etc., the following command entered will set an event as a break. In this example, a break is generated when the user writes 34(H) into location 24(H) of the external data memory;

```
>EV ST=MW A=24 D=34 <CR>  
>B/E ON <CR>  
>G <CR>  
PC MC OP PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP  
000 90 MOVX @R0,A 08 34 24 00 55 00 66 00 00 00 00  
<Break Event>  
>_____
```

- 17 EXAMPLE: DISPLAY THE CURRENT EVENT TRIGGER STATUS.

This command displays the current status of an event. In this example, MR (Memory Read) address is 34(H) and data is 44(H);

```
>EV St=MR A=34 D=44 <CR>  
>EV <CR>  
(ON)  
Status = MR  
Address = 034 (0000_0011_0100)  
Data = 44 (0100_0100)  
>_____
```

18 EXAMPLE: CLEAR THE CURRENT EVENT TRIGGER.

```
>EV CLR <CR>  
>EV <CR>  
Event is Clear  
>  
_
```

- NOTES:
- 1) A break by internal data memory access is not possible.
 - 2) If an event is set after clearing, the event is set to the ON status.
 - 3) Event specifications effected by changes take on the latest condition specified. The previous conditions not effected remain as they were.
 - 4) Event trigger pulse is not generated during the port write cycle of an instruction.

21 COMMAND: EXAMINE memory only

FUNCTION: Examines the memory contents and displays it in either ASCII or hex data.

SYNTAX: >E[/W] [/N] [memdata] addr <cr>

TERMS: /W :examine memory on a Word basis (default is a byte basis).
/N :does not verify change.

memdata >>> specifies type of data memory;

R: :internal (default)
D: :external
P: :program

addr :beginning address of examined memory
[phy x, where x= 8 if memdata= R: or D:
x=12 if memdata= P:].

22 COMMAND: EXAMINE and change

FUNCTION: Examines and changes the memory contents to hex data.

SYNTAX: >E[/W] [memdata] addr=mod_data <cr>

TERMS: /W :examine memory on a Word basis (default is a byte basis).

memdata >>> specifies type of data memory;

R: :internal (default)
D: :external
P: :program

addr :memory address to be changed [phy12].

mod_data :data to write to memory [hexdata].

Note: While in repeat mode use;

return (cr) :display next byte (word) of data.
comma ,(cr) :display same byte (word) of data.
caret ^ (cr) :display previous byte (word) of data.
slash /(cr) :exit Examine command.

EXAMINE

19 EXAMPLE: CHANGE THE CONTENTS OF MEMORY IN UNITS OF BYTES.

This command changes the contents of the memory in units of bytes. In this example, 55(H), 66(H) and 34(H) are written into locations 100(H), 101(H) and 102(H) using 1 byte units;

```
>E P:100 <CR>
100 04:55 <CR>
101 00:66 <CR>
102 00:34 <CR>
103 00:/ <CR>
>_
```

23 COMMAND: FILL

FUNCTION: Fills a block of memory with either hex or ASCII codes;

SYNTAX: **>F** [**/W**] [**/N**] [**memdata**] **beg_addr** [**,end_addr**],**data** <cr>

TERMS: /W :specifies to fill memory contents on a Word basis (default is a byte basis).

/N :does not verify memory contents.

memdata >>> specifies type of data memory;

R: :internal (default)

D: :external

P: :program

beg_addr :beginning block address to be filled [phy12].

end_addr :end block address [phy12/L byte] (default = length of area to be filled).

data :data that fills the block [hexdata/ASCII32].

20 EXAMPLE: FILL THE MEMORY.

This command fills the memory of a specific range with specified data. In this example, locations 0 -- 3FF(H) are filled with 4E71(H);

>F P:0,3FF,4E,71 <CR>

>D P:0,100 <CR>

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ASCII CODE
000	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	NqNqNqNqNqNqNqNqNq
010	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	NqNqNqNqNqNqNqNqNq
020	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	NqNqNqNqNqNqNqNqNq
030	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	NqNqNqNqNqNqNqNqNq
040	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	NqNqNqNqNqNqNqNqNq
050	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	NqNqNqNqNqNqNqNqNq
060	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	NqNqNqNqNqNqNqNqNq
070	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	NqNqNqNqNqNqNqNqNq
080	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	NqNqNqNqNqNqNqNqNq
>																	

24 COMMAND: GO

FUNCTION: Executes the user program. The GO command with no parameters continues execution after a break.

SYNTAX: `>G[beg_addr][,end_addr][,end_addr*]<cr>`

TERMS: beg_addr :beginning address to execute [phy12]
 (default=the current PC is used).
 end_addr :ending address to execute [phy12].
 end_addr* :optional second end address [phy12].

21 EXAMPLE: RESTART THE USER PROGRAM.

This command restarts the user program at the beginning (location 0) after clearing all registers. In this example, all the registers are cleared and the user program is executed starting at location 000(H);

```
>R RES <CR>
>G <CR>
PC MC OP
100 00 NOP PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP
<Break Monitor> 08 33 00 21 34 33 45 00 00 00 00
>_
```


COMMAND: HISTORY

FUNCTION (General): Specifies the trigger mode (6 types) for real-time tracing, and displays or searches for the user program operation to be traced.

25 COMMAND: Real-time trace status (HISTORY)

FUNCTION: Displays the current trace status.

SYNTAX: >H <cr>

26 COMMAND: Real-time trace counter reset (HISTORY)

FUNCTION: Clears (resets) the real-time trace counter;

SYNTAX: >H CLR <cr>

27 COMMAND: Monitor trigger storage (HISTORY)

FUNCTION: Specifies the trigger mode.

SYNTAX: >H EE|EM <cr>

TERMS: EE :end event trigger. Trace terminates when an event point is passed (range is 2045).

EM :end monitor trigger. Trace terminates when control returns to the monitor (maximum range is 2045).

28 COMMAND: Event trigger (HISTORY)

FUNCTION: Sets the trace specification.

SYNTAX: `>H BM|BE|CE|ME [,range] <cr>`

TERMS:

BM	:begin monitor. Trace section is initiated by emulation start and terminated after the number of words specified in the range parameter have been stored.
BE	:begin event. Trace begins when an event point is passed.
CE	:center event. The trace section is specified on either side of the event point.
ME	:multiple event. Trace is performed each time an event point is passed during loop processing.
range	:specifies trace range (number of words to be stored) [decimal 1 to 2045].

29 COMMAND: HISTORY format display

FUNCTION: Displays in disassembled or machine cycle format.

SYNTAX: `>H M|D [,int_point] [,term_point] <cr>`

TERMS:

M	:display in machine cycle format.
D	:display in machine cycle and disassembled format.
int_point	:initiation - point to begin display at [decimal 1 to 2047].
term_point	:termination - display termination pointer [decimal 1 (default) to 2047].

30 COMMAND: HISTORY search by machine cycle

FUNCTION: Searches for real-time trace contents.

SYNTAX: >HS,/addr|/port /[data]/[mem] [,int_point] [,term_point] <cr>

TERMS: addr :address to begin search [hex].

port >>> search port specification;

BUS or P1, P2, P3, P4, P5, P6, P7

data :data to be searched for [hexdata].

mem >>> specifies history memory;

MR: :memory read
MW: :memory write
PR: :port read
PW: :port write
OF: :opcode fetch

int_point :initiation - point to begin display at
[decimal 1 to 2047].

term_point :termination - display termination pointer
[decimal 1 (default) to 2047].

HISTORY

22 EXAMPLE: COUNT THE NUMBER OF EXECUTION MACHINE CYCLES.

This command calculates the time required for execution of an instruction by counting the number of clock cycles. In this example, the number of machine cycles required for the execution of the user program between 400(H) and 4FE(H) is 255 (decimal) and FF (hexadecimal);

```
>H CLR <CR>
>G 400,4FE <CR>
PC MC PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP
4FE FE MOV A,R6 4F 12 0D 04 07 02 24 0D 12 09 07
< Break Hardware A >
>H <CR>
LAE Counter = 00000FF/255
Storage mode = EM
Storage Size = 256/256
>_
```

23 EXAMPLE: RECORD THE PROGRAM CONTENT 2K MACHINE CYCLES BEFORE A BREAK IN THE USER PROGRAM.

This command sets a real-time trace at EM (End Monitor) and displays the memory contents before a break;

```
>H EM <CR>
>G <CR>
PC MC PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP
489 00 NOP 08 81 06 0A 07 10 10 10 08 02 00
< Break Monitor >
>HD <CR>
Point T Addr DT ST OP
2047 14E 4E ORL A,R6
2046 14F 4F ORL A,R7
2045 150 50 ANL A,@R0
2044 151 51 ANL A,@R1
2043 152 5253 JB2 53H
2041 154 5455 CALL 255H
>_
```

- 24 EXAMPLE: RECORD THE PROGRAM CONTENT 2K MACHINE CYCLES AFTER THE START OF THE USER PROGRAM.

This command sets a real-time trace at BM (Begin Monitor) and displays the contents of the program in disassembled format;

```
>H BM <CR>
>G <CR>
PC MC                PSW  A RO R1 R2 R3 R4 R5 R6 R7 SP
524 2425 JMP        125H  4A 24 03 05 0A 23 02 0B 0B 02
< Break Monitor >
>H D <CR>
Point T Addr DT      ST  OP
2047 * 48A 00        ST  NOP
2046   48B 00        ST  NOP
2045   48C 00        ST  NOP
2044   48D 00        ST  NOP
2043   48E 00        ST  NOP
2042   48F 00        ST  NOP
>_____
```

- 25 EXAMPLE: RECORD THE PROGRAM CONTENT 2K MACHINE CYCLES BEFORE AND AFTER THE OCCURRENCE OF THE EVENT TRIGGER.

In this example, the content of the program 2K cycles before and after the occurrence of the event (data 22(H)) is written into location 46(H) of the external data memory;

```
>EV ST=MR A=46 D=22 <CR>
>H CE <CR>
>G <CR>
PC MC                PSW  A RO R1 R2 R3 R4 R5 R6 R7 SP
131 31  XCHD        A,@R1  4B 06 05 0E 0E 0E 06 08 04 78 03
< Break Monitor >
>H D
Point T Addr DT      ST  OP
2047   4FD FD        ST  MOV    A,R5
2046   4FE FE        ST  MOV    A,R6
2045   4FF FF        ST  MOV    A,R7
2044   500 00        ST  NOP
>_____
```


HISTORY

26 EXAMPLE: RECORD THE USER PROGRAM CONTENTS FOR 256 CYCLES EACH TIME AN EVENT TRIGGER IS PASSED.

This command records the program contents for 256 machine cycles each time an event occurs [data is written into location 56(H)]:

```
>EV ST+MW A=56 D=XX <CR>
>H ME,256 <CR>
>G <CR>
PC MC PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP
281 81 MOVX A,@R1 08 00 01 03 07 00 23 11 08 02 00
< Break Monitor >
>H M <CR>
Point T Addr DT ST
0258 * 056 9F MW
0257 008 08 OF
0256 BUS PR
0255 009 09 OF
0254 P1 PR
0253 00A 0A OF
0252 P2 PR
>_
```

27 EXAMPLE: DETERMINE WRITE CYCLE FOR A SPECIFIC I/O PORT FROM THE REAL-TIME TRACE MEMORY.

The command used in this example analyzes machine cycles when data is written in the BUS port from the real-time trace memory;

```
>G <CR>
PC MC PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP
131 31 XCHD A,@R1 4B 06 05 0E 0E 0E 06 08 04 78 03
< Break Monitor >
>H S,/BUS//PW <CR>
Point T Addr DT ST
1858 BUS PW
1575 BUS PW
1292 BUS PW
1009 BUS PW
0726 BUS PW
0443 BUS PW
0160 BUS PW
>_
```

- NOTES:
- 1) The real-time counter uses 32 bits of ALE as 1 cycle.
 - 2) The display of the real-time trace start storage pointers include the pointers just after the program was started and those where the program was started and the PC specified. If these pointers exceed 2047, a "Full" message is displayed.

31 COMMAND: HOST

FUNCTION: Allows the use of symbolic debugging, "Z" command, etc. when the host computer is connected to the HOST/AUX port and the ICD is used in the LOCAL mode.

SYNTAX: >HOST ON|OFF <cr>

TERMS: ON :establishes the HOST-ON mode which connects the host system to the ICD in the LOCAL mode.

OFF :reverts to the normal HOST-OFF mode.

28 EXAMPLE: CONNECT THE ICD TO XICE IN THE HOST COMPUTER SYSTEM (LOCAL MODE).

This example command connects the ICD to XICE to perform Object file input from the host system, command input/output (HOST-ON), symbol/number conversions or number/symbol conversions, and to use the various commands expanded by the "Z" command;

```
>U ! <CR>
A>XICE <CR> !
>HOST ON <CR>
ZOS XICE V1.0
>_
```


32 COMMAND: IDENTIFICATION

FUNCTION: Displays the ICD device name and a version of the installed firmware.

SYNTAX: `>ID <cr>`

29 EXAMPLE: DISPLAY THE DEVICE NAME AND THE FIRMWARE VERSION.

This command shows that the ICD-178 is for use with the i48 processor and the version is V1.0;

```
>ID <CR>
ICD-178 for i48      V1.0
>_
```


33 COMMAND: INCIRCUIT status

FUNCTION: Displays the current incircuit mode.

SYNTAX: >I <cr>

34 COMMAND: INCIRCUIT specification

FUNCTION: Sets the ICD mapping mode.

SYNTAX: >I 0|1|2 <cr>

TERMS:

- 0 :System mode. Uses the ICD program memory. In this mode the target system I/O and interrupt signals are ignored.
- 1 :Partial mode. Uses the ICD program and target (user) system memories. In this mode, interrupt can be qualified by using the "PIN" command.
- 2 :Full mode. Uses only the target system memory. In this mode, the memories specified by the "MAP" command as RW or RO function as US memory. The target system I/O and interruption signal is effective regardless of the "PIN" command setting. However, if the EA signal of the "PIN" command is LOW, the program memory assumes the following status;

- Group 0: Entire area is US.
- Group 1: 0-3FF is RO and 400-FFF is US.
- Group 2: 0-7FF is RO and 800-FFF is US.
- Group 3: Entire area is RO

NOTES: In-circuit Mode and MAP/PIN command settings;

Command set Incircuit Mode		Mapping set Command							PIN Set Command	
		Program mem			XDM					
		RO	US	NO	RO	RW	US	NO	EN	DI
I 0	System Mode	RO	(RO)	NO	RO	RW	(RW)	NO	(DI)	DI
I 1	Partial Incircuit Mode	RO	US	NO	RO	RW	US	NO	EN	DI
I 2	Full Incircuit Mode	(US)	US	NO	(US)	(US)	US	NO	EN	(EN)

- (RW) - functions as RW regardless of "MAP" setting.
- (US) - functions as US regardless of "MAP" setting.
- (EN) - functions as EN regardless of "PIN" setting.
- (DI) - functions as DI regardless of "PIN" setting.

35 COMMAND: LOAD

FUNCTION: Downloads either a hex file on diskette or the object program (Intel format) from a specified port.

SYNTAX: >L [/T | /P | /A | /H] [filename] [,bias] [,message] <cr>

TERMS: T :use the terminal port (ignore software handshake)
 P :use the terminal port (perform software handshake)
 A :use the auxiliary port
 H :use the host port

filename :specifies the name of the file to download to the ICD (available with ZICE software only).

bias :specifies the memory address offset to be added to the object file being loaded. If omitted, starting address is specified [L byte].

message :load message [hexdata/ASCII 32]. "message" is entered in this form; 'message',ØD - where ØD represents a line feed.

The ICD outputs the Load message to the TERMINAL or AUX port prior to loading the object program (Intel format) if /T or /A is specified. "filename" may be used for the Load message when the /P or /H port is specified.

NOTES: Object File I/O Procedure

MODE	REMOTE	LOCAL
/T	x	x
/P	o	o
/A	x	x
/H	o	o
Default	o (HOST)	x (TERMINAL)

- o - The ICD inputs an Intel format file through handshaking. (See Section IV, COMMUNICATION PROTOCOL).
- x - The ICD inputs Intel format data only and does not perform handshaking. The data input ends upon receiving an end record (Intel format) or <EXT> (Control + C) through a specified port.

LOAD

30 EXAMPLE: READ A FILE IN INTEL FORMAT FROM THE HOST SYSTEM.

When the host system is connected to the HOST/AUX port in the LOCAL mode and it supports the protocol at the ICD loading time, this command loads a file from the host system in Intel format. In this example, the host system loads a file named FILE1.HEX (Intel format);

```
>L/H FILE1.HEX <CR>  
000  
>_
```

31 EXAMPLE: READ AN INTEL FORMAT FILE FROM ZICE IN THE HOST SYSTEM.

If ZICE software is being used in the REMOTE mode, this command loads a file from the host system (Intel format);

```
>L FILE1.HEX <CR>  
000  
>_
```

36 COMMAND: MAPPING status

FUNCTION: Displays the current map status.

SYNTAX: `>MA <cr>`

37 COMMAND: MAPPING specification

FUNCTION: Sets the ICD memory map. The target system or ICD program memory is used in 1K byte increments. (For the relationship between mapping and the in-circuit mode, refer to the IN-CIRCUIT command.)

SYNTAX: `>MA beg_addr [,end_addr]=memspec <cr>`

TERMS: beg_addr :beginning address of mapping [phy12].
 end_addr :ending address of mapping [phy12/L byte]
 memspec >>> specifies the program memory;
 RO :read only memory
 NO :non memory area (a break occurs in this area if accessed by a program)
 US :user (target system) memory

38 COMMAND: MAPPING - External data memory (XDM)

FUNCTION: Sets the external data memory mapping specification.

SYNTAX: `>MA XDM=memspec <cr>`

TERMS: memspec >>> specifies the program memory;
 RO :read only memory
 RW :read/write memory
 NO :no (non) memory
 US :user (target system) memory

MAPPING

- 32 EXAMPLE: SUBSTITUTE THE ICD MEMORY FOR A PART OF THE TARGET SYSTEM MEMORY.

If the target system memory is partially faulty, the defective portion can be replaced with the ICD memory by this command. In this example, the "I1" command sets the mapping mode, and the "MAP" command allocates 0 -- 3FF(H) to the program memory and 400FFF(H) to the user memory. The contents of the user memory can be transferred to the program memory by the "MOVE" command if necessary;

```
>I1 <CR>
>MA 0,3FF=RO <CR>
>MA 400,FFF=US, <CR>
>MOV P:0,FFF,P:),UP <CR>
>
_
```

- 33 EXAMPLE: MAP THE MEMORY.

The command in this example defines locations 0 -- FF(H) and 800 -- FFF(H) of the program memory and the external data memory as RO (Read Only). The location of the program memory 100 -- 7FF(H) is defined as NO (non-memory);

```
>MA 0,=RO <CR>
>MA 100,7FF=NO <CR>
>MA 800,FFF=RO <CR>
>MA XDM=RO <CR>
>
_
```

- 34 EXAMPLE: SHOW THE CURRENT MAPPING STATUS.

In this example, locations 0 -- FF(H) of the program memory are allocated to US (User memory), locations 100 -- 3FF(H) are allocated to RO (Read Only) and locations 400 -- 7FF(H) are allocated to NO (No memory);

```
>MA <CR>
In-Circuit Mode 1
Group 3
Program Memory
000-0FF = US
100-3FF = RO
400-7FF = NO
800-FFF = US
XDM
XDM = US
>
_
```

39 COMMAND: MOVE

FUNCTION: Moves memory between the ICD and the target system.

SYNTAX: **>M**[memdata] beg_addr,end_addr,[memdata]mov_addr[,UP|,PU]<cr>

TERMS: memdata >>> specifies type of data memory;

R: :internal (default)
D: :external
P: :program

beg_addr :beginning address of data source [phy12].
end_addr :ending address of data source [phy12/L byte].
mov_addr :beginning address for destination [phy12].

UP|PU >>> optional parameter (default = data is moved according to the memory map);

UP :source is USER memory, destination is ICD .
 program memory.

PU :source is ICD program memory, destination is
 USER memory.

NOTES: If "P:" is specified above, "PU" becomes invalid. When
 the "D:" or "R:" destination is specified, the maximum
 transfer range is 256 bytes and "mov_addr" is limited
 to 0 -- FF(H). If "D:" or "R:" is specified as the
 transfer area, the range of "beg_addr" is 0 -- FF(H)
 and the transfer range is 256 bytes.

35 EXAMPLE: TRANSFER THE CONTENTS OF THE TARGET MEMORY TO THE PROGRAM MEMORY.

This command transfers the contents of the user memory (residing in ROM) to the program memory, changing part of the contents. In this example, the contents of the user memory location is 0 -- FFF(H);

```
>I1 <CR>
>MO P:0,FFF,P:0,UP <CR>
>E PL100 <CR>
100 00:01
101 01:/
>_
```


40 COMMAND: NEXT

FUNCTION: Executes n-step All trace display from the current pointer.

SYNTAX: >N [steps] <cr>

TERMS: steps :specifies the number of instructions to execute [decimal 1 to 65535].

36 EXAMPLE: TRACE AND DISPLAY 5 STEPS BEGINNING AT THE CURRENT PC.

When an unconditional trace is desired, this command traces all the instructions depending on the number of times specified. This trace mode is independent of the current trace mode. In this example, five instructions are traced starting at the current PC;

```
>N 5 <CR>
PC MC
100 18 INC R0
101 F8 MOV A,R0
102 19 INC R1
103 1A INC R2
104 1B INC R3
>_____
```

PSW	A	R0	R1	R2	R3	R4	R5	R6	R7	SP
08	00	57	67	35	01	00	00	00	00	00
..	57
..	68
..	36
..	02

41 COMMAND: OFFSET status

FUNCTION: Displays the current offset value.

SYNTAX: >O <cr>

42 COMMAND: OFFSET specification

FUNCTION: Sets a value (1 of 4) in an offset register for relative addressing. All offset registers can be used for ICD memory address parameters.

SYNTAX: >O &0|&1|&2|&3=addr <cr>

TERMS: addr :address - offset to place in the register
[hex16]

&0 :offset register 0
&1 :offset register 1
&2 :offset register 2
&3 :offset register 3

37 EXAMPLE: SET AN OFFSET REGISTER FOR SPECIFYING RELATIVE ADDRESSING.

When debugging a program consisting of a number of different modules, assign the difference between the address on the assemble list and the physical address to an offset register. The address on the assembly list may be found by adding the name of the offset register at addressing. In this example, the offset register, "&0" is set to 325(H);

```
>O &0=325 <CR>
>D 10&0 <CR>
   5 6 7 8 9 A B C D E F 0 1 2 3 4 ASCII CODE
35 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
>_
```


OFFSET

38 EXAMPLE: DISASSEMBLE AND DISPLAY THE CONTENTS OF MEMORY LOCATIONS SPECIFIED BY RELATIVE ADDRESSING.

This command disassembles and displays the contents of memory locations starting at an address plus the value specified in the offset register. In this example, the contents of memory 20(H) bytes after the location 20(H) + &0 (offset register) are displayed;

```
>DI 20&0,L20 <CR>
355 55   STRT   T
356 5657 JT1    57H
358 58   ANL   A,R0
359 59   ANL   A,R1
35A 5A   ANL   A,R2
35B 5B   ANL   A,R3
35C 5C   ANL   A,R4
35D 5D   ANL   A,R5
35E 5E   ANL   A,R6
35F 5F   ANL   A,R7
360 60   ADD   A,@R0
361 61   ADD   A,@R1
362 62   MOV   T,A
363 00   NOP
364 6465 JMP    365H
366 00   NOP
367 67   RRC   A
368 68   ADD   A,R0
369 69   ADD   A,R1
36A 6A   ADD   A,R2
36B 6B   ADD   A,R3
36C 6C   ADD   A,R4
36D 6D   ADD   A,R5
36E 6E   ADD   A,R6
36F 6F   ADD   A,R7
370 70   ADDC  A,@R0
371 71   ADDC  A,@R1
372 7273 JB3    73H
374 7475 CALL  375H
>_
```

39 EXAMPLE: DISPLAY THE RELATIVE ADDRESSING (OFFSET REGISTER) STATUS.

In this example, the current offset registers contain values of 298(&0), 198(&1), 2EC(&2) and C62(&3);

```
>OF <CR>
&0 = 298
&1 = 192
&2 = 2EC
&3 = C62
>_
```

43 COMMAND: PIN status

FUNCTION: Displays the current pin status.

SYNTAX: `>PI <cr>`

44 COMMAND: PIN specification

FUNCTION: Masks or un.masks the target system interruption signal when the incircuit mode is I1. Interruptions are ignored in the I1 mode and are valid in the I2 mode.

SYNTAX: `>PI spec=EN|DI <cr>`

TERMS: spec >>> specifies either;

RESET :reset the pin specification

SS :single step

INT :interruption or hold signal enable

EA :CPU/MEMORY separator

EN DI :enable or disable the interrupt or request.

40 EXAMPLE: DISABLE THE INPUT OF INTERRUPT SIGNALS.

This command example disables all the external interrupts in the I1 mode;

```
>I1 <CR>
>PI RESET=DI <CR>
>PI SS=DI <CR>
>PI INT=DI <CR>
>PI EA=DI <CR>
>_____
```

41 EXAMPLE: DISPLAY THE CURRENT INTERRUPT STATUS.

In this example, the current interrupt signal and masking status shows that the RESET input and INT signals are disabled (DI) and that the SS and EA input signals are enabled (EN). Input for all four signals is H(1);

```
>PI <CR>
In-Circuit Mode 1
RESET/ (DI) = H
SS/    (EN) = H
INT/   (DI) = H
EA     (EN) = H
>_____
```


45 COMMAND: PORT examination

FUNCTION: Examines the port contents.

SYNTAX: >P port <cr>

TERMS: port :[specifies port; BUS, P1 thru P7].

46 COMMAND: PORT examination and change.

FUNCTION: Examines and changes the port contents.

SYNTAX: >P port=data <cr>

TERMS: port :[specifies port; BUS, P1 thru P7].

data :new value of data [hex16/ASCII 8].

NOTES: After inputing the port value, this command enters the repeat mode where the following control characters are recognized;

return	(cr)	:display next byte (word) of data.
comma	,(cr)	:display same byte (word) of data.
caret	^(cr)	:display previous byte (word) data.
slash	/(cr)	:exit the PORT mode.

42 EXAMPLE: WRITE DATA INTO THE BUS PORT.

This command writes data successively into the BUS port. In this example, Data, 01(H), 02(H), 03(H), 04(H) and 05(H) are written into the BUS port in succession;

```
>P BUS=01,02,03,04,05 <CR>  
> _____
```

43 EXAMPLE: CHANGE THE PORT DATA IN UNITS OF BYTES.

In this example, 55(H) is written into and then read out of the P1 port. Then 32(H) is written again into the P1 port;

```
>P P1 <CR>  
P1 55:55, <CR>  
P1 55:32 <CR>
```

47 COMMAND: PRINT

FUNCTION: Outputs data sent from the ICD to the TERMINAL or HOST/AUX ports simultaneously. The printer must be connected to the HOST/AUX port for this command to be effective.

SYNTAX: >PR ON|OFF <cr>

TERMS: ON :output data to both TERMINAL and HOST/AUX ports.

OFF :output data to the console only.

44 EXAMPLE: PRINT THE CONTENTS DISPLAYED ON THE CONSOLE SCREEN.

```

>PR ON <CR>
>D O,L20 <CR>
  0 1 2 3 4 5 6 7 8 9 A B C D E F ASCII CODE
00 57 68 36 02 00 00 00 00 00 00 00 00 00 00 00 Who.....
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
>_

```


48 COMMAND: QUIT

FUNCTION: Returns control to the host system (valid when using ZICE software only).

SYNTAX: **>Q <cr>**

45 EXAMPLE: REBOOT TO THE HOST COMPUTER SYSTEM.

```
>Q <CR>  
A>_____
```


52 COMMAND: ROM status

FUNCTION: Displays the current ROM type.

SYNTAX: >RO <cr>

53 COMMAND: ROM specification

FUNCTION: Writes the contents of the ICD program memory into the ROM area of the 8748/49 through the ROM Writer, or reads the contents of the ROM area of the 8048/49 or 8748/49 into the ICD program memory.

SYNTAX: >RO E|P|R|V|W [,beg_addr][,end_addr]<cr>

TERMS: E|P|R|V|W >>> specifies ROM operation;

E :erase check
 P :program - writes after check
 R :read - reads data from ROM
 V :verify - compares ICD program memory & ROM
 W :write - writes to ROM

beg_addr :start address of read/verify/write operation [phy12].

end_addr :end address of read/verify/write operation [phy12].

NOTES: When "beg_addr" and "end_addr" are omitted, the address range is specified as 0 -- 1023 (8748/8048) or 0 -- 2047 (8749/8049).

ROM

- 48** EXAMPLE: WRITE THE CONTENTS OF ROM INTO THE EMULATION MEMORY.

This command lets the contents of ROM be written into the emulation memory for alteration and/or disassembling;

```
>RO R <CR>
-Reading
-Verifying
-OK
>__
```

- 49** EXAMPLE: WRITE THE MEMORY CONTENTS INTO ROM.

This command stores a program in the memory into ROM;

```
>RO W <CR>
-Writing
-Verifying
-OK
>__
```

- 50** EXAMPLE: WRITE AN INTEL HEX FILE IN THE HOST SYSTEM INTO ROM.

This command stores an Intel Hex file in the host system into ROM;

```
>L FILE!.HEX <CR>
000
>RO W <CR>
-Writing
-Verifying
-OK
>__
```

- 51** EXAMPLE: DISPLAY THE CURRENT GROUP AND ROM TYPE.

In this example, the current group is set to 1 and the ROM is 87XXH;

```
>RO <CR>
Group 1
87XXH
>__
```

54 COMMAND: SAVE

FUNCTION: Saves the contents of a specified range of memory to a disk file on the host system. (The file format is the same as the LOAD command.)

SYNTAX: `>SA[/T | /P | /A | /H][file],beg_addr,end_addr,user_addr[,mes]`

TERMS:

T :use the terminal port (ignore software handshake)

P :use the terminal port (perform software handshake)

A :use the auxiliary port

H :use the host port

file :name of the file to be used for saving the memory contents (available with ZICE software only).

beg_addr :first address to save [phy24].

end_addr :last address to save [phy24/L byte].

user_addr :beginning user program address [phy24].

mes :save message [hexdata/ASCII 32]. "message" is entered in this form; 'message',ØD - where ØD represents a line feed.

The ICD outputs the save message to the TERMINAL or AUX port prior to saving the object program (Intel format) if the /T or /A port is first specified. "filename" may be used for the user defined save message when software handshaking is performed and the /P or /H port is specified.

NOTES: Object File I/O Procedure

MODE	REMOTE	LOCAL
/T	x	x
/P	○	○
/A	x	x
/H	○	○
Default	○ (HOST)	x (TERMINAL)

- - The ICD inputs an Intel format file through handshaking. (See Section IV, COMMUNICATION PROTOCOL).
- x - The ICD inputs Intel format data only and does not perform handshaking. The data input ends upon receiving an end record (Intel format) or <EXT> (Control + C) through a specified port.

SAVE

52 EXAMPLE: SAVE THE MEMORY CONTENTS IN THE HOST SYSTEM.

In this example, the memory content (locations 0 -- FFF) is saved in the FILE2.HEX (Intel format) file of the host system;

```
>SA/H FILE2.HEX,0,FFF,0 <CR>  
>_____
```

53 EXAMPLE: SAVE THE MEMORY CONTENTS IN THE HOST SYSTEM THROUGH ZICE.

When ZICE is being used (REMOTE mode), this command saves the memory contents. In this example, the memory contents (locations 0 -- FFF(H)) are saved in the FILE2.HEX (Intel format) file of the host system;

```
>SA FILE2.HEX,0,FFF,0 <CR>  
>_____
```

54 EXAMPLE: DISPLAY THE CONTENTS OF THE MEMORY IN INTEL HEX FORMAT.

This command displays the memory content on the console. The TERMINAL port (T) designates I/O from the console. In this example, the program memory contents of 0 -- 100 is displayed in Intel Hex format;

```
>SA/T,0,100,0 <CR>  
:2000000000102030405060708090A0B0C0D0E0F10111213141516171819A1B1C1D1E1FF0  
:20002000202122232425262728292A2B2C2D2E2F303132333435363738393A3B3C3D3E3FD0  
:20004000404142434445464748494A4B4C4D4E4F505152535455565758595A5B5C5D5E5B0  
:20006000606162636465666768696A6B6C6D6E6F707172737475767778797A7B7C7D7E790  
:2008008081828384858687888898A8B8C8D8E8F909192939495969798999A9B9C9D9E9F70  
:200A000A0A1A2A3A4A5A6A&A8AAAABACADAEAFAB0B1B2B3B4B5B6B7B8B9BABBBCBDBBEF50  
:2000C000C0C1C2C3C4C5C6C7C8C9CACBCCDCDCFC0D1D2D3D4D5D6D7D8D9DADBBDCDEDF30  
:2000E000E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEFF0F1F2F3F4F5F6F7F8F9FAFBFCFDFEF10  
:0101000000FE  
:00000001FF
```

55 COMMAND: SEARCH

FUNCTION: Searches for the memory contents and displays the matched or unmatched address.

SYNTAX: `>S[/W][/D][memdata]beg_addr,end_addr[,data]<cr>`

TERMS: /W :search for memory contents on a Word basis
(default is a byte basis).

/D :search for unmatching data (default is a
search for matched data).

memdata >>> specifies type of data memory;

R: :internal (default)

D: :external

P: :program

beg_addr :address to begin the search [phy12].

end_addr :address to end the search [phy12/L byte].

data :data for which to search [ASCII 32, hexdata].

55 EXAMPLE: PERFORM A SEARCH WHICH WILL DISPLAY THE LOCATION
ADDRESSES OF THE DATA SPECIFIED.

In this example, the beginning address is 0 and 100(H) bytes of memory are to be searched. The object of the search is the ASCII character string 'ERROR'. The addresses that contain data equal to 'ERROR' are the result of the search (048);

```
>S P:0,L100,'ERROR' <CR>
048
>__
```


56 COMMAND: SUPERVISOR status

FUNCTION: Displays the current Supervisor state.

SYNTAX: >SU <cr>

57 COMMAND: SUPERVISOR specification

FUNCTION: Specifies to use a breakpoint as a supervisor call. A supervisor call is used to output data, such as message, when a breakpoint specified as a supervisor call is passed.

SYNTAX: >SU/ C|7|U ON|OFF <cr>

TERMS: C :use hardware break C as a supervisor call.
 7 :use software break 7 as a supervisor call.
 U :user software break as a supervisor call.
 ON :use a supervisor call.
 OFF :do not use a supervisor call.

56 EXAMPLE: OUTPUT A MESSAGE TO THE ICD FROM THE USER PROGRAM.

In this example, a software break (00) is set and then turned ON (>SU/U ON). When the user program is executed, a software break occurs because NOP (00) is executed. Control is then returned to the user program after the I/O operation of the ICD SIO is completed (when the Supervisor command is set to ON);

```
>A 100 <CR>
100 MOV R1,#10H <CR>
102 MOV R7,#02H <CR>
104 MOV A,R1 <CR>
105 MOVP A,@A <CR>
106 NOP <CR>
107 INC R1 <CR>
108 ORL A,#0 <CR>
10A JNZ 2 <CR>
10C <CR>
>B S=EN <CR>
>B S=00 <CR>
>SU/U ON <CR>
>G 100 <CR>
```

```
HELP ! ! !
HELP ! ! !
```

SUPERVISOR

NOTES: See the FUNCTION CODE table.

FUNCTION CODE explanation;

- 1) Function Code 00, 10: 1 CHARACTER INPUT FROM A PORT;

Input parameter

Register R7: 00H specifies TERMINAL port.
10H specifies HOST/AUX port.

End status

Register A: Input characters.

The characters input through the specified port are stored in register A. If there is no input through the port when called, return to the target program is not effected until input is given.

- 2) Function Code 01, 11: FETCH PORT INPUT STATUS SIGNAL;

Input parameter

Register R7: 01H Specifies TERMINAL port.
11H Specifies HOST/AUX port.

End status

Register A: 00H No input data.
FFH Input data exists.

Displays information on the presence or absence of input data from the specified port.

- 3) Function Code 02, 12: 1 CHARACTER OUTPUT TO PORT;

Input parameter

Register R7: 02H Specifies TERMINAL port.
12H Specifies HOST/AUX port.

End status

Register A: Output character.

A character stored in register A outputs to the specified port. If the output of the previous data has not been completed when called, return to the target program does not occur until the output is completed.

- 4) Function Code 03, 13: FETCH PORT OUTPUT STATUS SIGNAL;

Input parameter

Register R7: 00H Specifies TERMINAL port.
13H Specifies HOST/AUX port.

End status

Register A: 00H Output is not completed.

Displays information on whether the output from the specified port has been completed.

FUNCTION	FUNCTION CODE	DATA OUT	DATA IN
	R 7 - reg	A - reg	
TERMINAL Port data in	0 0	-	Rec data
HOST/AUX Port data in	1 0	-	Rec data
TERMINAL Port input status read	0 1	-	Input status
HOST/AUX Port input status read	1 1	-	Input status
TERMINAL Port data out	0 2	Output data	-
HOST/AUX Port data out	1 2	Output data	-
TERMINAL Port output status read	0 3	-	Output status
HOST/AUX Port output status read	1 3	-	Output status

Function Code Table

58 COMMAND: TRACE status

FUNCTION: Displays the current Trace status.

SYNTAX: >T <cr>

59 COMMAND: TRACE designation

FUNCTION: Makes the current trace mode valid (ON), invalid (OFF) or clears (CLR) the trace setting.

SYNTAX : >T ON|OFF|CLR <cr>

60 COMMAND: TRACE specification

FUNCTION: Sets the ICD trace mode.

SYNTAX: >T [/S] A|J [,beg_addr] [,end_addr] <cr>

TERMS:

- S :specifies single step mode. In this mode, the All or Jump trace is executed each time the space bar or return (cr) is pressed.
- A :all commands are traced and displayed.
- J :jump trace (only the branch command is traced and displayed).
- beg_addr :beginning trace address [hex].
- end_addr :ending trace address [hex/Lbyte] (default=0).

TRACE

57 EXAMPLE: TRACE ALL INSTRUCTIONS.

This command traces all the instructions of the user program and displays register contents after execution of the program. To temporarily stop the execution or to restart the program, enter <SPACE>. To interrupt the program, press the <ESC> key.

```
>T A <CR>
>G 5A <CR>
  PC MC
05A 5A ANL A,R2 PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP
05B 5B ANL A,R3 08 00 55 68 36 02 00 00 00 00 00
05C 5C ANL A,R4 .. .. .. .. .. .. .. .. ..
05D 5D ANL A,R5 .. .. .. .. .. .. .. .. ..
05E 5E ANL A,R6 .. .. .. .. .. .. .. .. ..
05F 5F ANL A,R7 .. .. .. .. .. .. .. .. ..
060 60 ADD A,@R0 .. 55 .. .. .. .. .. .. .. <ESC>
>_____
```

58 EXAMPLE: TRACE ONLY BRANCH INSTRUCTIONS.

This command displays the branch instruction and the contents of registers when the JUMP trace mode is set (>TJ) and the program is executed by the GO command. To temporarily stop, restart, and interrupt the program execution, use the same procedure as for the trace ALL instruction;

```
>TJ <CR>
>G <CR>
  PC MC PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP
064 6465 JMP 365H 08 BD 55 68 36 02 00 00 00 00 00
064 6465 JMP 365H 08 BD 66 68 45 00 00 02 00 00 00
>_____
```

59 EXAMPLE: USE THE SINGLE STEP TRACE MODE.

This command performs a single step trace each time there is a key input from the console. Entering a return (cr) causes the program to proceed to the next instruction. ESC returns control to the command mode;

```

>T/S A <CR>
>G 4E <CR>
  PC MC          ORL      A,R6          PSW  A R0 R1 R2 R3 R4 R5 R6 R7 SP
04E 4E          ORL      A,R6          08 00 55 68 36 02 00 00 00 00 00 <CR>
04F 4F          ORL      A,R7          .. .. .. .. .. .. .. .. .. .. <CR>
050 50          ANL      A,@R0        .. .. .. .. .. .. .. .. .. .. <CR>
051 51          ANL      A,@R1        .. .. .. .. .. .. .. .. .. .. <CR>
052 5253       JB2      53H          .. .. .. .. .. .. .. .. .. .. <ESC>
> _____

```

60 EXAMPLE: DISPLAY THE CURRENT TRACE STATUS.

In this example, the current trace mode is the ALL trace, and the range of the instructions to be traced is 0 -- FFF(H);

```

>T <CR>
(ON) ALL 000-FFF
> _____

```

61 EXAMPLE: CLEAR THE TRACE MODE.

This command clears the trace mode and the program now runs in real-time when the GO command is given;

```

>T CLR <CR>
>T <CR>
Trace is clear
> _____

```


61 COMMAND: USER

FUNCTION: This command allows the console terminal (connected to the ICD TERMINAL port) to be used as the terminal for the host system. When this command is entered, the ICD does not treat any code, other than the terminator code coming through the TERMINAL port, as the ICD command. The codes which the ICD receives through the HOST/AUX port are all sent to the TERMINAL port. To switch the user terminal of the host computer system back to the console terminal of the ICD, enter the terminator code. This command will work when the ICD is in the LOCAL mode only.

SYNTAX: >U code <cr>

TERMS: code :terminator code

Note: <ESC>, <NAK>, <SP>, <BS> and <cr> cannot be used as terminator codes.

62 EXAMPLE: USE THE CONSOLE CONNECTED TO THE TERMINAL PORT AS THE CONSOLE FOR THE HOST SYSTEM.

```
>U ! <CR>
A>XICE <CR>
>HOST ON <CR>
ZOS XICE v1.0
>_____
```


62 COMMAND: VERIFY

FUNCTION: Compares the object in the Intel format (from the host computer file or specified port) with the ICD memory contents.

SYNTAX: >V[/T | /P | /A | /H][filename] [,bias] [,message]<cr>

TERMS:

T :use the terminal port (ignore software handshake)

P :use the terminal port (perform software handshake)

A :use the auxiliary port

H :use the host port

filename :object name to compare (valid with ZICE software only).

bias :address offset to apply to the object file being compared [hexdata].

message :verify message [ASCII32/hexdata]. "message" is entered in this form; 'message',ØD - where ØD represents a line feed.

The ICD outputs the verify message to the TERMINAL or AUX port prior to verifying the object program (Intel format) if the /T or /A port is first specified. "filename" may be defined for the user defined verify message when software handshaking is performed and the /P or /H port is specified.

NOTES: Object File I/O Procedure

MODE	REMOTE	LOCAL
/T	×	×
/P	○	○
/A	×	×
/H	○	○
Default	○ (HOST)	× (TERMINAL)

- - The ICD inputs an Intel format file through handshaking. (See Section IV, COMMUNICATION PROTOCOL).
- × - The ICD inputs Intel format data only and does not perform handshaking. The data input ends upon receiving an end record (Intel format) or <EXT> (Control + C) through a specified port.

VERIFY

63 EXAMPLE: COMPARE THE CONTENTS OF THE MEMORY AND AN INTEL FORMAT FILE IN THE HOST SYSTEM.

When the host system is connected to the HOST/AUX port (LOCAL mode) and it supports the protocol at the time of ICD verification, this command compares the contents of a file from the host system with the contents of the memory. In this example, the contents of memory are compared with the contents of the file, FILE3.HEX (Intel format) in the host system;

```
>V/H FILE3.HEX <CR>  
Addr M O
```

```
100  
>__
```

64 EXAMPLE: COMPARE THE CONTENTS OF A FILE (INTEL FORMAT) READ FROM ZICE IN THE HOST SYSTEM WITH THE MEMORY CONTENTS. In this example, the contents of the host system file, FILE3.HEX (Intel format) are compared with the memory contents;

```
>V FILE3.HEX <CR>  
Addr M O
```

```
100  
>__
```

NOTE: "Adrs" is the memory address, "M" is the memory, and "O" is the object file.

63 COMMAND: ZICE

FUNCTION: The ZICE command is used exclusively with Zice software. ZICE expands the commands not built into the ICD-178. When the ICD is used in the remote operation (host computer) the ZICE utility software is required for controlling the ICD.

SYNTAX: >Z (Z command & parameter) <cr>

TERMS: Z commands include;

- ZB :batch processing. Executes commands of the filename. BAT file in the host system by batch processing.
- ZC :comment. Inserts a comment in a batch file of the host system or in the middle of repeat processing and outputs it to the console.
- ZD :directory. Displays a directory of the *.BAT or *.HEX file in the host system by directory format.
- ZH :help. Displays command list.
- ZM :menu. Enter data by command menu method.
- ZP :print. Outputs the contents displayed on the console screen to a list device.
- ZR :repeat. Repeats a specified command.
- ZW :wait. Places a temporary pause in a repeat or batch subcommand.

65 EXAMPLE: OUTPUT THE CONTENTS OF THE CONSOLE DISPLAY ONTO A LIST DEVICE;

In this example, all commands executed after the "ZP ON" command and before the "ZP OFF" command, output the results to both console and printer;

```
>ZP,ON <CR>
>
>D 0,L30 <CR>
  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  ASCII CODE
00  55 68 36 02 00 00 00 00 00 00 00 00 00 00 00  Uh6.....
10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
>ZP,OFF <CR>
>_____
```


SECTION III

TECHNICAL REFERENCES

This section contains specific technical information on the ICD-178 for 8048 and is intended as a supplement for Sections I and II of the USER'S MANUAL. It is not necessary to read this section before using the ICD-178, however, you may find the information in this section useful if you require a detailed analysis on a particular emulator feature.

3.1 - EMULATION SELECT SWITCH

This chapter describes the various settings and functions of the 4-bit emulation switch. This switch must be set correctly for proper emulation.

3.2 - SERIAL INTERFACE

This chapter describes the components and functions of the SIO S-791 Serial Interface module, including specifications and switch settings. Included in this chapter are pin/signal designations for the TERMINAL and HOST/AUX; RS-232C interface, current loop interface, and TTL interface.

3.3 - MEMORY EMULATION

This chapter describes the ICD emulation memory which is used for downloading object files, altering the memory contents, etc. This chapter discusses each of the four types of memory area; Read/Write, Read Only, User and Non (No).

3.4 - CPU EMULATION

This chapter contains information on CPU emulation, including a detailed analysis of the control signals of the 8048 processor, including a discussion on the differences between the 8048 microcomputer unit (MCU) and the ICD's internal processor.

3.5 - ROM (EPROM) PROGRAMMER

This chapter describes the specifications of the EPROM programmer and the five types of EPROM processing that are available.

3.6 - REAL-TIME TRACE

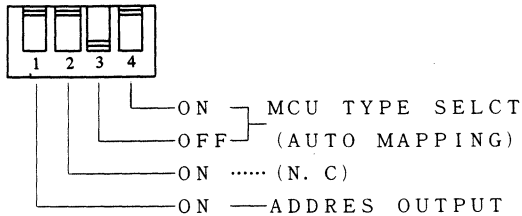
This chapter contains information on the theory and operation of the real-time trace buffer, including a detailed examination of the six trigger modes that are selected by the "HISTORY" command.

3.7 - PROBES

This chapter describes the function of the Map Control probe and Emulation Qualify probe (both used with the External Break cable), and the Event Trigger probe.

3.1 — EMULATION SELECT SWITCH

The Emulation Select switch effects the machine cycle operation of the target system, and selects the internal program capacity for the corresponding MCU.



Bit 3, 4 setting these bits (see below) effects the selection of the Object MCU and corresponding ICD internal program memory for the various processor families.

Bit 2 not connected.

Bit 1 OFF. This interfaces the ICD to memory devices that have slow access times.

Bit 1 ... ON. This interfaces the ICD to memory devices that have normal access times.

The ICD classifies the MCUs of the 8048 family into four groups according to their internal program memory capacity.

Bit 3	Bit 4	Group	Target MCU	Internal program memory
OFF	OFF	Group 0	8535, 8039, 8040	0KB
OFF	ON	Group 1	8048, 8748	1KB
ON	OFF	Group 2	8049, 8749	2KB
ON	ON	Group 3	8050	4KB

During emulation, the ICD control processor checks the status of the switches and takes the appropriate action below;

- 1) The in-circuit mode 2 (partial), determines the internal and external boundary of the program memory space.
- 2) When read/write is performed by ROM command, the address parameter value is checked for its validity. If the address parameter is missing, the range of read/write is determined automatically.
- 3) When the status of the IN-CIRCUIT, MAP and ROM command is displayed, the Group number is also displayed.

3.2 — SERIAL INTERFACE

3.2.1 INTRODUCTION

The Serial Interface module (SIO S-791) controls communication between the ICD-178 and various external devices through the TERMINAL and HOST/AUX terminals. This chapter shows the components of the S-791 module and the input/output signals that transmit through the TERMINAL and HOST/AUX ports (RS232, TTL or Current Loop interface). This chapter also discusses the four serial interface control signals which effect various terminals and host computer systems.

3.2.2 SERIAL INTERFACE SPECIFICATIONS

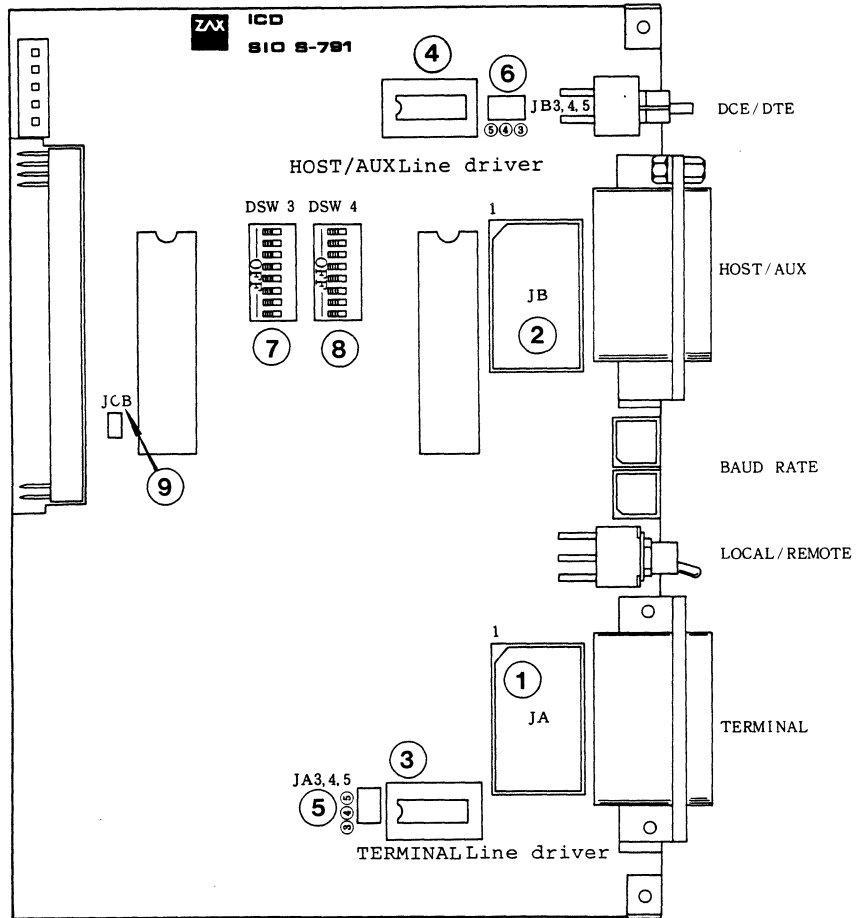
Interface channels:	2 channels, TERMINAL and HOST/AUX
Interface signals:	RS232C (standard) 20ma Current Loop TTL (Current Loop and TTL interface signals are used by changing the jumper sockets JA or JB or the line driver IC.)
Communication mode:	Full-duplex, start/stop
Transmission format:	start bit - 1 data bits - 7,8(factory setting) stop bit - 1,2(factory setting) parity bit- odd, even or no parity(factory setting) (data, stop and parity bits are set with the DIP switches DSW3 and DSW4.)
Transmission code:	ASCII codes
Baud rates:	19200, 960, 4800, 2400, 2000, 1800, 1200, 600, 300, 200, 150, 134.6, 110, 75. (Baud rates are set by the Baud rate select switch located on SIO S-791 module.)

3.2.3 SIO S-791 MODULE COMPONENTS

- 1 JA socket - the JA socket is used specifically for the TERMINAL port. This socket enables the current loop or TTL signals depending on the jumper connections (see the current loop and TTL interface diagrams).
- 2 JB socket - the JB socket is used specifically for the HOST/AUX port. This socket enables the current loop or TTL signals depending on the jumper connections (see the current loop and TTL interface diagrams).

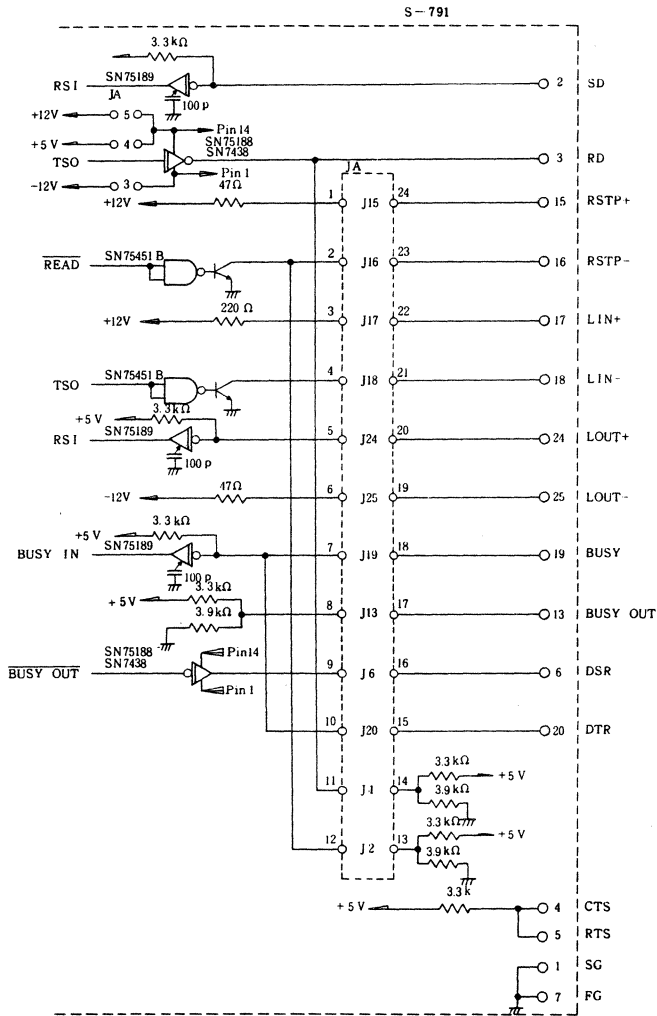
NOTE: When one group of signals are enabled (RS232, TTL or current loop) the other two groups are disabled.

- 3,4 TERMINAL and HOST/AUX line drivers - SN75188 is used with the RS232 and current loop interface. To use TTL interface, replace SN75188 with SN7438 and change the JA3/4/5 and JB3/4/5 jumpers.
- 3 TERMINAL line driver.
- 4 HOST/AUX line driver.
- 5 JA3/4/5 power supply jumpers - used specifically for the TERMINAL line driver IC. JA3 and JA5 supply +12V to the SN75188. JA4 supplies +5V to SN7438.
- 6 JB3/4/5 power supply jumpers - used specifically for the HOST/AUX line driver IC. JA3 and JA5 supply +12V to the SN75188. JA4 supplies +5V to SN7438.
- 7,8 DSW3 & DSW4 Transmission format switches - these switches are used to set data and stop bits for the TERMINAL (DSW3) and HOST/AUX (DSW4) ports.
- 7 DSW3 Transmission format switch (TERMINAL port)
- 8 DSW4 Transmission format switch (HOST/AUX port)
- 9 JCB Console break jumper socket - the JCB socket works the same as the MONITOR BREAK switch. (Factory setting is ON.)



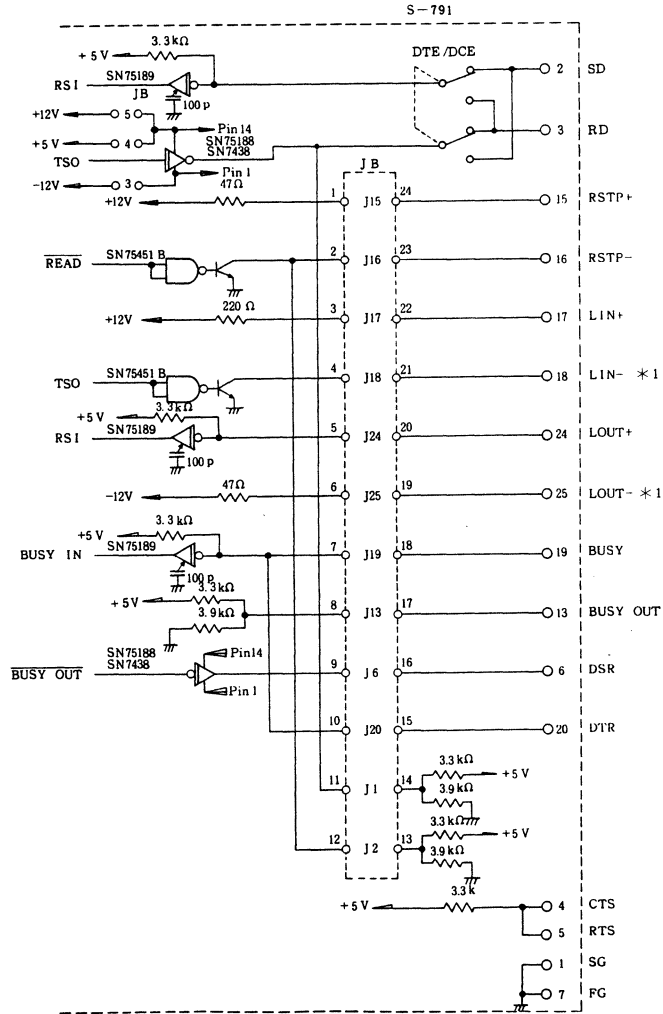
SIO S-791 Module

(1) TERMINAL



SIO S-791 Interface Block Diagram (Terminal Port)

(2) HOST/AUX



SIO S-791 Interface Block Diagram (HOST/AUX Port)

3.2.5 RS-232C INTERFACE

PIN NO.	SIGNAL NAME	MEANING	IN/OUT	JA No.
1	FG	FRAME GROUND		SN 75188N ^{*3}
2	SD	SEND DATA	IN	
3	RD	RECEIVE DATA	OUT	
4	RTS	REQUEST	IN ^{*1}	J6, J20 ^{*2} J6, J20
5	CTS	CLEAR TO SEND	OUT ^{*1}	
6	DSR	DATA SEND READY	OUT	
20	DTR	DATA TERMINAL READY	IN	
7	SG	SIGNAL GROUND		

RS-232C Interface (TERMINAL Port) Signals

- *1 CTS and RTS are looped back (null modem) within the ICD and pulled up to +5V.
- *2 DTR and DSR are looped back (null modem) within the ICD by connecting JA6 and 20 (pins 15 & 16) together. DTR is used as the BUSY signal for an ICD terminal by connecting JA20. DSR may be used as the BUSY signal by connecting JB6.

When JA6 is connected, the loopback jumper between JA6 and 20 (pins 15 & 16) is prohibited. The standard connection = JA6/20 and JA20.

- *3 SN75188N is installed for the TERMINAL line driver. In this configuration, JB3 and JB5 are connected with a jumper plug to supply $\pm 12V$ to SN75188N. JA4 may NOT be connected. The standard connection = JB3 and JB5.

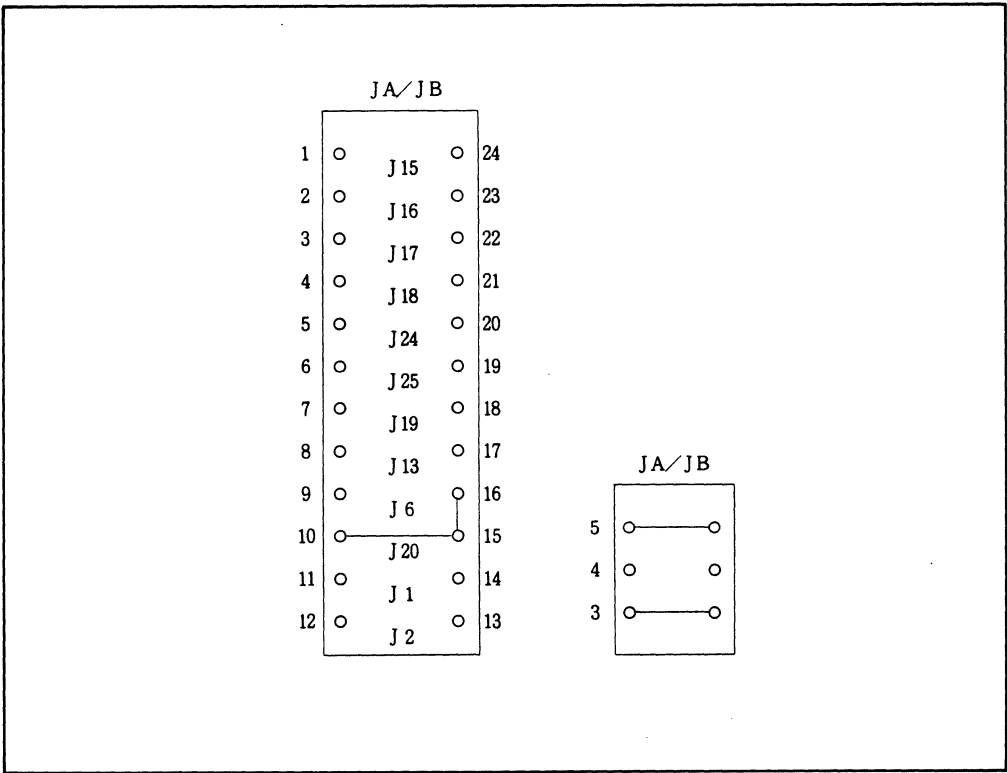
PIN NO.	SIGNAL NAME	MEANING	IN/OUT	JA No.
1	FG	FRAME GROUND	*1	SN 75188N *4
2	SD	SEND DATA	OUT(IN)	
3	RD	RECEIVE DATA	IN(OUT)	J6,20 *3 J6,20
4	RTS *2	REQUEST TO SEND	OUT(IN)	
5	CTS	CLEAR TO SEND	IN(OUT)	
6	DSR	DATA SEND READY	IN(OUT)	
20	DTR	DATA TERMINAL READY	OUT(IN)	
7	SG	SIGNAL GROUND		

RS-232C Interface (HOST/AUX Port) Signals

- *1 Values in () are assumed when the DCE/DTE select switch is set to DCE.
- *2 CTS and RTS are looped back (null modem) within the ICD by connecting JB6 and 20 (pins 15 & 16) together.
- *3 DTR and DSR are looped back (null modem) within the ICD by connecting JB6 and 20 (pins 15 & 16) together.

DTR is used as the BUSY signal for the ICD terminal by connecting JB20. DSR may also be used as the BUSY signal for the ICD terminal by connecting JB6. When JB6 is connected, the loopback jumper between JB6 and 20 (pins 15 & 16) is prohibited. Standard connection = JB6/20 and JB20.

- *4 SN75188N is installed for the HOST/AUX line driver. In this configuration, JB3 and JB5 are connected with a jumper plug to supply $\pm 12V$ to SN75188N. JB4 may NOT be connected. Standard connection = JB3 and JB5.



RS-232C Interface Jumper Settings

3.2.6 CURRENT LOOP INTERFACE

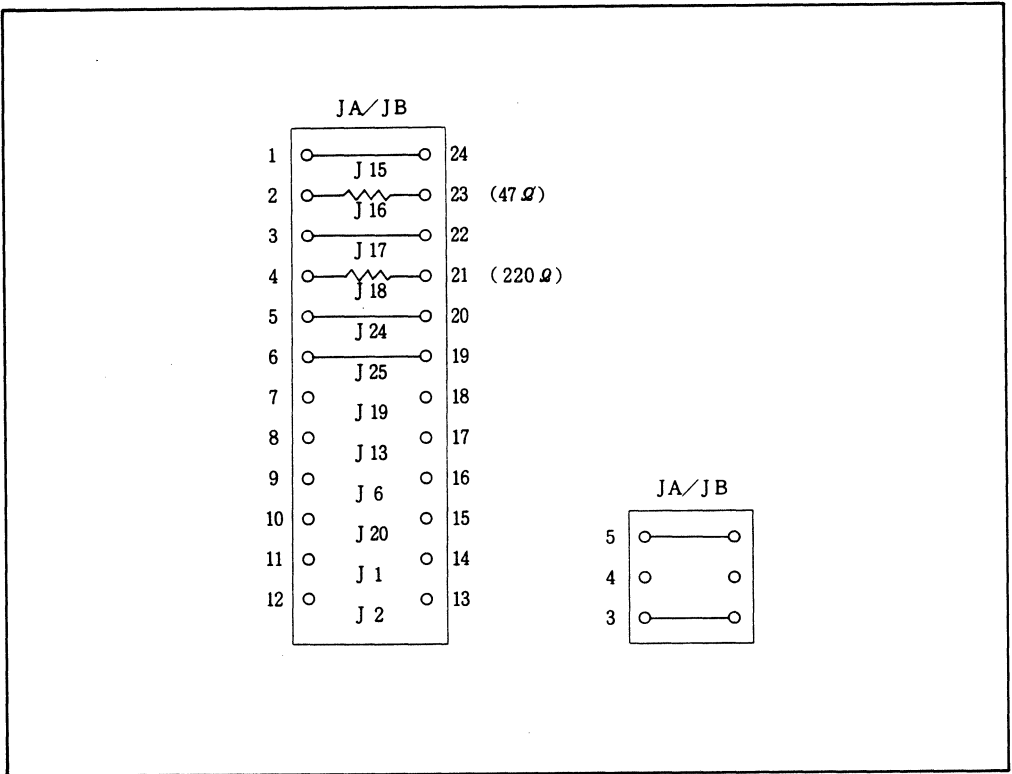
PIN NO.	SIGNAL NAME	MEANING	IN/OUT	JA No.
24	LOUT+	CURRENT LOOP OUT(+)	IN *1	J24
25	LOUT-	CURRENT LOOP OUT(-)	IN *1	J25
17	LIN+	CURRENT LOOP IN(+)	OUT *2	J17
18	LIN-	CURRENT LOOP IN(-)	OUT	J18/220 *3
15	RSTP+	READER STEP(+)	OUT *2	J15
16	RSTP-	READER STEP(-)	OUT	J16 *4

Current Loop Interface (TERMINAL and HOST/AUX Ports) Signals

- *1 Current source pin used for current loop input signals, -12V pull-down.
- *2 Current source pin used for current loop input signals, +12V pull-up.
- *3 Jumpers JA and JB18 are connected with a current limiting resistance of 220 ohms-1/4W -or- the resistance is altered to suit the associated circuit.
- *4 Jumpers JA and JB18 are connected with a current limiting resistance of 47 ohms-1/4W.

With a current loop interface, the HOST/AUX port is used only at the DCE setting (DCE/DTE select switch).

The baud rates over the current loop interface must NOT exceed 600bps.



Current Loop Interface Jumper Settings

3.2.7 TTL INTERFACE

PIN NO.	SIGNAL NAME	MEANING	IN/OUT	JA No.
1	FG	FRAME GROUND		
2	SD	SEND DATA	IN	SN 7438 ^{*2}
3	RD	RECEIVE DATA	OUT	
19	BUSY	BUSY INPUT	IN	J19
13	BUSYOUT	BUSY OUTPUT	OUT	J13, 6 ^{*1}
16	RSTP	READER STEP	OUT	J16
7	SG	SIGNAL GROUND		

TTL Interface (TERMINAL Port) Signals

*1 The BUSYOUT signal becomes a TTL level signal when JA6 and JA13 (pins 8 & 9) are joined together and then connected to JA13.

*2 The Terminal line driver contains an SN7438N. JA4 is connected with a jumper plug to supply +5V to SN7438N. JA3 and 5 must NOT be jumped.

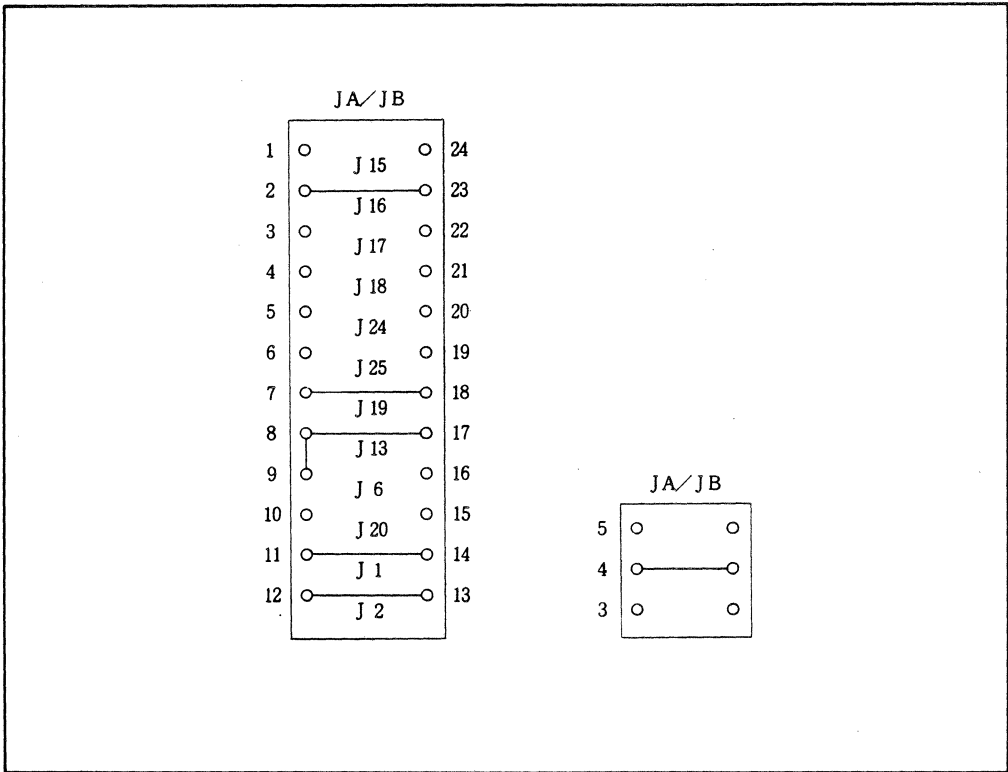
CAUTION: JA20 MAY NOT BE CONNECTED WHERE JA19 IS CONNECTED.

PIN NO.	SIGNAL NAME	MEANING	IN/OUT	JA No.
1	FG	FRAME GROUND		
2	SD	SEND DATA	OUT(IN)	SN 7438*3
3	RD	RECEIVE DATA	IN(OUT)	
19	BUSY	BUSY INPUT	IN	J19
13	BUSYOUT	BUSY OUTPUT	OUT	J13, 6*2
16	RSTP	READER STEP	OUT	J16
7	SG	SIGNAL GROUND		

TTL Interface (HOST/AUX Port) Signals

- *1 When the DCE/DTE switch is set to DCE, signals are routed in the directions indicated in ().
- *2 The BUSYOUT signal becomes a TTL signal when JB6 and JB 13 (pins 8 & 9) are connected, then outputs to the pin when JB13 is connected.
- *3 The HOST/AUX line driver contains SN7438N. JB4 is connected with a jumper plug to supply +5V to SN7438N. JB3 and 5 must NOT be connected together.

CAUTION: JB20 MAY NOT BE CONNECTED WHERE JB19 IS CONNECTED.



TTL Interface Jumper Settings

3.2.8 SERIAL INTERFACE CONTROL SIGNALS

The Serial Interface control signals effect the transmission between the ICD and various external communication devices. These signals include; XON/XOFF, BUSY, DTR, BUSYOUT, DSR, and RSTP.

XON and XOFF protocol

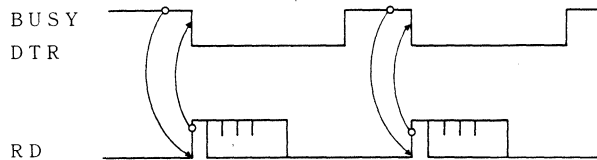
When the host computer or terminal cannot receive data sent from the ICD due to the ICD's high baud rate, the data output can be controlled with XON and XOFF.

XOFF ... DC3 (CTR-S: 13H)

XON ... DC1 (CTR-Q: 11H)

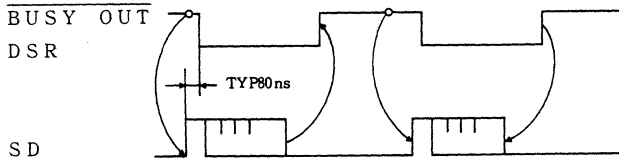
The host computer or terminal sends XOFF to the ICD before the reception buffer overruns. After XOFF is sent, the host computer or terminal sends XON to the ICD if the reception buffer is ready. The ICD then resumes data transmission after receiving XON.

BUSY and DTR (input signals)



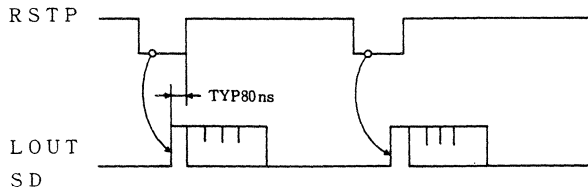
The ICD can stop data transmission if a BUSY signal is detected from a low-speed terminal. Normally, the terminal sets the BUSY signal low, from the leading edge of the receive data signal (RD) starting bit to the completion of data processing. The ICD suspends its data transmission to the terminal as long as the BUSY signal is low.

BUSYOUT and DSR (output signals)



When the host computer system sends data at a higher speed than the internal ICD monitor processor can accept, the BUSYOUT signal of the ICD must be monitored. The ICD sets the BUSYOUT signal to low until the ICD monitor reads transmitted data (SD) from the host system. During this time, the host computer waits for data transmission to the ICD.

RSTP (output signal)

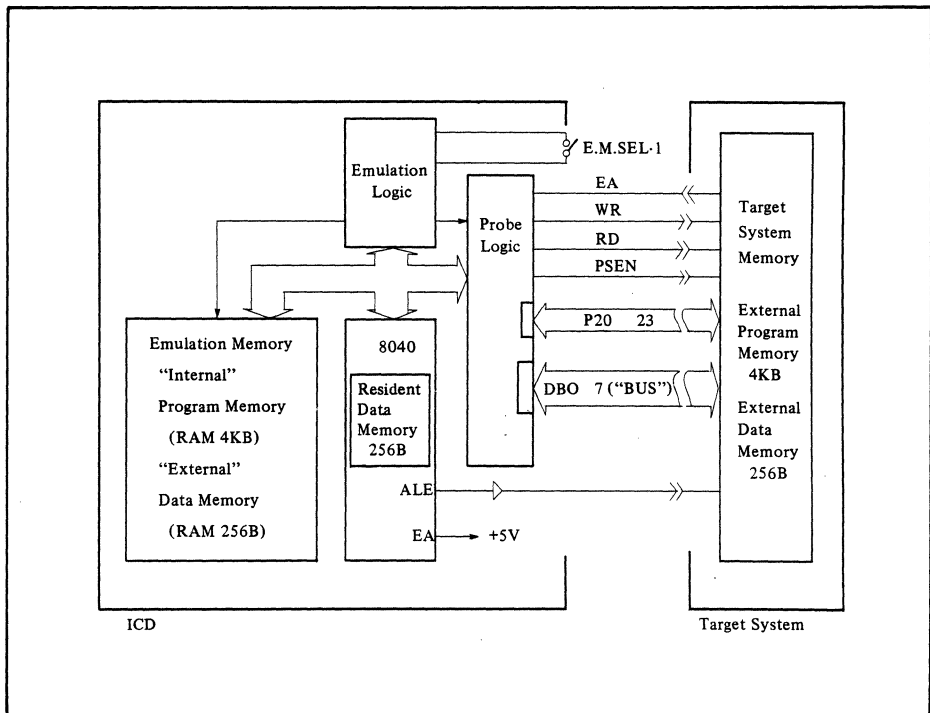


The ICD can transmit the RSTP signal to terminals that require a step signal for each data transmission. The ICD sets RSTP to low when data reading is requested. It then returns RSTP to high upon detection of the start bit signal from the terminal transmitted data.

3.3 MEMORY EMULATION

3.3.1 ICD EMULATION MEMORY

The ICD incorporates 4K-bytes of high-speed RAM which is called ICD program (emulation) memory. This memory can be used for downloading object files, altering the memory contents and as a read/write buffer for the ROM (EPROM) programmer. The ICD also contains "External Data Memory" (XDM) which allows programs to be executed without the ICD being connected to a target system.

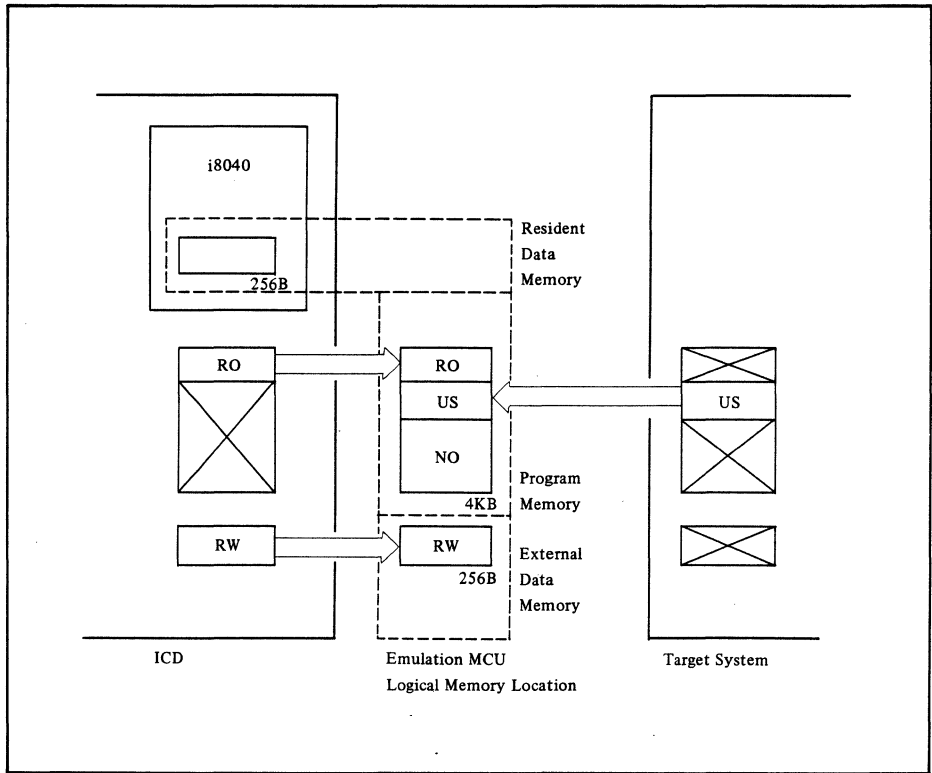


ICD/Target System Memory Interface Diagram

3.3.2 MEMORY MAPPING

When you are performing emulation in the I1 mode, it is possible to use the target system memory (mapped in units of 256 bytes) or ICD emulation memory. These conditions are set by the "MAP" command. There are four types of memory blocks to choose from; READ/WRITE, READ ONLY, USER (TARGET) and NON (NO MEMORY).

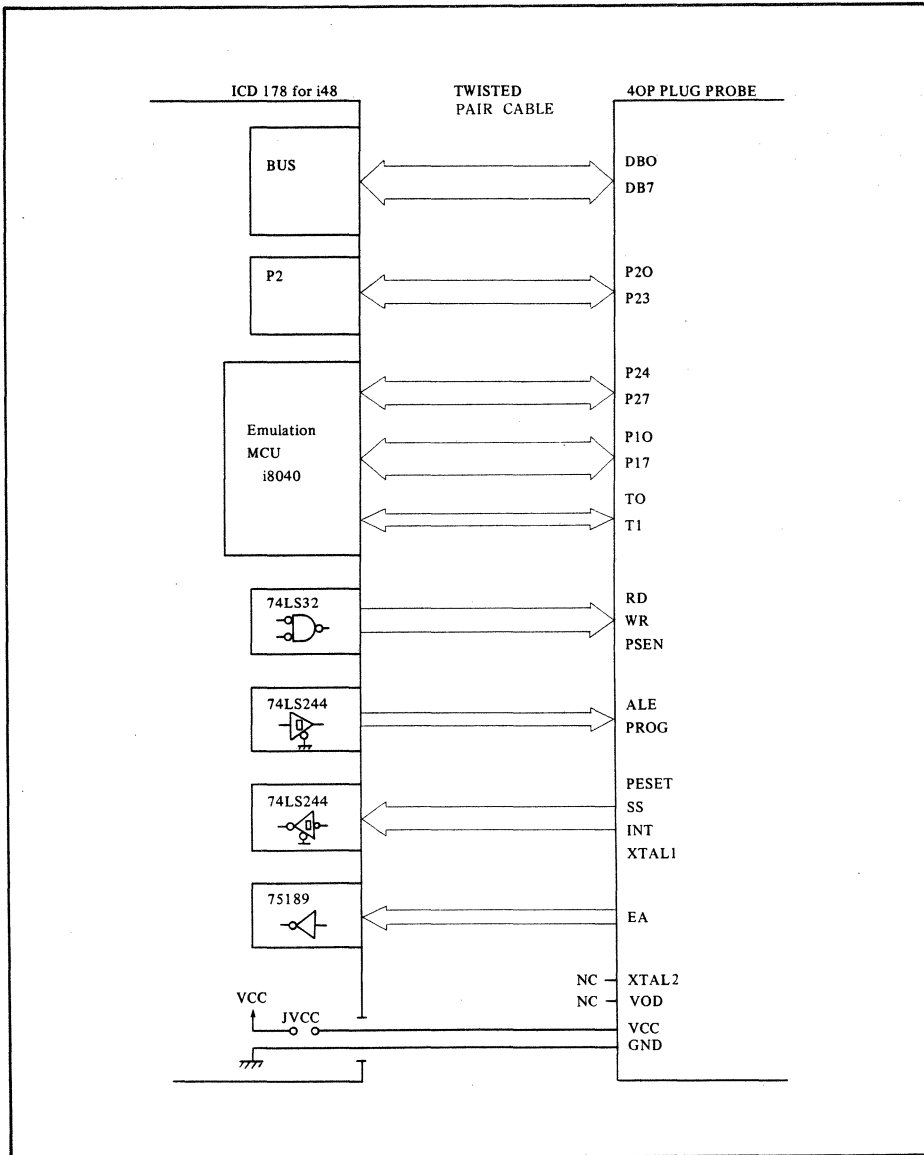
- READ/WRITE: This specification is valid for the External Data Memory (XDM) only. (You cannot use it with the ICD emulation memory.) Memory blocks having this specification can be accessed by read or write.
- READ ONLY: This specification uses the ICD internal emulator memory in place of the target system's memory. You cannot write into this memory area during emulation, however, it can be used as if it were ROM.
- USER: This specification uses the memory mounted in the target system.
- NON: This specification is for a memory area mapped as "non-existent." This means a place in the ICD where no memory resides. If the target program accesses this memory area during emulation, a break in the program will occur.



ICD/Target System Memory Mapping Diagram

3.4 CPU EMULATION

3.4.1 ICD/TARGET SYSTEM INTERFACE



3.4.2 MCU/ICD DEVIATIONS

The ICD-178 will not always perform the same actions as the Microcomputer Unit (MCU). It is important to note;

1) Internal data memory

The standard MCU mounted in the ICD is 8040. This chip has 256 bytes of internal data memory.

2) AC/DC characteristics

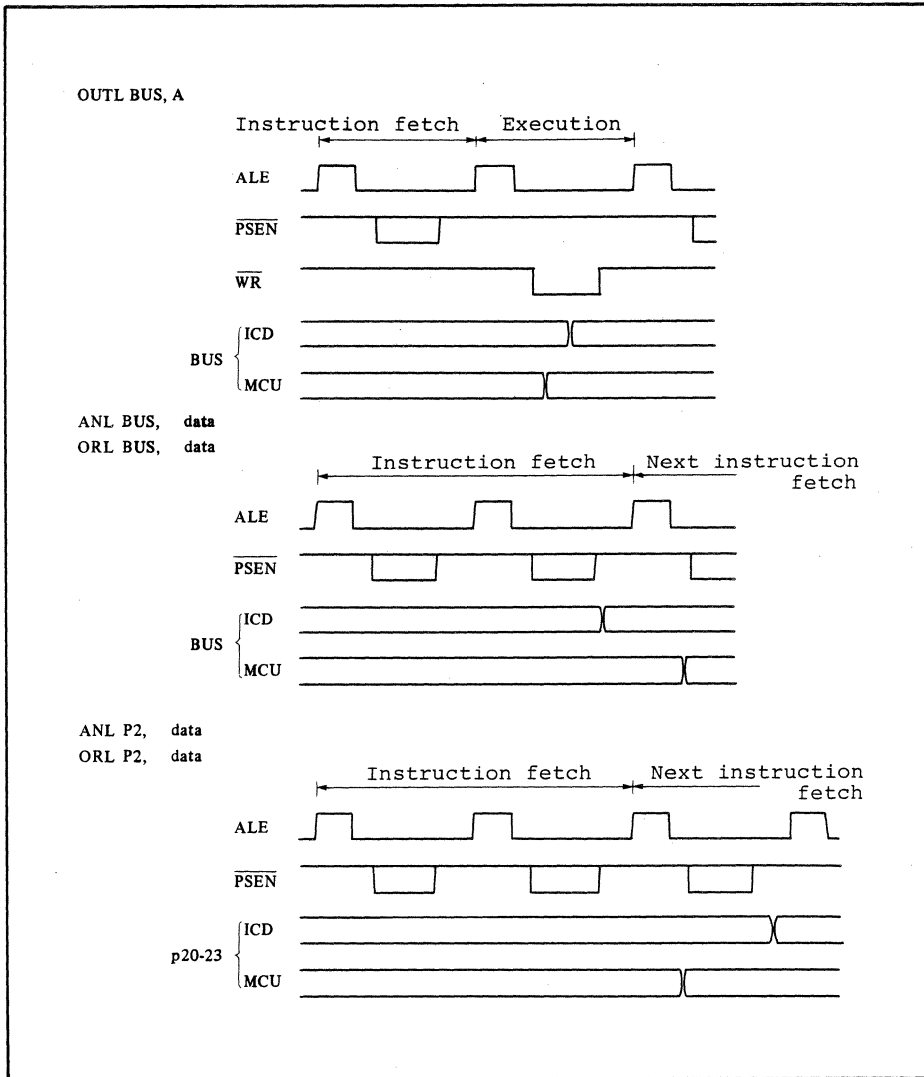
A - Propagation delay caused by TTL buffer. Not all signal lines are connected directly to the MCU and the target system. The TTL buffer is usually placed between the two. This causes the load current, drive current and rise/fall time to be different - causing a propagation delay in certain instances.

B - The BUS port and Port 2 are simulated within the ICD to generate their functions. This causes the output timing to differ between the MCU and ICD. The output status of the BUS port and Port 2 also differs depending on the Emulation Select switch setting.

C - Clock oscillator circuit. When the target system clock runs at the TTL level, proper emulation is assured. If the clock is supplied with an X'tal or RC circuit however, normal output is not guaranteed. Therefore, you may have to build the proper clock simulator board and connect it to the "CLKS" connector on the ICD.

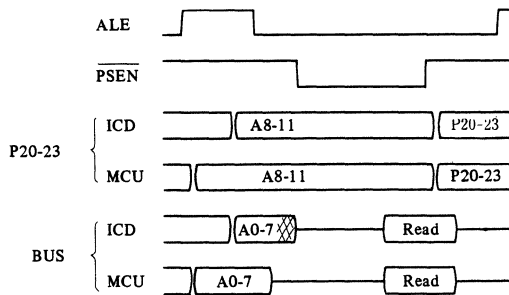
3) ALE output during MONITOR operation.

During MONITOR operation, the ALE output varies (H and L) and should not be used for a clock output.

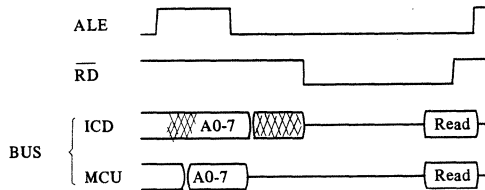


MCU/ICD Timing Deviations

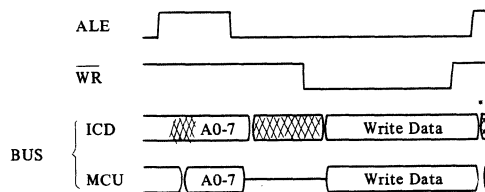
a) Read timing of program memory in target system



b) Read timing of external data memory in target system



c) Write timing of external data memory in target system



MCU/ICD Timing Deviations

d) Read/write timing of ICD internal emulation
(including program memory and external data memory)

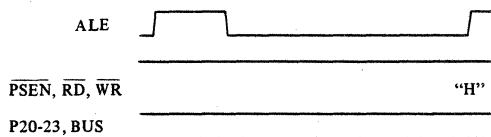
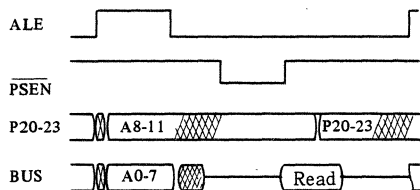
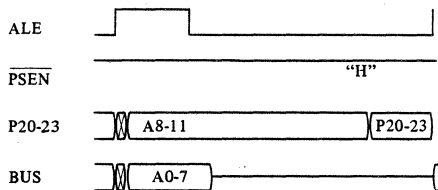


Fig. 5-2 When E.M.SEL.1="ON" (as shipped)

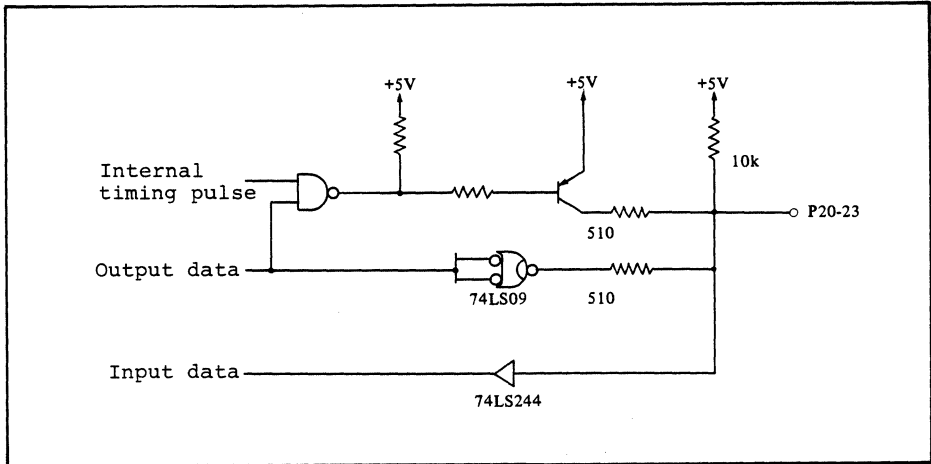
e) Read timing of program memory in target system



f) Read timing of ICD internal emulation memory (program memory)



MCU/ICD Timing Deviations

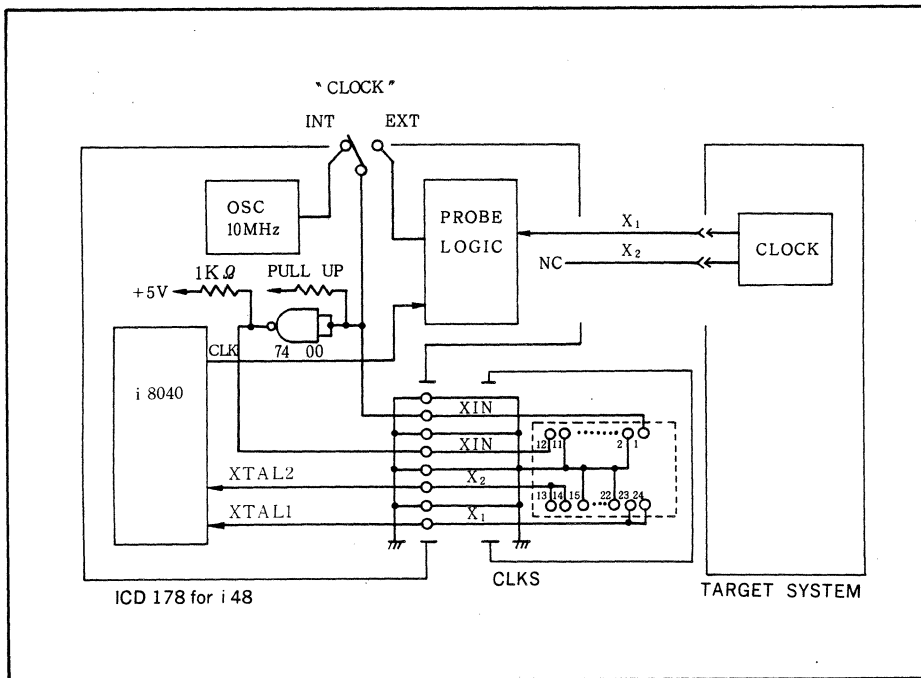


P20-23 Simulation Circuit

3.4.3 CLOCK SELECTION

The CLOCK switch specifies the use of either the;

- 1) ICD internal clock (10MHz TTL level input) [INT].
- 2) External (target) clock (TTL level input)[EXT].
- 3) Crystal/RC circuit CLKS board emulation.



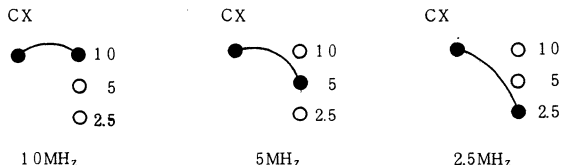
ICD/Target System Clock Configuration

CLKS DIP PLUG ASSIGNMENT

PIN No.	SIGNAL NAME	MEANING	IN/OUT
1	SIN	TTL clock input terminal *1	IN
2-11	GND	Ground	--
15-22	GND	Ground	--
12	\overline{XIN}	TTL clock XIN inverted input terminal *1	IN
13, 14	X2	XTAL2 of MCU Input	OUT
23, 24	X1	XTAL1 of MCU Input	OUT

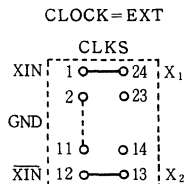
ICD Internal Clock

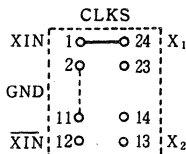
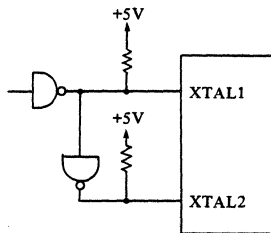
The ICD internal clock runs at a frequency of 10 MHz with a 50% duty cycle. This is the setting from the factory. You can change this clock speed to 2.5 MHz or 5 MHz by using the ICD internal jumper located on the S-792 board.



Target System External Clock

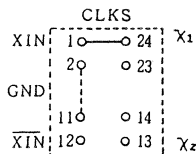
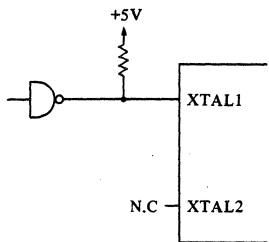
If you are using a crystal or RC oscillator circuit to drive the external clock, you must fabricate an equivalent circuit on the SCI DIP plug.



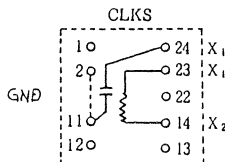
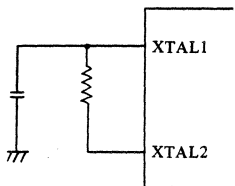


(As shipped)

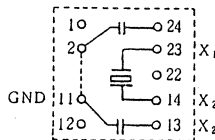
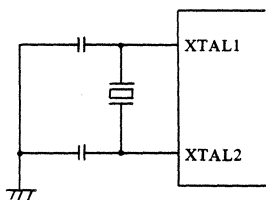
CLOCK switch = "EXT"



CLOCK switch = "EXT"



CLOCK switch = "INT" or "EXT"



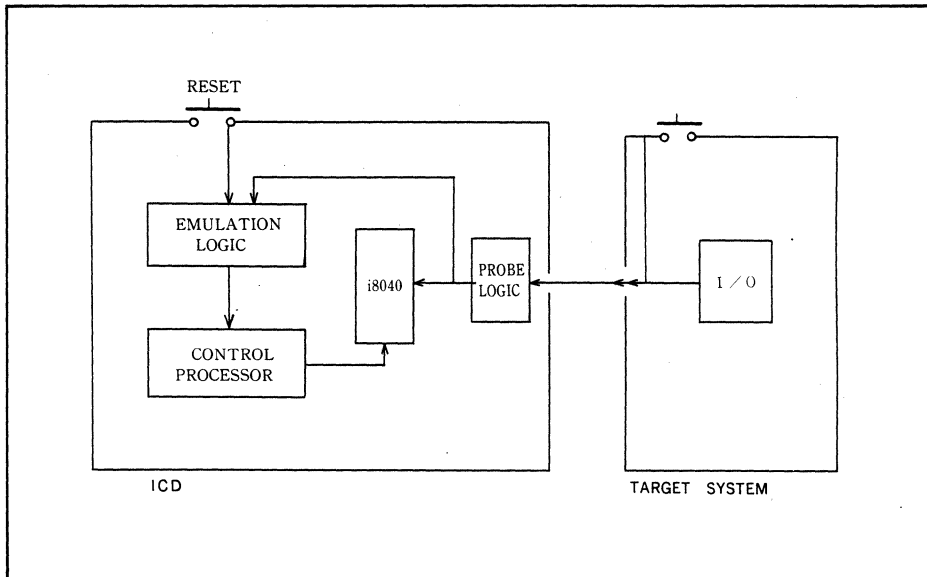
CLOCK switch = "INT" or "EXT"

Target System Clock Configurations

3.4.4 RESET Signal

The RESET switch located on the ICD Operator Panel resets the ICD, including the internal emulation Microcomputer Unit (MCU). The MCU will also accept a RESET input from the target system during emulation, however during MONITOR operation, this RESET to the MCU is ignored. To reset the MCU during MONITOR operation use the "REGISTER" RESET command. This command also initializes the contents of all registers.

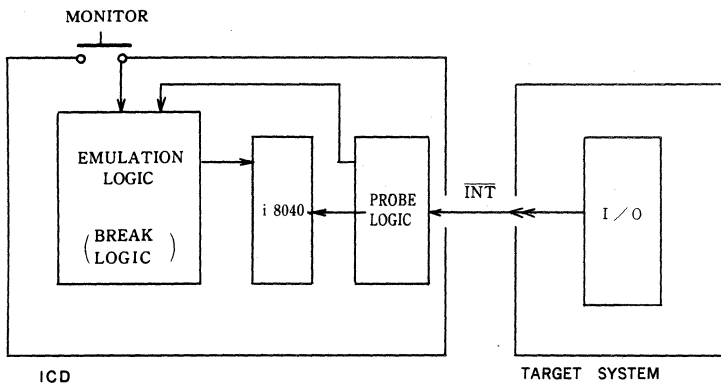
The RESET input from the target system may be enabled/disabled when emulation is performed in the In-circuit 1 mode.



RESET Circuit Block Diagram

3.4.5 MONITOR (Interrupt Signal)

The MONITOR switch returns control to the ICD monitor during emulation. When the MONITOR light is on, external interrupts (INT input) and timer/counter interrupts are prohibited. Although the operation of the timer/counter continues, caution should be exercised when it is used as a timer because the count rate is decreased.



MONITOR Circuit Block Diagram

	Incircuit Mode 0		Mode 1		Mode 2	
	Monitor	Emulatio	Monitor	Emulation	Monitor	Emulation
RESET	x *1	x	x *1	△	x *1	○
INT	x	x	x	△	x	○
SS	x	x	△	△	○	○
EA	x	x	△	△	○	○

○ : Effective

x : Ineffective

△ : Normally may be masked by PIn command.

*1 .. Use Register RESET command instead.

3.5 ROM (EPROM) PROGRAMMER

3.5.1 SPECIFICATIONS

Associated processor: EP-ROM processor
i8748/48H 1K bytes
i8749/49H 2K bytes

MASK-ROM processor
i8048/48AH 1K bytes
i8049/49AH 2K bytes
i8050AH 4K bytes

Programming capacity: 4K bytes, maximum

ROM processor selection: ROM module is changed according to ROM processor.

Operating method: Refer to ROM command in the Command Reference Guide.

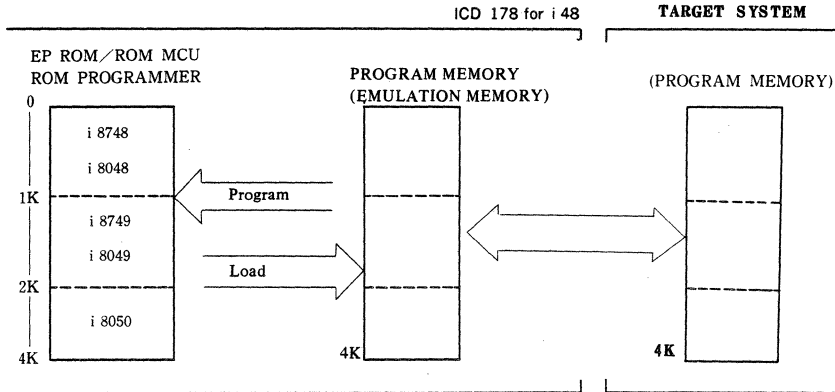
Programming voltage:

ROM Select Module	ROM Processor	VDD	EA	PROG	Unit
i87 × ×	i8748/49	25	23	23	Volt
i87 × × H	i8748H/49H	21	18	18	Volt
i80 × ×	i8048/49/50	5	12	—	Volt
i80 × × H	i8048H/49H/50H	5	12	—	Volt

NOTE ; PROG— =Floating

NOTE: USE THE PROPER ROM MODULE WITH THE CORRESPONDING ROM PROCESSOR. ROM MODULE/PROCESSOR MISMATCH MAY DAMAGE THE PROCESSOR.

3.5.2 ROM PROGRAMMING



The ROM command is >RO E R W V P address 1, address 2 <cr>. The parameters for this command are;

- E (erase) - EPROM processor erase status is checked.
- R (read) - The program memory content of the EPROM/ROM processor is read into the ICD program memory (with a read-after-check).
- W (write) - The ICD program memory content is read into the program memory of the EPROM/ROM processor (with a read-after-check).
- V (verify) - The program memory content of the EPROM/ROM processor is verified by the contents of the ICD program memory.
- P (program) - The parameters E, W and V are performed by this program command.

3.6 REAL-TIME TRACE

3.6.1 REAL-TIME TRACE SPECIFICATIONS

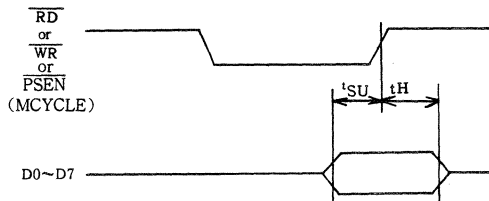
Word width	32 bits
Word size	2K
Acquisition cycle	CPU machine cycle (150ns min)
Trace section	2046 machine cycles (2.78ms min./10MHz)
Fixed trace data	A0-11, D0-7 PROGRAM MEMORY DATA MEMORY RD/WR/PORT
Trigger Mode	The start and end of tracing are specified by the "HISTORY" command.
Display	Machine cycle or disassembled display.

Real-time Counter Specifications

Count bit width	32 bits
Count clock	ALE clock
Effective count time	5841 seconds

3.6.2 REAL-TIME TRACE ACQUISITION

The storing of the data occurs on the leading edge of the RD/WR/PSEN signals during real-time tracing. The ICD ensures positive storing of the address, status and data when using the ICD program or user memory. However, storage of user memory or user I/O read is not guaranteed unless the data setup time is longer than 65ns.



t_{SU}	Real Time Trace User Data Read set up time	min 65 ns
t_{H}	Real Time Trace User Data Read hold time	min 0 ns

3.6.3 REAL-TIME TRACING - THEORY OF OPERATION

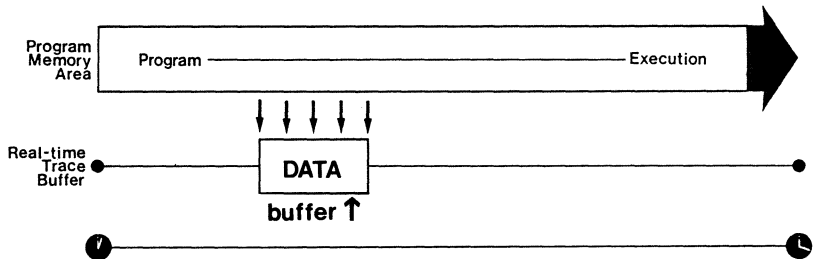
INTRODUCTION

The real-time trace records a portion of the program memory area. During emulation, a specified section of the program memory contents is recorded ("captured") and stored in the real-time trace buffer. This specific trace section can then be dumped and displayed. If there is a problem with the user program, a record of the program execution (history) can be reviewed.

DESCRIPTION

Real-time Trace Buffer

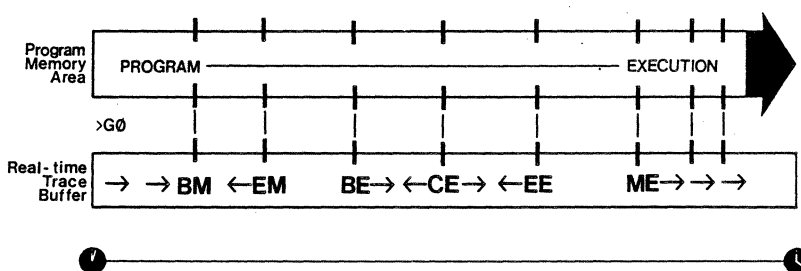
The real-time trace buffer can be thought of as a data storage facility (consisting of word width and word size) that moves along parallel to the user program. The buffer stores the same data that is being executed by the program. When the buffer becomes filled with data and the tracing operation is to continue, the real-time trace will begin to "write over" the previous data it has stored. The buffer will always display the latest data stored in it.



Trigger Modes

The ICD is equipped with a powerful real-time trace unit that features six different trigger modes. These modes act as "trigger points" and determine where (and when) the trace section is recorded in relation to the user program. The six trigger modes include; Begin Monitor (BM), End Monitor (EM), Begin Event (BE), Center Event (CE), End Event (EE) and Multiple Event (ME). These names designate which points in the user program will trigger the real-time trace (Event points, Break points, pushing the MONITOR switch, beginning the program with the "GO" command).

These modes are specified by the "HISTORY" command (see Section 2 - COMMAND REFERENCE GUIDE).



Trace Section Size

The trace section size is determined by specifying the "range" in the "HISTORY" command. This range extends backwards/forwards (or both), from one of the six trigger modes. When the range is not specified, the ICD assumes the default (maximum) condition. The range specification can be found under "Real-time Trace Specifications."

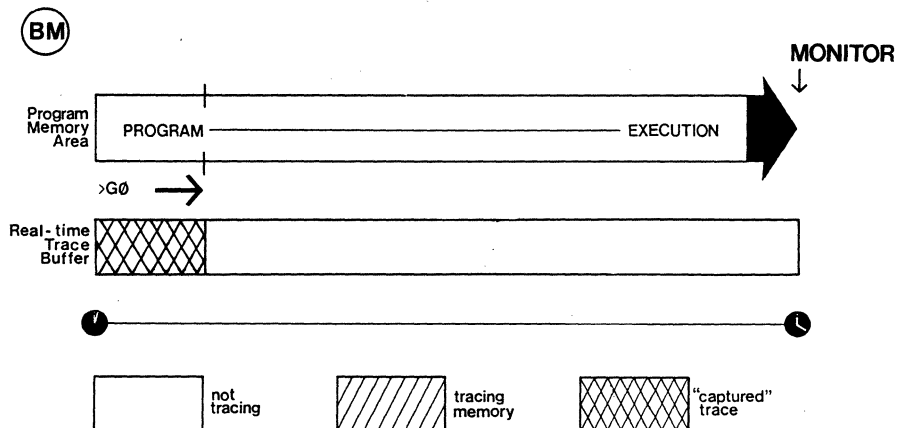
"Captured" Trace Data

The real-time trace buffer is used to record and display a specific section of the user program. This specific section is referred to as the "captured" trace section. It is this portion of the user program that will be displayed to you. The captured trace section always contains the latest data from the program.

Simplest Case: Begin Monitor Mode

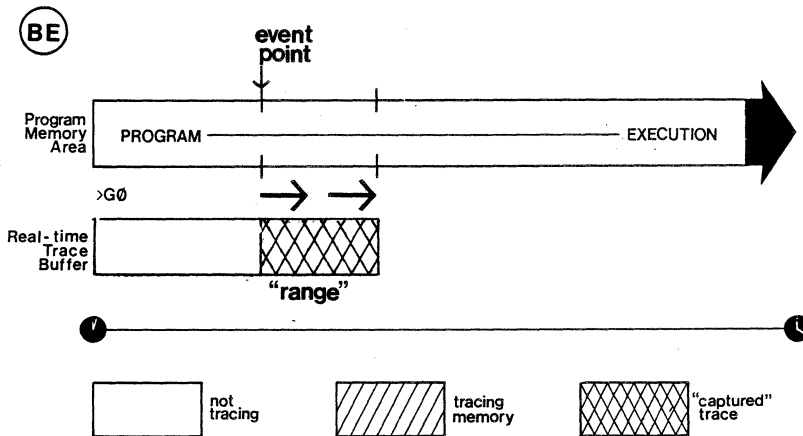
An easy way to understand how the real-time trace works, is to examine the Begin Monitor mode. In this mode, the "GO" command (which begins emulation), also triggers the start of real-time trace storing. The same data that is executed from the program memory area is transferred to the real-time trace buffer and stored.

After the user program executes (and the buffer stores) the data equivalent of the range, the trace buffer is considered full and the storing ends. The data that is now stored in the buffer is captured trace data. The real-time trace then enters a non-trace mode and stops when a MONITOR break (accomplished by pushing the MONITOR switch) or breakpoint is encountered.



Begin Event Mode

The Begin Event mode works in the same way as the Begin Monitor mode except that an event point triggers the real-time trace instead of the "GO" command. The buffer stores the amount specified by the range and then stops. The data now stored in the buffer is captured trace data.



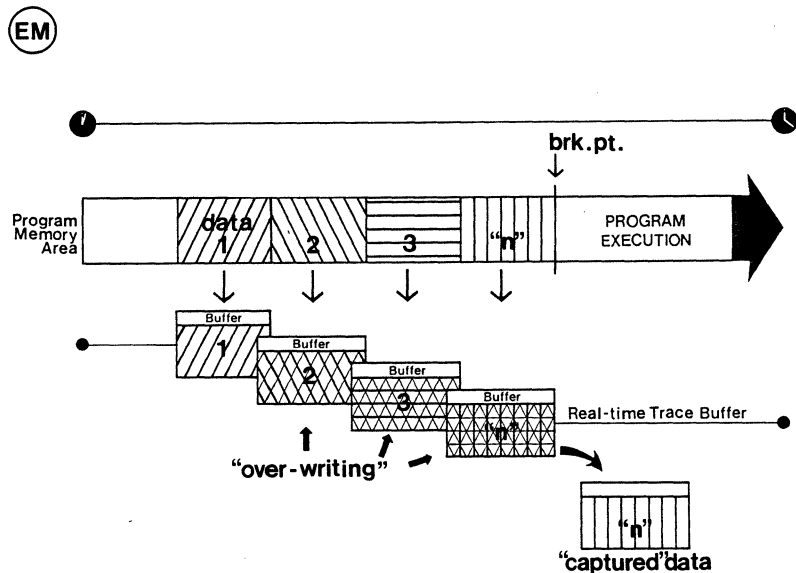
End Monitor Mode

The End Monitor mode begins storing all data and then terminates the storage process when a breakpoint is encountered, or when the MONITOR switch is pressed. The captured trace section is then found by subtracting the range from the break or MONITOR point.

For the ICD to accomplish this type of trace, it must begin recording data in the real-time trace buffer at the start of emulation, (even though the desired trace section might be toward the end of the user program).

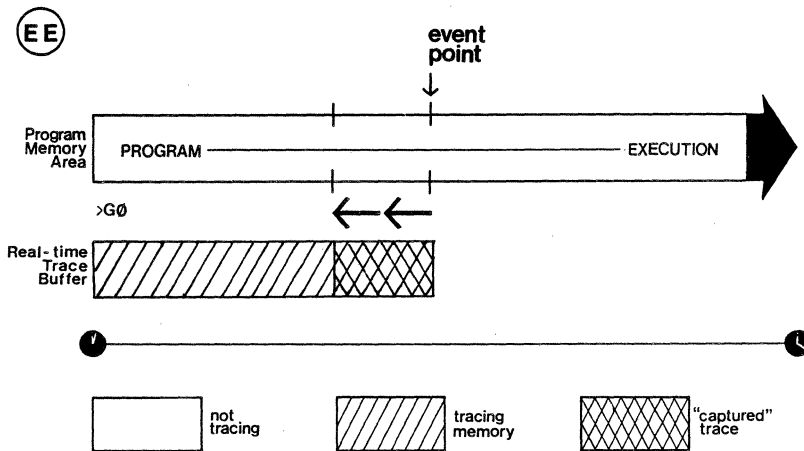
The ICD performs this type of trace by using a "snapshot" recording technique and then by "writing over" the data previously captured by the snapshot (see Real-time Trace Buffer diagram).

The End, Center and Multiple Event modes use this same snapshot and over-writing technique in their operation.



End Event Mode

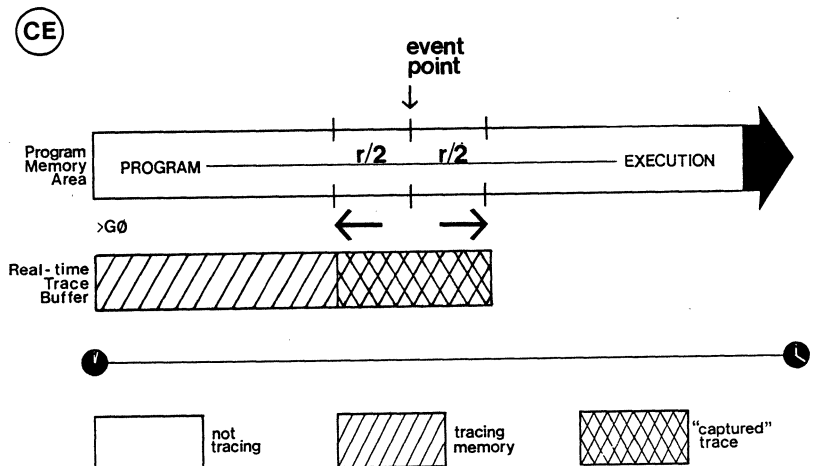
The End Event mode works in the same way as the End Monitor except that an event point (instead of a breakpoint) triggers the buffer to stop storing data. The captured trace section is then found by subtracting the range from the event point.



Center Event Mode

The Center Event mode is used when there is a single event point in the user program. The Center Event mode takes the range specification and "splits" it on either side of the event point. For example, if 2K is specified as the range, 1K of data would be captured before the event point and 1K would be captured after the event point.

Just like the End Monitor and End Event mode, the Center Event mode sets the real-time trace to begin recording data immediately after the "GO" command.

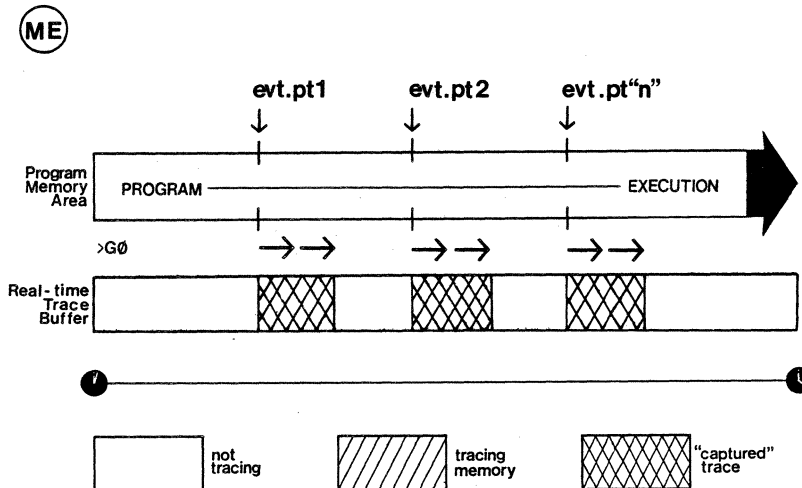


Multiple Event Mode

The Multiple Event mode is used when there are multiple event points in the user program. The Multiple Event mode takes the range specification and divides it by the total number of event points in the user program. The quotient is then equal to the size of each trace section. Each captured trace section initiates from each one of the event points.

For example, if 2K is specified as the range and there are four event points set in the program, the size of each captured trace section would be;

$$2K = 2(1024); 2048/4 = 512$$

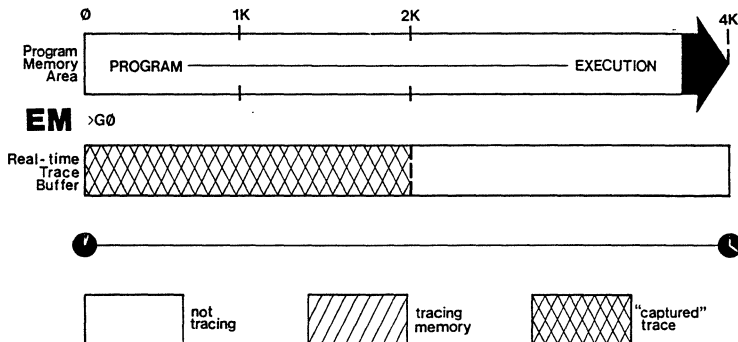


The following pages show the six trace modes available with the real-time trace. An example is included with each trace mode. There is also a short program included so that you can see how the real-time trace works. The program can be used with each trace mode.

TRACE MODE: End Monitor

COMMAND: >H EM <cr>

RANGE: 2K



EXAMPLE: End Monitor (This is the default condition after the ICD is turned ON and the RESET switch is pressed).

NOTE: For simplicity, all locations from 0 to 7FF are filled with NOP instructions.

NOP Instructions

>I0 (default condition)

>F P:0,7FF,0

>B/A OF,7FF

>G 0

;fills program memory from location 0 to 7FF with NOPs.
 ;sets a hardware breakpoint.
 ;starts emulation and initiates real-time History buffer storage.

PC	MC	OP	PSW	A	R0	R1	R2	R3	R4	R5	R6	R7	SP
770	00	NOP	0A	02	00	00	00	00	00	00	00	00	02

<Break Hardware A>
>

;program reaches break point and breaks.
 ;displays History contents in numeric code.

>HD

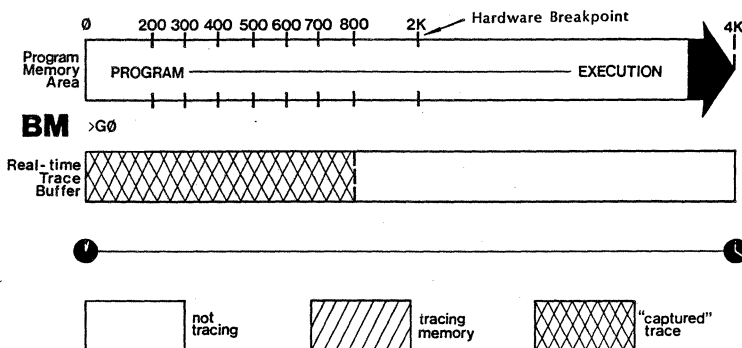
POINT	T	ADDR	DT	ST	OP
1921		000	00		NOP
1920		001	00		NOP
#					
0003		7FE	00		NOP
0002		7FF	00		NOP
0001	*	PAUSE			

* indicates trigger point.
 # real-time storage pointer range 1906 to 0001

TRACE MODE: Begin Monitor

COMMAND: >H BM <cr>

RANGE: 800



EXAMPLE: Begin Monitor

NOTE: This example uses the same program as the Event Monitor mode program (only the Begin Monitor specification changes).

```
>H CLR ;resets History command.
>H BM 800 ;sets the real-time storage
           to Begin Monitor mode.
>B/A OF,7FF
>G 0 ;starts emulation and initi-
      ates real-time trace buffer
      storage.
      ;emulation stops when program
      reaches a breakpoint.
```

```
PC MC OP PSW A R0 R1 R2 R3 R4 R5 R6 R7 SP
7FF 00 NOP 08 00 00 00 00 00 00 00 00 00 00
```

<Break Hardware A>

>

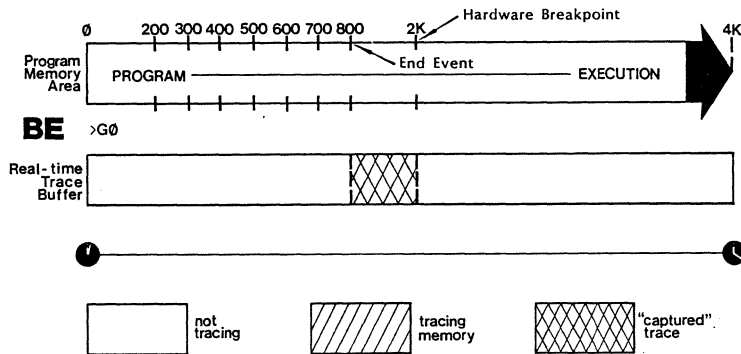
```
>HD ;displays History contents in
POINT T ADDR DT ST OP numeric code, starting from
0802 * 0000 00 NOP address 1.
0801 0001 00 NOP
0002 7FF 00 NOP
0001 PAUSE
```

*indicates trigger point.

TRACE MODE: Begin Event

COMMAND: >H BE <cr>

RANGE: 2K



EXAMPLE: Begin Event

```

>F P:0,7FF,00           ;fills program memory from
                        ;location 000 to FFFF
                        ;with NOP instructions.

>H CLR                 ;
>H BE                  ;sets Begin Event real-time
                        ;trace storage mode.

>EV ST=OF,A=0320       ;sets event point.
>B/A OF,77F           ;sets hardware breakpoint to
                        ;terminate emulation.

>G 0                   starts emulation from location
                        ;000.

```

```

PC MC OP PSW A R0 R1 R2 R3 R4 R5 R6 SP
077F 00 NOP 08 00 00 00 00 00 00 00 00
<Break Hardware A>
>

```

;program reaches hardware break A and stops.

```

>HD                   ;displays History contents in
POINT T ADDR DT ST OP mnemonic code.
1121 * 0320 00      NOP
1120      0321 00      NOP
0002      077F 00      NOP
0001      PAUSE

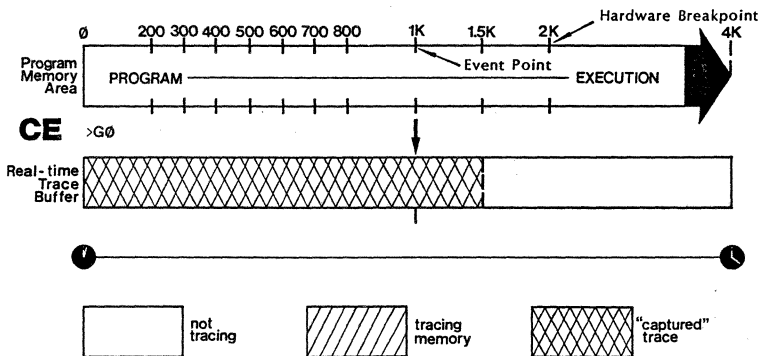
```

*indicates the Event point.

TRACE MODE: Center Event

COMMAND: >H CE <cr>

RANGE: 500



EXAMPLE: Center Event

```

>H CLR                               ;continues from Begin Event
>B INI                               example. Use same program
>H CE 500                             and hardware breakpoint.
                                       ;resets History command.
                                       ;resets event trigger.
                                       ;sets Center Event real-time
                                       trace storage mode.
>EV ST=OF,A=3FF                       ;sets event point.
>G 0                                   ;starts emulation from location
                                       000.
>B/A OF,7FF                           ;program will reach hardware
                                       breakpoint A and stop. Screen
                                       will display same data as
                                       Begin Event example.

>HD                                   ;displays the contents of
                                       History buffer in mnemonic
                                       codes.

```

POINT	T	ADDR	DT	ST	OP
1525		000	00		NOP
1524		001	00		NOP
0502	*	3FF	00		NOP
0501		400	00		NOP
0002		5F3	00		NOP
0001		PAUSE			

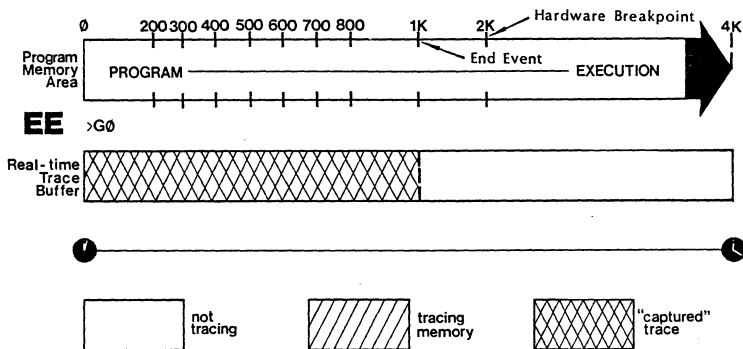
;see Begin Event example.

*indicates event point.

TRACE MODE: End Event

COMMAND: >H EE <cr>

RANGE: 2K



EXAMPLE: End Event

```

>F P:0,7FF,0
>H CLR
>B INI
>H EE

>EV ST=OF,A=3FF
>B/A OF,7FF
>G 0

;continues from Center Event
example. Use same program and
hardware breakpoint.
;resets History command.
;resets event trigger.
;sets End Event real-time
trace storage mode.
;sets event point.
;
;starts emulation.

;program will reach hardware
breakpoint A and stop. Screen
will display same data as
Begin Event example.

>HD

;displays the contents of
History buffer in mnemonic
codes.

POINT T ADDR DT ST OP
1027 000 00 NOP
1026 001 00 NOP ;see Begin Event example.

0005 3FE 00 NOP
0004 * 3FF 00 NOP
0003 400 00 NOP
0002 401 00 NOP
0001 PAUSE

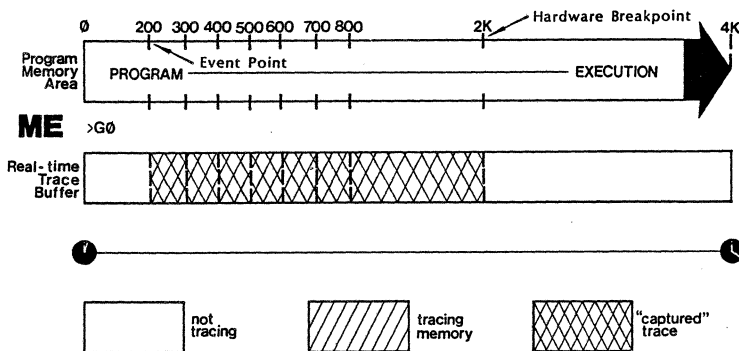
```

*indicates event point.

TRACE MODE: Multiple Event

COMMAND: >H ME <cr>

RANGE: 100



EXAMPLE: Multiple Event

```
>F P:0,7FF,0
>A 7FE
7FE JMP 0H
>B/A OF,7FE,10

>H CLR
>B INI
>H ME,100
>EV ST=OF A=0C8
```

;continues from End Event example. Use same program and hardware breakpoint. A JUMP instruction is added at location 7FE so that the program will loop during execution. (100 passing counts are also added to the breakpoint).

```
>G 0
>HD
```

```
;resets History command.
;resets event trigger.
;sets Multiple Event real-time trace storage mode and storage size as 100 instructions per loop.

;starts emulation.

;displays the contents of History buffer in mnemonic codes.
```

POINT	T	ADDR	DT	ST	OP
1020	*	0C8	00		NOP
1019		0C9	00		NOP
918	*	0C8	00		NOP
917		0C9			
103		PAUSE			
102	*	0C8			NOP
101		0C7			NOP

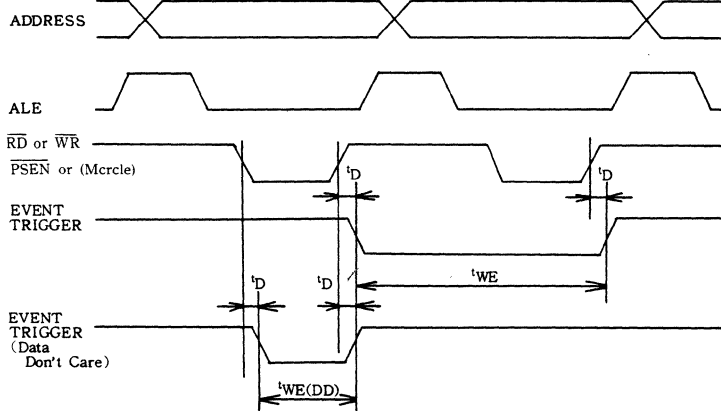
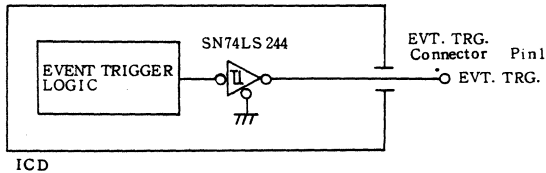
;see Begin Event example.

*indicates event point.

3.7 PROBES

3.7.1 EVENT TRIGGER PROBE

The Event Trigger probe is used to receive a low-level pulse (TTL) signal when an event point is passed during emulation. The duration of this low-level pulse is 250ns from the moment at which the RD/WR rises in a cycle with matching event conditions, and the moment at which the RD/WR signal in the next cycle rises. A low-level pulse synchronized with the RD/WR signal outputs if "don't care" is specified as an event condition.



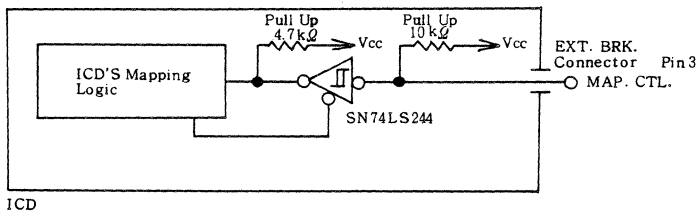
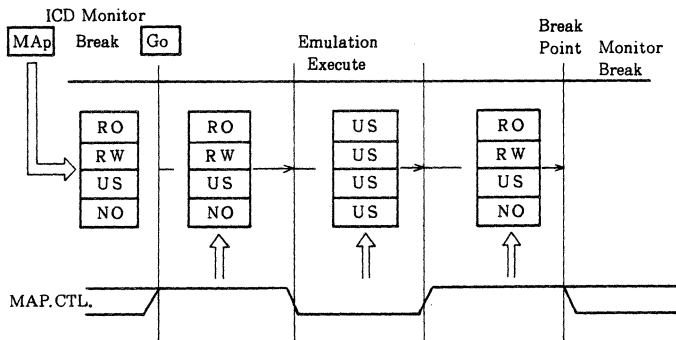
$t'WE$	Width of EVENT TRIGGER Time.	min 415 ns
$t'WE(DD)$	Width of EVENT TRIGGER Time. (Data Don't Care)	min 250 ns
$t'D$	Delay Time.	max 80 ns

Event Trigger Timing Diagram

3.7.2 MAP CONTROL PROBE

The Map Control probe allows the target memory to be used over the entire memory area (4K bytes + 256 bytes) regardless of the map specification. The Map Control probe of the External Break cable must receive the TTL signal and the in-circuit mode must be either 1 or 2 for the probe to operate correctly.

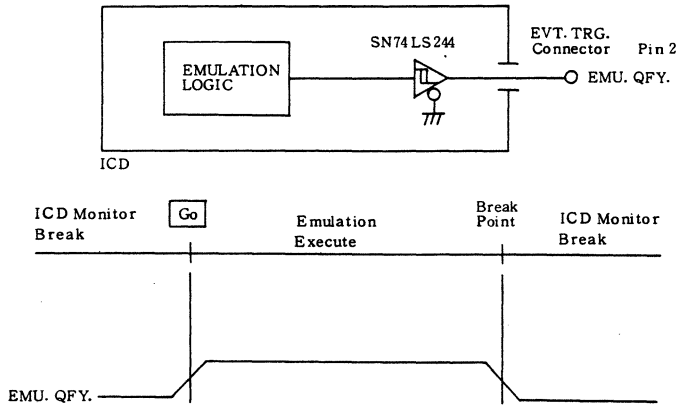
This input signal is similar to the EA input signal of the MCU chip, however it is different than that of the EDM (External Data Memory). This input signal is ignored when the MONITOR lamp is ON.



Map Control Timing Diagram

3.7.3 EMULATION QUALIFY PROBE

The Emulation Qualify probe indicates the status of the ICD emulating the target system. The Emulation Qualify output is at a high level (TTL) until control returns to the ICD monitor. Unwanted signals can be removed by connecting the probe to a logic analyzer or similar device.



Emulation Qualify Timing Diagram



Zax Corporation 2572 White Road, Irvine, California 92714
(714) 474-1170 • 800-421-0982 • TLX 183829