ICD-178
FOR
8086/88
USER'S MANUAL

ICD-178
FOR
8086/8088
USER'S MANUAL

Part No. ZTP-107-00, Rev. A. Printed: 1985

Limitations on Warranties and Liability

ZAX Corporation warrants this equipment to be free from defects in materials and workmanship for a period of one (1) year from the original shipment date from ZAX. This warranty is limited to the repair and replacement of parts and the necessary labor and services required to repair this equipment.

During the 1-year warranty period, ZAX will repair or replace, at its option, any defective equipment or parts at no additional charge, provided that the equipment is returned, shipping prepaid, to ZAX. The purchaser is responsible for insuring any equipment returned, and assumes the risk of loss during shipment.

Except as specified below, the ZAX Warranty covers all defects in material and workmanship. The following are not covered: Damage as a result of accident, misuse, abuse, or as a result of installation, operation, modification, or service on the equipment; damage resulting from failure to follow instruction contained in the User's Manual: damage resulting from the performance of repairs by someone not authorized by ZAX; any ZAX equipment on which the serial number has been defaced, modified, or removed.

Limitation of Implied Warranties

ALL IMPLIED WARRANTIES, INCLUDING WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE LENGTH OF THIS WARRANTY.

Exclusion of Certain Damages

IN NO EVENT WILL ZAX BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THIS EQUIPMENT. THIS EXCEPTION INCLUDES DAMAGES THAT RESULT FROM ANY DEFECT IN THE SOFTWARE OR MANUAL, EVEN IF THEY HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

Disclaimer

Although every effort has been made to make this User's Manual technically accurate, ZAX assumes no responsibility for any errors, omissions, inconsistencies, or misprints within this document.

Copyright

ZAX Corporation
Technical Publications Department
2572 White Road
Irvine, California 92714

The BOX refers to ZAX's 8- and 16-bit microcomputers.

Written by Mark Johnson of ZAX Technical Publications.

Changes are periodically made to the information herein: these changes will be incorporated in new editions of this publication. Updates to this manual will be sent to all manual recipients.

A Reader's Comments form is provided at the back of this publication. If this form has been removed, send comments to the address above.

Reorder User's Manual 20-107-00
Reorder Command Reference Guide 20-603-01

Contents

ICD-178 for 8086/8088 FEATURES

General Characteristics

- 8086/8088 CPU and 8087 NDP support
- All I/O ports (64K bytes) available
- Host computer support
- All memory available

User Interface

- You control all functions from terminal or computer
- Symbolic debugging available with ZICE
- Mnemonic command names
- Setup emulation controls from batch file on host computer
- In-line assembler

Emulation Controls

- Internal or external clock
- Disable interrupt inputs
- Disable bus request input

Memory Mapping

- 128K bytes standard emulation memory
- Read-only or read/write emulation memory
- Programmable wait states
- Map override input
- 1K-byte mapping resolution
- 1 Mbyte maximum emulation memory
- "No memory" mapping specification
- Control from keyboard

Address and Data Specifications

- Four offset registers
- One bit "don't care" resolution

Breakpoints

- Four hardware breakpoints
- Eight software breakpoints
- Break on a specified address or data
- Break on range
- Break on access to non-memory area
- Break on write to read-only area
- Sequential break (A then B)
- Break on opcode fetch only
- Break on instruction execution
- Break on Nth occurrence
- Break on wait state timeout
- External break input (triggers from HI or LO signal edge)

Non-Real-time Trace

- Single step
- Step n steps
- Trace Jump instructions only

Real-time Trace

- Stores addresses, data, and status
- 4K bytes deep x 40 bits wide trace memory size
- Real-time counter
- Adjustable delay
- Trace control modes include:
  - Begin Monitor
  - End Monitor
  - Begin Event
  - End Event
  - Center Event
  - Multiple Event

Disassembly Capabilities

- Disassemble from program memory
- Disassemble trace memory from any selected area

Special Features

- Assemble into memory
- Use ICD's serial interface from user program
- Search program memory for pattern
- Search trace memory for pattern

About This Manual

Thank you for choosing a **ZAX** in-circuit emulator! Your **ZAX** emulator is one of the most powerful and sophisticated microprocessor development tools in the industry—as you will soon discover. But for all the things your emulator can do, it's still very simple to use. In fact, you don't have to know a thing about **ZAX** emulators to use this manual. The information presented in this manual is structured for first-time users, so you'll be learning about emulation techniques and applications as well. If you're already familiar with the principles of emulation, you can use this manual now to learn a few basic emulator skills, and then use the section on commands as a reference.

What This Manual Will Show You

- How to identify the parts (controls, components, accessories) of your emulator and understand what they do (Section 1).
- How to connect the emulator to your terminal, host computer and target system (Section 1).
- How to find out more about special emulator controls and learn how to use them for your specific applications (Section 1).
- How to use the accessories that come with your emulator (Section 1).
- How to use each of the emulator commands (Section 2).
- How to learn more about how your emulator works, by examining the internal control modules (Section 3).
- How to write support software programs for interfacing the emulator with a host computer (Section 4).

How To Use This Manual

There are really only two things you must know to use a **ZAX** emulator: the first is how to connect it to your present system, and the second is how to control the emulator's operation by using the commands. These two subjects are presented in the first two sections of this manual, and of these two, you'll be using the section on "commands" particularly.

So first, read Section 1 to learn about the various controls and components of your emulator. (Before you can operate the emulator, you'll have to set certain switches and make some minor adjustments so that it performs correctly with your system.) Then, continue on to learn how to connect your emulator to other devices such as a console terminal or a host computer, and your target system.

Once your emulator is working properly, you can refer directly to Section 2 to find out how to enter any of the emulator commands. Each command's function is examined, along with the format needed to use the command. Once you're familiar with the command syntax, you can use the fold-out Command Reference Guide located in the front of the manual.

If you need a refresher course on emulation principles, turn to Appendix A. If you're not sure how to apply the commands in an actual emulation session (we call it "debugging"), turn to Appendix B for a demonstration. Use Section 3 for a reference (it contains technical information that you may find useful later on). You can use Section 4 if you're writing your own support software programs to interface your host computer (if it's not already supported by **ZAX**'s ZICE communication software) to the emulator.

Oh, by the way, any time a word or phrase is used and you don't understand its meaning, turn to the Glossary at the back of this manual. It contains definitions for a number of common engineering terms as well as many specialized microprogramming terms.

Emulator or ICD?

One last thing—the official name of your emulator is the ICD-178 for 8086/8088 (ICD stands for IN-CIRCUIT DEBUGGER; 178 is the model number), although we'll use the initials **ICD** whenever we mean the ICD-178, in-circuit debugger, emulator or in-circuit emulator.

Now turn to Section 1 and get started.

Introduction

In Section 1, you'll learn about the different parts of your ICD, what they do, and how to use them. You'll also learn how to connect the ICD to your system (terminal, host computer, target system) and find out how to use the accessories that come with the ICD. Your ICD has a few special features that you should know about, too; you can find information about these features in this section as well.

A Word Of Caution

You shouldn't try to attach the ICD to any external device before you finish reading this section. As long as the power cord is disconnected you can't hurt anything internally, but don't connect the ICD to your target system before you read "How To Connect Your ICD To Other Devices," later in this section. Although it's difficult, it is possible to get the cables to the target system reversed, which could result in damage to the ICD's internal components.

Getting Acquainted With Your ICD

Your **ZAX** ICD-series in-circuit emulator is a microprocessor emulation device that can be used for developing and maintaining 8086/8088 microprocessor-based systems as well as the 8087 Numeric Data Processor (NDP). It does this by letting you direct and test activities in your prototype ("target") system. You perform these operations by entering one or more debugger commands.

All **ZAX** ICD-series emulators are controlled by a separate terminal, or in conjunction with your existing host computer system. You can use the debugger commands for your hardware or software projects by simply inputting the command mnemonics and parameters from just about any terminal or popular computer you might own.

A Few Features

Here are just a few things you can do using the debugger commands:

- Use the ICD's emulation memory to simulate or take the place of memory (or future memory) in your target system.
- Use a single-step trace operation to move through your program, one step at a time, and examine the registers' contents after each step.
- Set a combination of hardware and software breakpoints to stop your program when: data is written or read into a specific address; an event point is passed; a non-existent memory access is attempted; or an interrupt is acknowledged by the CPU. Hardware breakpoints can also generate triggers for instruments such as logic analyzers and oscilloscopes.
- Record ("trace") a portion of your program (beginning and ending anywhere within the program) and store it in the ICD's real-time trace buffer without affecting the emulation process. Later you can display the recorded memory contents in either machine code or in its disassembled format.
- Translate symbolic codes into machine instructions, item for item, using the in-line assembler.
- Selectively enable and disable the interrupt or bus request inputs—including non-maskable interrupts.

You can turn to Section 2 for a complete list of the ICD's debugger commands. To find out about other things your ICD can do, turn to "More About Your ICD," in this section.

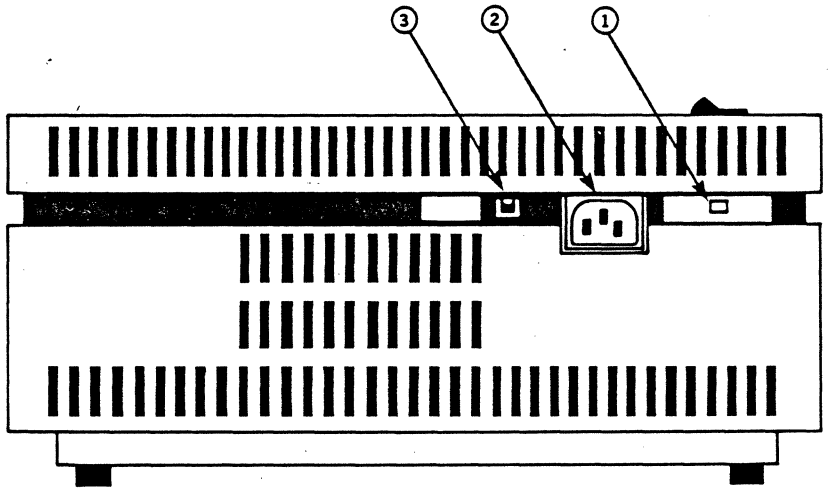
Now turn the page to learn about the parts of your ICD.

The Controls And Component Functions Of Your ICD

① AC POWER SELECT Switch. This switch is used to select the power requirements for the ICD. Set the switch to 110V/117V to run on a power supply of 110-120 VAC, or select 200V/240V to run on a power supply of 200-240 VAC.

② AC POWER CORD Receptacle. Accepts female end of the supplied three-wire power cord. Be sure to disconnect the power cord before moving the ICD.

③ DC OUT 24V (FAN Receptacle). Accepts connector end of the 24V DC fan.



④ TERMINAL Port Connector. Accepts male end of an RS-232 cable, to attach the ICD to a terminal when the ICD is controlled by a terminal (LOCAL operation mode). When a host computer is used to control the ICD (REMOTE operation mode), this port can be used as an auxiliary I/O.

⑤ HOST/AUX Port Connector. Accepts male end of an RS-232 cable, to attach the ICD to a host computer system when the ICD is controlled by a host computer (REMOTE operation mode). ICD commands can then be entered using the computer's keyboard. When using the ICD in the LOCAL operation mode, this port dumps object code, registers or memory to a host computer or printer.

⑥ LOCAL/REM (Local/Remote) Select Switch. This switch is used to select which port (TERMINAL or HOST/AUX) the ICD will use to receive commands.

⑦ BAUD RATE Switches (TERMINAL and HOST/AUX ports). These switches are used to set the baud rates for the TERMINAL and HOST/AUX ports. The factory setting is #1 (9600 bps). To change the baud rates for the ports, see "Changing the Baud Rate Settings," in Section 3.

⑧ DCE/DTE Select Switch. This switch is used to set the HOST/AUX port to either RS-232 data terminal equipment (DTE) or data communications equipment (DCE). Use the DTE setting if the ICD is used with a host computer; use the DCE setting if a printer is connected to the HOST/AUX port. (The TERMINAL port is always DCE.)

⑨ Top CPU In-circuit Probe Receptacle. Accepts female end of the Top CPU In-circuit Probe.

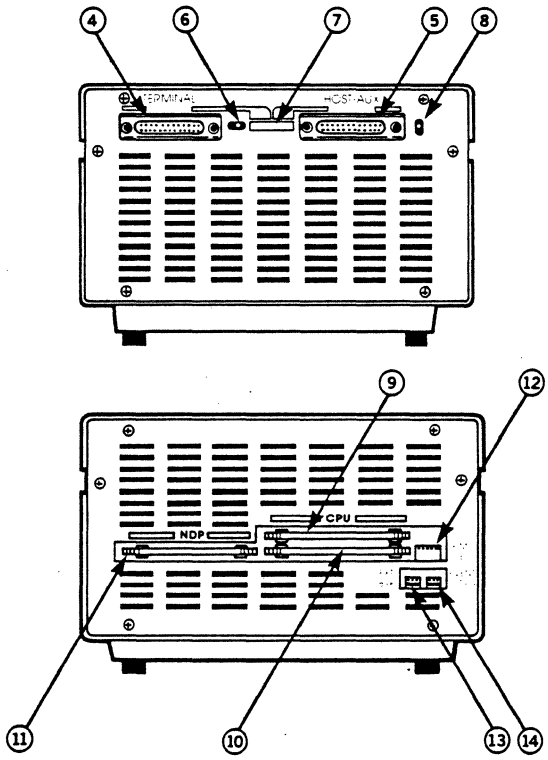
⑩ Bottom CPU In-Circuit Probe Receptacle. Accepts female end of the Bottom CPU In-circuit Probe.

⑪ NDP In-circuit Probe Receptacle. Accepts female end of the NDP (8087) In-circuit Probe.

⑫ E.M. SEL (Emulation Method Select) Switch. This switch is used to set the machine cycle operation to the target system. (See "More About Your ICD," in this section, for details on what this switch does.)

⑬ EXT.BRK. (External Break) Connector. Accepts female end of the External Break/Map Control cable. (See "More About Your ICD," in this section, for details about how to use this cable.)

⑭ EVENT TRG. (Event Trigger) Connector. Accepts female end of the Event Trigger/Emulation Qualify Cable. (See "More About Your ICD," in this section, for details about how to use this cable.)



⑮ POWER ON/OFF SWITCH. This switch is used to supply power to the ICD.

⑯ CLOCK INT/EXT Switch. This switch is used to select either the ICD's internal clock (INT) or the target system's clock (EXT).

⑰ HALT Lamp. This LED comes on after the ICD's CPU has stopped executing a HELP instruction or when a BUSAK (BUS ACKNOWLEDGE) is in progress.

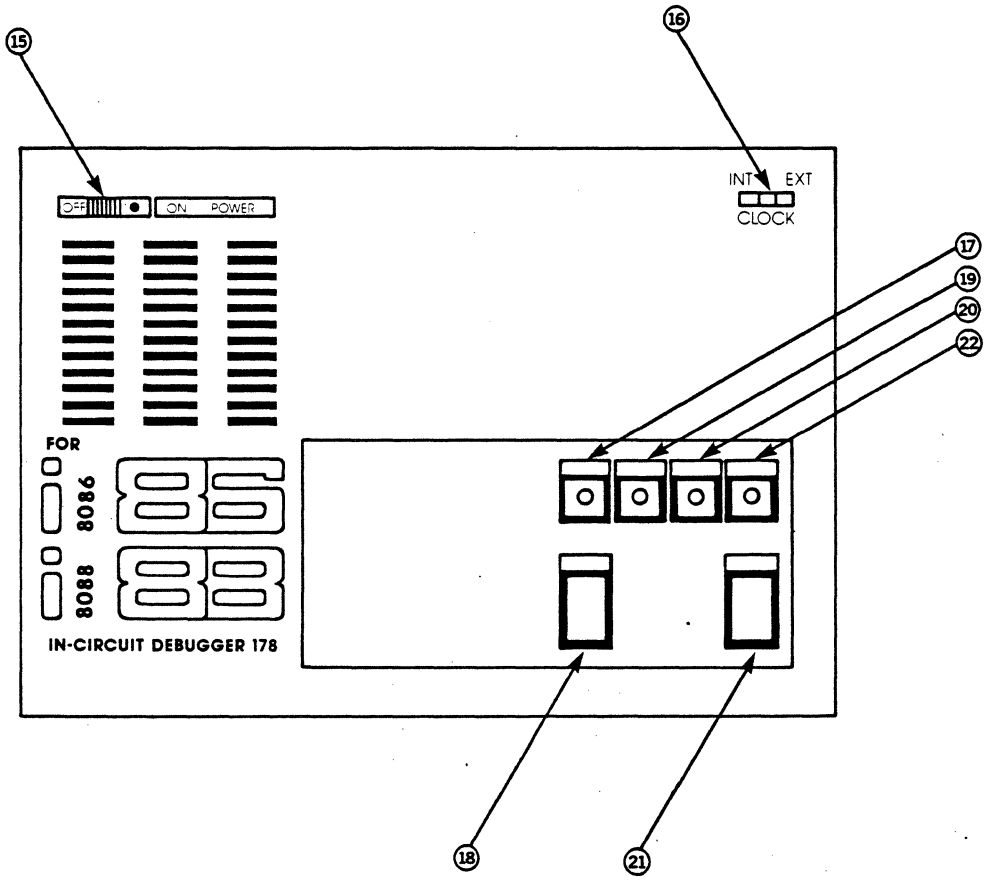
⑱ RESET Switch. This switch is used to reset the ICD monitor. You can push it any time the MONITOR lamp is lit. After you push the RESET switch, you'll see the ICD's identification message on your terminal's monitor.

⑲ MONITOR Lamp. This LED comes on to indicate that control is currently in the ICD's monitor. It will not be lit during emulation.

⑳ ICE (In-Circuit Enable) Lamp. This LED comes on when the ICD is operating in the I1 or I2 in-circuit mode.

㉑ MONITOR Break Switch. This switch is used to return control to the ICD monitor during emulation.

㉒ POWER Lamp. This LED comes on to indicate that power is being supplied to the ICD.



Now turn to the next chapter to learn how to connect the ICD to your system.

How To Connect Your ICD To Other Devices

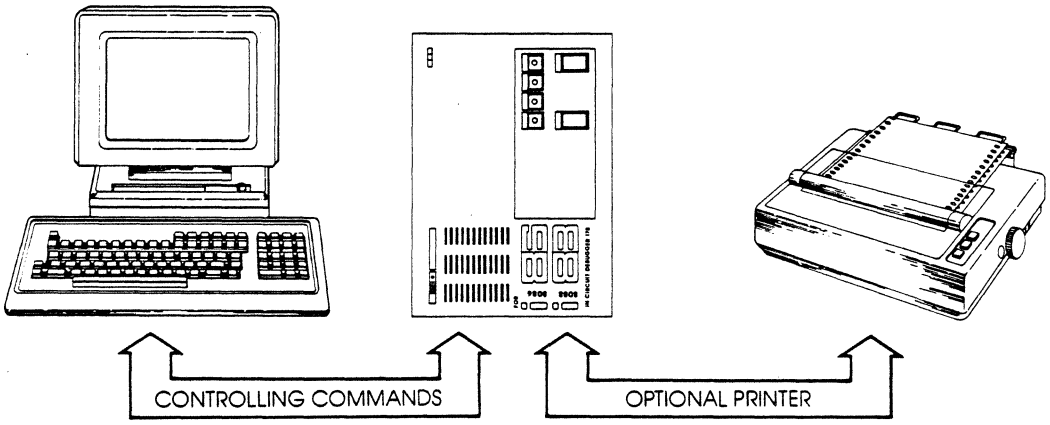
In the main introduction, you read that properly connecting the ICD to your system was one of the most important things you would learn in this manual. The following information will show you how to connect the ICD's components, what cables to connect and where they go, and which switches are set to what positions. Once you've completed the procedures outlined in this section, you'll have what is called a "Microprocessor Development System" (MDS). By using the commands and applications found in Section 2, you'll be able to perform a remarkable variety of debugging operations with your MDS.

Your System's Environment

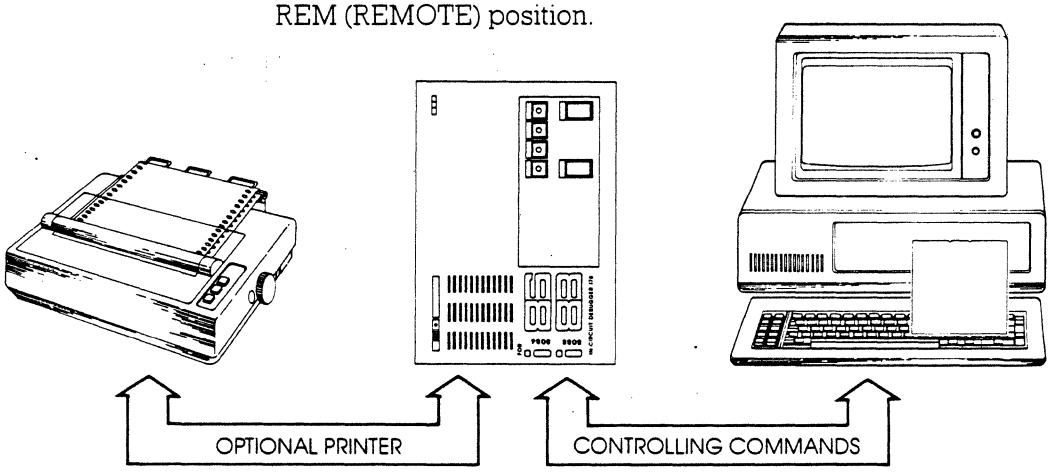
Before you connect your ICD to anything, you'll need to answer three questions about your system's environment. First, will you control the system with a **terminal** or **a host computer**? Second, if a terminal is used to control the ICD, will a host computer be used as a **source** for data files? And third, will your system be used to develop/debug **hardware or software**?

Terminal or Host Computer Controlled?

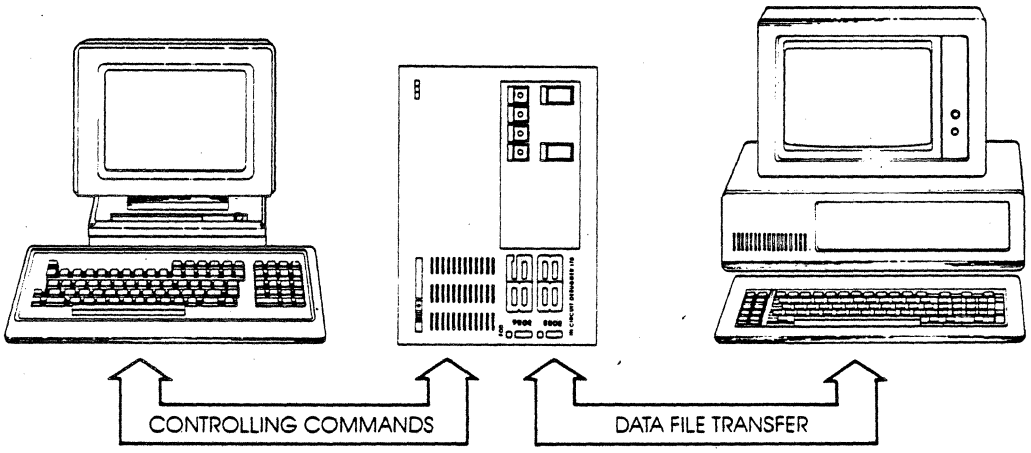
If you'll be controlling the ICD by a console terminal, it's called **TERMINAL CONTROL OF THE ICD**. In this configuration, the ICD "stands alone" (hence the name, stand-alone emulator), or apart from the auxiliary control of a host computer system. The ICD assumes a stand-alone mode of operation when you place the LOCAL/REM switch to the LOC (LOCAL) position.



If you'll be controlling the ICD with a host computer and using the utility software program ZICE, it's called **HOST COMPUTER CONTROL OF THE ICD**. The ICD assumes this mode of operation when you place the LOCAL/REM switch to the REM (REMOTE) position.



You may choose to control the ICD with a terminal and use a separate host computer to store data files. A printer can also be connected to the host computer to dump data for hard copies. This configuration is called **TERMINAL CONTROL OF THE ICD (WITH HOST DATA FILES)**. In this configuration, the ICD is still under direct control of the terminal, which the host computer serves as a data storage device. You can also cause the ICD to assume a "transparent" condition, which allows direct communication between the terminal and host computer.

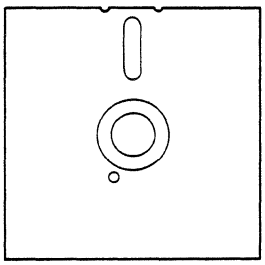


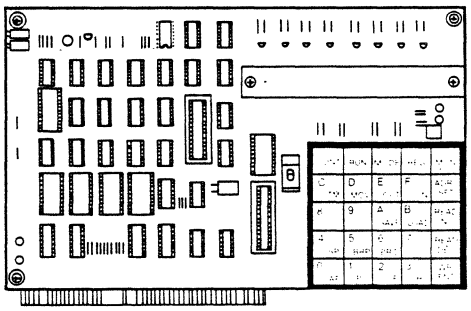
*NOTE: ZICE software may be used in the LOCAL mode—**TERMINAL CONTROL OF THE ICD (WITH HOST DATA FILES)**—for accessing the ZICE commands (help files, "Z" commands, etc.). To use this LOCAL "host computer assisted" mode, see the HOST command, in Section 2.*

Hardware or Software?

Your **hardware** is called a "target system." By physically removing the 8086/8088 CPU (or 8087 NDP) in your system and electronically replacing it with the ICD's internal microprocessor, you can control, test and check almost all possible functions in your target system.

Can you use your ICD without a target system? Of course!. Whenever you develop and debug **software**, you'll be doing it without the use of a target system. This mode is also an effective way to demonstrate some of your ICD's features.





System Configuration Characteristics

SYSTEM CONFIGURATION CHARACTERISTICS	TERMINAL CONTROL OF THE ICD	TERMINAL CONTROL OF THE ICD (WITH HOST DATA FILES)	COMPUTER CONTROL OF THE ICD
Operation Mode	LOCAL	LOCAL	REMOTE
Controlling Device	Console Terminal	Console Terminal	Computer
Recommended Baud Rate (bps)	9600	TERMINAL = 9600 HOST/AUX = 4800	9600
Memory Storage Facility	ICD Internal Only	Computer	Computer
Computer's Role	Not Used	Memory storage, ZICE access	Controlling device, ZICE access, memory storage
Optional Target System?	Yes	Yes	Yes
Optional Printer?	Yes	Yes, if connected to computer	Yes, if connected to computer
Can Emulate CPU and NDP?	Yes	Yes	Yes
Number of RS-232 Cables Needed	1 (2 if printer is used)	2	1 (2 if printer is used)
Uses ZICE Software?	No	Optional	Yes
Is ZICE Software Used For ICD Interface?	No	No	Yes
Can Access ZICE Commands?	No	Yes	Yes

Summing It All Up . . .

- Your ICD can function in any of three different system configurations.
- Your ICD can be used to debug hardware or software.
- Your ICD can operate with or without a target system.
- Your ICD can dump data directly to a printer.
- Your ICD can dump data to a printer attached to a host computer.
- Your ICD can be controlled by just a terminal or by a host computer.
- Your ICD can be controlled by a terminal and use a separate host computer for storing data files.
- Your ICD can be controlled by a terminal and use a separate host computer for accessing the ZICE commands.

Now turn the page and read about preparing a site for your system.

System Preparation Read this chapter before you connect anything!

Grounds Your ICD is equipped with a 3-wire polarized receptacle that accepts a 3-wire cord. This cord connects to a power source and protective ground. Make sure that you plug the power cord into a properly grounded 115 VAC receptacle. Do not try to bypass the 3-prong plug with an adaptor (3- into 2-prong adaptor).

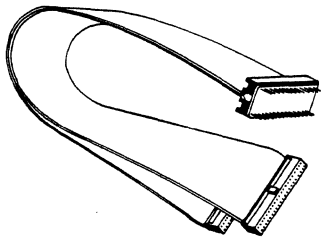
WARNING: THE GROUND TERMINAL OF THE 3-PRONG PLUG IS USED TO PREVENT SHOCK HAZARDS—DO NOT BYPASS IT!

Power Your ICD is normally set to operate on a voltage supply of 110-120 VAC, but this can be changed to 200-240 VAC by setting the Power Select switch to the 200V/240V position.

In most cases a multiple power outlet strip should be used to provide voltage to the entire system (host computer, terminal, printer, target system). Most power outlet strips are equipped with a circuit breaker in case of an overload, and all are properly grounded.

No matter what type of power source you use, **always** apply power after connecting the ICD to an electrical outlet, and always apply power in the same sequence: switch on the power supply first, and then press the POWER ON/OFF switch to ON.

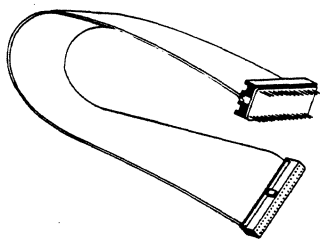
Important Facts About The CPU In-Circuit Probe



The CPU in-circuit probe is used to connect the ICD to your target system when you are emulating the 8086 or 8088 CPUs. The probe consists of a 20-inch ribbon cable with three end-connectors. The 40-pin connector end of the probe plugs into the target system's microprocessor socket. On the other end of the probe are two sockets which plug into the ICD's in-circuit probe receptacles. The sockets are labeled TOP and BOTTOM and **MUST** be placed in the corresponding top and bottom receptacles. THE LONGEST CABLE GOES INTO THE TOP RECEPTACLE.

CAUTION: DO NOT REVERSE PROBE CONNECTIONS. MISMATCHING THE TOP AND BOTTOM SOCKET CONNECTORS WILL CAUSE SEVERE DAMAGE TO THE ICD AND TARGET SYSTEM.

NDP (8087) In-circuit Probe



The NDP in-circuit probe is used to connect the ICD to your target system when you are emulating the 8087 Numeric Data Processor. This probe fetures a single cable and 40-pin connector and must be plugged into the receptacle labeled NDP.

NOTE: In many cases the NDP in-circuit probe may not need to be used because both the CPU and NDP are connected, internally, within the ICD. By adjusting the settings of the Emulation Method Select switch, the NDP in-circuit probe can usually be omitted, and NDP emulation can be performed using only the CPU in-circuit probe.

For more information on this subject, and to see if you can omit using the NDP in-circuit probe, see the chapter on "NDP Emulation," in Section 3.

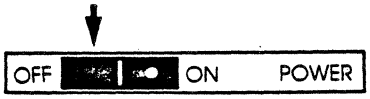
Now turn to the next page to learn how to prepare your ICD for operation.

Preparing Your ICD

Before you attach a system-controlling device (terminal or host computer) or your target system to the ICD, complete the following steps:

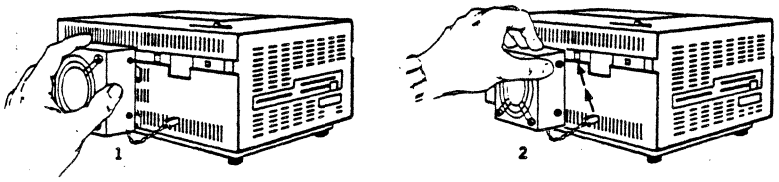
First—

Make sure that the POWER ON/OFF switch is set to OFF.



Now—

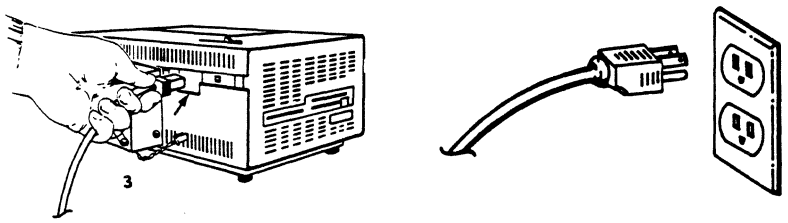
Attach the COOLING FAN to the ICD and plug the fan's connector into the receptacle labled DC OUT 24V.



Optional: The ICD's baud rates for the TERMINAL and HOST/AUX communication ports are factory-set at 9600bps. To change the baud rates, see "Changing The Baud Rate Settings," in Section 3.

Then—

Plug the AC POWER CORD into the ICD's power receptacle and connect the other end of the cable to a power source.



Now turn to the appropriate heading—which you'll find on one of the following pages—to construct *your* microprocessor development system.

System Configuration .....Terminal Control Of The ICD
Operation Mode .....LOCAL
Controlling Device .....Console Terminal
Optional Printer? .....Yes
Optional Target System? .....Yes
Number of RS-232 Cables Needed .....1 — 2 if printer is used
Recommended Baud Rates (bps) 9600
Uses ZICE Software? .....No

Use the illustration on the opposite page and the information below to construct this system configuration. Then adjust the switches as indicated in the bottom-right column.

CONSTRUCT YOUR SYSTEM

1) Connect your terminal to the ICD by using an RS-232 cable. Attach the cable from your terminal's serial (EIA RS-232) port to the ICD's TERMINAL port connector. The ICD defaults to 9600 baud, 8 data bits, 2 stop bits and no parity: set your terminal to these specifications.

2) [Optional] Connect your printer to the ICD by using an RS-232 cable. Attach the cable from your printer to the ICD's HOST/AUX port connector.

3) [Optional] If you're debugging a target system, remove the existing CPU (8086/8088) from your target system and insert the CPU IN-CIRCUIT PROBE (40-pin end) into the target system's CPU socket (pin 1 of the ICD's CPU in-circuit probe goes into pin 1 of the target system's CPU socket). Connect the other end of the CPU IN-CIRCUIT PROBE to the ICD's TOP and BOTTOM CPU in-circuit probe receptacles. THE LONGEST CABLE MUST BE CONNECTED TO THE TOP CPU IN-CIRCUIT PROBE RECEPTACLE.

If you're debugging a target system containing an NDP (8087) and you wish to use the NDP in-circuit probe, remove the existing NDP from your target system and insert the NDP IN-CIRCUIT PROBE (40-pin end) into the target system's NDP socket.

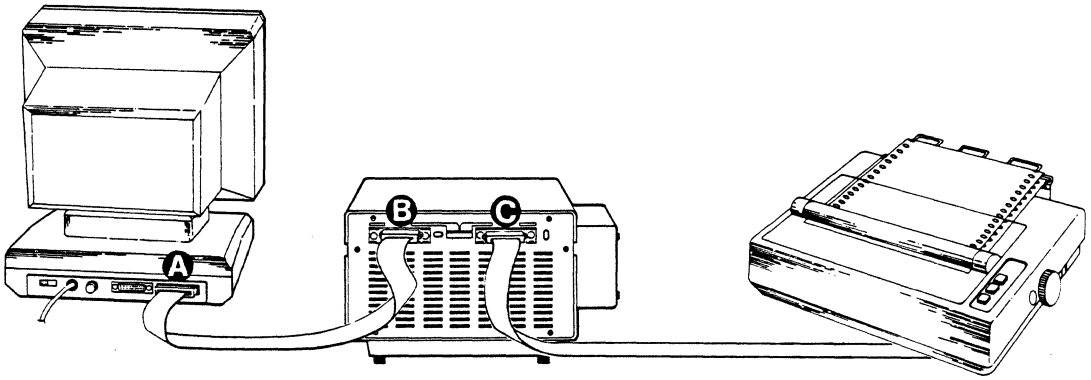
The following message should now appear on your monitor's screen (you may have to press the RESET switch on the ICD):

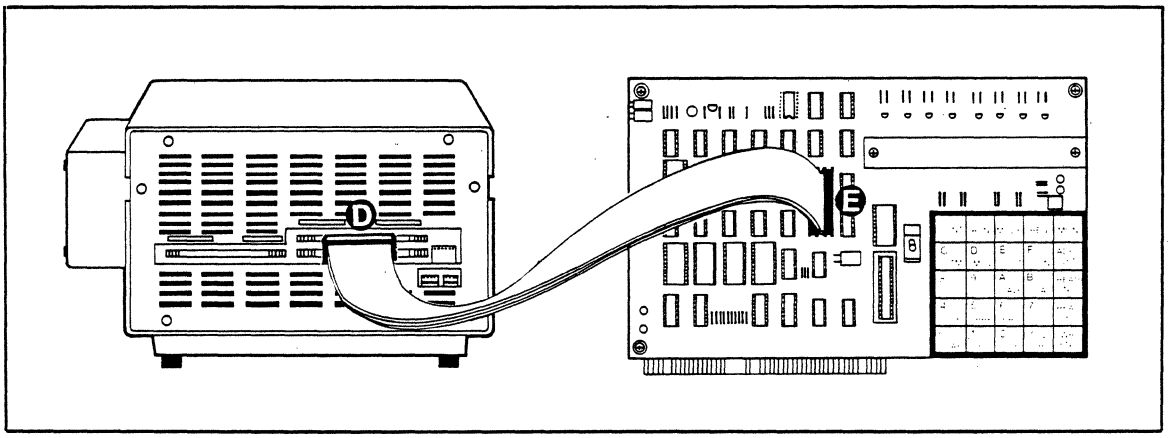
Now turn to "What Can You Do With Your MDS?" in this section.

ADJUST THESE SWITCHES

1	100V	117V
2	INT (EXT if target is connected)	
3	ON	
4	LOCAL	
5	DCE	

Ⓐ Terminal's EIA RS-232 port
Ⓑ ICD's TERMINAL port
Ⓒ ICD's HOST/AUX port
Ⓓ ICD's in-circuit probe receptacle (CPU/NDP)
Ⓔ Target system's CPU/NDP socket





OPTIONAL CONFIGURATION

System Configuration	**Terminal Control Of The ICD (With Host Data Files)**
Operation Mode	LOCAL
Controlling Device	Console Terminal
Optional Printer?	Yes
Optional Target System?	Yes
Number of RS-232 Cables Needed	2
Recommended Baud Rates (bps)	Terminal: 9600, Host: 4800
Uses ZICE Software?	Optional
Can Access ZICE Commands?	Yes

Use the illustration on the opposite page and the information shown below to construct this system configuration.

CONSTRUCT YOUR SYSTEM

1) Connect your terminal to the ICD by using an RS-232 cable. Attach the cable from your terminal's serial (EIA RS-232) port to the ICD's TERMINAL port connector. The ICD defaults to 9600 baud, 8 data bits, 2 stop bits and no parity: set your terminal to these specifications.

2) Connect your host computer to the ICD by using an RS-232 cable. Attach the cable from your host com puter's serial (EIA RS-232) port to the ICD's HOST/AUX port connector.

3) [Optional] If you're debugging a target system, remove the existing CPU (8086/8088) from your target system and insert the CPU IN-CIRCUIT PROBE (40-pin end) into the target system's CPU socket (pin 1 of the ICD's CPU in-circuit probe goes into pin 1 of the target system's CPU socket). Connect the other end of the CPU IN-CIRCUIT PROBE to the ICD's TOP and BOTTOM CPU in-circuit probe receptacles. THE LONGEST CABLE MUST BE CONNECTED TO THE TOP CPU IN-CIRCUIT PROBE RECEPTACLE.

If you're debugging a target system containing an NDP (8087) and you wish to use the NDP in-circuit probe, remove the existing NDP from your target system and insert the NDP IN-CIRCUIT PROBE (40-pin end) into the target system's NDP socket.

The following message should now appear on your monitor's screen (you may have to press the RESET switch on the ICD):

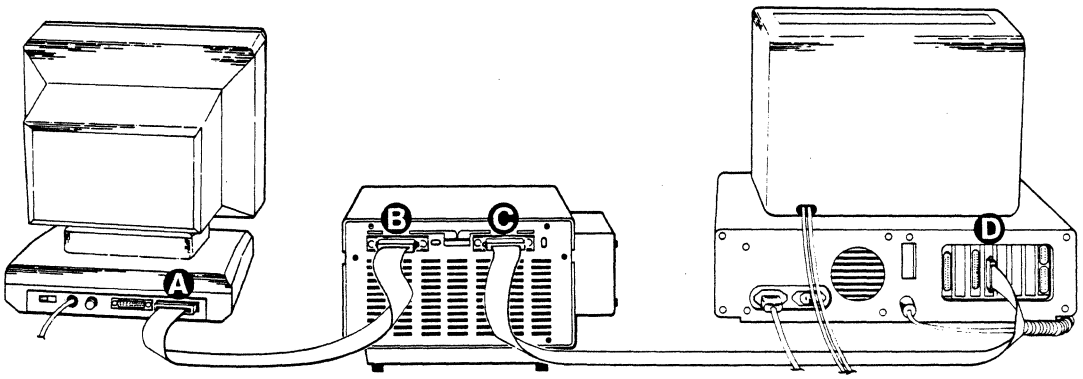
ICD-278 for i86 V2.0B

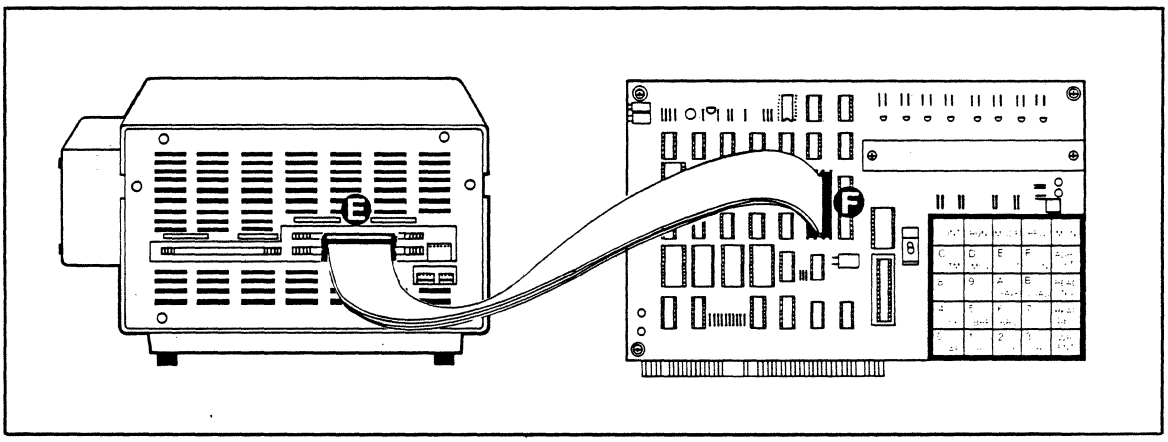
Now turn to "What Can You Do With Your MDS?" in this section.

ADJUST THESE SWITCHES

1	100V 117V
2	INT (EXT if target is connected)
3	ON
4	LOCAL
5	DTE (with ZAX's BOX). DCE (for personal computers)

Ⓐ Terminal's EIA RS-232 port
Ⓑ ICD's TERMINAL port
Ⓒ ICD's HOST/AUX port
Ⓓ ICD's in-circuit probe receptacle (CPU/NDP)
Ⓔ Target system's CPU/NDP socket





OPTIONAL CONFIGURATION

System Configuration .....Host Computer Control Of The ICD

Operation Mode .....REMOTE
Controlling Device .....Host Computer
Optional Printer? .....Yes
Optional Target System? .....Yes
Number of RS-232 Cables Needed .....1 — 2 if printer is used
Recommended Baud Rates (bps) .....Terminal: 9600, Host: 4800
Uses ZICE Software? .....Yes

Use the illustration on the opposite page and the information below to construct this system configuration.

CONSTRUCT YOUR SYSTEM

1) Connect your terminal to the ICD by using an RS-232 cable. Attach the cable from your terminal's serial (EIA RS-232) port to the ICD's TERMINAL port connector. The ICD defaults to 9600 baud, 8 data bits, 2 stop bits and no parity: set your terminal to these specifications.

2) Connect your host computer to the ICD by using an RS-232 cable. Attach the cable from your host computer's serial (EIA RS-232) port to the ICD's HOST/AUX port connector.

3) [Optional] If you're debugging a target system, remove the existing CPU (8086/8088) from your target system and insert the CPU IN-CIRCUIT PROBE (40-pin end) into the target system's CPU socket (pin 1 of the ICD's CPU in-circuit probe goes into pin 1 of the target system's CPU socket). Connect the other end of the CPU IN-CIRCUIT PROBE to the ICD's TOP and BOTTOM CPU in-circuit probe receptacles. THE LONGEST CABLE MUST BE CONNECTED TO THE TOP CPU IN-CIRCUIT PROBE RECEPTACLE.

If you're debugging a target system containing an NDP (8087) and you wish to use the NDP in-circuit probe, remove the existing NDP from your target system and insert the NDP IN-CIRCUIT PROBE (40-pin end) into the target system's NDP socket.

At this point, you will have to load the ZICE software program necessary for interfacing the ICD with your host computer. Execute the program loading commands as outlined in the ZICE software documentation.

The following message should now appear on your monitor's screen (you may have to press the RESET switch on the ICD):

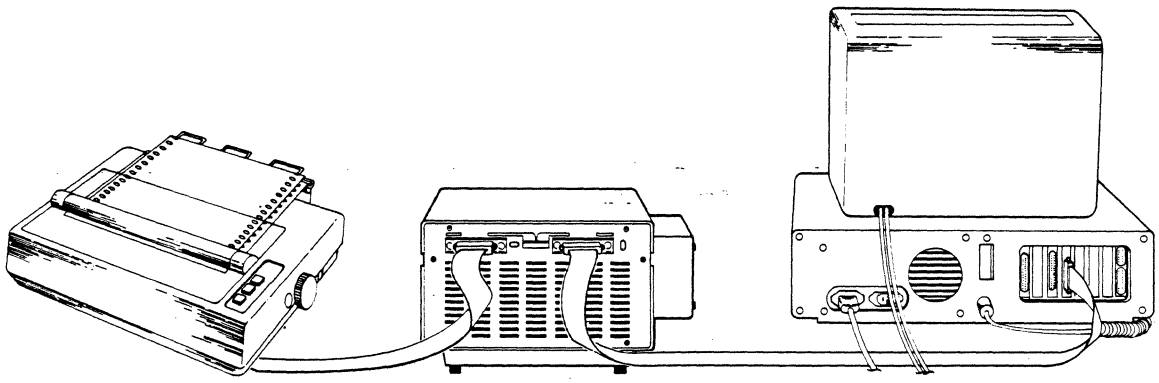
ICD-178 for i86 V2.0B

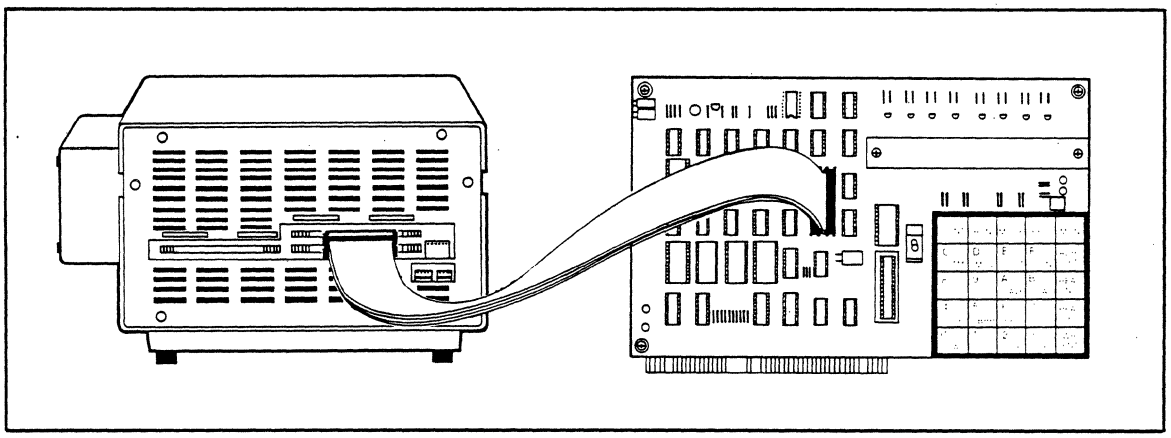
Now turn to "What Can You Do With Your MDS?" in this section.

ADJUST THESE SWITCHES

1	100V 117V
2	INT (EXT if target is connected)
3	ON
4	REMOTE
5	DTE (with ZAX's BOX), DCE (for personal computers)

Ⓐ ICD's TERMINAL port
Ⓑ ICD's HOST/AUX port
Ⓒ Computer's SIO port
Ⓓ ICD's in-circuit probe receptacle (CPU/NDP)
Ⓔ Target system's CPU/NDP socket



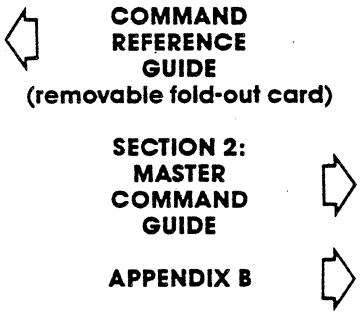


OPTIONAL CONFIGURATION

What Can You Do With Your MDS?

You should now have a fully operational Microprocessor Development System (MDS) capable of developing and debugging your hardware or software designs. If your MDS is functioning correctly, and the ICD's identification message appears on your monitor's screen, you can now:

- Turn to the "Master Command Guide," Section 2, for a complete analysis of your ICD's debugger commands.
- Turn to Appendix B for a demonstration of the features and functions of your ICD.
- Use the fold-out "Command Reference Guide" (from the front of this manual) as a source for the various command formats.



What To Do If Your MDS Is Not Working

If your MDS is not functioning correctly or gives you problems during emulation, turn to "Trouble Shooting," on the next page. Start by reading "Checking Electrical Connections," and then proceed to "Diagnosing ICD Interface Problems" if you encounter problems when you're emulating.

Trouble Shooting

Introduction: The Problem . . .

Your ICD must be controlled by either a separate terminal or a host computer's keyboard. Because you must connect the ICD to these external devices to form your development system, there's always the possibility of misplacing a cable, setting a switch to the wrong position, or bypassing a procedure.

. . . And The Solution!

"Trouble Shooting" is designed to get you through the problems you may have encountered in "How To Connect Your ICD To Other Devices." It begins with a typical example of what the ICD should do when the system is operating correctly. Then the ICD by itself is tested, followed by testing the ICD and terminal, together. ICD, terminal and target system configuration is then tested.

What Should Happen

When the ICD is connected to a terminal the following should happen:

When the ICD's POWER ON/OFF switch is pressed to ON, the PWR (power) and MONITOR lamps should come on and the external cooling fan should be operating. The terminal's monitor should then show the ICD's identification message:

ICD-178 for i86 V2.0B

If the ID message does not appear, try pressing the RESET switch. A prompt (>) should also appear, indicating that the system is working properly and the ICD is ready to accept commands. At this point, any of the "status commands" (command name followed by a RETURN) can be entered.

They include: B, EV, H, I, MA, O, PI, R, SU, T

Try entering a few of the status commands. If the response from the ICD is the command's status, then the system is probably functioning properly. Otherwise, continue reading and following the procedure outlined in this chapter.

How To Get Your ICD Working

In this trouble-shooting session, you'll start by disconnecting the ICD from all external devices such as the target system, host computer or terminal. You'll check the ICD by itself (just connect its power cord), and then attach a terminal. If that configuration works properly, you can connect your target system for final testing.

NOTE: If you're using a host computer to control the ICD, be sure to check the ICD and host computer operation (together) BEFORE connecting your target system.

Now begin with "Checking Electrical Connections."

Checking Electrical Connections

1. Press the ICD's POWER ON/OFF switch to OFF.
2. Turn the power OFF on all externally attached devices (terminal, host computer, target system, etc.).
3. Disconnect all externally attached devices from the ICD.
4. Unplug the AC power cord from the ICD and from the wall outlet or power supply.
5. Check the wall outlet or power supply by plugging in a working device (lamp, terminal, logic analyzer, etc.). If the outlet or power supply is controlled by a switch, is the switch ON?
6. Disconnect and reconnect each device's AC power cord to ensure a proper electrical connection.

Proceed with "Diagnosing ICD Interface Problems," on the next page.

Diagnosing ICD Interface Problems

ICD and External Cooling Fan

Connect the External Cooling Fan to the ICD and then connect the ICD's power cord to a voltage source.

PROBLEM:
The external cooling fan doesn't work.

SOLUTION:
What's Probably Wrong:
The fan is not getting power.

What To Do:
Make sure that the fan connector is firmly pressed into the ICD's fan receptacle and that the POWER ON/OFF switch is in the ON position.

The fan works but the lamps on the Operator Panel don't come on.

What's Probably Wrong:
There is an internal problem with the ICD.

What To Do:
Return the ICD for servicing.

ICD and Terminal

Before you begin, make sure your terminal is working properly (i.e., the curser on the screen should be visible). Then use an RS-232 cable to connect the ICD to the terminal.

PROBLEM:
The terminal does not respond at all when the RESET switch is pressed.

SOLUTION:
What's Probably Wrong:
There is either an interface problem or a defective component in the system.

What To Do:
First make sure that the RS-232 cable is firmly attached to both the ICD and terminal connectors. Is the cable defective? If the cable is OK, check that the INT/EXT clock switch is set to INT and that the LOCAL/REM switch is set to LOCAL. Make sure that both the ICD and terminal are transmitting at the same baud rates.

Terminal responds with "gibberish" when the RESET switch is pressed.

What's Probably Wrong:
The baud rates for the ICD and terminal are different.

What To Do:
Make sure that the baud rates for the ICD and the terminal are the same (your ICD's baud rate was factory-set at 9600).

Terminal responds with a C?> error message when any of the commands are entered.

What's Probably Wrong:
On some terminals the ICD will only recognize a command that is stated with capital letters (e.g. R not r).

What To Do:
Press the Lock or Caps Lock key on your keyboard to the locked position.

If you've reached this point with no problems, your difficulty probably lies in the ICD failing to emulate your target system. Now connect the ICD to your target system and read through the next check-out procedure.

ICD With Target System Connected

Connect the target system to the ICD, using the CPU in-circuit probe. Use a terminal to control the ICD.

PROBLEM:
Terminal doesn't work

SOLUTION:
What's Probably Wrong:
There is either an interface problem or a defective component in the system.

What To Do:
Check that the ICD is properly connected to your target system, that the target system has power, and that the terminal is adjusted correctly. Select the EXTERNAL (EXT) clock, and press the RESET switch on the ICD. The ICD's identification message and prompt should appear. If a prompt fails to appear when the clock is set to EXT, switch to the INTERNAL (INT) clock and press RESET again. (With INT selected, the ICD and terminal should work independently of your target system.)

If the ICD operates on the INT setting, the problem is probably a poor clock signal from your target system. It is possible to use the ICD with the INT setting, but you will lose real-time operation.

What To Do If The ICD Still Doesn't Work

In most cases, the procedures just listed will solve all but the most stubborn problems. However, it is possible that the ICD is still not functioning correctly. If this is the case, you should consult directly with **ZAX** Corporation.

More About Your ICD

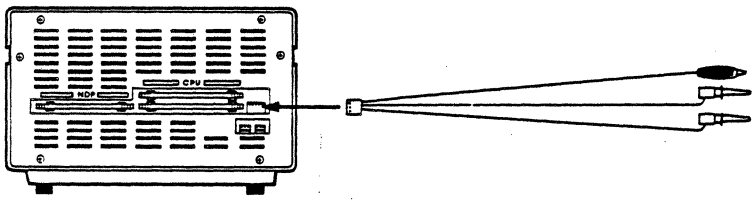
Introduction

In this chapter, you'll learn how to use the accessories that come with your ICD (#1). You'll also learn what the Emulation Method Select switch does. By using the accessories and adjusting the settings on the Emulation Method Select switch, you'll be able to further expand your ICD's debugging capabilities.

The two accessory cables can be used to input and output pulses to and from the ICD. By using the four probes that are attached to the ends of these cables, you can:

- Determine if the ICD is emulating.
- Cause a breakpoint in your program to output a pulse to an external device.
- Selectively access either ROM or RAM.
- Cause the ICD to insert a break in your program when an external pulse is sensed.

The Emulation Method Select switch lets you: suppress output signals from the ICD, interface the NDP (8087) to the host CPU (8086/8088), and insert wait states.



Accessory Cables & Probes

To use the probes, see the chart on the opposite page. Plug the connector with the appropriate colored probe into the ICD's EXT.BRK. or EVENT TRG. receptacle, and then connect the probe to the desired peripheral device.

Probe Functions

Probe Name	Probe Color	Probe Location	What The Probe Does	How It's Used
Emulation Qualify	WHITE	BLUE wire of the Event Trigger cable	Outputs a HIGH level signal from the ICD to the Emulation Qualify probe during emulation. During the MONITOR mode (breakpoint encountered or MONITOR button pressed) the signal level is LOW.	The EQ signal can be used as an "emulation in progress" indicator or to remove unwanted signals during emulation.
Event Trigger	GREEN	BLUE wire of the Event Trigger cable	Outputs a LOW level signal from the ICD to the Event Trigger probe when an event point is passed during emulation.	The Event Trigger output is useful when a timing analysis of some external circuitry (not controlled by the ICD) is desired. In this application, the LOW level signal could be used to trigger a logic analyzer or oscilloscope.
Map Control	YELLOW	RED wire of the Extenal Break cable	Accepts a LOW level input signal from the target system to dynamically select between ROM and RAM. A LOW level input signal causes the ICD to set all memory as user (target) memory.	The ROM/RAM selection process is helpful when developing a system which uses phantom ROM (ROM that operates for the system bootsrap procedure and then hides behind the main memory). The Map Control signal lets you access the same user memory address space that is occupied by the phantom ROM.
External Break	RED	RED wire of the External Break cable	Accepts a LOW level input signal from an external component to trigger a break during the program execution.	The External Break input is useful in capturing information (usually on the hardware level) that exists outside of the control of the microprocessor.

Emulation Method Select Switch

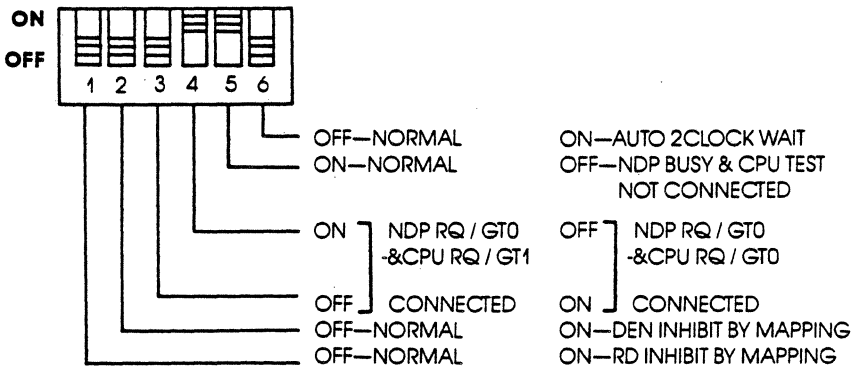
Description The Emulation Method Select switch is a 6-bit, ON/OFF type switch.

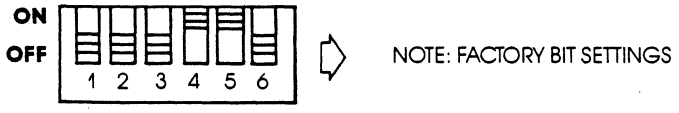
Location The E.M.SEL switch end of the ICD. (See "The Controls And Component Functions Of Your ICD," in this section.)

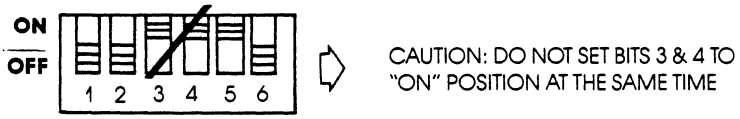
Function Bit 1 of the Emulation Method Select switch sends or suppresses the RD signal; bit 2 sends or suppresses the DEN signal; bits 3 and 4 internally connect the RQ/GT lines of the NDP (8087) to the RQ/GT lines of the CPU (8086/8088); bit 5 connects the BUSY signal of the NDP to the TEST signal of the CPU processor; and bit 6 inserts 1, 2 or 3 wait states into each machine cycle.

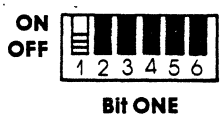
Application See the individual bit settings that follow.

Using The Emulation Method Select Switch Set the bits to the ON or OFF position with a small, pointed tool.





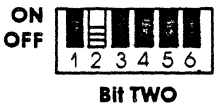




ON Suppresses the ICD's RD signal to the target system.

OFF Outputs the RD signal to the target system. (This is the normal setting.)

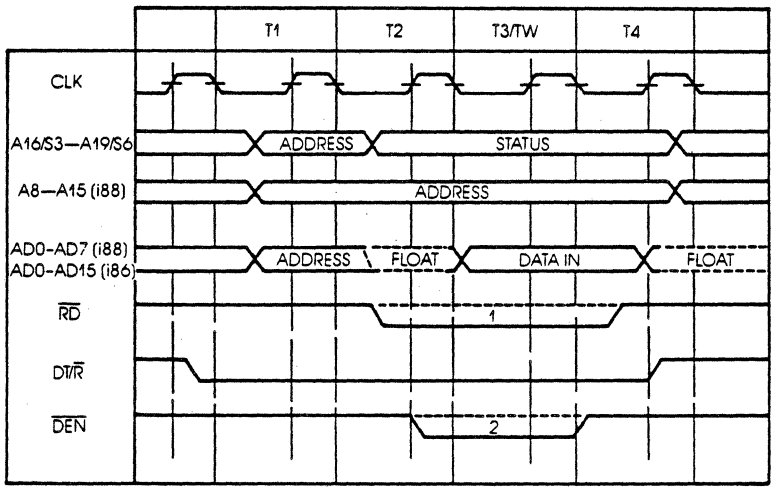
NOTE: RD signal (Active Low-Level Output) indicates that the processor is performing a memory or I/O read cycle.

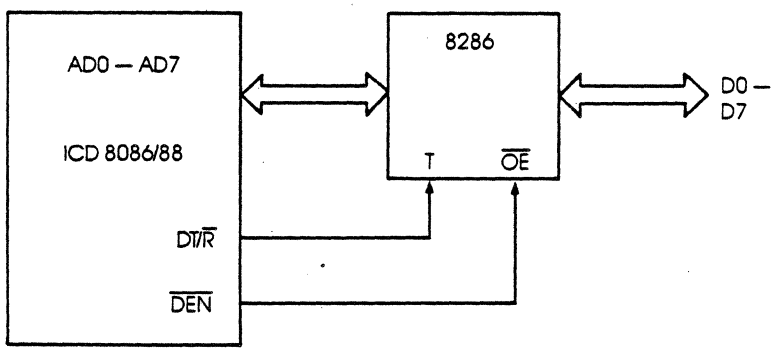


ON Suppresses the ICD's DEN signal to the target system.

OFF Outputs the DEN signal to the target system. (This is the normal setting.)

NOTE: The DEN signal is used to enable the data bus transceiver, and is generated by the ICD when the target system is in the minimum mode.





Bit THREE

3 OFF
4 ON

Connects the NDP RQ/GT0 signal lines to the CPU RQ/GT1 signal lines. This forms an internal connection between the NDP and CPU request/grant lines.

Bit FOUR

3 ON
4 OFF

Connects the NDP RQ/GR0 signal lines to the CPU RQ/GR1 signal lines. This forms an internal connection between the NDP and CPU request/grant lines.

NOTES: The ICD accepts the RQ/GT signals in the in-circuit modes I1 and I2. These signals allow the bus operation to be performed regardless of an emulation break, in which the target system executes a local bus operation by RQ/GT.

It is possible to enable or disable the RQ/GT signals from the target system by using the PIN command. (See the PIN command in the "Master Command Guide," Section 2.)

Facts about the 8086's Request/Grant signals:

The following pin function description is for the 8086 processor operating in the maximum mode (i.e., MN/MX = Vss).

Signal symbol: RQ/GT0, RQ/GT1
Type: I/O
Pin No. 30, 31

The Request/Grant pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bi-directional, with RQ/GT0 having higher priority than RQ/GT1. RQ/GT has an internal pull-up resistor, so it may be left disconnected. The request/grant sequence is as follows:

1. A pulse 1 CLK wide from another local bus master indicates a local bus request ("hold") to the 8086 (pulse 1).

2. During a T4 or T1 clock cycle, a pulse 1 CLK wide from the 8086 to the requesting master (pulse 2) indicates that the 8086 has allowed the local bus to float. The pulse also indicates that the 8086 will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge."

3. A pulse 1 CLK wide from the requesting master indicates to the 8086 (pulse 3) that the "hold" request is about to end, and that the 8086 can reclaim the local bus at the next CLK.

Each master-master exchange of the local bus is a sequence of 3 pulses. There must be one dead CLK cycle after each bus exchange. Pulses are active LOW.

If the request is made while the CPU is performing a memory cycle, it will release the local bus during T4 of the cycle, when all the following conditions are met:

1. Request occurs on or before T2.
2. Current cycle is not the low byte of a word (on an odd address).
3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence.
4. A locked instruction is not currently executing.

Facts about the NDP's Request/Grant signals.

Signal symbol: RQ/GT0
Type: I/O

The RQ/GT0 pin is used by the NDP to gain control of the local bus from the CPU for operand transfers or on behalf of another bus master. It must be connected to one of the two processor request/grant pins. The request/grant sequence on this pin is as follows:

1. A pulse one clock wide is passed to the CPU to indicate a local bus request by either the NDP or the master connected to the NDP's RQ/GT1 pin.

2. The NDP waits for the grant pulse and, when it is received, will either: initiate bus transfer activity in the clock cycle following the grant; or pass the grant out on the RQ/GT1 pin in this clock if the initial request was for another bus master.

3. The NDP will generate a release pulse to the CPU one clock cycle after the completion of the last NDP bus cycle, or on receipt of the release pulse from the bus master on Q/GT1.

Signal symbol: RQ/GT1
Type: I/O

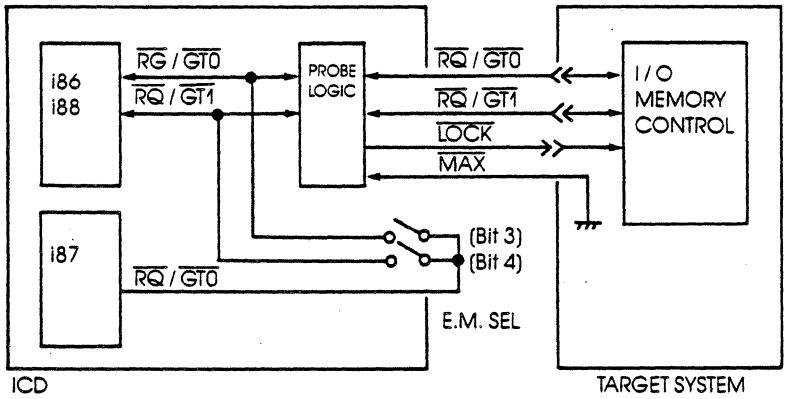
The RQ/GT1 pin is used by another local bus master to force the NDP to request the local bus. If the NDP is not in control of the bus when the request is made, the request/grant sequence is passed through the NDP on the RQ/GT0 pin one cycle later. Subsequent grant and release pulses are also passed through the 8086 with a 2- and 1- clock delay, respectively, for resynchronization. RQ/GT1 has an internal pull-up resistor, and so may be left disconnected. If the NDP has control of the bus, the request/grant sequence is as follows:

1. A pulse 1 CLK wide from anothr local bus master indicates a local bus request to the NDP (pulse 1).

2. During the NDP's next T4 or T1, a pulse 1 CLK wide from the NDP to the requesting master (pulse 2) indicates that the NDP has allowed the local bus to float, and that it will enter the "RQ/GT acknowledge" state at the next CLK. The NDP's control unit is disconnected logically from the local bus during "RQ/GT acknowledge."

3. A pulse 1 CLK wide from the requesting master indicates to the NDP (pulse 3) that the "RQ/GT request" is about to end, and that the NDP can reclaim the local bus at the next CLK.

Each master-master exchange of the local bus is a sequence of 3 pulses. There must be one dead CLK cycle after each bus exchange. Pulses are active LOW.



ON Connects the NDP BUSY signal to the CPU TEST signal. This forms an internal connection between the NDP and the CPU. (This is the normal setting.)

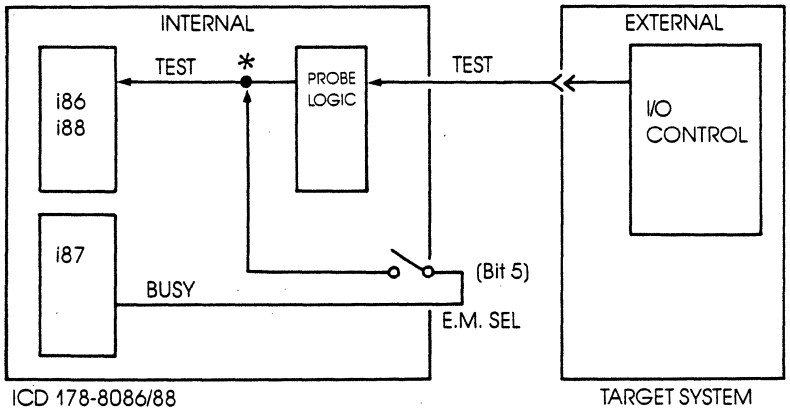
OFF Disconnects the NDP TEST signal and the CPU BUSY signal internally.

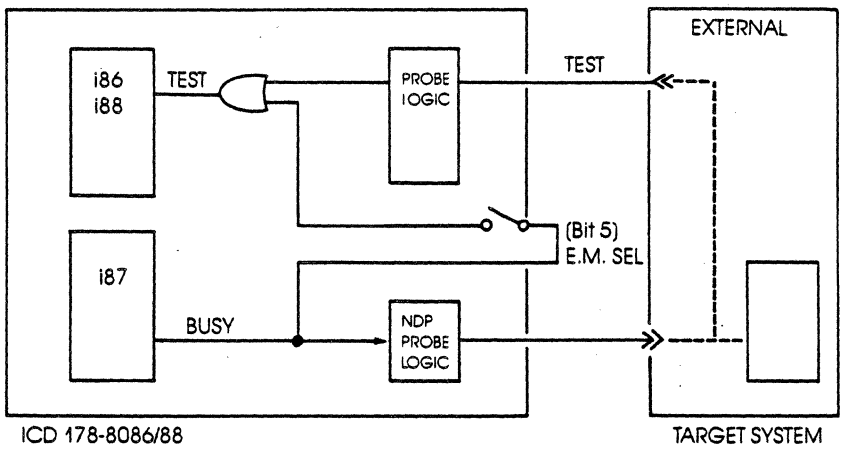
NOTES: TEST signal (Active Low Input) is examined by the "WAIT" instruction. If the TEST input is low, execution continues; otherwise, the processor waits in an idle state. This input is synchronized internally during each clock cycle on the leading edge of clock.

The TEST input of the target system is "ORed," with the BUSY signal of the NDP to be used as the TEST input to the ICD's CPU. The ICD is able to accept the TEST signal when the ICD is operating in the I1 or I2 in-circuit mode.

BUSY Signal (Active High Output): Whenever the NDP executes a numeric instruction the BUSY line is activated. The BUSY signal of the NDP can be used for the target system when the in-circuit mode is I1 or I2.

WAIT Instruction. The CPU interprets the WAIT instruction as "wait while TEST is active." The CPU examines the TEST pin every 5 clock cycles; if the TEST is inactive, execution proceeds with the instruction following the WAIT. If TEST is active, the CPU examines the pin again. The effective execution time of a WAIT can range from 3 wait states (3 wait states are required for decoding and setup) to infinite—as long as TEST remains active. The WAIT instruction then prevents the CPU from decoding the next instruction until the NDP is not BUSY. The instruction following a WAIT is decoded simultaneously by both processors.





ON A 2-clock wait state is automatically inserted in every memory of I/O cycle.

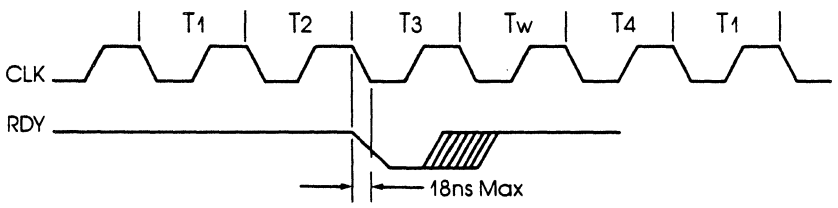
OFF No wait state is generated. (This is the normal setting.)

NOTES: A WAIT condition is generated to the CPU by controlling the READY input.

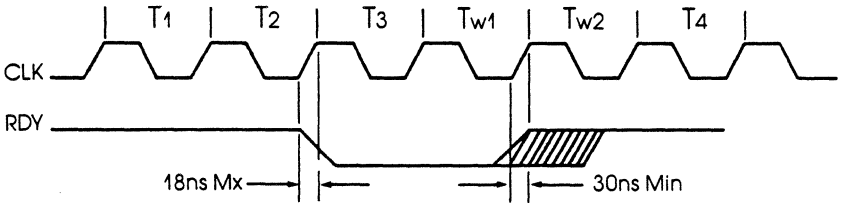
READY (Active High Input): There are two modes of operation for this signal: Active READY and Inactive READY. An Active READY will automatically add an additional clock cycle to a current bus cycle.

The READY signal must go low before the leading edge of T3. The WAIT cycle will appear after T3.

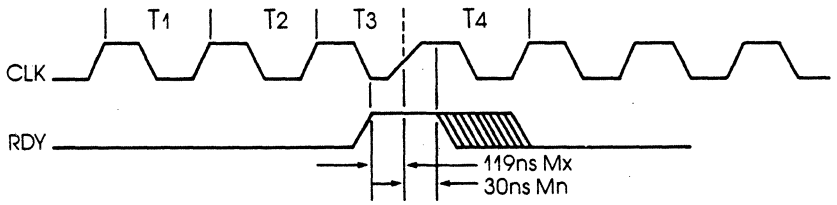
ONE WAIT CYCLE



TWO WAIT CYCLES



NO WAIT CYCLES



NOTE: More information is available on the BUSY, TEST and RQ/GT signals. See the chapter on "NDP Emulation," in Section 3.

ICD COMMANDS

Program Control

GO—Starts the program execution

BREAK—Stops the program execution on a variety of different parameters

EVENT—Signals an event in the program, triggers the trace feature, or sends out an external signal at a point in the program

HISTORY—Records the program execution in real time, and then displays it in either machine or disassembled format

TRACE—Displays program execution in non-real time

NEXT—Displays "n" instruction lines as executed in non-real time

OFFSET—Sets an offset in the emulator for relative program addressing

Memory Control

ASSEMBLE—Converts the mnemonics entered from the keyboard to machine language in memory

DISASSEMBLE—Converts the memory contents to assembly language mnemonics

DUMP—Displays the memory contents in hexadecimal/ASCII format

ALLOCATION—Arranges ICD's available memory into the selected CPU memory space

COMPARE—Compares the memory contents and displays the non-matching data

MOVE—Moves the memory contents between the ICD and the target system

EXAMINE—Examines one or more memory locations and optionally modifies them

FILL—Fills the memory contents with data

SEARCH—Searches the memory contents for either matched or unmatched data

Debug/ Emulation Control

REGISTER—Displays or changes the registers' data

SUPERVISOR—A "system call" to allow access to the ICD's serial input/output ports

PRINT—Sends the display to a printer

PIN—Enables or disables selected CPU input signals

PORT—Examines one or more I/O port locations and optionally modifies them

IDENTIFICATION—Identifies the type of emulator in use and the firmware version

IN-CIRCUIT—Sets the ICD mapping mode

USER—Allows one terminal to communicate with both the ICD and a host computer

MAP—Sets the ICD/target system memory map

CALCULATION—Performs subtraction, addition, and conversion of hex and decimal data

Host & File Handling Commands

LOAD—Loads an Intel Hex file from the host computer to the ICD memory

SAVE—Saves an Intel Hex file to the host computer

VERIFY—Checks a file in the host computer against a file in the ICD

†HOST—Initiates or terminates LOCAL "Host Computer Assisted" mode

†QUIT—Exits ZICE control and returns control to the host computer operating system

† Available with ZICE software only.

Introduction

ZAX ICD-series emulators respond to commands which you enter from a console terminal or host computer. The commands enable the ICD to perform a variety of complex debugging tasks for you. In this section, you'll learn how to use the debugger commands and how to perform actual debugging and development operations.

In order to use the commands effectively, you'll need to become familiar with three different areas:

- **The language needed to implement the commands**
- **What each command does**
- **How to use the commands to perform debugging or development operations**

Command Language

All ZAX ICD-series emulators execute operations in response to "command statements" made up of the "command name" and "parameters." The command name refers to a symbol or group of symbols that designate the basic emulation operation to be performed (e.g., G for GO, MA for MAP, T for TRACE, etc.). Parameters refer to any additional information that complements the command name, such as a specific address, an address range, or a base value. Together, the command name and the parameters can be combined to execute a variety of complex debugging operations.

The control firmware within the ICD requires that the command statements be entered in a concise and logical manner, and that all required elements of the command statement be used. The elements of the command statement are described on the next page. The elements shown here represent all possible items within a command statement. Of course, not all commands require the presence or absence of each element.

Elements Within A Command Statement

The Prompt Character. The prompt character lets you know that the ICD is ready to accept a command statement. The prompt character is supplied by the ICD—you do not enter it—and is always displayed on the left side of the console's screen.

Example of prompt character: >

The Command Name. Commands are represented by the first, or first two, letters of the command name. The commands are displayed by upper-case typeface and should be entered using capital letters.

Examples of command names: B (for BREAK), CO (for COMPARE), SA (for SAVE)

Command Qualifiers. The slash key (/) acts to signal a qualifier for the command whenever it appears immediately following the command mnemonic.

Examples of qualifiers: B/O B/E F/W

The Space Character. The space character is an invisible character that not only improves the readability of a sentence, but in the case of the command format, it is recognized as a delimiter for the command name. Spaces must be interpreted from the command format; there is no symbol used to indicate spacing.

Example of space character in use: EV ON

In this example, the space between EV and ON allows the ICD to interpret EV as the EVENT command, and ON as a directive to enable the command.

Keywords are items which you must enter as shown. These items are displayed by upper-case typeface, but usually any combination of upper-case or lower-case letters may be used to enter them.

NOTE: Some terminals must use upper-case letters only. If the ICD responds with an error message, try using upper-case letters.

Examples of keywords: UP EN LO ON OFF

User-Supplied Items. Lower-case letters in *italic typeface* show items which you may supply; these are called user-supplied items.

Examples of user-supplied items include the name of your *file* (TEST.HEX), **a *beginning address*** (0), **an *ending address*** (3FF), **a *comparison address*** (100), **and *data*** (55).

Address and Data Parameters. The command numerical parameters for the ICD commands are described below:

addr, beg_addr, comp_addr, mov_addr, stop_addr, search_addr = Hexadecimal numbers in 16 bits (0-FFFF), relative to the current code segment (CS:) or two 16-bit addresses arranged as a logical address (0000:FFFF), where a segment register name may be used to indicate its current value (e.g., ES:1234). With some commands, a 20-bit (0-FFFFF) physical address may be used. These parameters specify a memory address with 16-bit or 20-bit hexadecimal characters. These parameters can be specified in an addition or subtraction equation, or a bias can be added if offset registers (0, 1, 2, or 3) are provided.

"Don't care" conditions may be specified for the BREAK and EVENT commands, on a bit or nibble basis, by entering X at the desired position. Examples include:

X1A3X—**Don't care condition in hexadecimal notation. May be specified in 4-bit units (0-F, or X).**

1000_101X_X1XX_010X_1XX0—**Don't care condition in binary notation. May be specified in 1-bit units (0, 1, or X).**

end_addr = Hexadecimal numbers in 16 bits (0-FFFF) for logical address, 20 bits for physical address, or number of bytes in 16 bits (0-FFFF).

NOTE: The byte format is: Lnnnn where nnnn = (0-FFFF).

data, mod_data, and search_data = Hexadecimal/binary number in 8 or 16 bits (0-FFFF). These parameters can be specified in an addition or subtraction equation, but the offset registers cannot be used.

"Don't care" conditions may be specified for the EVENT command, on a bit or nibble basis, by entering X at the desired position. Examples include:

7X—**Don't care condition in hexadecimal notation. May be specified in 4-bit units (0-F, or X).**

01XX-X001—**Don't care condition in binary notation. May be specified in 1-bit units (0, 1, or X).**

The Equal Sign. The equal sign (=) causes the value or information on its right to assume a relationship with the value on its left.

Example of the equal sign: P 100=55

In this example, the ICD does not display anything in response to this entry, but the value entered on the right (which represents a data value of 55H) is now assigned a relationship with the value on the left (an address value of 100H).

The Comma Character. The comma character (,) is used to separate parameters when more than one parameter is required to form a command statement.

Example of the comma character: DI 0,100

NOTE: A space may be substituted for a comma (e.g., DI 0 100=DI 0,100*), but a space cannot be used where a comma acts as the separator (e.g.,* DI 0, 100*).*

Brackets. Items in square brackets ([ ]) are optional. If you choose to include the information, you should not enter the brackets, only the information inside the brackets.

Examples of brackets: [D=*data*] [,*bias*]

The Return Key. The return key is used to terminate statements and execute commands, and it must be entered after every statement. It is assumed that the return key must be pressed after the command statement is entered; there is no symbol used to indicate the return key in the command format.

NOTE: Other parameters are defined and explained in each command. See **Terms** *and* **Notes** *for an explanation about these parameters.*

Example Of The Command Format

Each command is presented in the same format as shown below. This format makes it easy to find the name of a command and what it does, and then how to enter it correctly. An example (sometimes more than one) shows how the command is used in a debug/development session.

The example below illustrates the DUMP command and includes many elements of a typical command statement. This command is also used as the syntax example in "How To Enter A Command."

Command ① DUMP

Operation ② Displays the memory contents in both hexadecimal and ASCII code.

Syntax ③ [/W]*beg_addr*[,*end_addr*]

Terms ④ W = Displays the memory contents in word units arranged in MSB/LSB order (default is byte units).

beg_addr = Beginning address of display.

end_addr = Ending address of display.

Syntax Example ⑤

```
D/W  100,1FF
D  120
```

Notes ⑥ The *end_addr* is an optional parameter. If it is omitted, 16 bytes are displayed starting with *beg_addr*.

Command Example ⑦ See **Syntax Example** above. The first example shows that the memory contents are displayed in word units, beginning with address 100 and ending with address 1FF. The second example shows that the last 16 bytes are displayed beginning at address 120.

Explanations

① **Command Name.** The command name is always found at the top of the page. If a command performs more than one task, a description of the various command functions can be found after the command name, for example, "OFFSET: Specification" and "OFFSET: Status."

② **Operation** describes the action of the command, and emulation practices and principles that involve the command.

③ **Syntax** shows the characters and elements that are needed to implement the command. However, the characters and elements in **Syntax** may not provide enough information in themselves to correctly enter the command (the parameters may only represent an address or data value). The information in **Terms** should then be used to define the parameters.

④ **Terms** describes the characters and elements used in **Syntax**. The lower-case characters in *italic typeface* show items which you must supply. Upper-case characters show what these items are and how they should be entered.

⑤ **Syntax Example** shows how the command might be entered using various characters and elements, and the correct spacing between them.

NOTE: If a command cannot be entered, or the ICD responds with an error message, try entering the example shown in **Syntax Example.**

⑥ **Notes** explains important facts about the command. It usually contains information about the parameters shown in **Terms**, or it may include an explanation of how the command is used in a debug/developmernt application. **Spacing** describes the correct spacing of the elements of the syntax.

⑦ **Command Example** shows how the command might be used in an actual debug/development session.

How To Enter A Command

Before you can enter a command, you'll need to know what operation(s) the command performs. This can be found in two different places: "ICD COMMANDS" and "HOST & FILE HANDLING COMMANDS," which is shown on the first few pages of this section, and **Operation**, found in the Command Format.

After selecting the command, examine the information in **Syntax** and **Terms**. Enter the parameters needed to perform the task you desire. Examine the **Syntax Example** to see the proper spacing and how the characters and elements are used. An example of this procedure is shown below using the DUMP command.

Command Example

The syntax for the DUMP command is:

D[/W] *beg_addr*[,*end_addr*]

The terms used in the syntax are:

W = Display the memory contents in word units (default is byte units).

beg_addr = Beginning address of display.

end_addr = Ending address of display.

Entering The Command Example

To use this command, first enter D (the mnemonic for DUMP). Now decide (after examining the definitions in **Terms**) if the memory contents should be displayed in word or byte units. Since W is in brackets, it represents an optional parameter (if it was omitted, the display would be in byte units). For this example, we'll use a word display and enter W, preceded by a slash, and followed by a space. The first user-supplied item is the *beginning address* for the display (we'll supply the value of 100). The next item is an optional (because it's in brackets, [ ]) *ending address*. In this example we'll specify 1FF for this parameter, preceded by a comma (,).

At this point, the display on the console's screen should look like:

```
>D/W  100,1FF
```

This input now forms a command statement, complete with the command mnemonic, usable parameters, elements, and proper spacing. To send the command statement to the ICD for execution, press the return key on your keyboard.

What To Do If You Make An Input Error

If you make an error when entering a command statement, merely backspace over the error (which cancels the character) and enter the new information. You can also press the Delete (Del) key, which not only cancels out the error, but displays the cancelled character as well.*

If you've already entered a command statement into the ICD but you meant something else, press Ctrl-U (Control-U),* then just re-enter the correct command statement, and the ICD will execute the latest command.

**NOTE: These features are available in the LOCAL mode only (i.e., when a console terminal is used to control the ICD directly).*

ERROR MESSAGES If you enter a parameter incorrectly, use an invalid address, or forget to use a space at the appropriate place, the ICD will respond with an error message. The error messages and causes are shown below and on the back of the fold-out Command Reference Guide.

Error Message	Displayed when
C?>	an unrecognizable command is entered
P?>	a parameter code error occurs
/?>	a modifier code error occurs
**Break Busy	the break specification exceeds the limit
**Unable Soft Break	a software break is set at the address presently not mapped in RAM
**Multi-Break Address	a software break is set at the same address
**Input Error	an input error occurs
**Check Sum Error	a check sum error occurs
**File Name Error	a parameter code error occurs with the LOAD or VERIFY commands
**Not Local Mode	a LOCAL mode command is used when the system is in the REMOTE mode
**Not Remote Mode	a REMOTE mode command is used when the system is in the LOCAL mode
**Memory Write Error at *####*	there is a memory modification error
**I/O Timeout Error at *####*	a timeout error occurs at a specific address
**Memory Timeout Error at *####*	memory or I/O in the target system does not respond to an ICD access
**Memory Guarded Access Error at *####*	when a user program attempts to access an area mapped as NO memory
**Software Break Instruction Misrecovered at *####*	an error has occurred while attempting to replace original contents of a software break location

NOTE: #s refer to address locations in the program.

ALLOCATION

Command ALLOCATION: Status

Operation Displays a logical block or sequence of blocks by the address range and by the corresponding beginning physical block number for the block series.

Syntax AL

Command Example See the "ALLOCATION: Specification" command.

ALLOCATION

Command ALLOCATION: Specification

Operation Allocates any 1K-byte block from the ICD program memory to any address space in the CPU's memory.

Syntax AL *beg_addr*[,*end_addr*]=*block_no*

Terms *beg_addr* = Transposes the supplied address to the first 1K-block address space where the address resides.

end_addr = Transposes the supplied address to the last 1K-block address space where the address resides.

block_no = The allocation beginning block number (range = 0 to 75H).

Syntax Example AL 0,1FF=3F

Notes Ending Address. If the *end_addr* exceeds the first 1K-block specification, the end of the entire block in which the address resides will be assigned as the ending address. For example:

```
>AL 0,400=5      ← 400 EXCEEDS FIRST 1K-BLOCK SPECIFICATION.
                   END OF SECOND 1K BLOCK NOW ASSUMES POSITION
                   AS THE ENDING ADDRESS. ALLOCATION STATUS
                   WILL THEN SHOW:

>AL
0000—0007FF = 005   ← FIRST ALLOCATION BLOCK IS NOW
00800—FFFFF = 002     ACTUALLY 2K LONG
>
```

Block Number. Each 1K block of memory that is available in the ICD is assigned a sequential block number beginning with 0; these physical memory blocks may be allocated to any logical block address. The block number parameter represents the beginning of the block, but if the allocation beginning and

ALLOCATION

ending address parameters define more than one logical block, sequential block numbers will be assigned to each subsequent logical block. It is possible to assign more than one logical block address to a single physical memory block, so beware.

The block number range depends on the amount of emulation memory in the ICD:

128K memory: range = 0 to 7FH
256K memory: range = 0 to 0FFH
384K memory: range = 0 to 17FH
512K memory: range = 0 to 1FFH
1M memory: range = 0 to 3FFH (maximum)

Spacing: A space is required between AL and *beg_addr*. No spaces are permitted after *beg_addr*.

Command Example

Press the RESET switch on the ICD to initialize the allocation block number to 0, then enter:

```
>AL          ← DISPLAYS THE CURRENT ALLOCATION STATUS
00000—FFFFF = 000          ← SHOWS ALL MEMORY TO BE UNALLOCATED
>
>AL 0,3FF=5          ← ASSIGNS THE FIRST 1K MEMORY SPACE TO BLOCK #5
>AL          ← DISPLAYS NEW ALLOCATION STATUS
00000—003FF = 005          ← SHOWS BLOCK #5 MEMORY ASSIGNMENT
00400—FFFFF = 001
>AL 400,8FF=12          ← ASSIGNS NEXT 4FFH MEMORY SPACE TO BLOCK #12
>AL          ← DISPLAYS NEW ALLOCATION STATUS
00000—003FF = 005
00400—00BFF = 012          ← NOTICE HOW MEMORY RANGE
00C00—FFFFF = 003            INCLUDES NEXT 1K BLOCK AS WELL
>
```

ASSEMBLE

Command ASSEMBLE

Operation Translates simple-to-understand mnemonic instructions into machine language. The opposite translation (machine language to assembly language mnemonics) is accomplished using the DISASSEMBLE command.

Applications Note: The In-Line Assembler in the ICD is a powerful software tool that can be used for writing patches into program code that has either been downloaded from a host computer or originated in the target system. This feature also allows you to quickly write your own routines, develop small programs, etc.

Syntax

A *mem_addr* <cr>
xxxx:xxxx (8086/8088 assembly instruction) <cr>
xxxx:xxxx <cr>

Terms *mem_addr* = The beginning memory address where assembled code is stored. The logical address assumes the current code segment is used unless otherwise specified.

xxxx:xxxx = The next storage location.

8086/8088 assembly instruction = The mnemonic instruction to be assembled and stored. Operand may include number or .sym (.sym value must be pre-defined).

<cr> = Exits the assemble mode.

Syntax Example >A 100

Notes All number operands are assumed to be decimal unless specified as hexadecimal.

Spacing: A space is required between A and *mem_addr*. A space is required between opcode and operand of mnemonic instruction (no tab).

ASSEMBLE

Command Example Execute this sequence:

```
>A0:0        ←STARTS ASSEMBLING THE PROGRAM INTO ADDRESS 0
0000:0000  MOV  BX,1000H
0000:0003  MOV  AL,0
0000:0005  MOV  [BX],AL
0000:0007  INC  BX
0000:0008  INC  AL
0000:000A  JNZ  5H
0000:000C  HLT
0000:000D          ←PRESS THE RETURN KEY HERE TO END THE PROGRAM INPUT
>
>DI  0:0,000C       ←DISPLAYS THE PROGRAM JUST ENTERED
>
```

BREAK

Command BREAK

Introduction

The best way to safely stop a moving car is to use the brakes. In emulation, the best way to stop a program for examination is by using BREAKpoints. You can use the BREAK commands to set breakpoints anywhere within a program, and you can specify many different types of breaks to stop the program execution. Breakpoints differ from event points (see the EVENT command) in that they actually cause the program to stop execution; event points are used to trigger various external events, including stopping execution, without necessarily affecting the emulation process.

Software breakpoints replace program instructions automatically with monitor calls, in order to stop the program execution at a particular point in the program. This provides real-time operation until the break. Several software breakpoints can be set throughout the program and selectively enabled and disabled. Also, an unlimited number of user breakpoints can be assembled into the code throughout the program.

The ICD can also implement hardware breakpoints, which recognize machine cycles but do not disturb normal software execution. Hardware breakpoints can cause the ICD hardware to monitor the address and status signals for a specified condition. When the conditions are met, a break occurs.

Both hardware and software breakpoints can be activated (enabled), and then temporarily deactivated (disabled), without affecting their location addresses within the program or their parameter specifications.

Another break feature allows the ICD to use a probe to receive a signal from a peripheral, which can then cause a break in the program. (See "More About Your ICD," in Section 1.)

There are 16 different BREAK command formats. See each format for an explanation and an example.

BREAK

Command BREAK: Status

Operation Displays the current status of the break command. Use this command to check the condition of the breakpoint settings.

Syntax B

Command Example This command example shows what the break status might reveal after several break parameters are defined:

```
>B
A  (ON)   EX   0040:0600   1   0  IND  BOTH  (0000_0000_1010_0000_0000)
B  (OFF)  MR   003XX/DS    1   0  ARM  NDP   (0000_0000_0011_0XXX_XXX0)
0  (ON)        0040:0213   1   0
1  (ON)        01000       1   0
2  (ON)        01FFE       1   0
S  (DI)   HLT
X  (ON)   HI               1   0
E  (OFF)                   1   0
T  (ON)
W  (ON)
>
```

ADDRESS
BREAK OPERATION
BREAK STATUS
BREAK IDENTIFICATION

NOTE: A,B,C = hardware break names, 0,1,2 = event break names (up to 8 names), S = software break opcode, X = external break at high edge of signal, E = event break, T = ready time-out break, W = write-protect break.

BREAK

Command BREAK: Hardware Breakpoint Qualification

Operation Enables, disables, or clears the settings of the hardware breakpoints.

Applications Note: This command can be used to temporarily disable pre-set hardware breakpoints without affecting their locations within the program or their parameter specifications.

Syntax B[/*name*] *switch*

Terms *name* = A, B, or C

switch = ON, OFF, or CLR

Syntax Example
B/A ON
B OFF

Notes A, B, or C identifies hardware breakpoint names, and more than one name can be specified at a time (e.g., B/A/C CLR). If the breakpoint *name* is omitted, all hardware and software breakpoints are affected.

ON enables the breakpoint(s), OFF disables the breakpoint(s), and CLR clears the break condition.

Hardware breakpoints automatically default to "ON" after they are specified by the "BREAK: Hardware Breakpoint Specification" command.

Spacing: A space is required between *name* and *switch*. If *name* is omitted, a space is required between B and *switch*.

Command Example See Syntax Example and the "BREAK: Hardware Breakpoint Specification" command.

BREAK

Command BREAK: Hardware Breakpoint Specification

Operation Sets a hardware breakpoint within the user program. Setting a hardware break configures the emulator hardware to monitor the address and status signals for the specified condition to occur. When the conditions are met in the program, a break occurs.

Syntax B[*/name*] *status,addr*[*,passcount*]

Terms *name* = A, B, or C

status = Any one of eight types of break status, including:

M (memory access)
P (port access)
MR (memory read)
MW (memory write)
PR (port read)
PW (port write)
OF (operation code fetch)
IA (interrupt acknowledge)
EX (command execution)

addr = The address to break on.

segment = Any one of four segments for the address, including:

CS (code segment)
DS (data segment)
SS (stack segment)
ES (extra segment)

passcount = The number of times the condition occurs before breaking, from 1 to 65535.

BREAK

Syntax Example B/C M,1111_0011_XX10_110X

Notes A, B, or C identifies hardware breakpoint names.

If *name* is omitted, the next available breakpoint is used; if all the breakpoints are in use, an error message will be displayed.

The *addr* can be specified by a binary or hexadecimal notation. To specify a "don't care" condition in 1-bit units (binary notation), or in 4-bit units (hexadecimal notation), write X at the required position.

If *passcount* is specified, real-time operation is momentarily lost each time the condition occurs. If the *passcount* specification is omitted, 1 is assumed.

Spacing: A space is required between *name* and *status*. If *name* is omitted, a space is required between B and *status*. Spaces are not permitted where commas are used to separate the parameters.

BREAK

Command Example

Execute this sequence:

```
>B/B OF,200      ←SPECIFIES HARDWARE BREAKPOINT
>B       ← CHECKS BREAKPOINT STATUS
B  (ON)    OF 00200          1    0 IND BOTH (0000__0000__0010__0000__0
S  (DI)    HLT
E  (OFF)                     1    0
T  (ON)
W  (ON)
>B/B OFF     ← DISABLES HARDWARE BREAKPOINT B
>B     ← CHECKS THE BREAKPOINT STATUS AGAIN
B  (OFF)   OF 00200          1    0 IND BOTH (0000__0000__0010__0000__0
S  (DI)    HLT
E  (OFF)                     1    0
T  (ON)
W  (ON)
>
```

This example shows a hardware breakpoint is placed at address 200 in the program and that the status to break on is an opcode fetch. The "BREAK: Status" command is then used to verify the breakpoint setting. Next, the breakpoint is temporarily disabled using the B/B OFF command. Again, the "BREAK: Status" command is used to verify the change.

BREAK

Command BREAK: Event then Hardware Breakpoint

Operation Causes a break in the program at a hardware breakpoint (A, B, or C), but only after an event point is also passed (see EVENT command). The arm feature creates a simple level of sequencing: A-then-B relationship.

Applications Note: This command can be used to trigger a peripheral device (such as a logic analyzer) when an event point is passed in the program. The program then stops when a breakpoint is encountered.

Syntax B[/*name*] *switch*

Terms *name* = A, B, or C

switch = ARM or IND

Syntax Example
B/C ARM
B IND

Notes A, B, or C identifies hardware breakpoint names, and more than one name can be specified (e.g., B/A/C IND). If the breakpoint name is omitted, all three hardware breakpoints are affected.

If ARM is selected, the break occurs after an event trigger takes place. If IND is selected, the break occurs independently of any event trigger.

The ARMing event is not automatically reset. See the "BREAK: ARM Initialize" command.

Spacing: A space is required between *name* and *switch*. If *name* is omitted, a space is required between B and *switch*

Command Example See Syntax Example.

BREAK

Command BREAK: ARM Initialize

Operation Clears (initializes) the event pass condition and resets the ARM specification of the "BREAK: Event then Hardware Breakpoint" command.

Syntax B INI

Notes Once the ARMing event has occurred, the condition will remain ARMed until cleared by this command.

Spacing: A space is required between B and INI.

BREAK

Command Example Execute this sequence:

```
>A0:0
0000:0000 MOV  BX,1000H
0000:0003 MOV  AL,0
0000:0005 MOV  [BX],AL
0000:0007 INC  BX
0000:0008 INC  AL
0000:000A JNZ  5H
0000:000C HLT
0000:000D
>EV ST=MW,A=1012  ← SETS EVENT TO TRIGGER ON MEMORY WRITE TO
                    LOCATION 1012H
>B/A EX,B      ← SETS BREAKPOINT A TO BREAK ON EXECUTION OF LOCATION 8H
>B/A ARM       ← ARMS BREAKPOINT A BY EVENT TRIGGER
>B INI         ← CLEARS (INITIALIZES) EVENT'S ARMING CONDITION
>G 0:0         ← BEGINS EXECUTION

  00008        FECO                                INC     AL
CS       IP  ODITSZAPC   AX   BX   CX   DX   SP   BP   SI   DI   SS   D
0000:0005  000000010  0014 1014 0000 0000 0000 0000 0000 0000 0000 0C
<Break  Hardware A>
G  0:0

  00008        FECO                                INC     AL
CS       IP  ODITSZAPC   AX   BX   CX   DX   SP   BP   SI   DI   SS   D
0000:0005  000000010  0001 1001 0000 0000 0000 0000 0000 0000 0000 0C
<Break  Hardware A>
>B  INI        ← CLEARS EVENT'S ARMING CONDITION AGAIN
>G  0:0

  00008        FECO                                INC     AL
CS       IP  ODITSZAPC   AX   BX   CX   DX   SP   BP   SI   DI   SS   D
0000:0005  000000010  0014 1014 0000 0000 0000 0000 0000 0000 0000 00
<Break  Hardware A>
>
```

BREAK

Command BREAK: Software Breakpoint Specification

Operation Sets a software breakpoint within the user program.

Setting a software breakpoint causes the ICD to automatically replace the opcode at the specified address with a HLT or INT3 instruction opcode (see the "BREAK: Software/User Breakpoint Code" command). When this code is encountered during execution, a temporary break will occur, the original contents of this location will be replaced, and execution will restart at that same location for the duration of that one instruction. The ICD will then enter the monitor mode.

Setting a software breakpoint is a two-step process requiring both Specification and Recognition commands—see the "BREAK: Software Breakpoint Recognition" command.

Syntax B[/*name*] *addr*[,*passcount*]

Terms *name* = 0, 1, 2, 3, 4, 5, 6, or 7

addr = The address to break on.

passcount = The number of occurrences before a break, from 1 to 65535.

Syntax Example

```
B/4  100,3
B/7  1000
```

BREAK

Notes 0, 1, 2, . . . or 7 identifies sotware breakpoint names.

If *name* is omitted, the first available break name is used; if all available breakpoints are in use, an error message will be displayed.

For software breakpoints, the *addr* is specified by a hexadecimal notation and must be a single, specific address.

If *passcount* is specified, real-time operation is momentarily lost each time the condition occurs. If the *passcount* specification is omitted, 1 is assumed.

A software breakpoint cannot be specified in a USER-ROM area since the breakpoint requires changing the memory contents (at the specified location) to a HLT or INT3 instruction, and ROM cannot be changed. A hardware breakpoint must be used in this situation.

A software breakpoint must be specified for a location containing the first byte of an opcode; otherwise, the ICD will not break, and unpredictable results will occur within the program execution.

The monitor call is automatically placed at the specified locations when the program code is executed, but the program display will only show the original contents at that location. Anything that causes the contents of a location to be changed during program execution destroys the monitor call instruction.

Spacing: A space is required between*name* and *addr*. If *name* is omitted, a space is required between B and *addr*.

BREAK

Command Example Execute this sequence:

```
>B/5 1000          ← SETS SOFTWARE BREAKPOINT AT ADDR 1000
>B S=EN            ← ENABLES THE SOFTWARE BREAKPOINTS
>B                 ← CHECKS THE STATUS OF THE BREAKPOINTS
5 (ON)  OF 01000        1       0      ← SHOWS THAT
S (EN)  HLT                            SOFTWARE BREAKPOINT
E (OFF)                 1       0      #5 IS ACTIVE AT
T (ON)                                 ADDR 1000
(ON)
>
```

This example shows that a software breakpoint labeled 5 is set at address 1000 in the program. The software breakpoint is enabled (software breakpoints must be enabled to function), and then the "BREAK: Status" command is used to verify the change.

BREAK

Command BREAK: Software Breakpoint Recognition

Operation Enables or disables all software and user breakpoints. Setting a software breakpoint is a two-step operation requiring the software and user breakpoint to be enabled before any software breakpoints become operational.

Syntax B S=*switch*

Terms *switch* = EN or DI

Syntax Example B S=EN

Notes EN enables the software and user breakpoints, causing a break in the program based on the software breakpoint specification or when a user break is encountered. DI disables the software and user breakpoints, causing them to be temporarily disabled, although their initial specification remains unaffected.

The ICD defaults to DI upon power-up or reset.

Spacing: A space is required between B and S. No spaces are permitted after S; the equal sign acts as the separator.

Command Example See Syntax Example and the "BREAK: Software Breakpoint Specification" command.

BREAK

Command BREAK: Software/User Breakpoint Code

Operation Specifies which code the ICD uses to implement a software or user break.

Applications Note: The ICD can use HLT (0F4H), INT3 (OCCH) or any code from 0 to 0FFH (if specified by its correct hexadecimal code) to cause a software break within the user program. This allows you to conveniently cause a program break without continuously specifying breakpoint parameters.

Syntax B S=*op_code*

Terms *op_code* = HLT,INT3, or any hexadecimal code

Syntax Example B S=INT3

Notes The ICD defaults to HLT upon power-up or reset.

Spacing: A space is required between B and S. No spaces are permitted after S; the equal sign acts as the separator.

Command Example

Execute this sequence:

```
>B          ← CHECKS THE BREAKPOINT STATUS
S  (DI)  HLT        ← SHOWS SOFTWARE BREAK CODE IS CURRENTLY HLT
E  (OFF)            1       0
T  (ON)
W  (ON)
>B  S=INT3          ← CHANGES SOFTWARE BREAK CODE TO INT3
>B  S=EN            ← ENABLES ALL SOFTWARE BREAKPOINTS
>B                  ← CHECKS THE BREAKPOINT STATUS AGAIN
S  (EN)  INT3       ← SHOWS THE SOFTWARE BREAK CODE
E  (OFF)            1       0
T  (ON)
W  (ON)
>
```

This example shows how the software break code is changed from HLT to INT3 and then enabled. The "BREAK: Status" command verifies the change.

BREAK

Command BREAK: Software Breakpoint Qualification

Operation Enables, disables, or clears the software breakpoints.

Applications Note: This command can be used to temporarily disable pre-set software breakpoints without affecting their address locations within the program or their parameter specifications.

Syntax B[/*name*] switch

Terms 0, 1, 2, 3, 4, 5, 6, or 7

switch = ON, OFF, or CLR

Syntax Example
B/3 ON
B OFF

Notes 0, 1, 2, . . . or 7 identifies software breakpoint names and more than one name can be specified at a time (e.g., B/1/2/3/4 OFF). If the breakpoint *name* is omitted, all the hardware and software breakpoints are affected.

ON enables the breakpoint, OFF disables the breakpoint, and CLR clears the break condition.

Spacing: A space is required between *name* and *switch*. No spaces are permitted between B/*name*.

BREAK

Command Example Execute this sequence:

```
>B          ← CHECKS THE BREAKPOINT STATUS
S  (DI)     HLT
E  (OFF)                  1        0
T  (ON)
W  (ON)
>B/2  7FF        ← SETS A SOFTWARE BREAKPOINT AT ADDR 7FF
B  S=EN          ← ENABLES THE SOFTWARE BREAKPOINTS
>B          ← CHECKS THE BREAKPOINT STATUS AGAIN
2  (ON)  007FF            1        0        ← SHOWS THAT SOFTWARE
S  (EN)  HLT                                   BREAKPOINT #2 IS
E  (OFF)                  1        0          ACTIVE AT ADDR 7FF
T  (ON)
W  (ON)
>B/2 OFF         ← DISABLES SOFTWARE BREAKPOINT #2
B           ← CHECKS THE STATUS AGAIN
2  (OFF) 007FF            1        0        ← SHOWS SOFTWARE
S  (EN)  HLT                                   BREAKPOINT #2 IS
E  (OFF)                  1        0          INACTIVE
T  (ON)
W  (ON)
>
```

This command shows how a software breakpoint is set, enabled, and then disabled. After each operation, the status of the breakpoints is checked against the changes.

BREAK

Command BREAK: Processor Access

Operation Specifies which processor access causes a hardware break in the user program.

Syntax B/*name processor*

Terms *name* = A, B, or C

processor = CPU, NDP, or BOTH

Syntax Example B/B NDP

Notes A, B, or C identifies hardware breakpoint names, and more than one name can be specified at a time (e.g., B/A/B/C CPU).

CPU means that accessing the main (8086/8088) processor causes a break; NDP means that accessing the Numeric Data (8087) Processor causes a break; and BOTH means that accessing both processors (8086/8088 and 8087) causes a break. (BOTH is the default when a hardware breakpoint condition is specified.)

Spacing: A space is required between *name* and *processor*. No spaces are permitted between B/*name*.

Command Example

Execute this sequence:

Press the RESET switch on the ICD, then enter:

```
>B          ← CHECKS THE BREAKPOINT STATUS
S  (DI)     HLT
E  (OFF)                    1   ·   0
T  (ON)
W  (ON)
>B/A OF,1FF          ← SETS A HARDWARE BREAKPOINT
>B          ← SHOWS THE REVISED BREAKPOINT STATUS
A  (ON)  OF 001FF      1          0 IND BOTH (0000__0000__0001__1111__11
S  (DI)  HLT          ← SHOWS BOTH PROCESSORS ARE SPECIFIED
E  (OFF)                    1          0
T  (ON)
W  (ON)
>B/A NDP          ← CHANGES BOTH SPECIFICATION TO NDP PROCESSOR ONLY
>B          ← CONFIRMS THE CHANGE
A  (ON)  OF 001FF      1          0 IND NDP (0000__0000__0001__1111__111
S  (DI)  HLT          ← SHOWS NDP IS SPECIFIED
E  (OFF)                    1          0
T  (ON)
W  (ON)
>
```

BREAK

Command BREAK: External Signal Qualification

Operation Allows the ICD to sense a signal (using the accessory probes) from an external source and cause a break in the user program. This command specifies how the break is triggered, from either the high-going or low-going edge of the external signal. To enable or disable this command, see the "BREAK: External Breakpoint Qualification" command.

Syntax B/X *edge*[,*passcount*]

Terms *edge* = HI or LO

passcount = The number of occurrences before a break, from 1 to 65535.

Syntax Example B/X LO

Notes HI causes the breakpoint to occur on the rising edge of the signal; LO causes the breakpoint to occur on the falling edge of the signal.

When *edge* is specified, the parameters for the "BREAK: External Breakpoint Qualification" command become effective.

If *passcount* is specified, real-time operation is momentarily lost each time the condition occurs. If the *passcount* specification is omitted, 1 is assumed.

Spacing: A space is required between B/X and *edge*. No spaces are permitted between B/X.

Command Example See the "BREAK: External Breakpoint Qualification" command.

BREAK

Command BREAK: External Breakpoint Qualification

Operation Allows the ICD to sense a signal (using the accessory probes) from an external source and trigger a break in the user program during emulation. This command enables, disables, or clears that feature. (For more information on how to use the accessory probes, see "More About Your ICD," in Section 1.)

Syntax B/X *switch*

Terms *switch* = ON, OFF, or CLR

Syntax Example B/X CLR

Notes ON enables the recognition of an external trigger, OFF disables the recognition of an external trigger, and CLR clears the external trigger specification.

Spacing: A space is required between B/X and *switch*. No spaces are permitted between B/X.

BREAK

Command Example Execute this sequence:

```
>B          ← CHECKS THE BREAKPOINT STATUS
S   (DI)  HLT
E   (OFF)          1       0
T   (ON)
W   (ON)
>B/X  HI          ← SETS SIGNAL RECOGNITION TO HIGH EDGE OF SIGNAL
>B
S   (DI)  HLT
X   (ON) HI        1       0        ← SHOWS EXTERNAL BREAK FEATURE
E   (OFF)          1       0          IS ACTIVE
T   (ON)
W   (ON)
>B/X  OFF         ← DISABLES EXTERNAL BREAK FEATURE
>B          ← CHECKS BREAKPOINT STATUS AGAIN
>B
S   (DI)  HLT
X   (OFF)  HI      1       0        ← SHOWS EXTERNAL BREAK FEATURE
E   (OFF)          1       0          IS INACTIVE
T   (ON)
W   (ON)
>B/X  CLR         ← CLEARS THE EXTERNAL BREAKPOINT FEATURE
>B          ← VERIFIES THE CHANGE
S   (DI)  HLT
E   (OFF)          1       0
T   (ON)
W   (ON)
>
```

This example shows how the external breakpoint specification is set to occur at the high edge of an external signal. The external breakpoint is then temporarily disabled and finally cleared.

BREAK

Command BREAK: Event Breakpoint

Operation Allows the ICD to use an event trigger as a breakpoint (see the EVENT command). This command enables or diables the event break feature but does not affect the event point specification in any way.

Syntax B/E *switch*

Terms *switch* = ON or OFF

Syntax Example B/E OFF

Notes ON enables the event breakpoint and OFF disables the event breakpoint.

Spacing: A space is required between B/E and *switch*. No spaces are permitted between B/E.

BREAK

Command Example Execute this sequence:

```
>EV          ← DISPLAYS EVENT STATUS
Event is Clear          ← SHOWS ABSENCE OF EVENT POINTS
>EV  ST=OF   A=7FF          ← SETS AN EVENT POINT IN PROGRAM
>EV          ← DISPLAYS NEW EVENT POINT SETTING
(ON)
Status    = OF
Address = 007FF     (0000__0000__0111__1111__1111)
>B/E  ON          ← ENABLES THE EVENT POINT TO CAUSE A BREAK IN EXECUTION
>B
S   (DI)  HLT
E   (ON)          1       0          ← SHOWS EVENT POINT SETTING IS ACTIVE
T   (ON)
W   (ON)
>
```

This example shows how an event in the program can be used to send out a signal to a peripheral device. First, the event point is set in the program at address 7FF, and then the status command is used to verify the setting. Next, the event break point is enabled by using a breakpoint command. The "BREAK: Status" command is used again to verify that the event point is enabled.

BREAK

Command BREAK: Event Breakpoint Passcount

Operation Sets the passcount for the event breakpoint.

Syntax B/E *passcount*

Terms *passcount* = The number of occurrences before a break, from 1 to 65535 (default = 1).

Syntax Example B/E 4

Notes Spacing: A space is required between E and *passcount*. No spaces are permitted between B/E.

BREAK

Command BREAK: Write Protect Breakpoint

Operation Causes a break in the user program if the program attempts to write into a protected memory area (see the MAP command). After the break, the ICD responds with an error message that reads: Break Write Protect.

If this break is disabled, any attempt to write to a protected memory location will fail, thereby preserving its integrity; however, program execution will continue without causing a break.

Syntax B/W *switch*

Terms *switch* = ON or OFF

Syntax Example B/W ON

Notes ON enables the write protect feature and OFF disables the write protect feature. (This feature is automatically activated when the ICD boots up.)

Spacing: A space is required between B/W and *switch*. No spaces are permitted between B/W.

BREAK

Command Example

Execute this sequence:

```
>MA 0,FFF=RO      ← SETS MEMORY AS READ-ONLY FROM ADDRESS 0 TO FFF
>MA
In-Circuit Mode 0 (US=>RW)
00000-00FFF = RO (000)      ← SHOWS STATUS OF MEMORY IS READ-ONLY
01000-FFFFF = RW (004)        FROM ADDR 0 TO FFF
>
>B/W ON      ← ENABLES THE WRITE PROTECT FEATURE
>B
S  (DI)  HLT
E  (OFF)      1      0
T  (ON)
W  (ON)      ← SHOWS WRITE PROTECT FEATURE IS ACTIVE
>
```

This example shows how the write protect feature might be used. First, memory within the ICD is mapped from 0 to FFF as read-only. Because the in-circuit status is I O (debugging using the ICD's memory only), any area mapped as user (target system) memory is now re-mapped as read/write memory in the ICD; this causes the remaining memory areas (1000-FFFFF) to act as read/write memory. The write protect feature is then enabled using the "BREAK: Write Protect Breakpoint" command. Finally, the break status is checked to verify the changes. The ICD now causes a break if an attempt is made to write into memory locations 0 to FFF.

BREAK

Command BREAK: Timeout Breakpoint

Operation Causes a break in the user program when the ICD is unable to access the target memory contents within a certain time period (128 clock cycles). If the READY signal is negated for more than 128 clock cycles, a time-out condition will occur. After the break, the ICD responds with an error message that reads: Break Timeout.

Applications Note: This break command can be used to flag a failure by the target system to re-assert a ready condition. The failure could be caused by a problem in the hardware, or it could be inherent in the design. If the problem lies in the design, the Timeout Breakpoint feature should be disabled; but if it is a hardware problem, disabling this feature could cause the ICD to "lock-up" due to a continuously negated ready condition.

This feature can also act as a safeguard for the target's refresh period if dynamic RAMs are being used.

Syntax B/T *switch*

Tems *switch* = ON or OFF

Syntax Example B/T OFF

Notes ON enables the timeout feature and OFF disables the timeout feature. (This timeout feature is auttomatically activated when the ICD boots up.)

Spacing: A space is required between B/T and *switch*. No spaces are permitted between B/T.

Command Example See Syntax Example.

CALCULATION

Command CALCULATION

Operation Performs subtraction and addition of hexadecimal and/or decimal numbers, and performs hexadecimal-to-decimal or decimal-to-hexadecimal conversions. The results of the particular operation are displayed in both decimal and hexadecimal notation.

Syntax C operand *#1*[±*operand#2*]] . . . ±*operand#n*]

Terms *operand #1, #2 . . . #n* = −2147483648 to 2147483647 (signed) or 0 to 4292967295 (unsigned); or 0 to 0FFFFFFFFH.

Syntax Example C 427+351−2FFH

Notes Both addition and subtraction may be performed on the same line.

Negative decimal results are displayed as "unsigned/signed."

Spacing: A space is required between C and *operand #1*. No spaces are permitted after *operand #1.*

Command Example

```
>C 283-350          ← SUBTRACTION OF DECIMAL NUMBERS
FFFFFFBDH
4294967229-67
>C 100              ← CONVERSION OF DECIMAL NUMBER
00000064H
100
>C FFH+1FF50H       ← ADDITION OF HEXADECIMAL NUMBERS
0002004FH
131151
>C 429-2EH+8+FF3DH  ← MIXED CALCULATIONS
000100C4H
65732
>
```

COMPARE

Command COMPARE

Operation Compares the contents of specified memory blocks within the ICD or target system, and then displays the non-matching data. The comparison can be made between different memory blocks as mapped to the ICD, or between one block of memory within the ICD and one in the target system.

Syntax CO *beg_addr,end_addr,comp_addr*[,*direction*]

Terms *beg_addr* = The beginning address for comparison.

end_addr = The ending address for comparison.

comp_addr = The beginning memory address to be compared.

direction = UP or PU.

Syntax Example CO 100,3FF,1000,UP

COMPARE

Notes

If UP is selected, *beg_addr* is user memory, and *comp_addr* is ICD program memory. If PU is selected, *beg_addr* is ICD program memory and *comp_addr* is user memory.

If *direction* is omitted, memory locations are specified by the MAP command.

This command displays non-matching data on a line-for-line basis. To control the scrolling of the display, alternately press the space bar. To exit the display, press the Escape (Esc) key.

Spacing: A space is required between CO and *beg_addr*. No spaces are permitted after *beg_addr*; commas are used to separate the remaining parameters.

Command Example

See Syntax Example. This example shows that a memory block (100 to 3FF) in the target system is compared with a block of memory in the ICD, beginning at address 1000. Any unmatching data will be displayed, along with the location addresses.

DISASSEMBLE

Command DISASSEMBLE

Operation Translates the memory contents from machine language to assembly language mnemonics, and then displays the converted contents. The opposite translation (assembly language mnemonics to machine language) is accomplished by using the ASSEMBLE command.

Syntax DI [*beg_addr*] [,*end_addr*]

Terms *beg_addr* = The beginning memory address in the program.

end_addr = The ending memory address in the program.

Syntax Example
```
DI 100,1A6
DI FFF
DI
DI ,L40
```

Notes If *beg_addr* is omitted, disassembly begins at the current program counter (PC). If *end_addr* is omitted, 11 lines of instructions are automatically displayed.

This command displays items on a line-for-line basis. To control the scrolling of the display, alternately press the space bar. To exit the display, press the Escape (Esc) key.

Spacing: A space is required between DI and *beg_addr* (if *beg_addr* is used). Spaces are not permitted where commas are used to separate the parameters.

Command Example See Syntax Example. The first example shows that the memory contents in the ICD are disassembled beginning from address 100 to address 1A6. In the second example, the ending address is omitted, which causes 11 lines of the memory contents to be disassembled starting from address FFF. The third example illustrates that 11 instruction lines are displayed from the current PC. The fourth example displays the current PC to PC + 40.

DUMP

Command DUMP

Operation Displays the memory contents in both hexadecimal and ASCII code.

Syntax D[/W] *beg_addr*[,*end_addr*]

Terms W = Displays the memory contents in word units arranged in MSB/LSB (Most Significant Bit/Least Significant Bit) order (the default is byte unit display).

beg_addr = Beginning address of display.

end_addr = Ending address of display.

Syntax Example

```
D/W 100,1FF
D 1FFF
```

Notes The *end_addr* is an optional parameter; if it is omitted, 16 bytes are displayed starting with *beg_addr*.

The 8086/8088 arranges word data in memory as: low address = LSB, high address = MSB; therefore, the /W option effectively swaps the two bytes of each word.

This command displays items on a line-for-line basis. To control the scrolling of the display, alternately press the space bar. To exit the display, press the Escape (Esc) key.

Spacing: A space is required between D or D/W and *beg_addr*. Spaces are not permitted where commas are used to separate the parameters.

Command Example See Syntax Example. The first example shows that the memory contents are displayed in word units, beginning with address 100 and ending with address 1FF. The second example shows that the last 16 bytes are displayed beginning at address 1FFF.

Command EVENT:

Introduction

An event can be defined as a significant occurrence in time. That is, events take their respected place at a point in time, without affecting the passing of time itself. And, of course, the ICD's EVENT command works on the same principle.

This command allows an event to occur during the execution of a program, without necessarily stopping the program. In this way, an event point differs from a breakpoint because breakpoints always stop the program execution.

The EVENT command can enact four different operations: trigger a peripheral device, such as a logic analyzer; trigger the real-time trace feature—which is defined by the HISTORY command; arm a hardware breakpoint in an A-then-B type sequence; and stop the program in a manner similar to the BREAK command.

Unlike the BREAK command, the EVENT command has the advantage of allowing you to specify a certain data pattern on the data bus, in addition to the normal address parameters, memory access, and I/O access conditions.

Events point can be enabled and disabled, just like breakpoints. This feature allows you to temporarily disable the event setting without affecting its address location within the program or its parameter specifications.

Using The Event Command

There are three EVENT commands: Status, Qualification, and Specification. To see how to use an event point as a breakpoint, see the "EVENT: Specification" and "BREAK: Event Breakpoint" commands. To arm a hardware breakpoint, see the "BREAK: Event Then Hardware Break" command. To use an event point to trigger the real-time trace, see the HISTORY command. To use an event point to trigger a peripheral device, see "More About Your ICD," in Section 1.

Command EVENT: Status

Operation Displays the current event point specifications. When changes are made to the event point specifications by using the "EVENT: Specification" command, this command is used to display the latest changes.

Syntax EV

Command Example

```
>EV
Event is Clear
```

This is the default condition for the EVENT command. The display shows the absence of any event points in the program. After specifying an event point, the "EVENT: Status" command might reveal a display such as the one shown below:

```
>EV
(ON)
Status   = MW
Address  = F035:0000 (1111_0000_0101_1000_0000)
Segment  = CS
Mode     = CPU
Data     = 55        (0101_0101)
>
```

This status display shows that the EVENT command is enabled (ON), the status of the event point is a memory write (MW), the port is located at address F035:0000 (which is also represented by its physical address in bit-wise notation), the segment value is CS (code segment), the EVENT command will access the CPU only, and the data to match for the event is 55. When these conditions are satisfied, an event occurs.

Command EVENT: Qualification

Operation Enables, disables, or clears an event trigger.

Applications Note: This command can be used to temporarily disable an event point without affecting its location within the program or its parameter specifications. Use this command after setting an event point with the "EVENT: Specification" command.

Syntax EV *switch*

Terms *switch* = ON, OFF, or CLR

Syntax Example EV CLR

Notes ON enables the event trigger recognition feature, OFF disables the event trigger recognition feature, and CLR clears the event setting. (ON is the default when an event is set using the "EVENT: Specification" command.)

Spacing: A space is required between EV and *switch*.

Example Command See Syntax Example and the "EVENT: Specification" command.

Command EVENT: Specification

Operation Sets the conditions for an event point trigger.

Syntax EV [ST=*status*][,A=*addr*][,SEG=*seg*][,D[/W]=*data*][M=*mode*]

Terms *status* = The type of cycle to trigger event on. This can be one of nine different names, including:

M (memory access)
P (port access)
MR (memory read)
MW (memory write)
PR (port read)
PW (port write)
OF (operation code fetch)
IA (interrupt acknowledge)
EX (command execution)
ANY (any operation)

addr = Specifies the address value to match for the event.

seg = Specifies one of four different segments (or any of them), including:

CS (code segment)
DS (data segment)
SS (stack segment)
ES (extra segment)
ANY (any segment)

W = Qualifier to specify word data.

data = Specifies the data value to match for the event.

mode = CPU, NDP, or BOTH

Syntax Example

```
EV  ST=MR
EV  A=1111_1111_0000_0XXX_XXX0
EV  ST=MW,A=EFD04,SEG=ES
```

Notes

All parameters for this command are optional; all parameters not defined remain unchanged.

Both *addr* and *data* may be specified as "don't care," in 1-bit units (binary) or in 4-bit units (hex), by writing X at the required position. Any undefined parameter defaults as "don't care."

If data is specified other than "don't care," the address parameter must have the lowest address bit (A0) defined as 0 or 1 (not as "don't care").

When specifying a P, PR, or PW cycle for the event, and the port address is defined, the address should be defined as a 16-bit address, with the upper 8 bits defined as "don't care." (Example: port address 34 = XX34.)

If CPU is specified, event occurs on main processor access (8086/8088). If NDP is specified, event occurs on co-processor access (8087). If BOTH is specified, event occurs on either processor access.

Spacing: A space is required between EV and the first parameter. Spaces are not permitted where commas are used to separate the parameters.

Command Example

```
>EV  A=FF0XX          ← SPECIFIES AN ADDRESS (X ON NIBBLE BASIS)
>EV  A=0000_0011_1111_1111_1110   ← SPECIFIES AN ADDRESS
                                    (X ON BIT BASIS)
>EV  ST=MR            ← SPECIFIES AN EVENT STATUS
>EVV A=CS:0,D=0       ← SPECIFIES ADDRESS AND DATA FOR EVENT
>EV  ST=MW,A=EFD05,SEG=ES   ← SETS EVENT USING STATUS,
                              ADDRESS, AND SEGMENT
>EV  ST=PR,D=XX46     ← SETS EVENT FOR PORT READ OF SPECIFIED DATA
```

EXAMINE

Command EXAMINE

Operation Examines one or more memory locations and optionally modifies them. The locations can be displayed and changed with either ASCII or hexadecimal values.

Syntax E[/W][/N] *beg_addr*[=*mod_data*]

Terms W = Use the word mode (the default is the byte mode).

N = No-verify (the default is to read-verify after write).

beg_addr = Starting address for display.

mod_data = Modified (new) data for this location.

Syntax Example

```
E/W 100=5555
E FFE
```

Notes When /W option is selected, the word will be displayed or entered in LSB/MSB (Least Significant Bit/Most Significant Bit) order (bytes swapped).

If *mod_data* is omitted, the command enters a repeat mode, which allows several locations to be changed.

The repeat mode includes:

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

Spacing: A space is required before *beg_addr*. No spaces are permitted between *beg_addr* and *mod_data*; the equal sign acts as the separator.

```
>E  0
00000 FF=74,      ← CHANGES VALUE TO 74H; RE-EXAMINE
00000 74=         ← LEAVES VALUE UNCHANGED; GO TO NEXT ADDRESS
00001 BF=         ← LEAVES VALUE UNCHANGED; GO TO NEXT ADDRESS
00002 BF='A'      ← CHANGES VALUE; GO TO NEXT ADDRESS
00003 72=34^      ← CHANGES VALUE; GO TO PREVIOUS ADDRESS
00002 41=^        ← LEAVES VALUE UNCHANGED; GO TO PREVIOUS ADDRESS
00001 BF=,        ← LEAVES VALUE UNCHANGED; RE-EXAMINE ADDRESS
00001 BF=^        ← LEAVES VALUE UNCHANGED; GO TO PREVIOUS ADDRESS
00000 74=/        ← LEAVES VALUE UNCHANGED; EXIT COMMAND
>E/W 20
00020 A9BF=4455   ← CHANGES WORD VALUE; RE-EXAMINE
00020 4455=       ← LEAVES VALUE UNCHANGED; GO TO NEXT LOCATION
00022 FDB2='HI',  ← CHANGES VALUE (ASCII); RE-EXAMINE
00022 494B=       ← LEAVES VALUE UNCHANGED; GO TO NEXT LOCATION
00024 CFED=       ← LEAVES VALUE UNCHANGED; GO TO NEXT LOCATION
00026 F7F5=^      ← LEAVES VALUE UNCHANGED; GO TO PREVIOUS LOCATION
00024 CFED=0/     ← CHANGES VALUE; EXIT COMMAND
>E 30
00030 06=         ← EXAMINES ONLY
00031 A0=
00032 00=
00033 64=
00034 0C=
00035 0E=/        ← EXITS COMMAND
```

Command FILL

Operation Fills a block of memory with either hexadecimal or ASCII codes.

Syntax F[/W][/N] *beg_addr,end_addr,data*

Terms W = Fill memory contents of a word basis (the default is a byte basis).

N = No-verify (the default is to read-verify after write).

beg_addr = The block beginning address to be filled.

end_addr = The block ending address to be filled.

data = Data that fills the block.

Syntax Example
```
F 100,3FF,55
F/N 4000,4FFF,0
F/W DS:0,FF,3412
```

Notes When /W option is selected, the word will be displayed or entered in LSB/MSB (Least Significant Bit/Most Significant Bit) order (bytes swapped).

Spacing: A space is required before *beg_addr*. No spaces are permitted where the commas act as separators.

Command Example See Syntax Example. The first example shows how memory is filed from address 100 to address 3FF, with a data value of 55; the second example shows how memory is filled without verifying the write; and the third example shows how memory is filled on a word basis, including a data segment value.

Command GO

Operation Executes the user's program.

Syntax G [*beg_addr*][,*end_addr*][,*end_addr #2*]

Terms *beg_addr* = The address to begin execution.

end_addr = The last address to execute.

end_addr #2 = Optional second ending address

Syntax Example

```
G
G 100
G 0,2FFE
```

Notes All parameters for this command are optional. If *beg_addr* is omitted, the program continues from the current program counter. If *end_addr* is omitted, the program continues until a breakpoint or a monitor break. When *end_addr #2* is specified, the first location reached by execution (*end_addr* or *end_addr #2*) will cause a break. Two hardware breakpoints must be available to activate the *end_addr* or *end_addr #2* parameters.

Spacing: A space is required between 6 and any additional parameters. Spaces are not permitted where commas are used to separate the parameters.

Example Command See Syntax Example. The first example starts the program from the current program counter; the second example starts the program from address 100; and the third example starts the program from 0 and stops it at address 2FFE.

Command HISTORY (Real-time Tracing)

Introduction

The real-time trace is one of the most powerful and useful features of your ICD. It allows you to record (hence the name "History" command) and then analyze a specific section of program execution rather than sift through the entire program looking for one particular problem.

Event points (see the EVENT command) can trigger the real-time trace buffer to start or stop the data storage process when program execution begins. By specifying storage modes, the event points control the start/stop action of the real-time trace.

By using the various storage modes, the real-time trace can effectively capture any set of instructions within a program. The program execution can then be stopped, and the address, data, and control bus of the latest series of machine cycles can be displayed (in either machine cycle or disassembled format) on the console screen, or dumped to a printer. Thus, if a problem develops during the program execution, the real-time trace provides a record that can be reviewed to determine what and where the problem is.

Trace Width and Depth

An emulator's trace memory should be wide enough to accommodate the processor's address, data, and status lines. With the ICD-178 for 8086/8088, the trace memory is 40 bits wide (16 bits data/16 bits address/8 bits status).

When it comes to the trace memory's depth, more is not always better. If too much depth is specified, it may be difficult to sift through all the data; if the trace memory depth is insufficient, the chances of recording the trace section where the problem exists are diminished. Your ICD has a maximum trace memory depth of 4K (4095) machine cycles; this may be reduced by specifying the "range" in the HISTORY command (except for the End Monitor and End Event modes). The ability to alter the size of the trace storage permits very specific tracing.

Real-time Trace Buffer

The data that is recorded from the program execution is stored in the ICD real-time trace buffer. The real-time trace bufer can be thought of as a data storage facility that moves along parallel to the user program, storing the same data that is executed by the user program.

The maximum storage capacity of the real-time trace buffer is 4K machine cycles, but by using a "First-In/First-Out" (FIFO) recording technique, the buffer captures the latest program execution by discarding old data and replacing it with new data. By using this technique, the display reveals the latest data the buffer has stored.

(diagram of program and real-time trace buffer)

Trigger Name	Begin Monitor	End Monitor	Begin Event
Command format	H BM	H EM	H BE,trace__range
Activated by	GO command	GO command	An event point
Terminates when	Buffer filled	Break in execution	Buffer is full
FIFO when buffer full?	No	Yes	Yes
Range affects	Storage size	Nothing (ignored)	Storage size
End result in buffer	First 4K cycles executed	Last 4K cycles executed	4K cycles following event

Trigger Name	Center Event	End Event	Multiple Event
Command format	H CE,trace__range	H EE	H ME,trace__range
Activated by	GO command	GO command	An event point
Terminates when	Event point + range of cycles is reached	An event point occurs	Buffer is full
FIFO when buffer full?	Yes	Yes	No
Range affects	Offset of event from center	Nothing (ignored)	Temporary storage termination until
End result in buffer	4K surrounding events	Event point + 4K cycles	Several "snapshots" triggered by event points

Using the Real-time Trace

The ICD's real-time trace feature defaults to the End Monitor mode upon initialization; therefore, it is always active, that is, it records the program execution even if the HISTORY command parameters are omitted. The ICD can also display the recorded memory contents in four different modes (using the "HISTORY: Real-time Format Display" command).

The options, then, for the HISTORY command involve selecting the proper command format to trigger or halt the real-time trace feature. A discussion of each storage mode follows.

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 tracing so that the data executed from the program memory area is simultaneously transferred to the real-time trace buffer.

After the user program executes (and the buffer stores) the data equivalent of the range, the trace buffer fills to that point and then stops. The data that is now stored in the buffer is the "captured" trace section (the section that the ICD displays). The real-time trace then enters a non-trace mode and stops when a MONITOR break (accomplished by pressing the MONITOR switch) or breakpoint is accomplished.

(diagram of Begin Monitor mode)

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 (up to 4K) and then stops.

NOTE: The event itself is not stored in the buffer, but triggers the buffer to begin storing.

(diagram of Begin Event mode)

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 the last 4K before the breakpoint or MONITOR break.

The ICD accomplishes this type of tracing by recording and storing data on a First-In/First-Out (FIFO) basis after the buffer is filled. By using this technique, the ICD displays the latest data in the trace buffer.

The End and Center Event modes use this same FIFO recording technique in their operation.

(diagram of End Monitor mode)

End Event Mode

The End Event mode works in the same way as the End Monitor mode except that an event point (instead of a breakpoint) triggers the buffer to halt data storage. The captured trace section is the last 4K before and including the event point.

(diaggram of End Event)

Center Event Mode

The Center Event mode is used when you desire the trace to surround a single event point in the program. It performs this task by reading the range specification and recording that number of cycles after the event point occurs. The remainder of the 4K buffer then contains cycles just prior to and including the event point. For example, if 1K is specified as the range, 1K of data would be captured after the event point, and the remaining 3K would be captured before the event point. If the specified range is 4000, 4000 cycles would be captured after the event, and the remaining 95 cycles would be captured before the event point. (4K = 4095 cycles.)

Just like the End Monitor and End Event modes, the Center Event mode causes the real-time trace to start recording data immediately after the GO command.

(diagram of Center Event mode)

Multiple Event

The Multiple Event mode is identical to the Begin Event mode, except that when the trace-range specification is reached, the tracing temporarily stops until another event point occurs. Then the buffer resumes storing another trace-range number of cycles. When the 4K buffer is completely filled, the event points are then ignored, and the buffer remains in a non-storage mode. This allows several event occurrences to trigger the History buffer, giving successive "snapshots" of a particular routine.

NOTE: The smaller the trace-range, the more times an event can retrigger the buffer to store data.

(diagram of Multiple Event mode)

Command HISTORY: Real-time Trace Status

Operation Displays the current status of the real-time trace buffer.

Applications Note: The real-time trace status can be used to analyze the condition of the real-time trace buffer (i.e., storage mode name, size of the trace range, number of cycles executed, and number of cycles stored in the History buffer).

When the real-time trace specifications are changed, the "HISTORY: Status" command will display their latest settings.

Syntax H

Command Example Press the RESET switch on the ICD to initialize the HISTORY command, then enter:

```
>H
Clock Counts  = 00000000/0     ← NUMBER OF CLOCK CYCLES
Storage Mode = EM      ← MODE AND TRACE RANGE
Storage Size   = 0/0      ← NUMBER OF CYCLES PASSED
>
```

The ICD defaults to this condition upon initialization. It automatically resets the clock counter to 0, selects End Monitor as the storage mode, sets the trace range to maximum, and initializes the storage size to 0.

In this example, Clock Counts shows the number of clock cycles (T-states) since the real-time trace was cleared. The number to the left of the slash (/) is the hexadecimal number of clock cycles, and the number to the right is its decimal equivalent. Storage Mode shows that the default specification is the End Monitor mode (the trace range for this mode is automatically set to 4095). Storage Size shows the number of cycles since the program was started (to the right of the slash) or since the program was resumed (to the left of the slash). If the Storage Size displayed Full, it would indicate a full buffer, or 4095 cycles.

In the next example, the Begin Event mode is selected and the trace range is omitted. The status command now shows:

```
>H
Clock Counts   = 00000000/0
Storage Mode = BE  4093
Storage Size    0/0
>
```

The trace range for the Begin Event mode defaults to 4093 (4095, the maximum, must be specified).

Command HISTORY: Real-time Trace Counter Reset

Operation Clears (resets) the clock counter.

Syntax H CLR

Notes Spacing: A space is required between H and CLR.

Command Example See "HISTORY: Real-time Trace" examples.

Command HISTORY: Real-time Trace Format Display

Operation Allows the contents of the real-time trace buffer to be displayed in either machine cycle format or disassembled format.

Syntax H *mode*[,*int_point*][,*term_point*]

Terms *mode* = M, D, C, or X

int_point = Initial point of display, from 1 to 4095.

term_point = Point at which display terminates, from 1 to 4095.

Syntax Example
```
H M,200,100
H D
```

Notes M specifies to display the program execution in machine cycle format. D displays the program execution in disassembled format (excludes opcode fetch), C specifies clock cycle format, and X specifies machine cycle with disassembly.

The *int_point* must be greater than or equal to *term_point*. The storage pointer is numbered by bus cycles—displayed from high to low—where "1" is the most recent bus cycle.

This command displays items on a line-for-line basis. To control the scrolling of the display, alternately press the space bar. To exit the display, press the Escape (Esc) key.

Spacing: A space is required between H and *mode*. Spaces are not permitted where commas are used to separate the parameters.

Command Example

See the following and the "HISTORY: Real-time Trace" examples.

If H M (machine cycle format) or H C (clock cycle format) is selected, the following headings will be shown on the display:

Point T Address St Data Seg IF QS_Instruction

Where: Point = address in HISTORY buffer
T = event point indicator
Address = cycle address
St = cycle status (type of cycle operation)
Data = cycle data
Seg = segment used in this cycle to derive address
IF = interrupt flag status
QS_Instruction = queue status_Hex machine code
F_xx = first instruction fetched from queue
S_xx = subsequent instruction fetched from queue
Emp = queue flush

If H D (disassembled format) or H X (machine cycle format with disassembly) is selected, the following headings will be shown on the display:

Point T Addr. Machine Code Prefix Opcode Operand

Where: Point = address in HISTORY buffer
T = event point indicator
Addr. = cycle address
Machine Code = entire instruction in Hex code
Prefix = disassembled prefix mnemonic, e.g., LOCK, REPNE, etc.
Opcode = disassembled opcode mnemonic
Operand = disassembled operand (may be symbolic if ZICE software is used)

Command HISTORY: Real-time Trace Storage Mode

Operation Specifies the trace mode for the real-time trace buffer. This is the command that specifies what type of mode activates the real-time trace feature.

Syntax H mode[,*range*]

Terms *mode* = Trace mode. This can be one of six different modes, including:

BM (begin monitor mode)
EM (end monitor mode)
BE (begin event mode)
CE (center event mode)
EE (end event mode)
ME (multiple event mode)

range = The trace range, from 1 to 4095.

Syntax Example H ME,2027

Notes The range specified for the EM and EE modes will be ignored; it defaults to the maximum 4K size.

Spacing: A space is required between H and *mode*. No spaces are permitted where commas are used to separate the parameters.

Command Examples See "HISTORY: Real-time Trace" examples.

HISTORY: Real-time Trace Command Examples

NOTE: To illustrate the following examples, memory locations 0 through 1FFF are first filled with NOP instructions. NOPs will be displayed in all the examples.

Example trace mode: End Monitor
Command format: H EM
Trace range: 4K

The ICD defaults to the End Monitor mode when it boots up.

Execute the following:

```
>I 0        ← SPECIFIES IN-CIRCUIT MODE 0
>F 0,1FFF,90    ← FILLS 0 TO 1FFF WITH NOPs (IT TAKES A FEW SECONDS FOR THE
                  ICD TO DO THIS)
>B/A EX,1770    ← SETS A HARDWARE BREAKPOINT TO TERMINATE EMULATION
>G 0:0      ← STARTS EMULATION AND INITIATES REAL-TIME TRACE STORAGE.
              ICD RUNS PROGRAM, STOPS AT BREAKPOINT A, AND DISPLAYS:
01770     90                                NOP
CS      IP ODITSZAPC  AX   BX   CX   DX   SP   BP   SI   DI   SS   S
0000:1772 000000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
<Break Hardware A>
>
```

Now enter:

```
>H D    ← DISPLAYS REAL-TIME TRACE CONTENTS IN DISASSEMBLED FORMAT. (USE
          THE SPACE BAR TO CONTROL SCROLLING; PRESS THE ESC KEY TO EXIT.)
```

Point T	Addr.	Machine Code	Prefix	Opcode Operand
4095	?????	90		NOP
4092	?????	90		NOP
4090	?????	90		NOP
4089	?????	90		NOP
4088	?????	90		NOP
4087	?????	90		NOP
8045	?????	90		NOP

Example trace mode: Begin Monitor
Command format: H BM
Trace rane: 4K

This example continues from the Event Monitor example and uses the same program.

Execute the following:

```
>CLR      ← RESETS THE CLOCK COUNTER; NOT NECESSARY UNLESS CHECKING CLOCK
             COUNT WITH THE STATUS COMMAND
>H BM     ← SETS THE REAL-TIME TRACE TO THE BEGIN MONITOR MODE
>B/A EX,0FA0   ← SETS A HARDWARE BREAKPOINT TO TERMINATE EMULATION
>G 0:0    ← STARTS EMULATION AND INITIATES THE REAL-TIME TRACE STORAGE.
             ICD RUNS PROGRAM, STOPS AT BREAKPOINT A, AND DISPLAYS:

00FA0      90          NOP
CS      IP ODITSZAPC  AX   BX   CX   DX   SP   BP   SI   DI   SS   DS
0000:0FA2 000000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000(
<Break Hardware A>
>
```

Now enter:

```
>H D      ← DISPLAYS REAL-TIME TRACE CONTENTS IN DISASSEMBLED FORMAT. (USE
             THE SPACE BAR TO CONTROL SCROLLING; PRESS THE ESC KEY TO EXIT.)
```

Point T	Addr.	Machine Code	Prefix	Opcode Operand
4095 *				
4092	?????	90		NOP
4090	?????	90		NOP
4089	?????	90		NOP
4087	?????	90		NOP
4084	?????	90		NOP
4082	?????	90		NOP
4081	?????	90		NOP

*NOTE: Indicates trigger point.

Example trace mode: Begin Event
Command format: H BE
Trace range: 4K

This example continues from the Begin Monitor example.

Execute the following:

```
>H BE      ← SETS THE REAL-TIME TRACE TO THE BEGIN EVENT MODE
>EV ST=EV;A=1770      ← SETS AN EVENT POINT
>B/A EX,1F40      ← SETS A HARDWARE BREAKPOINT TO TERMINATE EMULATION
>G 0:0      ← STARTS EMULATION. ICD RUNS PROGRAM, STOPS AT BREAKPOINT A,
              AND DISPLAYS:

01F40      90          NOP
CS      IP ODITSZAPC  AX   BX   CX   DX   SP   BP   SI   DI   SS      S
0000:0FA2 000000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0  )0(
<Break Hardware A>
>
```

Now enter:

```
>H D      ← DISPLAYS REAL-TIME TRACE CONTENTS IN DISASSEMBLED FORMAT. (USE THE
            SPACE BAR TO CONTROL SCROLLING; PRESS THE ESC KEY TO EXIT.)
```

Point	T	Addr.	Machine Code	Prefix	Opcode Operand
3343	*				
3342		?????	90		NOP
3341		?????	90		NOP
3339		01776	90		NOP
3336		01777	90		NOP
3334		01778	90		NOP
3333		01779	90		NOP
3332		0177A	90		NOP
3331		0177B	90		NOP
3329		0177C	90		NOP

*NOTE: Indicates event point.

Example trace mode: Center Event
Command format: H CE
Trace range: 4K

This example continues from the Begin Event example.

Execute the following:

```
>H CE      ← SETS THE REAL-TIME TRACE TO THE CENTER EVENT MODE
             (RANGE DEFAULTS TO 2046)
>EV ST=EX,A=1770      ← SETS AN EVENT POINT
>B/A EX,1F40      ← SETS A HARDWARE BREAKPOINT TO TERMINATE EMULATION
>G 0:0      ← STARTS EMULATION AND INITIATES THE REAL-TIME TRACE STORAGE. ICD
              RUNS PROGRAM, STOPS AT BREAKPOINT A, AND DISPLAYS:

01F40      90          NOP
CS      IP ODITSZAPC  AX   BX   CX   DX   SP   BP   SI   DI   SS   DS
0000:0FA2 000000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000(
<Break Hardware A>
>
```

Now enter:

```
>H D      ← DISPLAYS REAL-TIME TRACE CONTENTS IN DISASSEMLED FORMAT. (USE
            THE SPACE BAR TO CONTROL SCROLLING; PRESS THE ESC KEY TO EXIT.)

Point T   Addr.  Machine Code          Prefix   Opcode Operand
0750      00000  90                             NOP
0748      00001  90                             NOP
0747      00002  90                             NOP
0745      00003  90                             NOP
0742      00004  90                             NOP
0740      00005  90                             NOP
0739      00006  90                             NOP
0738      00007  90                             NOP

0005      001BE  90                             NOP
0002      Pause
0001  *
>
```

*NOTE: Indicates event point.

Example trace mode: End Event
Command format: H EE
Trace range: 4K

This example continues from the Center Event example.

Execute the following:

```
>H EE      ← SETS THE REAL-TIME TRACE TO THE END EVENT MODE
>EV ST=EX,A=1770      ← SETS AN EVENT POINT
>B/E ON      ← ENABLES AN EVENT POINT BREAK
>G 0:0      ← STARTS EMULATION AND INITIATES THE REAL-TIME TRACE STORAGE. ICD
               RUNS PROGRAM, STOPS AT EVENT POINT, AND DISPLAYS:

01770      90          NOP
CS      IP ODITSZAPC  AX   BX   CX   DX   SP   BP   SI   DI   SS      S
0000:0FA2 000000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0  00
<Break Event>
>
```

Now enter:

```
>H D      ← DISPLAYS REAL-TIME TRACE CONTENTS IN DISASSEMBLED FORMAT. (USE
             THE SPACE BAR TO CONTROL SCROLLING; PRESS THE ESC KEY TO EXIT.)

Point T   Addr.  Machine Code          Prefix   Opcode Operand
2045      00000  90                             NOP
2043      00001  90                             NOP
2042      00002  90                             NOP
2040      00003  90                             NOP
2037      00004  90                             NOP
2035      00005  90                             NOP
2034      00006  90                             NOP
2033      00007  90                             NOP

0003      004C9  90                             NOP
0002      Pause
0001   *
>
```

*NOTE: Indicates event point.

Example trace mode: Multiple Event
Command format: H ME
Trace range: 100

This example continues from the End Event example. For this example, a Jump (JP) instruction is added at location FFE so that the ICD will loop during execution. (Loop passing counts are added to the breakpoint.)

Execute the following:

```
>A FFE      ← ASSEMBLES FROM ADDRESS FFE

ICD Displays:     Your Response:
0000:0FFE           JMP OH <cr>
0000:1001             <cr>
>
>H ME,100      ← SETS THE REAL-TIME TRACE TO THE MULTIPLE EVENT MODE
                  AND THE STORAGE SIZE AS 100 INSTRUCTIONS PER LOOP
>B/A .EX,F00   ← SETS A HARDWARE BREAKPOINT TO TERMINATE EMULATION
>EV ST=EX,A=0700   ← SETS AN EVENT POINT
>B/E OFF       ← DISABLES PROGRAM BREAK BY AN EVENT POINT
>G 0:0         ← STARTS EMULATION
```

ICD runs program, stops at hardware breakpoint, and displays:

```
00F00     90                                              NOP
CS      IP ODITSZAPC  AX   BX   CX   DX   SP   BP   SI   DI   SS   DS
0000:0F02 000001011  0000 0000 0001 FFF2 54AB 0000 5E54 0000 0000 0000
<Break Hardware A>
>
```

Now enter:

```
>H D      ← DISPLAYS REAL-TIME TRACE CONTENTS IN DISASSEMBLED FORMAT. (USE
             THE SPACE BAR TO CONTROL SCROLLING; PRESS THE ESC KEY TO EXIT.)

Point  T Addr.  Machine Code          Prefix   Opcode Operand
0102   *
0100     ?????  90                             NOP
0100     ?????  90                             NOP
0099     ?????  90                             NOP
0098     ?????  90                             NOP
0097     ?????  90                             NOP
0095     ?????  90                             NOP
0092     ?????  90                    NOP
0090     ?????  90                    NOP

0007     ?????  90                             NOP
0005     ?????  90                             NOP
0002     ?????  90                             NOP
0001     Pause
>
```

*NOTE: Indicates event point every 100 points in the buffer.

Command HISTORY: Real-time Trace Search By Machine Cycle

Operation Searches through the History trace buffer for certain specified operations. For example, "find all of the times a memory write operation to memory location 1234H occurred."

Syntax H S,/[*addr*]/[*cycle*]/[*data*]/[*seg*][,*int_point*][,*term_point*]

Terms *addr* = Value to search for ("*addr*_W" means to search for a word address).

cycle = Type of machine cycle, and includes one of the following:

MR (memory read)
MW (memory write)
PR (port read)
PW (port write)
IA (interrupt acknowledge)
HA (halt acknowledge)
OF (operation code fetch)
NR (NDP read)
NW (NDP write)

data = Data to search for. (Must also specify address.)

seg = Segment qualifier, and includes one of the following:

CS (code segment)
DS (data segment)
ES (extra segment)
SS (stack segment)

int_point = Initial point of display, from 1 to 4095.

term_point = Point at which display terminates, from 1 to 4095.

Syntax Example H S,/1000/OF/90/CS,200,100

Notes If data is specified, *addr* specification is also required. The *int_point* defaults to 4095, and *term_point* defaults to 1; otherwise, *int_point* must be specified as greater than or equal to *term_point*.

The storage pointer is numbered by bus cycles—displayed from high to low—where "1" is the most recent bus cycle.

This command displays items on a line-for-line basis. To control the scrolling of the display, alternately press the space bar. To exit the display, press the Escape (Esc) key.

Spacing: A space is required between H and S, and thereafter no spaces are permitted; slashes and commas are used to separate information. Slashes must still be present (e.g. H S,///ES) if *addr*, *data*, or *cycle* is excluded or if address is singularly specified (e.g. HS,1234///).

Command Example See Syntax Example.

Command IDENTIFICATION

Operation Displays the current ICD device name and the firmware version.

Syntax ID

Notes This display is also shown when the RESET switch is pressed on the ICD.

Command Example

```
>ID
ICD-178 for i88        v2.0
```

This example shows that the ICD emulates the 8088 processor and that the firmware version within the ICD is 2.0 (if the 8086 processor was installed, "i86" would be displayed). Your firmware version may be different than 2.0, depending on your purchase date.

Command IN-CIRCUIT: Status

Operation Displays the current in-circuit status, either 0, 1, or 2. The in-circuit status is also displayed when the "MAP: Status" command is used.

Syntax I

Command Example See the MAP command.

Command IN-CIRCUIT: Specification

Operation Sets the ICD mapping code. See Notes (below) and the MAP command for an explanation and example of the different mapping modes.

Syntax I [*mode*]

Terms *mode* = 0, 1, or 2

Syntax Example I 0

Notes 0 = System mode. Debugging is performed using the ICD program memory only. The area specified as US (user memory) by the MAP command acts as RW (read/write memory) in the ICD. Target system I/O and interrupt signals are ignored.

1 = Partial mode. Debugging is performed using the ICD program memory and user (target system) memory, as defined by the MAP command. Interrupts can be disqualified by using the PIN command.

2 = All mode. Debugging is performed using only the target system memory. Memory now mapped as read/write and read-only act as user (target system) memory. I/O and interrupts are enable. Any area mapped as NO (non-memory) will act as NO memory regardless of the in-circuit mode.

In-circuit mode settings and memory specifications are shown below.

In-circuit Mode/Description	Memory Type RO	RW	US	NO	PIN Functions EN	DI
IO/System Mode	RO	RW	(RW)	NO	(DI)	DI
I1/Partial Mode	RO	RW	US	NO	EN	DI
I2/All Mode	(US)	(US)	US	NO	EN	(EN)

( ). Items in parentheses show the revised memory or PIN specification for that particular in-circuit mode.

Spacing: A space is required between I and *mode*.

Command Example See the MAP command.

Command LOAD

Operation Downloads an Intel-Hex file from the host computer to the ICD's memory (or through the ICD to user memory).

Applications Note: This command can be used in both LOCAL (ICD controlled by a terminal, using a computer for storage) and REMOTE (ICD controlled by a host computer running ZICE software) mode.s

Syntax L[/*source*] *filename*[.*ftype*] [,*bias*]

Terms *source* = T, P, A, or H

filename = Name of the file to download to the ICD.

.*ftype* = Optional filetype (.abs is the default).

bias = Memory address offset to be added to the object file being loaded (default is 0).

Syntax Example

```
L/H  TEST.H86,100
L/A  ,200
L/A
```

Notes If *source* is omitted, command defaults to H in the REMOTE mode or LOCAL with HOST ON mode, and T in the LOCAL mode.

T specifies to use the TERMINAL port and X-ON/X-OFF protocol. P specifies to use the TERMINAL port and software protocol. A specifies to use the HOST/AUX port and X-ON/X-OFF protocol. H specifies to use the HOST/AUX port and software protocol. (see software specifications in Section 4 for a description of the software protocol.)

When using XON-XOFF protocol options (T, A), it is necessary for the host to either recognize XON-XOFF, or delays must be inserted after each carriage return (end of each record). Otherwise, every second record may be lost. Also, if recognition of XOFF by the host computer is slow (more than two characters), the problem could exist as well. In certain instances, a slower baud rate may help to correct the problem (but is usually undesirable, due to extended download times, especially with long files).

Spacing: A space is required before *filename*; no spaces are permitted where commas act as separators.

Command Example

See Syntax Example. The first example shows how the LOAD command is used with ZICE (host software utilizing software protocol). If ZICE is used, H becomes the default, and may therefore be omitted. With this example, a bias of 100 is added to the load address.

The second example loads a file from a host computer not using ZICE software. For this application, the ICD's HOST/AUX port must be connected to a port on the host computer normally designated for a terminal (one having access to the OS command language).

The third example is used when the host computer's OS command language cannot be accessed via the SIO port, but rather from a separate terminal. This command will be given to the ICD first, then the ICD will wait—ready to receive input prompted from the host terminal.

Command MAP: Status

Operation Displays the current memory assignments and address parameters as defined by the "MAP: Specification" command.

Syntax MA

Command Example Execute this sequence:

```
>I  O          ← USES ICD'S MEMORY RESOURCES
>MA            ← SHOWS HOW MEMORY IS CATEGORIZED
In-Circuit Mode 0 (US=>RW)
00000-FFFFF = RW (000)
>
```

In this example (default condition), the in-circuit mode is first set to 0 (debugging using ICD memory only), and then the MAP status command is entered. The display shows that the in-circuit mode is 0, that user (target system) memory now acts as read/write memory (US=>RW), and that the entire memory area (from 0D to FFFFF) is categorized as read/write memory. The (000) ranges from 0 to FFF and indicates the block number in 1K-block increments. 0 to 32 (1F) is the standard range.

A second example is shown below:

```
>I  2          ← USES TARGET SYSTEM'S MEMORY RESOURCES
>MA            ← SHOWS HOW THE MEMORY IS CATEGORIZED
In-Circuit Mode 2 (RW,RO=>US)
00000-FFFFF = RW (000)
>
```

In this example, the I 2 mode (debugging using target system memory only) is selected, and then the MAP status is requested. The display shows that the in-circuit mode has changed to 2, and that all memory categorized as read/write or read-only (from 0 to FFFFF) now functions as user (target system) memory.

Command MAP: Specification

Operation Categorizes the target system's memory functions as either read-only, read/write, user (target system) or non-memory area.

Applications Note: This command can be used to develop your target system's firmware (ROM) by allowing code in a main-frame system to be downloaded to the ICD, mapped as RO, and tested before being burned into the target's ROM.

Syntax MA *beg_addr*[,*end_addr*]=*area*

Terms *beg_addr* = The beginning address of mapping.

end_addr = The ending address of mapping.

area = RO, RW, US, or NO

Syntax Example

```
MA 1000,1FFF=US
MA 150=RO
```

Notes The target system or ICD memory is used in 1K-byte blocks. The parameters are only valid when the in-circuit mode is I1. (See IN-CIRCUIT command.)

If the *beg_addr* or *end_addr* does not coincide with the beginning or ending of a 1K-block location, the beginning or ending area is assigned a location that includes *beg_addr* or *end_addr*.

Two of the areas, RO and RW, refer to ICD user memory, and RW gives the user program free access to this memory. RO enables the user program to read this memory, but any attempt to write to this area will be blocked, and (unless the B/W break-point is disabled) will also cause a break during program execution.

US acts as target system memory area (US being RAM, ROM, I/O, etc.—whatever resides at those locations in the target). NO memory assignment is useful in debugging by causing a break in the emulated program if an attempt is made to access this non-existent memory area. A NO memory area is recognized as such, regardless of the in-circuit mode.

Spacing: A space is required between MA and *beg_addr*. No spaces are permitted after *beg_addr*; the comma and equal sign act as the separators.

Command Example

Execute this sequence:

```
>I  1          ← USES BOTH ICD AND TARGET SYSTEM MEMORY RESOURCES
·□  π☎▫▫▫ ±
>MA 1000,1FFF = US        ← CATEGORIZES MEMORY BLOCKS
>MA  2000,FFFF = RW
>
>MA          ← SHOWS HOW THE MEMORY IS CATEGORIZED
In-Circuit Mode 1
00000-00FFF = RO (000)
01000-01FFF = US
02000-FFFFF = RW (008)
>
```

In this example, the I 1 (debugging using both ICD memory and target system memory) is selected, and then the memory blocks are categorized as read-only (0 to FFF), user (1000 to 1FFF), and read/write (2000 to FFFFF). The MAP status command is then entered, showing how the memory was just specified. A second example is shown below:

```
>I 2          ← USES TARGET SYSTEM MEMORY RESOURCES
>MA           ← SHOWS HOW THE MEMORY IS CATEGORIZED
In-Circuit Mode 2 (RW,RO=>US)
00000-00FFF = RO (000)
01000-01FFF = US
02000-0FFFF = RW (008)
>
```

In this example, the I 2 (debugging using target system memory only) is selected, which automatically categorizes read/write and read-only memory areas (from 0 to FFFFF) as user (target) memory (RW,RO=>US).

Command MOVE

Operation Moves the memory contents between different locations within the ICD, or between the ICD and the target system.

Syntax M *beg_addr,end_addr,mov_addr*[,*direction*]

Terms *beg_addr* = Beginning address of data source.

end_addr = Ending address of data source.

mov_addr = Beginning address for destination.

direction = UP or PU

Syntax Example M 100,3FF,100,UP

Notes UP means that the source is user (target system) memory and the destination is ICD program memory. PU means that the source is ICD program memory and the destination is user (target system) memory. If direction is omitted, data is relocated within the memory areas as specified by the MAP command.

Spacing: A space is required between M and *beg_addr*. No spaces are permitted where commas are used as separators.

Command Example

See Syntax Example. In this example, a block of memory in the target system, beginning at address 100 and ending at address 3FF, is moved to the ICD, beginning at address 100.

For an application of the MOVE command, carry out the demonstration below:

```
>F  0,3FF,55        ← FILLS A 1K MEMORY BLOCK WITH 55s
>D  0,3FF           ← DISPLAYS THE MEMORY CONTENTS
>M  290,3A0,2EFF    ← MOVES A SECTION OF THE MEMORY
                      TO ADDRESS 2EFFH
>
>D  2EFF,300E       ← DISPLAYS THE TRANSFERRED MEMORY CONTENTS
>                     AT THE NEW LOCATION
```

Command NEXT

Operation This command is a subcommand of the TRACE command. It allows the next 1 to 65,535 instructions to be executed and traced in non-real time from the current CS:IP.

Syntax N [*steps*]

Terms *steps* = 1 to 65,535

Syntax Example N 5

Notes The *steps* means the number of instructions to execute from the current program counter, and may be any integer from 1 to 65,535. IF *steps* is omitted, only a single instruction line is displayed.

When the registers' contents are displayed as a series of periods (. . . .), it indicates that the contents of the registers are unchanged. The registers' contents are displayed fully, however, at least once every 22 lines.

Spacing: A space is required between N and *steps*.

Command Example

Press the RESET switch on the ICD, then execute this sequence:

```
001F0       IP ODITSZAPC  AX    B    CX    DX    SP    BP    SI    DI    SS
0000:01F2      000000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0
>Break Hardware A<
>N  3        ← SHOWS THE NEXT THREE INSTRUCTION LINES

CS          IIP ODITSZAPC  AX    BX    CX    DX    SP    BP    SI    DI    SS
001F2           90                                NOP
0000:01F3      000000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0
001F3           90                                NOP
0000:01F4      .......... .... .... .... .... .... .... .... .... .... ....
001F4           90                                NOP
0000:01F5      .......... .... .... .... .... .... .... .... .... .... ....
>
```

This example illustrates how the NEXT command is used after program execution halts. When the program stops at address 1F0, entering N 3 causes the next three instruction lines to be displayed.

Command OFFSET: Status

Operation Displays the status of the "OFFSET: Specification" command.

Syntax 0

Command Example

```
>0          ← SHOWS THE STATUS OF THE OFFSETS
&0 = 0000   ← SHOWS THE DEFAULT CONDITIONS (ALL OFFSET REGISTERS = 0)
&1 = 0000
&2 = 0000
&3 = 0000
>
```

This example shows the default condition of the OFFSET command. Changing the address of any one of the four offset values (0-3) causes a change in the 0000 display.

Command OFFSET: Specification

Operation Sets an offset in the ICD for relative program addressing.

Applications Note: This command is useful when debugging a program that consists of a number of different modules. The procedure would be to assign the physical base address for each module to one of the offset registers. Any location in a module may be addressed by specifying its relative address to that module's base address, plus an offset register. The address parameter of any command will then be interpreted as the sum of the relative address and the offset register (physical base address).

Syntax 0 &*number*[=*addr*]

Terms *number* = 0, 1, 2, or 3

addr = Offset to place in the register.

Syntax Example 0 &2=FFF

Notes Any of the four offset registers can be used with any of the ICD command memory addressing parameters.

When *addr* is omitted, the offset register is cleared to zero.

Spacing: A space is required between 0 and &. No spaces are permitted between &*number*=*addr*; the equal sign (=) acts as the separator.

Command Example Execute this sequence:

```
>0  &1=351      ← SETS #1 VALUE TO OFFSET OF 351
>0      ← SHOWS CURRENT OFFSET VALUES
&0 = 0000
&1 = 0351
&2 = 0000
&3 = 0000
C  1377H+351H      ← USE CALCULATION COMMAND TO FIND OFFSET
000016C8H            ADDRESS LOCATION (1377H+351H=16C8H)
5832
>
>DI  0:1377&1      ← DISASSEMBLES FROM ADDRESS 1377H + THE
                     OFFSET VALUE
0000:16C8  0000          ADD . . . .
0000:16CA  0000          . . .  . . . .
0000:16CE  0000          . . .  . . . .
0000:16CF  0000          . . .  . . . .
. . . . . . . .  . . . . etc.
```

Command PIN: Status

Operation Displays the current status of the "PIN: Specification" command.

Syntax PI

Command Example

```
>PI      ← SHOWS STATUS OF INPUT SIGNALS IN THE I O MODE
All Pin Disable
MAX
NMI       (EN) = L
INTR      (EN) = H
TEST/          = H
RQGT/     (EN)
>
```

This example shows the status of the inputs signals when the in-circuit mode is 0 (in this mode, the input signals cannot be enabled). H shows that the current logic levels of the signal are high. The slash (X) after the signal name signifies an "active-low" signal.

If the ICD was operating in the I 1 mode, the display would show:

```
>PI
In-Circuit Mode 1
NMI       (EN) = L
INTR      (EN) = H
TEST/          = H
RQGT/     (EN)
>
```

If the in-circuit mode was 2, all input signals would automatically be disabled.

Command PIN: Specification

Operation Masks or unmasks selected input signals when the in-circuit mode is 1.

Syntax PI *signal=switch*

Terms *signal* = NMI (non-maskable interrupt)
INTR (interrupt request)
RQGT (request grant—MAX mode only)
HOLD (DMA hold request—MIN mode only)

switch = EN or DI

Syntax Example PI INTR=DI

Notes The parameters for this command are only valid when the in-circuit mode is 1. When the in-circuit mode is 2, all signals are valid. When the in-circuit mode is 0, all target system signals are ignored.

EN is used to enable the signal and DI is used to disable the signal.

Spacing: A space is required between PI and *signal*. No spaces are permitted after *signal*.

Command Example Execute this sequence:

```
>I  1      ← SETS MODE TO PERMIT PIN ALTERATION
>PI        ← SHOWS STATUS OF INPUT SIGNALS
In-Circuit Mode 1
NMI        (EN) = L
INTR       (EN) = H
TEST/           = H
RQGT/      (EN)
>
>PI  NMI=DI      ← DISABLES THE NMI SIGNAL
>PI        ← SHOWS THE STATUS OF INPUT SIGNALS AGAIN
In-Circuit Mode 1
NMI        (DI) = L      ← VERIFIES THE CHANGE
INTR       (EN) = H
TEST/           = H
RQGT/      (EN)
>
```

In this example, the in-circuit mode 1 is selected (ICD and target system memory resources) to manipulate the various input signals. The PIN status then shows that all the inputs are active (ENabled). Next, the NMI signal is disabled, and the PIN status is used again to verify the change.

Command PORT

Operation Examines one or more I/O port locations and optionally modifies them. The locations can be displayed and replaced with either hexadecimal or ASCII values.

This command works on the same principle as the EXAMINE command, except that the port address accesses the I/O port space.

Syntax P[/W] *port_addr*[=*mod_data*]

Terms W = Word mode (default is the byte mode).

port_addr = Starting address for display.

mod_data = New data for this location.

Syntax Example
```
P FF=23
P 55
```

Notes If *mod_data* is omitted, the command enters a repeat mode, which allows several locations to be changed.

The repeat mode includes:

return (cr) to display the next byte (word) of data.
comma (,) to display the same byte (word) of data.
caret () to display previous byte (word) of data.
slash (X) to exit the PORT command.

Spacing: A space is required between P and *port_addr*. No spaces are permitted between *port_addr* and *mod_data*; the equal sign (=) acts as the separator.

Command Example

See Syntax Example. The first example illustrates how the port located at address FF is changed to a data value of 23. The second example allows the ports to be modified, beginning at address 55.

Now examine the following display:

```
>P  12      ← STARTS BY EXAMINING PORT #12
0012  12=23,      ← CHANGES VALUE TO 23; RE-EXAMINE
0012  12=         ← LEAVES VALUE UNCHANGED; GO TO NEXT ADDRESS
0013  00=         ← LEAVES VALUE UNCHANGED; GO TO NEXT ADDRESS
0014  14='21'     ← CHANGES VALUE; GO TO NEXT ADDRESS
0015  00=34^      ← CHANGES VALUE; GO TO PREVIOUS ADDRESS
0014  14=^        ← LEAVES VALUE UNCHANGED; GO TO PREVIOUS ADDRESS
0013  00=,        ← LEAVES VALUE UNCHANGED; RE-EXAMINE ADDRESS
0013  00=17       ← CHANGES VALUE; GO TO NEXT ADDRESS
0014  14=         ← LEAVES VALUE UNCHANGED; GO TO NEXT ADDRESS
0015  00=         ← LEAVES VALUE UNCHANGED; GO TO NEXT ADDRESS
0016  16=00/      ← CHANGES VALUE; EXIT COMMAND
>
```

Commnd PRINT

Operation Controls logging of ICD commands by sending the terminal display to an external serial printer.

Syntax PR *switch*

Terms *switch* = ON or OFF

Syntax Example PR ON

Notes ON enables the printing feature and OFF disables the printing feature.

The printing is routed to the HOST/AUX port when the ICD is in LOCAL mode, and to the host printer when the ICD is in REMOTE mode (using ZICE, or the LOCAL "HOST ON" mode using ZICE).

Spacing: A space is required between PR and *switch*.

Command Example See Syntax Example.

Command REGISTER: 8086/8088 Status

Operation Displays the current status of the 8086/8088 registers and any changes made after using the "REGISTER: Examine and Change" command.

Syntax R

Notes The 8086/8088 register contents can be changed by using the "REGISTER: Specification" command.

Command Example

```
>R
  CS    IP   ODITSZAPC  AX   BX   CX   DX   SP   BP   SI   DI   SS   DS
0000:0000  000000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
>
```

This example shows the status of the 8086/8088 registers (currently all 0). Changing any of the registers with the "REGISTER: Examine and Change" command affects this display.

Command REGISTER: 8087 Status

Function Displays the current status of the 8087 registers.

Syntax R[//*status*]

Terms *status* = E or N

Syntax Example
R/E
R/N

Notes E displays the 8087 registers ST(0) — ST(7), and N displays all 8087 registers.

The 8087 register contents can be changed by using the "REGISTER: Specification" command.

Spacing: No spacing is allowed between parameters.

Command Example

```
R/N
Control Word:
  X  X  X  IC  -RC-  -PC-  IEM  X  PM UM OM ZM DM IM
  0  0  0  0   0     0     0    0  0  0  0  0  0  0  0
Status Word:
   B   C3   --TOP--  C2 C1 C0  IR  X  PE  UE  OE  ZE  DE  IE
  0  0  0  0    0     0  0  0   0  0  0   0   0   0   0
Instruction Address:
  00000
Last Operation:
  D8  00:
Operand Address:
  00000
Stack Register:
  ST(0)  00  00  00  00  00  00  00  00  00  00    Tag=0
  ST(1)  00  00  00  00  00  00  00  00  00  00    Tag=0
  ST(2)  00  00  00  00  00  00  00  00  00  00    Tag=0
  ST(3)  00  00  00  00  00  00  00  00  00  00    Tag=0
  ST(4)  00  00  00  00  00  00  00  00  00  00    Tag=0
  ST(5)  00  00  00  00  00  00  00  00  00  00    Tag=0
  ST(6)  00  00  00  00  00  00  00  00  00  00    Tag=0
  ST(7)  00  00  00  00  00  00  00  00  00  00    Tag=0
>

>R/E
  ST(1)  +0.00000000000000000E+0          Valid
  ST(2)  +0.00000000000000000E+0          Valid
  ST(3)  +0.00000000000000000E+0          Valid
  ST(4)  +0.00000000000000000E+0          Valid
  ST(5)  +0.00000000000000000E+0          Valid
  ST(6)  +0.00000000000000000E+0          Valid
  ST(7)  +0.00000000000000000E+0          Valid
>
```

Command REGISTER: Reset

Operation Sets all the registers to zero (except for CS, which remains FFFF).

Syntax R RESET

Notes To reset CS to 0000, use the "REGISTER: Specification" command: R CS=0.

Spacing: A space is required between R and RESET.

Command Example Execute this sequence:

```
>R      ← SHOWS THE STATUS OF THE REGISTERS
>R CX=2FFE      ← SETS REGISTER CX TO A VALUE OF 2FFEH
>R      ← SHOWS THE STATUS OF THE REGISTERS AGAIN
>R RESET      ← RESETS ALL REGISTER VALUES TO 0
>R      ← VERIFIES THE CHANGE TO 0
>
```

This example shows how register CX is changed from 0000 to 2FFE, and then set back to 0000 using the "REGISTER: Reset" command.

Command REGISTER: Examine and Change

Operation Examines and changes the contents of the 8086/8088 or 8087 internal registers.

Syntax R[/N] *reg_name*=[*data*]

N = No-verify (the default is to read-verify after write).

Terms *reg_name* = Any one of the following registers:

AX	AH	AL	BX	BH	BL
CX	CH	CL	DX	DH	DL
CS	DS	ES	SS	SP	IP
BP	SI	DI	FL	O D I T S Z A P C	

data = New value for register contents.

Syntax Example

```
R HL=A000
R DE
```

Notes If R *reg_name* is entered, this command displays the current contents of the specified register. If *data* is used, this command changes the contents of the specified register to the new value.

For *reg_names* ODITSZAPC, only 0 and 1 are valid data entries.

Spacing: A space is required between R and *reg_types*. No spaces are permitted after *reg_types*; the equal sign (=) acts as the separator.

Command Example

Execute this sequence:

```
>R  RESET      ← INITIALIZES REGISTERS TO 0
 CS       IP   ODITSZAPC  AX   BX   CX   DX   SP   BP   SI   DI   SS
FFFF:0000     000000000 0000 0000 0000 0000 0000 0000 0000 0000 000(
>
>R  DX=2FFF    ← SETS THE DX REGISTER TO 2FFF
>R  DX         ← SHOWS THE VALUE OF THE DX REGISTER
2FFF           ← VALUE OF DX REGISTER
>R             ← SHOWS THE VALUES OF ALL THE REGISTERS
 CS       IP   ODITSZAPC  AX   BX   CX   DX   SP   BP   SI   DI   SS
FFFF:0000     000000000 0000 0000 0000 2FFF 0000 0000 0000 0000 000(
```

This example illustrates how a register is changed to a new value, and the two ways in which it can be checked.

Command SAVE

Operation Saves an Intel Hex file from the ICD memory to the host computer. (The file format is the same as the LOAD command.)

Syntax S[/*destination*] *filename*[.*ftype*],*beg_addr*,*end_addr*, *entry_addr*

Terms *destination* = T, P, A, or H.

filename = Name of the file to be used for saving the memory contents.

.*ftype* = Optional three-letter filetype (.abs is the default).

beg_addr = First address to save.

end_addr = Last address to save.

entry_addr = Starting address of the user program.

Syntax Example SA/H TEST.H86,0,3FF,1000

Notes

If *destination* is omitted, command defaults to H in the REMOTE (host computer control of the ICD) mode or LOCAL with HOST ON (host computer assisted) mode, and T in the LOCAL (terminal control of the ICD) mode.

T specifies to use the TERMINAL port and X-ON/X-OFF protocol. P specifies to use the TERMINAL port and software protocol. A specifies to use the HOST/AUX port and X-ON/X-OFF protocol. H specifies to use the HOST/AUX port and software protocol. (See software specifications in Section 4 for a description of the software protocol.)

Either XOFF-XON or DTR-DSR flow control will be accepted by the ICD when the *destination* option is T or A. If the host computer does not provide input flow-control, its input buffer will probably overflow.

Spacing: A space is required before *destination*; no spaces are permitted where commas act as separators.

Command Example

See Syntax Example.

Command SEARCH

Operation Searches the memory contents and displays the matching or unmatching data, if any.

Syntax S[/W] [/D] *beg_addr,end_addr,search_data*

Terms W = Word search (if omitted, byte search is made).

D = Search for unmatching data (if omitted, search is made for matching data).

beg_addr = Address to begin search.

end_addr = Address to end search.

search_data = Data to search for.

Syntax Example S/D 100,7FF,55

Notes This command displays items on a line-for-line basis. To control the scrolling of the display, alternately press the space bar. To exit the display, press the Escape (Esc) key.

Spacing: A space is required before *beg_addr*. No spaces are permitted where the commas act as separators.

Command Example See Syntax Example. This example illustrates that a search of the memory contents is made from address 100 to address 7FF. The display will show all locations that contain data other than 55.

Command SUPERVISOR: Specification

Operation Provides a way to access the ICD's serial ports (TERMINAL or HOST/AUX) from the emulated program by using specified breakpoints as supervisor calls to the ICD system.

The breakpoints in the emulated program do not stop the program being emulated, but perform input/output to the ICD serial interface only.

IMPORTANT! Do not use the HOST/AUX port to output data during a supervisor call if ZICE software is being used, or the communication protocol will be disturbed.

Syntax SU[/*break switch*]

Terms *break* = C, 7, or U

switch = ON or OFF

Syntax Example
```
SU/7  ON
SU
```

Notes C specifies to use hardware breakpoint C as a supervisor call, 7 specifies to use software breakpoint 7 as a supervisor call, and U specifies to use a user software breakpoint as a supervisor call. ON enables the specified breakpoint (C, 7, or U), and OFF disables it.

If a user software breakpoint is specified, the supervisor call will occur at each user software breakpoint. In this way, multiple calls can be used throughout a program.

The function code of the supervisor call is specified in the DL register, and the I/O data is transferred via the AL register. (See Function Code Key chart.)

Omitting all prameters will display the current supervisor call settings.

Spacing: A space is required between *break* and *switch*. No spaces are required before *break*.

Command Example Execute this sequence:

```
>R  RESET      ← RESETS THE REGISTERS TO 0
>R  CS=0       ← SETS THE CS REGISTERS TO 0
>A  100        ← STARTS ASSEMBLING THE SAMPLE PROGRAM FROM ADDRESS 100H
0000:0100  MOV  SI,120H
0000:0103  MOV  DL,2
0000:0105  CLD
0000:0106  LODSB
0000:0107  OR   AL,AL
0000:0109  JNZ  106H
0000:010B  HLT
0000:010C      ← <RETURN> HERE TO TERMINATE INPUT
>
>B  S=EN       ← ENABLES ALL SOFTWARE BREAKPOINTS
>B/C  EX  0:107      ← SETS HARDWARE BREAKPOINT C AT ADDRESS 107H
>SU/C  ON      ← USES BREAKPOINT C AS A SUPERVISOR CALL
>F   120,139,'THIS IS A SUPERVISOR CALL'      ← CALL MESSAGE
>F   13A,143,'MESSAGE',0D,0A,00      ← CALL MESSAGE
>G  100        ← RUNS PROGRAM FROM ADDRESS 100H
THIS IS A SUPERVISOR CALL      ← ICD ISSUES MESSAGE THEN STOPS AT BREAKPOINT C

 010B          F4                                        HLT
  CS       IP   ODITSZAPC   AX   BX   CX   DX   SP   BP   SI   DI   SS
0000:010B      000001010  0000 0000 0000 0002 0000 0000 0012 0000 0000
<Break Software User>
>
```

Port Input Status Fetch

Entry Conditions:
Register DL = 01H Get input status from TERMINAL port
Register DL = 11H Get input status from HOST/AUX port

Exit Conditions:
Register DL = Unchanged
Register AL = OH No data is available at specified port
Register AL = FFH Data has been received at specified port

Input Character from Port

Entry Conditions:
Register DL = 00H Input character from TERMINAL port
Register DL = 10H Input character from HOST/AUX port

Exit Conditions:
Register DL = Unchanged
Register AL = Character received from specified port

NOTE: If no character is available at the specified port, control will not return from the supervisor call until a character has been received.

Port Output Status Fetch

Entry Conditions:
Register DL = 03H Get output status from TERMINAL port
Register DL = 13H Get output status from HOST/AUX port

Exit Conditions:
Register DL = Unchanged
Register AL = 00H Port transmit buffer is busy (not ready)
Register AL = FFH Port transmit buffer is empty (ready)

Output Character from Port

Entry Conditions:
Register DL = 02H Output character to TERMINAL port
Register DL = 12H Output character to HOST/AUX port

Exit Conditions:
Register DL = Unchanged
Register AL = Unchanged

NOTE: If transmit buffer is busy when this call is made, control will not be returned until buffer is ready and character has been sent.

FUNCTION	FUNCTION CODE DL-reg	DATA OUT AL-reg	DATA IN
TERMINAL Port data in	0 0	—	RECEIVE DATA
HOST/AUX Port data in	1 0	—	RECEIVE 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

Command TRACE: Status

Operation Displays the current trace setting.

Syntax T

Command Example Execute the following:

```
>T      ← DISPLAYS THE CURRENT TRACE
Trace is Clear      ← SHOWS INACTIVE TRACE
>T  A      ← SETS TRACE TO ALL DISPLAY
>T      ← DISPLAYS NEW TRACE SETTING
(ON)    ALL 0000:0000-FFFF:000F (0000-FFFFF)
>T  J      ← SETS TRACE TO JUMP ONLY DISPLAY
>T      ← DISPLAYS NEW TRACE SETTING
(ON)    ALL 0000-FFFF      ← SHOWS JUMP SPECIFICATIONS
```

Command TRACE: Qualification

Operation Enables, disables, or clears the trace setting.

Applications Note: This command can be used to temporarily disable the software trace feature without affecting its location within the program or the parameter specifications.

Syntax T *switch*

Terms *switch* = ON, OFF, or CLR

Syntax Example T ON

Notes If ON is specified, the trace specification is valid. If OFF is specified, the trace specification is disabled. If CLR is specified, the trace specification is cleared.

Spacing: A space is required between T and *switch*.

Command Example See the Syntax Example and the "TRACE: Specification" command.

Command TRACE: Specification

Operation Performs a software trace of the program in non-real time.

Applications Note: This command allows a section of the user program to be displayed in a step-by-step manner by either automatically scrolling through the program, or moving through the program one line at a time.

Syntax T[/S] *mode*[,*beg_addr*] [,*end_addr*]

Terms S = Single step mode.

mode = AAn or J

beg_addr = Beginning address of memory to trace (default = 0).

end_addr = Ending address of memory to trace (default = FFFFF).

Syntax Example
```
T/S  J,100,300
T  A,200,FFF
```

Notes S causes a single instruction to be executed each time the space bar is pressed. The *mode* must be defined as either A or J. A means that all commands are traced and displayed, and J means all instructions are traced but only Jump instructions are displayed.

If *beg_addr* is omitted, the trace starts from address 0. If *end_addr* is omitted, the trace ends at address FFFFF. When *beg_addr* or *end_addr* is specified, all the instructions are traced, but only the instructions within the specified address range are displayed. The instructions that are located outside of the address parameters are executed in non-real time as well.

Spacing: A space is required between T and *mode* (or T/S and *mode*). No spaces are permitted where commas act as separators.

Command Example

Execute this sequence:

```
>F  0:100,2FF,90   ← FILLS MEMORY WITH NOPs
>F  0:300,,E9      ← FILLS ONE BYTE WITH A JUMP INSTRUCTION
>D  0:0,300        ← DISPLAYS MEMORY TO ADDRESS 300H
>T  A              ← TRACES AND DISPLAYS ALL INSTRUCTIONS
>G  0:100          ← DISPLAYS ALL OF PROGRAM AS IT RUNS
>T  A  100,11F     ← TRACES ALL INSTRUCTIONS FROM ADDRESS 100H TO 11FH
>G  0:100          ← DISPLAYS PROGRAM PER TRACE SPECIFICATION
>T/S  A  100,120   ← TRACES ALL INSTRUCTIONS FROM ADDRESS 100H
                     TO 120H AND DISPLAYS ONE LINE AT A TIME
>G  0:100          ← DISPLAYS ONE INSTRUCTION LINE EACH TIME SPACE BAR IS PRESSED
>T  J              ← DISPLAYS ONLY JUMP INSTRUCTIONS
>G  0:100          ← RUNS PROGRAM AND DISPLAYS JUMP INSTRUCTION
>T  CLR            ← CLEARS THE TRACE FEATURE
```

This example first fills a range of memory with NOPs so that a trace can be performed on the data. After the data is entered it is inspected, and then the trace parameters are specified. The first trace is of all instructions, the second trace is of all instructions from address 100 to 11F, the third trace is of all instructions and display is line-by-line, and the fourth is of Jump instructions only. Finally, the trace feature is cleared from the ICD memory.

Command USER

Operation Allows a single console terminal to communicate with either the ICD or a host computer.

Applications Note: This command enables the ICD to assume a "transparent" condition when it is positioned between a console terminal and a host computer. In this mode, a console terminal (connected to the ICD's TERMINAL port) can communicate directly with a host computer (connected to the ICD's HOST/AUX port). Essentially, the transparent mode uses the ICD as an interface or conduit between the two ports.

Syntax U [*code*]

Terms *code* = A single printing character used to signal the ICD to terminate the transparent communication mode. Control returns to the ICD command mode when this character is entered from the terminal's keyboard.

Notes The Terminal-to-ICD baud rate should be at least double that of the ICD-to-Host baud rate (recommended: host computer = 9600; terminal = 19,200).

U initiates the transparent mode and U *code* terminates the transparent mode.

Spacing: A space is required between U and *code*.

Syntax Example

```
U
U !
U ~
```

Command VERIFY

Operation Compares an Intel Hex format file on the host computer to the ICD memory (or through the ICD to the target memory).

NOTE: All parameters and uses are identical to the LOAD command, with the exception that the VERIFY command does not alter memory; it only compares the memory contents against the file and displays the difference.

Syntax V[/*source*] *filename*[.*ftype*] [,*bias*]

Terms *source* = T, P, A, or H

filename = Name of the file to download to the ICD.

.*ftype* = Optional three letter filetype (.abs is the default).

bias = Memory address offset to be added to the object file being compared (default is 0).

Syntax Example V/H TEST.HEX,100

Notes

T specifies to use the TERMINAL port and X-ON/X-OFF protocol. P specifies to use the TERMINAL port and software protocol. A specifies to use the HOST/AUX port and X-ON/X-OFF protocol. H specifies to use the HOST/AUX port and software protocol. (See software specifications in Section 4 for a description of the software protocol.)

If *source* is omitted, command defaults to H in the REMOTE (host computer controlled) mode and T in the LOCAL (terminal controlled) mode.

See the LOAD command Notes for additional information.

Spacing: A space is required before *filename*; no spaces are permitted where commas act as separators.

Command Example

See Syntax Example and the LOAD command examples for additional information.

Zice Commands—available with ZICE software only.

ZICE Command HOST

Operation Initiates or terminates LOCAL "Host Computer Assisted" mode.

Applications Note: This command enables the ICD to operate as though it is in the REMOTE mode when connected to a host computer running in the LOCAL mode (terminal control of the ICD with host computer access), Using this configuration, only one SIO port is required of a multi-user host computer (e.g., VAX), rather than two ports as required in the REMOTE mode.

Syntax HOST *switch*

Terms *switch* = ON or OFF

Syntax Example HOST ON

Notes This command is only available with firmware versions 2.0 or greater, and only recognized when the ICD is in the LOCAL mode.

ON enables the HOST feature and OFF disables the HOST feature.

The QUIT command will also perform the equivalent of the HOST OFF command, but the HOST OFF command does not terminate ZICE.

Spacing: A space is required between HOST and *switch*.

Command Example See Syntax Example.

ZICE Command QUIT

Operation Exits ZICE software control and returns control to the host computer system, or to the ICD if used in the LOCAL "Host Computer Assisted" mode (see the HOST command).

Syntax Q

Introduction

In this section, you'll learn about the eight internal control modules (including the optional Expansion Memory module) which, with the power supply, make up your ICD. These modules are used to control the various processes that are required for emulation, including electronically substituting your target system's microprocessor with the ICD's processor, controlling communication between the ICD and host computer or terminal, and tracing (and storing) a portion of the program memory contents for analysis.

Special Environments

Although it's not necessary to read this section to use your ICD, you may find the information helpful if you require an examination of how the ICD operates under certain conditions and in particular environments. In certain instances, modules may need to be modified to permit the ICD to operate at peak performance. All possible modifications are detailed in the module "Descriptions," on the following pages.

In order to modify the components and controls, or to change certain settings on the modules, the ICD must be partially or fully disassembled. At the end of this section is a procedure which explains how to disassemble your ICD and remove (and replace) the eight control modules.

IMPORTANT!

▲ This symbol defines the adjustments and modifications to the ICD which are permitted under the Warranty Policy. In order to preserve the warranty on this equipment, do not adjust, modify, and/or in any way alter the controls or components on the modules unless the written procedure for manipulating a particular module is marked by this symbol.

Overview: The Eight Control Modules

Indicator/Control Module (PANEL S-730)

This module contains the Operator Panel switches and indicator lamps. All controls are externally accessible. (There are no user-serviceable controls on this module.)

Serial Interface Output Module (SIO S-771)

This module contains the RS-232 serial interface connectors for the TERMINAL and HOST/AUX ports. A 20mA current loop or TTL level terminal may also be used by changing the configuration of this module. (There are several user-serviceable controls, components, and switches on this module—see "How To Disassemble Your ICD," at the end of this section, after reading about the module's components on one of the following pages.)

Expansion Memory Module (EXM-12 S-766)

This optional module expands the ICD's memory capabilities to 256K bytes (128K standard + 128K expansion). Components on this module include a 60-pin bus receptacle to connect with the Memory Mapping Unit module, and two 8-bit switches that control the module's functions. (To gain access to these components, see "How To Disassemble Your ICD," at the end of this section, after reading about the module's components on one of the following pages. To install the module, see the chapter on the Expansion Memory module.)

Break Comparator Memory Module (BRX S-778)

This module qualifies the conditions (address, data, status) for the BREAK command. (There are no user-serviceable controls on this module.)

CPU Control Module (CPU S-773)

This module contains the connectors, circuitry, CPU (8086/8088) processor, and NDP (8087) co-processor, which allow the ICD to emulate the target system's processors. (There are a few user-serviceable components on this module—see "How To Disassemble Your ICD," at the end of this section, after reading about the module's components on one of the following pages.)

Emulator Control Module (EMU S-775)

This module controls the emulation mode or monitor mode of operation for the ICD. (There are no user-serviceable components on this module.)

Real-time Storage Module (RTS S-775)

This module's circuitry includes the controller, memory, and real-time counter for performing tracing and storage of the user program. (There are no user-serviceable components on this module.)

Memory Mapping Unit Module (MMU S-776)

This module contains 128K bytes of high-speed static RAM (known as "emulation memory"), which can be used for down-loading files, altering the memory contents, and loading future memory into the target system. (There are a few user-serviceable components on this module—see "How To Disassemble Your ICD," at the end of this section, after reading about the module's components on one of the following pages.)

Internal/Control Module Description

The Indicator/Control module (PANEL S-730) contains three switches, four indicator lamps, one 60-pin bus receptacle, and intermediary circuitry. Switch SW1 selects between the internal (INT) or external (EXT) clock; switches SW2 and SW3 activate the RESET and MONITOR functions, respectively. The indicator lamps D1, D2, D3, and D4 show the condition of the HALT, MONITOR, ICE (in-circuit enable), and POWER functions.

The three switches and four indicator lamps are all accessible for operation (and viewing) from outside the ICD; there are no user-serviceable controls or components on this module.

(photo of module)

Serial Interface Output Module Description

The Serial Interface Output (SIO) module (S-791) controls communication between the ICD and various external devices (host computer, terminal, printer) through the TERMINAL and HOST/AUX ports. The SIO module's internal components feature jumper sockets and line drivers that can be modified to permit either RS-232, current loop, or TTL interface operation. There are also two transmission format switches (DSW3 and DSW4) that are used to set the data format and stop bits for the TERMINAL and HOST/AUX ports, and a special socket that allows any key on the console keyboard to activate the MONITOR break switch in the ICD.

These components are all user-serviceable; the ICD must be disassembled before they can be adjusted or modified. (See "How To Disassemble Your ICD," at the end of this section.)

The module's remaining components are all externally accessible. These include the DCE/DTE and LOCAL/REMOTE switches, the TERMINAL and HOST/AUX port connectors, and two rotary switches that set the communication baud rates for the ports.

(SIO halftone photo here)

SIO S-791 Module Components

① **JA Socket.** By connecting different pins with jumpers, this socket is used to select either RS-232, current loop, or TTL interface for the TERMINAL port.

② **JB Socket.** Used the same way as the JA socket, but selects the interface for the HOST/AUX port.

③ **TERMINAL Port Line Driver.** The standard line driver is an SN75188, and is used with RS-232 and current loop interface operation. When TTL interface is used, the standard line driver must be replaced with an SN7438 line driver.

④ **HOST/AUX Port Line Driver.** Functions the same as the TERMINAL port line driver, except controls the HOST/AUX port.

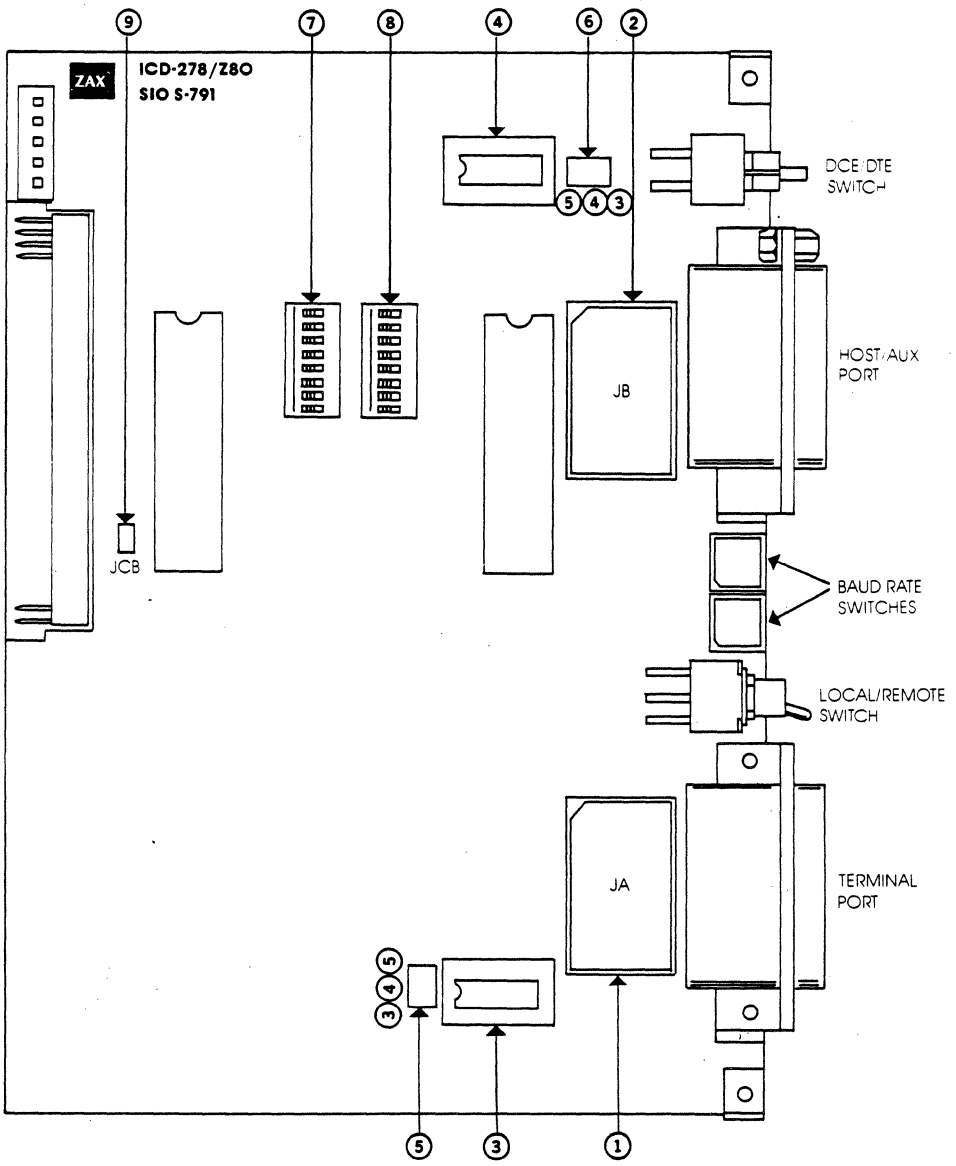
⑤ **JA 5/4/3 Power Supply Jumpers.** Supplies power to the TERMINAL port line drivers. Pins 3 and 5 supply +12V to the SN75188 line driver (when using RS-232 or current loop interface), and Pin 4 supplies +5V to the SN7438 line driver (when TTL interface is used).

⑥ **JB 5/4/3 Power Supply Jumpers.** Functions the same as JA 5&3, but supplies power to the HOST/AUX port line driver.

⑦ **DSW3 Transmission Format Switch.** Sets the data format and stop bits for the TERMINAL port. (See "How To Set The Transmission Format Switches.")

⑧ **DSW4 Transmission Format Switch.** Sets the data format and stop bits for the HOST/AUX port (See "How To Set The Transmission Format Switches.")

⑨ **JCB Console Break Jumper Socket.** When the pins of this socket are connected together, it allows any key on the terminal keyboard to activate the MONITOR break switch; it is essentially the same as pressing the MONITOR switch on the ICD. (The MONITOR switch is used to return control to the ICD monitor during emulation.)



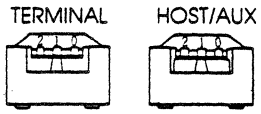
SIO S-791 SERIAL INTERFACE OUTPUT MODULE

Baud Rate Switches

The Baud Rate switches are used to set the baud rates for the TERMINAL and HOST/AUX ports. The factory setting is #1 (9600 bps) for both ports. There are 13 other baud rate settings available; do not set the baud rate switches to E or F.

▲ Changing The Baud Rate Settings

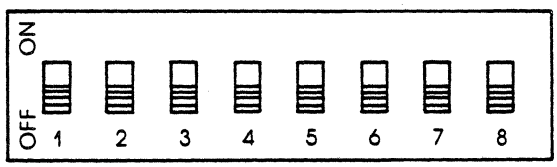
The Baud Rate switches are rotary-type switches., To change the baud rates, turn the dials to the number or letter shown in the Baud Rate diagram below. Use a pointed object such as a pen tip or a small screwdriver.



Baud Rate Switch No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Baud Rate (bps)	19.2K	9.6K	4.8K	2.4K	1.2K	600	300	150	75	110	134.5	200	1.8K	2K	—	—

▲ How To Set The Transmission Format Switches

The transmission format switches are used to set the data format and stop bits for the TERMINAL and HOST/AUX ports. Both 8-bit, ON/OFF type switches can be set by inserting a small, pointed tool and sliding the bits to the ON or OFF position.



Bit	OFF	ON
1	Data bit 8	Data bit 7
2	No parity bit	Enable parity bit
3	Even parity	Odd parity
4	Stop bit 2	Stop bit 1
5	Bit 8 always 0	Bit 8 always 1
6	Multi-ICD I/O disable	Multi-ICD I/O enable
7	Multi-ICD I/O disable	Multi-ICD I/O enable
8	TBMT & TEOC	TBMT only

Factory Settings All bits = OFF

NOTE 1: When bit 8 is set to OFF, the ICD transmits on a single buffer basis for monitoring the BUSY state. When this bit is set to ON, the ICD transmits on a double buffer basis without monitoring the BUSY state.

NOTE 2: Facts about TBMT and TEOC signals:

TBMT—Transmitted Buffer Empty. The transmitted buffer empty flag goes to a logic "1" when the data bits holding register may be loaded with another character.

TEOC—Transmitted End of Character. This line goes to a logic "1" each time a full character is transmitted. It remains at this level until the start of transmission of the next character.

▲ Multiple ICDs Signals for multiple ICDs can I/O through the HOST/AUX port by setting bits 6 and 7. When this feature is enabled, the External Break, Emulation Qualify, and Event Trigger signals can be monitored by more than one ICD. (I/O level is EIA.)

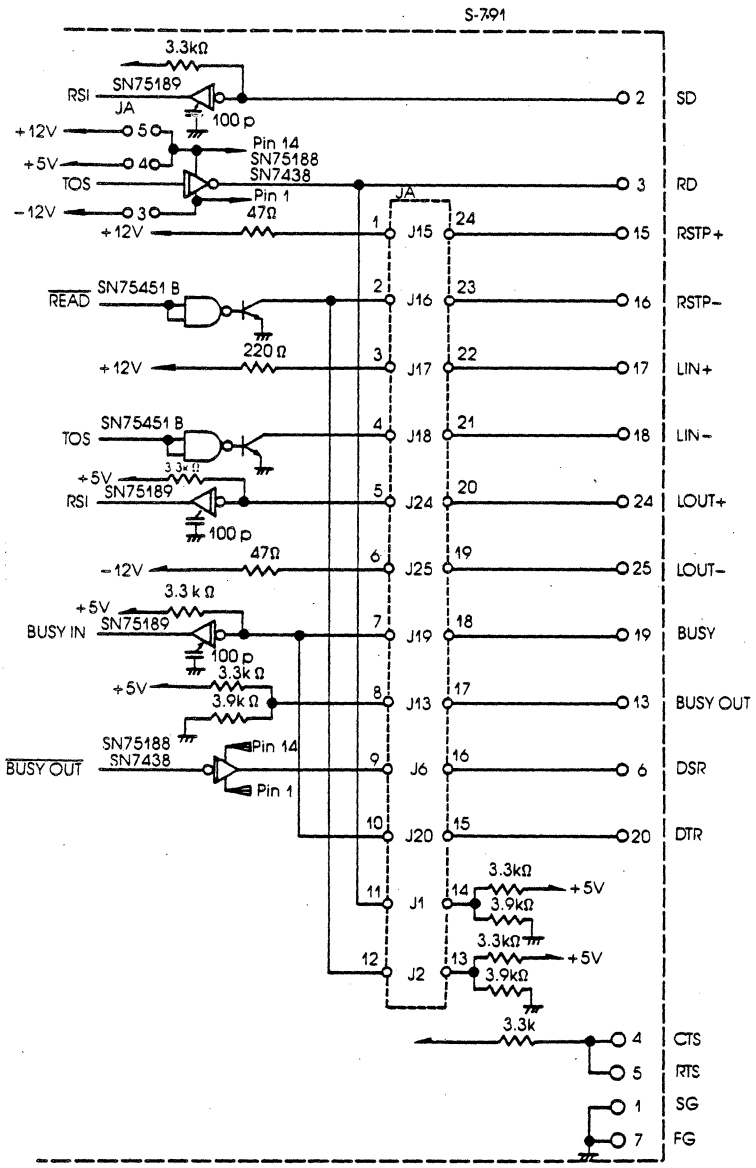
To activate this feature, set the following bits:

DSW3 bit 6 = ON DSW4 bits 6 & 7 = ON

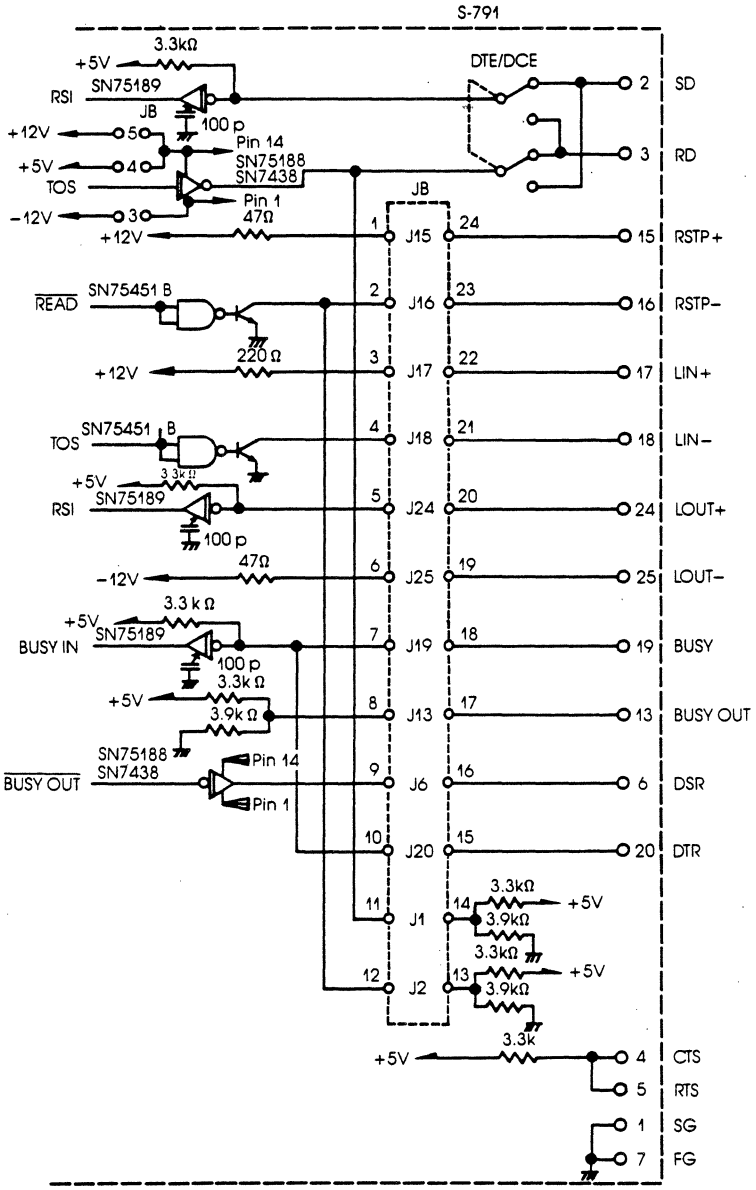
This feature effects the following pins of the HOST/AUX port:

Pin No.	Signal Name	I/O
11	External Break	IN
18	Emulation Qualify	OUT
25	Event Trigger	OUT

NOTE: The multiple ICD feature is available on ICDs which use SIO module S-771B.



SIO S-791 DIAGRAM (TERMINAL PORT)

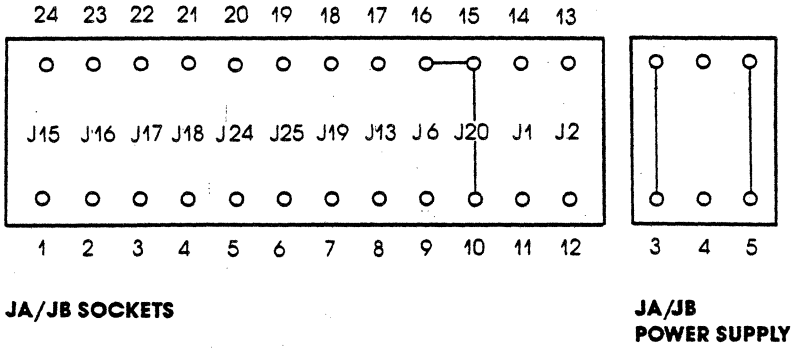


SIO S-791 DIAGRAM (HOST/AUX PORT)

RS-232 Interface

The RS-232 interface is the normal configuration for the ICD. The diagram below shows how the pins on the JA/JB sockets are arranged for the RS-232 settings. The two tables show the status of the signals for both the TERMINAL and HOST/AUX ports.

RS-232 Pin Configuration
(Standard connection is shown)



RS-232 Interface I/O Signals—TERMINAL Port

PIN No.	SIGNAL NAME	DESCRIPTION	IN/OUT	JA No.
1	FG	Frame Ground		
2	SD	Send Data	IN	SN 75188N
3	RD	Receive Data	OUT	
4	RTS	Request To Send *2	IN	
5	CTS	Clear To Send *2	OUT	
6	DSR	Data Set Ready	OUT	
20	DTR	Data Terminal Ready	IN	J 6, J 20 *3
7	SG	Signal Ground		

RS-232 Interface I/O Signals—HOST/AUX Port

PIN No.	SIGNAL NAME	DESCRIPTION	IN/OUT	JB No.
1	FG	Frame Ground		
2	SD	Send Data	OUT (IN) *1	SN 75188N
3	RD	Receive Data	IN (OUT)	
4	RTS	Request To Send *2	OUT (IN)	
5	CTS	Clear To Send *2	IN (OUT)	
6	DSR	Data Set Ready	IN (OUT)	
20	DTR	Data Terminal Ready	OUT (IN)	J 6, J 20 *3
7	SG	Signal Ground		

NOTE 1: Values in ( ) enabled when the DCE/DTE select switch is set to DCE.

NOTE 2: CTS and RTS signals are looped back (null modem) within the ICD and pulled up to +5V.

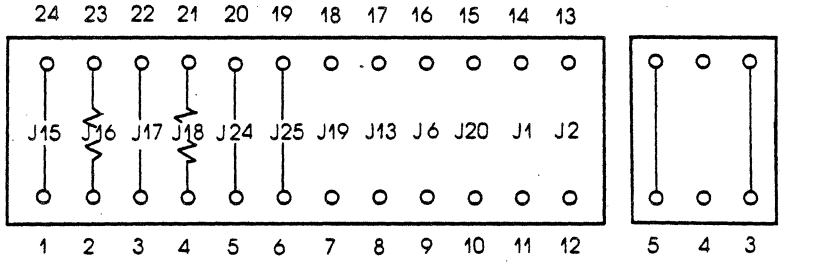
NOTE 3: Connecting pins 15 and 16 (JA/JB socket) causes the DTR and DSR signals to be looped back (null modem) within the ICD.

NOTE 4: Connecting pins 10 and 15 (JA/JB socket) causes the DTR signal to be used as the BUSY signal to the terminal. Connecting pins JA6/JB6 causes the DSR signal to be used as the BUSY signal to the terminal.

Current Loop Interface

The current loop interface is an optional configuration that is enabled when the JA and JB sockets are modified. The diagram below shows how the pins on the JA/JB sockets are arranged for the current loop setting. The table shows the status of the signals for both the TERMINAL and HOST/AUX ports.

Current Loop Interface
(Modified connection is shown)



▲ Using The Current Loop Interface

a) Connect pin 4 to pin 21 (JA18/JB18) with a 220 ohm, 1/4 watt resistor, or adjust the resistance to the associated circuit.

b) Connect pin 2 to pin 23 (JA16/JB16) with a 47 ohm, 1/4 watt resistor.

c) Connect the other pins as shown in the Current Loop Interface diagram.

d) Set the ICD's DCE/DTE select switch to DCE.

e) Adjust the baud rates for the TERMINAL and HOST/AUX ports to a maximum of 600 bps.

NOTE: Do not change the jumpers on the line driver power supply (JA3/JB3, JA5/JB5).

Current Loop Interface I/O Signals—
TERMINAL & HOST/AUX Ports

PIN NO.	SIGNAL NAME	DESCRIPTION	IN/OUT	JA/JB No.
24	LOUT+	Current Loop OUT(+)	IN	J 24
25	LOUT–	Current Loop OUT(–) *1	IN	J 25
17	LIN+	Current Loop IN(+) *2	J 17	
18	LIN–	Current Loop IN (–)	OUT	J 18 220 Ω
15	RSTP+	Reader Step (+)	OUT	J 15
16	RSTP–	Reader Step (–)	OUT	J 16 47 Ω

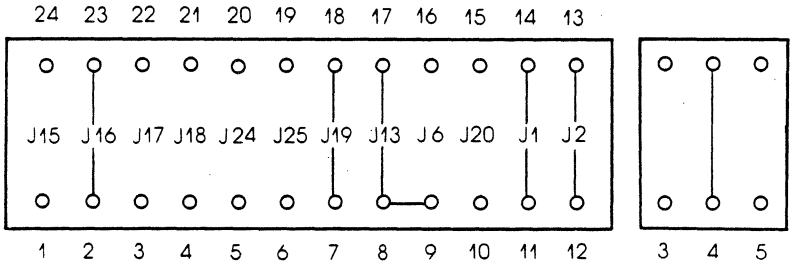
NOTE 1: Pin 25 is the current source pin for current loop input signals pulled down to –12V.

NOTE 2: Pin 17 is the current source pin for current loop input signals pulled up to +12V.

TTL Interface

The TTL interface is an optional configuration that is enabled when the JA/JB sockets are modified. The diagram below shows how the pins on the JA and JB sockets are arranged for the TTL Interface setting. The table shows the status of the signals for both the TERMINAL and HOST/AUX ports.

TTL Interface
(Modified connection is shown)



JA/JB SOCKETS

JA/JB POWER SUPPLY

▲ Using The TTL Interface

a) Remove the jumpers from JA3/JB3 and JA5/JB5 of the line driver power supply, and insert a single jumper into JA4/JB4.

b) Connect the pins as shown in the TTL Interface diagram.

TTL Interface I/O Signals—TERMINAL Port

PIN No.	SIGNAL NAME	DESCRIPTION	IN/OUT	JA No.
1	FG	Frame Ground		
2	SD	Send Data	IN	SN 7438
3	RD	Receive Data	OUT	
19	BUSY	BUSY Input	IN	J 19
13	BUSYOUT	BUSY Output	OUT	J 13, J 6 *2
16	RSTP	Reader Step	OUT	J 16
7	SG	Signal Ground		

TTL Interface I/O Signals—HOST/AUX Port

PIN No.	SIGNAL NAME	DESCRIPTION	IN/OUT	JA No.
1	FG	Frame Ground		
2	SD	Send Data	OUT (IN) *1	SN 7438
3	RD	Receive Data	IN (OUT)	
19	BUSY	BUSY Input	IN	J 19
13	BUSYOUT	BUSY Output	OUT	J 13, J 6 *2
16	RSTP	Reader Step	OUT	J 16
7	SG	Signal Ground		

NOTE 1: Values in ( ) enabled when the DCE/DTE select switch is set to DCE.

NOTE 2: Connecting pins 8 and 9 (JA/JB socket) causes the DTR signal to be used as the BUSY signal to the terminal.

Serial Interface Control Signals

XON and XOF Protocol

XON/XOFF allows terminals or host computer systems to receive data from the ICD even if the baud rates between these devices are different.

The XON/XOFF protocol works in the following manner:

1. The host computer or terminal sends XOFF to the ICD before the reception buffer overruns.

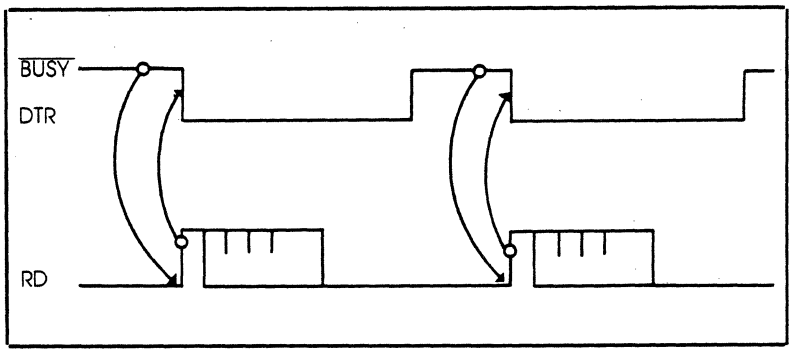
2. When the reception buffer is ready, the host computer or terminal sends XON to the ICD and resumes reception.

The control codes for XON/XOFF signals are:

XON —DC3 (CTRL-S: 13H)
XOFF—DC1 (CTRL-Q: 11H)

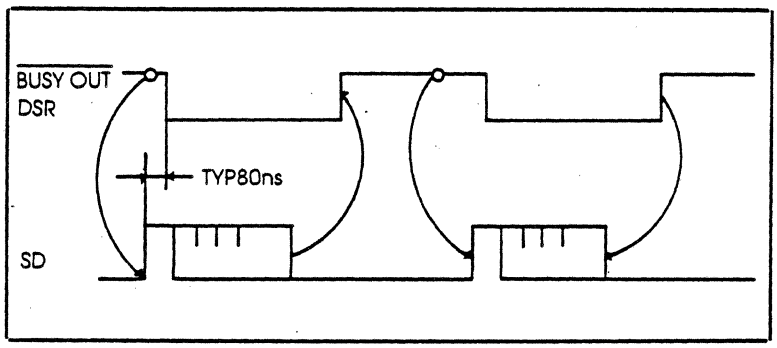
BUSY and DTR Input Signals

The BUSY signal sent from a low-speed terminal can be used to stop the ICD from transmitting data. Under normal conditions, the terminal sets the BUSY signal to low, from the leading edge of the RD-signal starting bit, to the completion of data processing. The ICD suspends data transmission to the terminal as long as the BUSY signal is low.



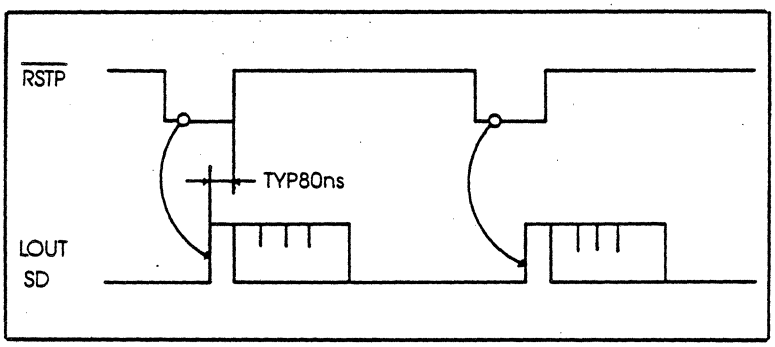
BUSYOUT and DSR Output Signals

When a host computer sends data at a higher speed than the ICD's internal monitor processor can accept, the BUSYOUT signal from the ICD must be monitored. The ICD sets the BUSYOUT signal to low until the ICD monitor reads the SD signal from the host computer.



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 it requests data to be read, and then returns RSTP to high when it detects the start bit signal from the terminal.



Expansion Memory Module Description

The optional Expansion Memory module (EXM-12 S-766) expands the ICD's memory capabilities to 256K bytes (128K standard + 128K expansion). Components on this module include a 60-pin bus receptacle to connect with the Memory Mapping Unit (MMU) module, a 5-pin power-supply connector, and two 8-bit switches that control the module's functions.

(half-tone of module)

▲ Installing The Module

If your ICD does not contain the optional Expansion Memory module and you wish to install it, follow the procedure below:

a) Remove the ICD's top, bottom, and side covers as described in "How To Disassemble Your ICD," in this section. It is not necessary to remove the other modules from the mainframe to install the Expansion Memory module.

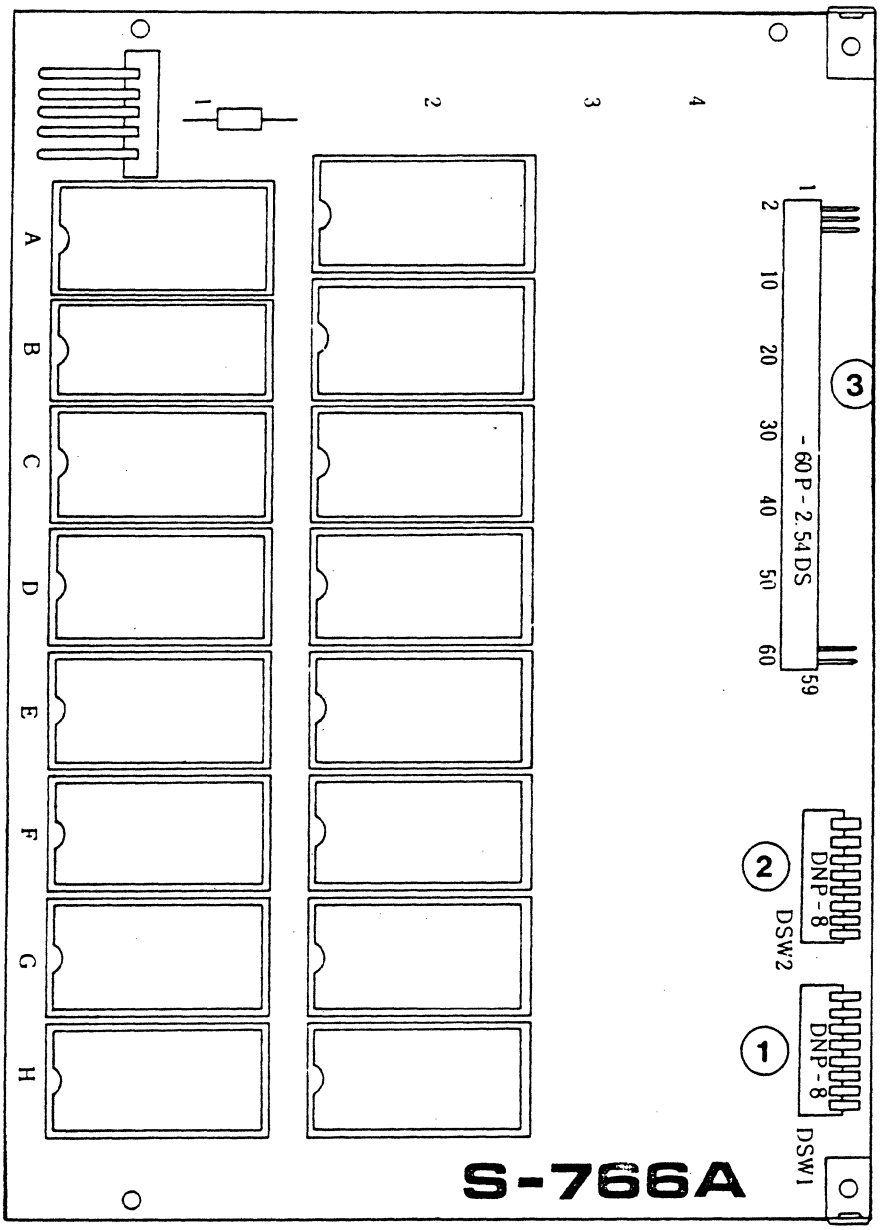
b) Slide the Expansion Memory module into the open slot that is located just below the SIO module. Position the module so that it fits between the aligning tabs, then fit the two small screws which attach the module to the mainframe.

c) Connect the power-supply socket to the module. (Use a pair of needle-nose pliers to push the socket onto the module's 5-pin plug.)

d) Connect the auxiliary bus cable to the module's 60-pin bus receptacle.

e) Attach the other end of the auxiliary bus cable to the 60-pin bus receptacle on the MMU module, located on the bottom of the ICD mainframe.

f) Replace the top, bottom, and side covers on the ICD mainframe.


1
2
3
4
A
B
C
D
E
F
G
H
1
2
10
20
30
40
50
60
59
- 60 P - 2.54 DS
3
2
DNP - 8
DSW2
1
DNP - 8
DSW1
S-766A

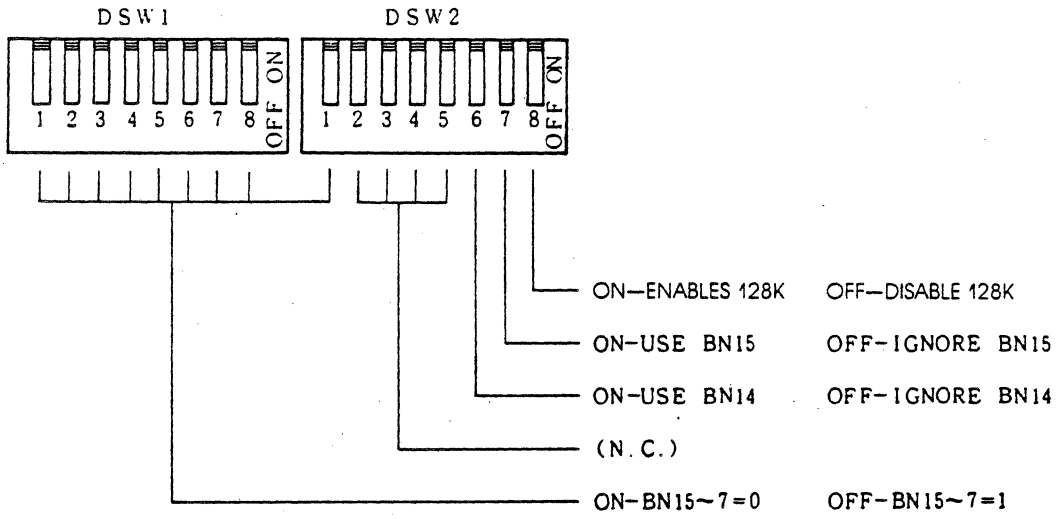
EXM-12 S-766 Module Components

1) & 2) DSW1 & DSW2 ICD Progrma Memory Block Number switches. Sets the allocation block number of the 128K-byte Expansion Memory module in 1K-byte blocks. (For more information on memory allocation, see the ALLOCATION command in Section 2.)

3) EXM-12 60-pin Bus Connector. Connects the EXM-12 module with the MMU S-776 Memory Mapping Unit module.

▲ DSW1 & DSW2 Switch Settings

The following diagram shows the factory-adjusted switch settings for the DSW1 and DSW2 ICD Program Memory Block Number switches.



Notes on bit functions:

Bit 8 of DSW2—Enables and disables the Expansion Memory module.

Bit 7 and bit 1 of DSW2—When both bits are OFF, the A25M address signal is suppressed. Signals A17M through A24M remain active.

Bit 6 of DSW2 and bit 7 of DSW1—When both bits are OFF, the A24M address signal is suppressed. Signals A17M through A23M and A25M remain active.

If bits 1 through 7 of DSW1 and bit 1 of DSW2 are ON, block numbers 000-07F are selected. If bit 1 of DSW1 is OFF and all other bits are ON, block numbers 080-OFF are selected.

The following diagram shows the allocation block numbers and corresponding bit settings:

Bit 1 (DSW2) = BN 15 (A25M)
Bit 8 (DSW1) = BN 14 (A24M)
Bit 7 (DSW1) = BN 13 (A23M)
Bit 6 (DSW1) = BN 12 (A22M)
Bit 5 (DSW1) = BN 11 (A21M)
Bit 4 (DSW1) = BN 10 (A20M)
Bit 3 (DSW1) = BN 9 (A19M)
Bit 2 (DSW1) = BN 8 (A18M)
Bit 1 (DSW1) = BN 7 (A17M)

Break Comparator Memory Module Description

The Break Comparator Memory module (BRX S-778) qualifies the conditions (address, data, status) for the BREAK command.

The BREAK command is used to control the functions of the Break Comparator Memory module; there are no user-serviceable controls or components.

(half-tone of break comparator memory module)

CPU Control Module Description

The CPU Control module (CPU S-773) contains the connectors, circuitry, and 8086 or 8088 microprocessors and 8087 co-processor, which allow the ICD to emulate the target system's microprocessor and co-processor.

The user-serviceable components on this module include: the CPU socket, which contains the 8086 or 8088 processor, and the NDP socket, which houses the 8087 co-processor; H, CX, and L jumpers, which allow you to set the ICD's internal clock speed to either 5 MHz or 10 MHz; and jumper pins that can be used to insert wait states into the machine cycle operation.

To gain access to these components and/or to change the processors, see "How To Disassemble Your ICD," located at the end of this section.

The remaining components are all externally accessible. These include the CPU probe connectors (which connect the ICD's internal processor to the target system), the NDP probe connector, and Emulation Method Select switch #1.

(halftone photo of module)

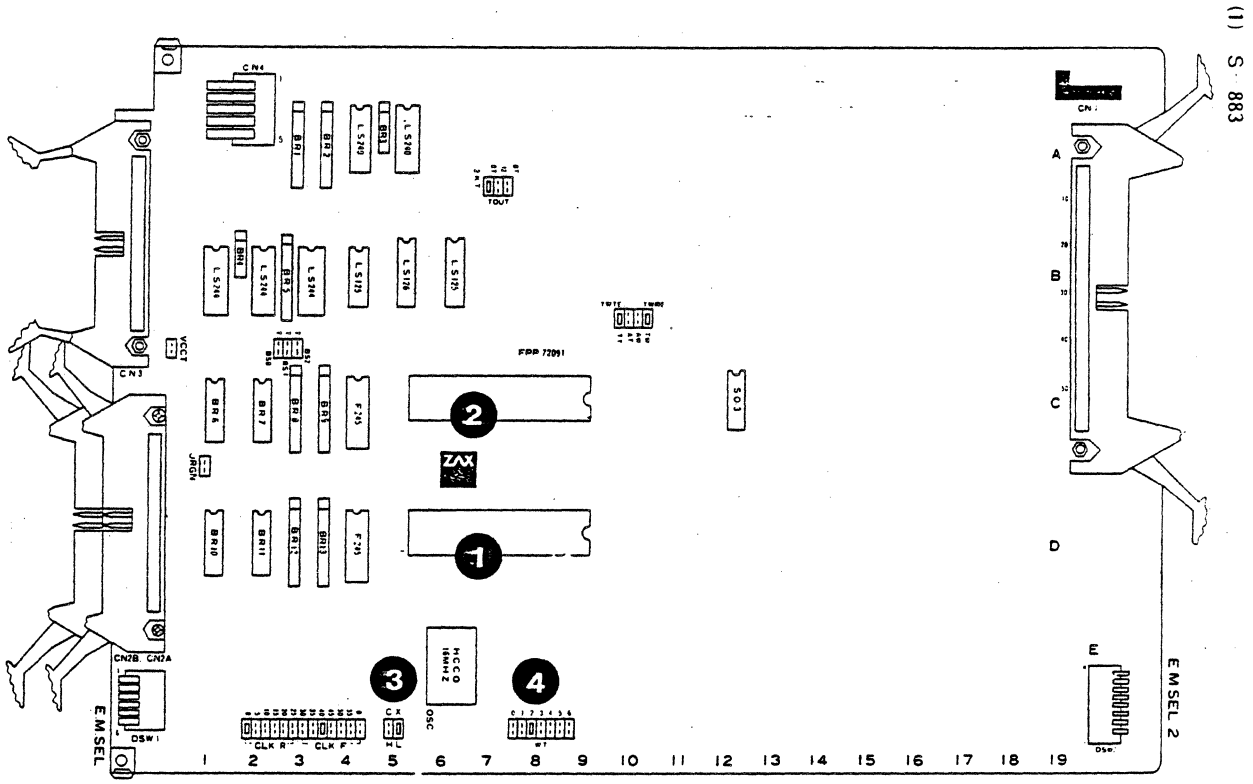
CPU Control Module Components

1) CPU Socket. Accepts either the 8086 or 8088 CPU.

2) NDP Socket. Accepts the 8087 Numeric Data Processor.

3) Internal Clock Jumper Pins. Changes the ICD's internal clock to either 5 or 10 MHz.

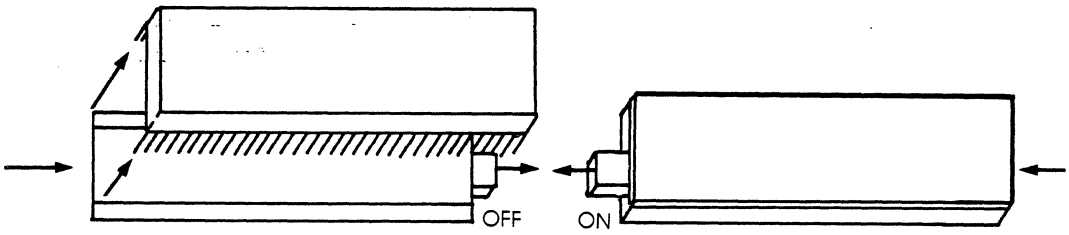
4) Wait State Jumper Pins. Inserts 1, 2, 3, 4, 5, or 6 wait states into each machine cycle.



▲ Changing CPUs

To change CPUs (8086 or 8088), disassemble the ICD as shown at the end of this section, and then remove the CPU Control module from the ICD. To remove the existing CPU, slide the tab marked "ON" into the socket until the tab marked "OFF" appears on the other end of the socket. Carefully remove the CPU from the socket, insert the new CPU into the socket, and then push the tab marked "OFF" into the socket until it locks into the original "ON" position.

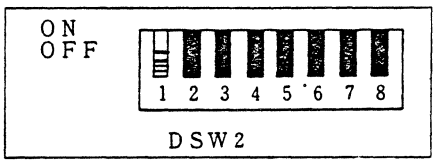
NOTE: Removing and installing the 8087 Numeric Data Processor is identical to the above procedure.



Chip is smaller and fits inside

Emulation MEthod Select Switch #2

The ICD contains two Emulation Method Select switches: switch #1 and switch #2 (the functions of switch #1 have already been discussed in Section 1). Switch #2 is accessible by removving the side panel on the TERMINAL and HOST/AUX port end of the ICD. (See "How To Disassemble Your ICD," located at the end of this section.)



▲ Switch Description And Functions

Emulation Method Select switch #2 is an 8-bit ON/OFF toggle-type switch located on the CPU Control module within the ICD. Bits 1, 2, 3, and 4 are functional; bits 5 through 8 are not connected.

Bit 1 is used to modify the clock phase between the ICD and the target system. With systems that operate at clock speeds over 8 MHz, emulation problems occur from an out-of-spec clock duty cycle and emulation propagation delays. By advancing the clock signal within the ICD to compensate for propagation delays, the ICD can effectively operate with target systems running at 8 MHz and over.

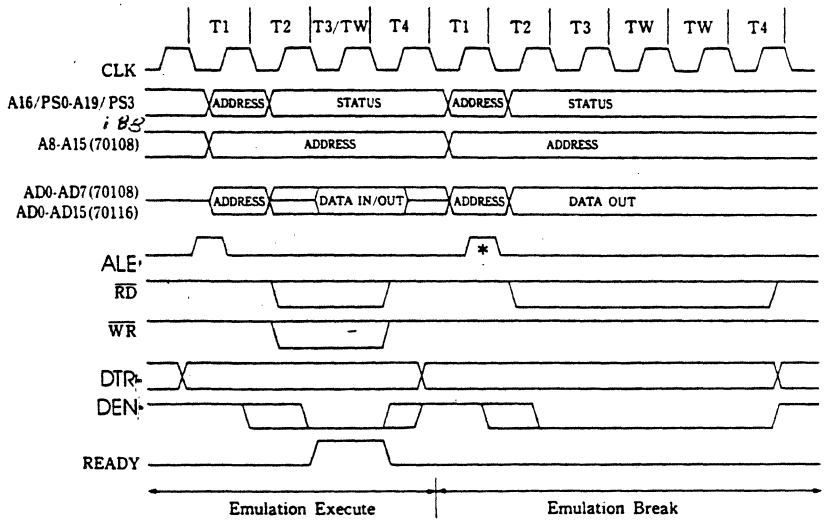
Function: OFF—modifies the clock phase for use with target systems that operate between 2 and 7 MHz.

ON—modifies the clock phase for use with target systems that operate over 6 MHz.

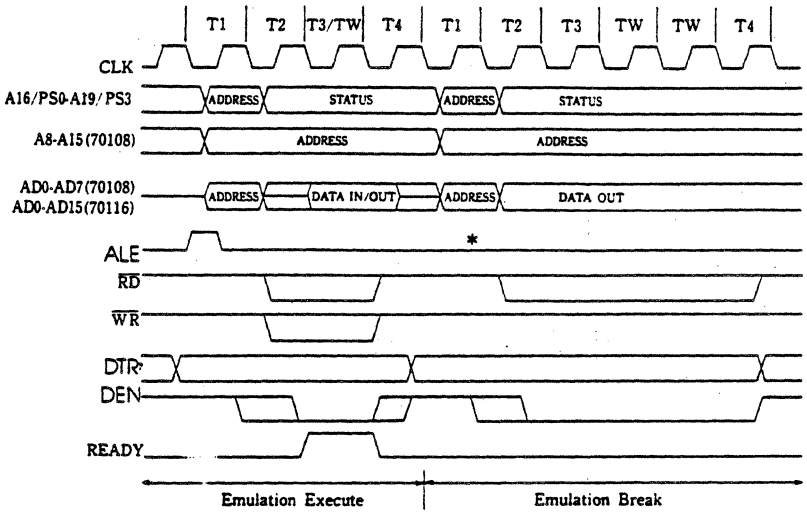
NOTE: On target systems that operate at clock speeds between 6 and 7 MHz, ON or OFF may be selected; choose the setting which works best with your system.

Bit 2 is used to control the output of the Address Latch Enable (ALE) signal, during a break status or when the ICD is operating in the emulation mode. The ALE signal is provided by the processor to latch the address into the 8282/8283 address latch. (Control of the ALE signal is effective in the minimum mode only, since the ALE signal is generated by an 8288 Bus Controller in the maximum mode.)

Function: OFF—ICD outputs the ALE signal every machine cycle—during emulation or during a break status.



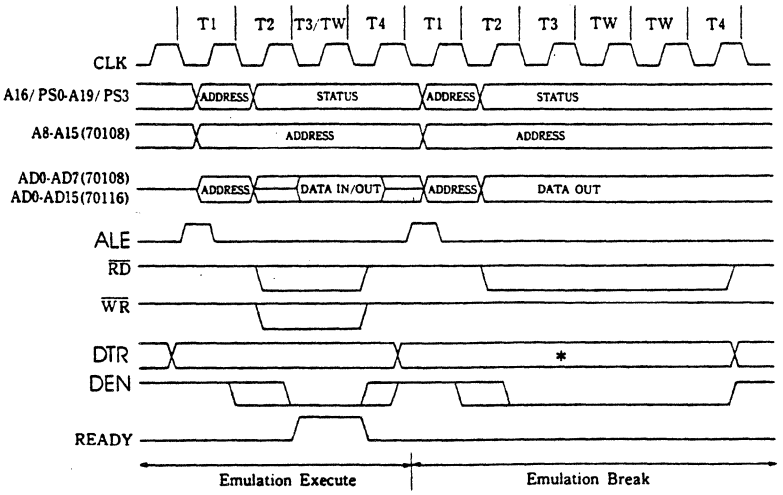
ON—ICD outputs the ALE signal during emulation, but suppresses it during a break status. (The signal is fixed to a low level.)



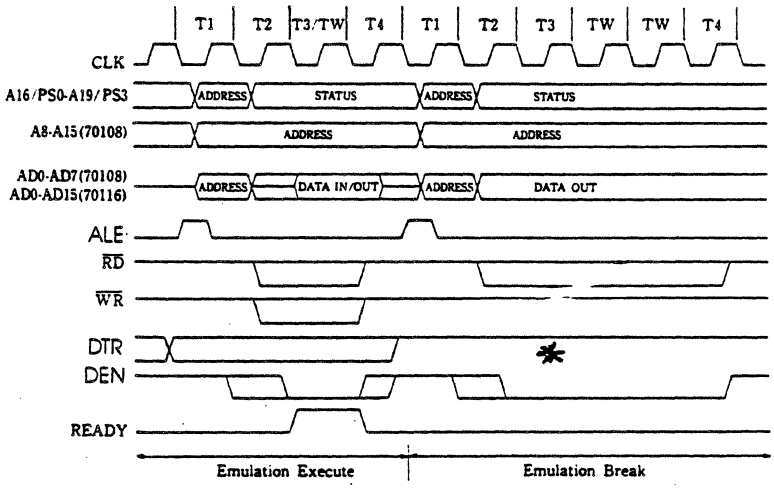
NOTE: The ALE signal can be generated during a break status, while in the maximum mode, by connecting S0/S1/S2 jumper. (See "CPU Control Module Jumpers.")

Bits 3 and 4 function together to control the output and level of the Data Transmit/Receive (DT/R) signal when the ICD is operating in the minimum mode. The DT/R signal is used with a minimum system that requires the use of an 8286/8287 data bus transceiver. It is used to control the direction of data flow through the transceiver.

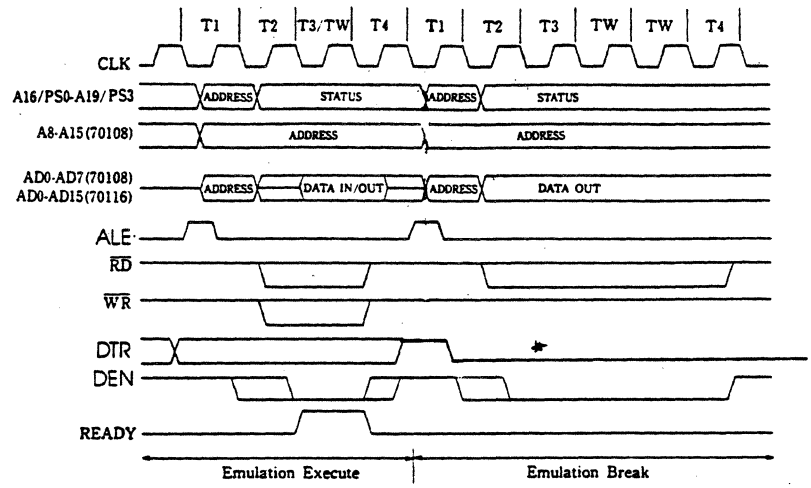
Function: Bit 3 OFF & Bit 4 "don't care"—The ICD generates the DT/R signal every machine cycle during emulation or when the ICD is in a break status mode (minimum mode only).



Bit 3ON & Bit 4 OFF—The DT/R signal is fixed to a high level when the ICD is in a break state.



Bit 3 ON & Bit 4 ON—Th DT/R signal is fixed to a low level when the ICD is in a break state.



▲ CPU Control Module Jumpers

The CPU Control module contains a number of plastic-encased, gold-plated connectors called "jumpers." The jumpers provide a convenient way of connecting (as opposed to soldering) the various pins on the module, which in turn control clock speeds, wait states, time outs, and control signal I/O.

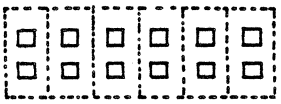
Internal Clock Jumpers. Sets the ICD's internal clock speed to either 5 or 10 MHz. (The ICD internal clock is selected by setting the INT/EXT switch to INT.)

The internal clock normally runs at a speed of 5 MHz, with a 33% duty cycle, but it can be changed to 10 MHz by modifying the jumpers on the CPU Control module. The clock jumper is identified by CX, and the H and L jumpers specify the high (H = 10 MHz) or low (L = 5 MHz) clock speed.

Settings: CX-L—Sets ICD's internal clock speed to 5 MHz.
CX-H—Sets ICD's internal clock speed to 10 MHz.

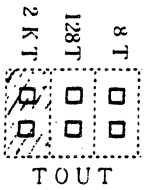
Factory Setting: CX-L

External Clock. Selecting the EXT setting on the INT/EXT switch enables the ICD to use an external clock. The external clock setting allows the peripheral LSI of the target system and the emulation CPU to be synchronized for simultaneous operation. *NOTE: To ensure accurate operation of the emulation CPU, a 33% duty cycle is required for high speed clocks greater than 5 MHz.*



Wait State Jumpers. Inserts 0 to 6 wait states in each memory or I/O cycle. To activate the wait state feature, use Emulation Method Select switch #1 and set bit 6 to the ON position.

Settings: WT-0—Sets the wait state to 0.
WT-1—Inserts 1 wait states.
WT-2—Inserts 2 wait states.
WT-3—Inserts 3 wait states.
WT-4—Inserts 4 wait states.
WT-5—Inserts 5 wait states.
WT-6—Inserts 6 wait states.



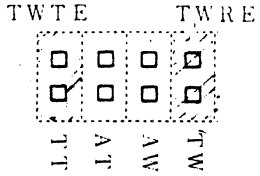
Timeout Jumpers. These jumpers set the timeout (delay from a READY signal response) to 2048, 128, or 8 clock cycles: This delay constitutes a wait state. Wait states can be used to cause a break (using the "BREAK: Timeout" command) in the user program when the ICD is unable to access the target system contents within a certain time period.

Settings: TOUT-8T—Sets the timeout to 8 clock cycles.

TOUT-128T—Sets the timeout to 128 clock cycles.

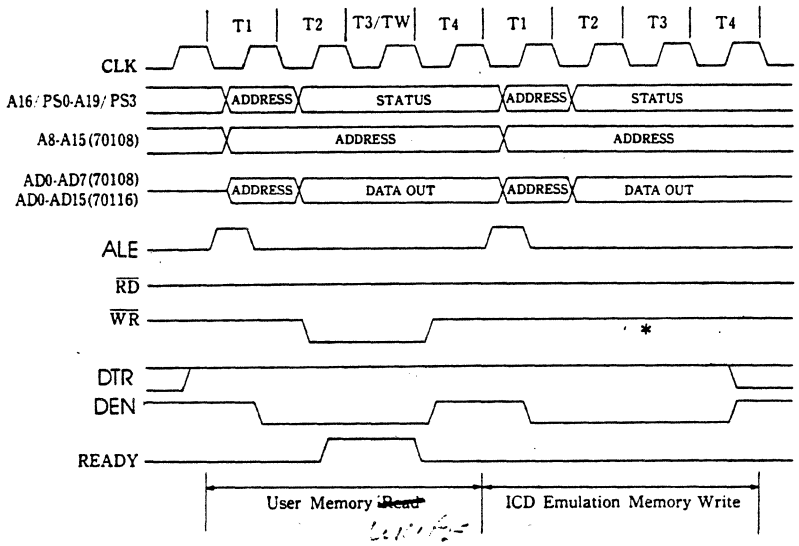
TOUT-2KT—Sets the timeout to 204B clock cycles.

Factory Setting: TOUT-2KT

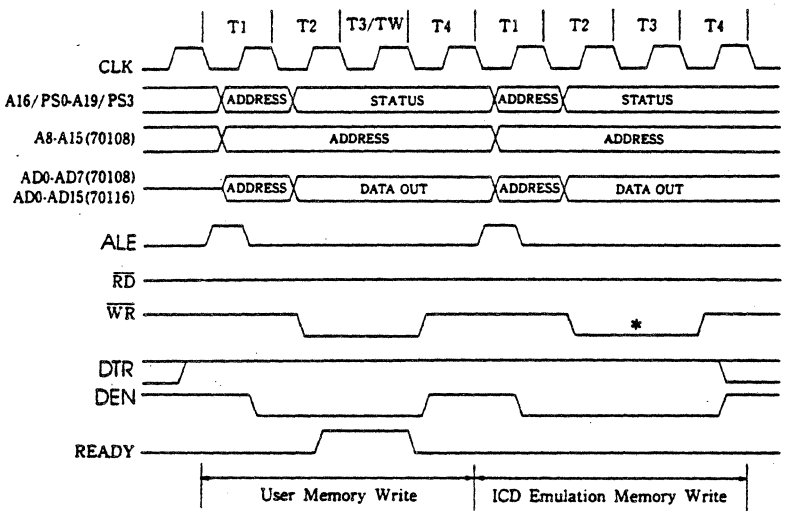


Write/Ready Jumpers. These jumpers control the READY and WRITE signals from the ICD during emulation, memory writing, and break conditions. The signals are active in the minimum mode only.

Settings: TWRE-TW—Generates the WRITE signal to the target system during emulation, and suppresses the WRITE signal to the ICD during an emulation memory write operation. (WRITE is suppressed during a break.)



TWRE-AW—Generates the WRITE signal to the target system during emuation, and to the ICD during an emulation memory write operation. (WRITE is suppressed during a break.)

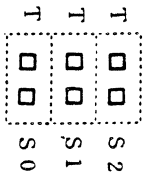


TWTE-TT—ICD accepts READY signal when reading or writing to the target system during emulation, and ignores READY signal when reading or writing to the ICD emulation memory. (ICD ignores READY during a break status.)

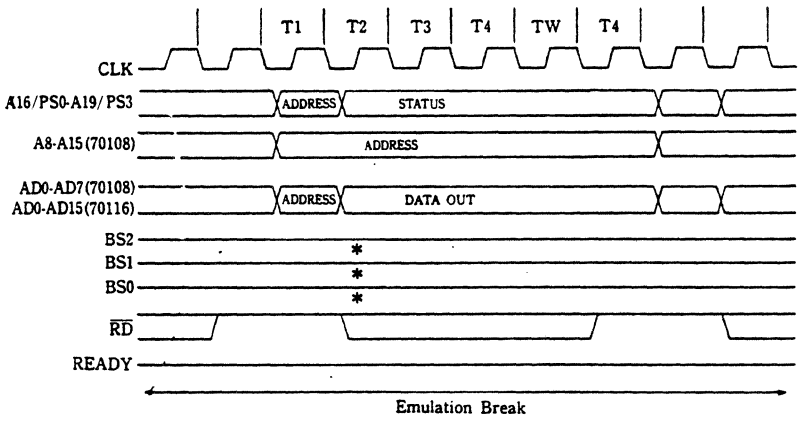
TWTE-AT—ICD accepts READY signal when reading or writing to the target system and to the ICD emulation memory. (ICD ignores READY during a break status.)

Factory Settings: TWRE-TW and TWTE-AT

NOTE: Set the TWRE-TW/AW jumper to either TW or AW; an open connection is not recommended.

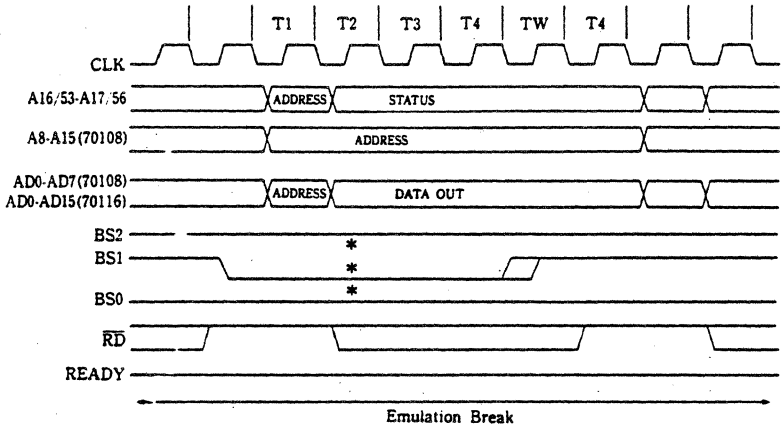


Status Jumpers: These jumpers control the output of the 8086 processor's S0, S1, and S2 status lines during an emulation break, while in the maximum mode. Normally, the S0, S1, and S2 signals are suppressed during an emulation break.



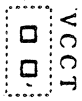
Settings: SOT-S0—Activates the S0 signal during an emulation break.

SOT-S1—Activates the S1 signal during an emulation break. When S1 is selected, the ICD outputs the RD and ALE signal to the target system. This can be useful for systems which require memory refresh.



SOT-S2—Activates the S2 signal during an emulation break.

Factory Setting: all open (not connected)



VCCT Jumper. Provides a +5 volt reference line to the target system.

Settings: VCCT (connected)—outputs +5 volts to the target system.

VCCT (open)—reference line is not connected to the target system.

Factory Settings: open (not connected)

JRGN Jumper. Connects the RQ/GTO signal from the ICD's 8087 co-processor to the 8087 target socket via the NDP in-circuit probe.

Settings: JRGN (connected)—enables the above function.

JRGN (open)—disables the above function.

Factory Setting: open (not connected)

CLK.R/CLK.F Jumpers. Permits synchronization of the clock signals between the ICD and target system, by adjusting the rising edge and failing edge of the CPU clock."CLK.R" controls the rising edge of the signal, and "CLK.F" controls the falling edge. The rising/falling times can be specified in 5 nanosecond increments. The range is 0—25 nanoseconds for the rising time and 30—55 for the falling edge.

Settings:	Rising Edge	Falling Edge
	CLK.R— 0 ns	CLK.F—30 ns
	CLK.R— 5 ns	CLK.F—35 ns
	CLK.R—10 ns	CLK.F—40 ns
	CLK.R—15 ns	CLK.F—45 ns
	CLK.R—20 ns	CLK.F—50 ns
	CLK.R—25 ns	CLK.F—55 ns
		CLK.F— 0 ns

Factory Settings: CLK.R—0 (0 nanoseconds)
CLK.F—40 (40 nanoseconds)

ICD/Target System Interface

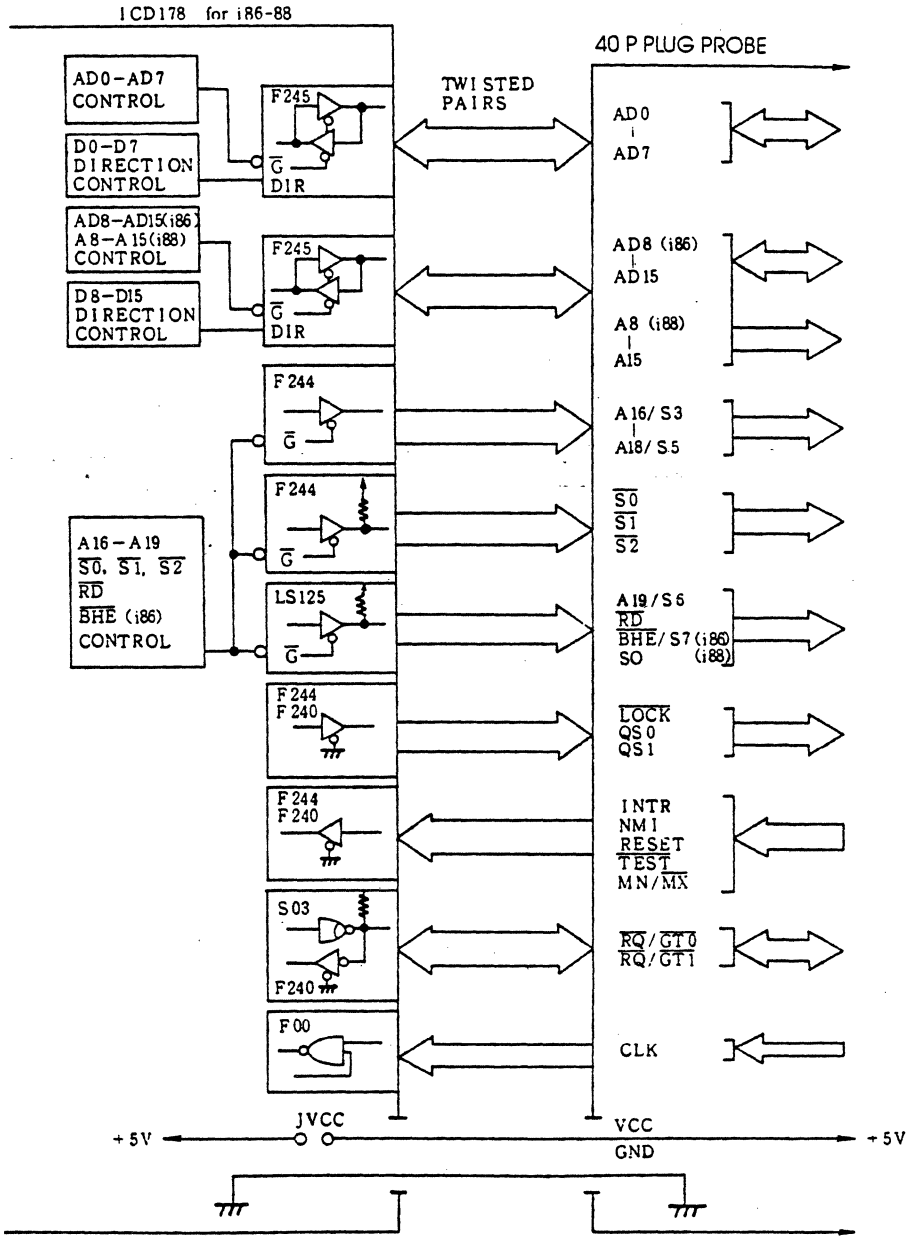
Minimum And Maximum Modes

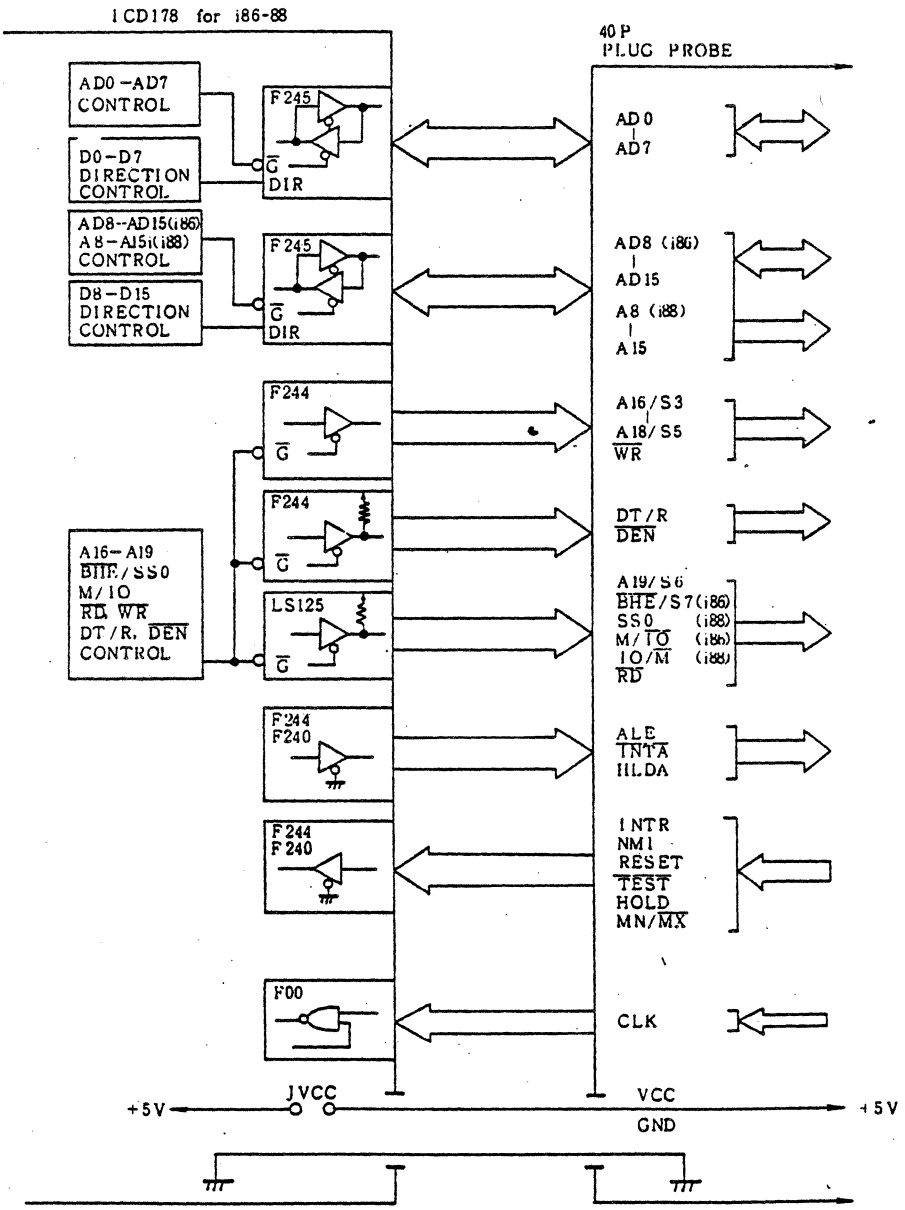
A unique feature of the 8086/8088 CPU allows a user to define a subset of the CPU's control signal outputs to adapt the system to a particular environment. Two different modes are available; minimum and maximum (min/max).

In the minimum mode, the CPU supports small, single-processor systems that consist of a minimum of devices, and typically use a local bus rather than support the multibus architecture. In this mode, the CPU itself generates all bus control signals and the command output signal. It also provides a mechanism for requesting bus access that is compatible with bus master type controllers (systems with a bus structure in which control of data transfers on the bus is shared between the CPU and associated peripheral devices).

In the maximum mode (used with multi-board systems, a bus controller (Intel 8288) acts as a sophisticated bus control function to provide compatibility with multi-bus architecture. In this mode, the bus controller, rather than the CPU, provides all bus control and command outputs, and allows pin previously delegated to these functions to be redefined to support multiprocessing functions.

The following diagrams show the relationship between the ICD and the target system interface when operating in the minimum and maximum modes.


ICD178 for i86-88
40 P PLUG PROBE
TWISTED PAIRS
AD0-AD7 CONTROL
D0-D7 DIRECTION CONTROL
F245
G
DIR
AD8-AD15(i86) A8-A15(i88) CONTROL
D8-D15 DIRECTION CONTROL
A16-A19
S0, S1, S2
RD
BHE (i86)
CONTROL
F244
F240
LS125
S03
F00
AD0
AD7
AD8 (i86)
AD15
A8 (i88)
A15
A16/S3
A18/S5
S0
S1
S2
A19/S6
RD
BHE/S7 (i86)
SO (i88)
LOCK
QS0
QS1
INTR
NMI
RESET
TEST
MN/MX
RQ/GT0
RQ/GT1
CLK
JVCC
+5V
VCC
GND


ICD178 for i86-88
40P
PLUG PROBE
AD0-AD7
CONTROL
D0-D7
DIRECTION
CONTROL
F245
G
DIR
AD8--AD15(i86)
A8-A15(i88)
CONTROL
D8-D15
DIRECTION
CONTROL
F245
G
DIR
F244
G
F244
G
A16-A19
BHE/SS0
M/IO
RD, WR
DT/R, DEN
CONTROL
LS125
G
F244
F240
F244
F240
F00
AD0
AD7
AD8 (i86)
AD15
A8 (i88)
A15
A16/S3
A18/S5
WR
DT/R
DEN
A19/S6
BHE/S7(i86)
SS0 (i88)
M/IO (i86)
IO/M (i88)
RD
ALE
INTA
HLDA
INTR
NMI
RESET
TEST
HOLD
MN/MX
CLK
+5V
JVCC
VCC
GND
+5V

Machine Cycle Operation—Min Mode

MACHINE CYCLE	STATUS CONTROL								A16-A19, BHE (i86) A8-A19, (i88)	ADO-AD15 (i86) ADO-AD7 (i88)	
	M/IO (i86)	IO/M (i88)	SSO (i88)	DT/R	DEN	WR	RD	INTA		Address	Data
Interrupt Acknowledge	0	1	0	0	0	1	1	0	OUT	IN	IN
Read I/O	0	1	1	0	0	1	0	1	OUT	OUT	IN
Write I/O	0	1	0	1	0	0	1	1	OUT	OUT	OUT
Halt	0	1	1	1	0	1	1	1	OUT	OUT	OUT
Instruction Fetch	1	0	0	0	0	1	0	1	OUT	OUT	IN
Read Data from Memory	1	0	1	0	0	1	0	1	OUT	OUT	IN
Write Data to Memory	1	0	0	1	0	0	1	1	OUT	OUT	OUT
Passive	1	0	1	1	0	1	1	1	TS	TS	TS
HOLD State	TS	TS	TS	TS	TS	TS	TS	1	TS	TS	TS
RESET State	TS	TS	TS	TS	TS	TS	TS	1	TS	TS	TS
ICD Program Memory Fetch *1	1	0	0	0	0*3	1	0*2	1	OUT	OUT	TS
ICD Program Memory Read *1	1	0	1	0	0*3	1	0*2	1	OUT	OUT	TS
ICD Program Memory Write *1	1	0	0	1	0*3	1	1	1	OUT	OUT	OUT
IO	X	X	TS	X	0	1	X	1	TS	TS	TS

Signal level: 0=L, 1=H, TS=3-state

*1 In this cycle, the target system is not accessed though the ICD program memory is mapped out, or an emulation break occurs.

*2 Setting Bit 1 of the EM.SEL switch to the ON position suppresses the RD signal.

Machine Cycle Operation—Max Mode

MACHINE CYCLE	STATUS CONTROL A8-A19 (I88) S2	S1	S0	RD	A16-A19, BHE (i86) ADO-AD7 (I88)	ADO-AD15 (i86) Address	Data
Interrupt Acknowledge	0	0	0	1	OUT	IN	IN
Read I/O	0	0	1	0	OUT	OUT	IN
Write I/O	0	1	0	1	OUT	OUT	OUT
Halt	0	1	1	1	OUT	OUT	OUT
Instruction Fetch	1	0	0	0	OUT	OUT	IN
Read Data from Memory	1	0	1	0	OUT	OUT	IN
Write Data to Memory	1	1	0	1	OUT	OUT	OUT
Passive (no bus cycle)	1	1	1	1	TS	TS	TS
Bus GRANT State	TS	TS	TS	TS	TS	TS	TS
RESET State	1	1	1	1	TS	TS	TS
ICD Program Memory Fetch *1	1	0	0	0*2	OUT	OUT	TS
ICD Program Memory Read *1	1	0	1	0*2	OUT	OUT	TS
ICD Program Memory Write *1	1	1	0	1	OUT	OUT	OUT
Emulation Break	1	1	1	1	TS	TS	TS
IO	TS	TS	TS	X	TS	TS	TS

Signal level: 0=L, 1=H, TS=3-state
*1 In this cycle, the target system is not accessed though the ICD program memory is mapped out.
*2 Setting Bit 1 of the EM.SEL switch to the ON position suppresses the RD signal.

ICD Program Memory Cycles

The ICD generates a cycle which does not access memory in the target system when the ICD program memory is selected (by memory mapping), or emulation is interrupted by a break status. This is the machine cycle of the ICD program memory fetch, read and write signals. The maximum and minimum modes affect the machine cycle operation differently, as described in the following:

ICD Program Memory Cycles—Maximum mode

In the maximum mode, the machine cycle is different from that during an emulation break. During the ICD program memory fetch and read cycles, each of the S0, S1, and S2 signals indicate the memory fetch/read cycle, but the ICD does not receive any data by setting the data bus to a 3-state.

During the write cycle, the S0, S1, and S2 signals directly indicate the memory write cycle, and the write data outputs on the data bus. This allows the ICD's program memory write cycle to perform the same operations as the target memory write cycle.

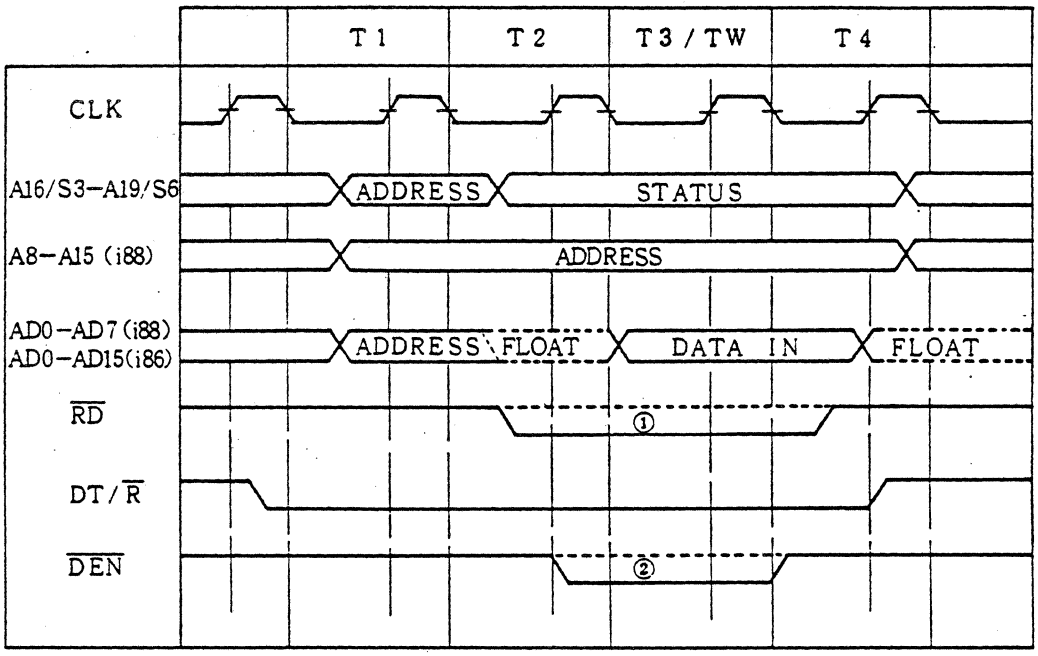
When emulation is temporarily halted with a break status, the S0, S1, and S2 signals are prohibited, and the target system is not accessed, but the address and data buses are set to 3-state. These conditions constitute a passive state.

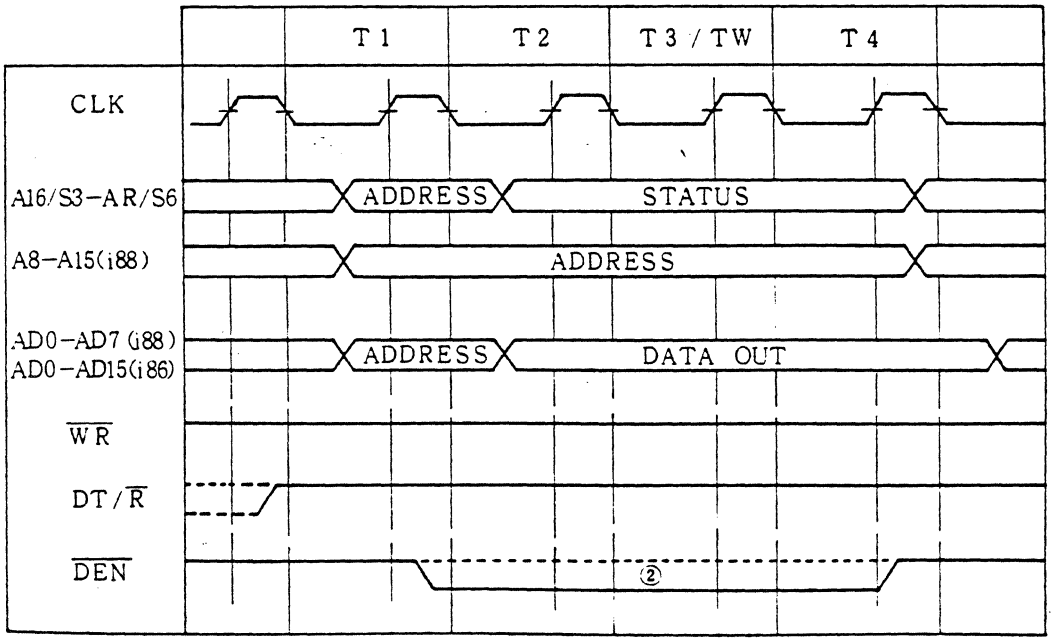
ICD Program Memory Cycles—Minimum mode

In the minimum mode, the machine cycle operation is identical to that during an emulation break. The ICD program memory write cycle prohibits the output of the WR signal and prevents the target system's memory from being accessed, although the ICD outputs the DEN signal.

The ICD program memory fetch and read cycles set the data bus to 3-state, but suppresses the output of the RD and DEN signals.

If a particular target system utilizes a memory-mapped I/O method in which the I/O operation is initiated by the RD signal, the ICD's program memory cycle may be manipulated by using the Emulation Method Select switch #1. See "More About Your ICD," in Section 1.




T1
T2
T3 / TW
T4
CLK
A16/S3-AR/S6
ADDRESS
STATUS
A8-A15(i88)
ADDRESS
AD0-AD7 (i88)
AD0-AD15(i86)
ADDRESS
DATA OUT
WR
DT/R
DEN
②

ICD Signals Examined

SIGNAL NAME	PIN No.	PIN No.	SIGNAL NAME
GND	2	1	GND
A1M	4	3	GND
A3M	6	5	A2M
A5M	8	7	A4M
A7M	10	9	A6M
A9M	12	11	A8M
A11M	14	13	A10M
A13M	16	15	A12M
A15M	18	17	A14M
A17M	20	19	A16M
A19M	22	21	A18M
A21M (GND)	24	23	A20M (GND)
A23M (GND)	26	25	A22M (GND)
A25M (GND)	28	27	A24M (GND)
GND	30	29	GND
D1M	32	31	D0M
D3M	34	33	D2M
D5M	36	35	D4M
D7M	38	37	D6M
D9M	40	39	D8M
D11M	42	41	D10M
D13M	44	43	D12M
D15M	46	45	D14M
GND	48	47	GND
GND	50	49	WRLM
GND	52	51	WRHM
GND	54	53	RDM
GND	56	55	GND
+5V	58	57	+5V
GND	60	59	GND

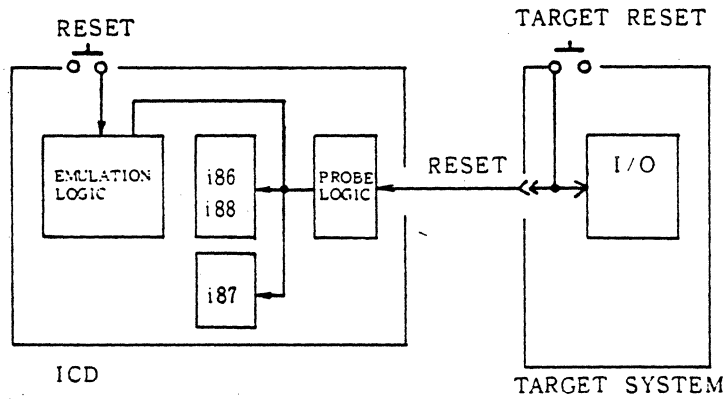
Emulator Bus Connector—Pin Assignment

RESET Signal

The RESET signal is used to reset the ICD monitor. The signal is sent by pushing the Reset switch on the Indicator/Control panel. This action resets the ICD monitor, but does not reset the target system. Typically, the target system will have a manual reset switch that resets the entire system.

Resetting the target system also causes a hardware reset of the ICD's CPU registers. However, if an emulation break is in progress, resetting the target system will not have any effect on the ICD's CPU registers. The CPU registers must be reset by entering the REGISTER RESET COMMAND.

NOTE: Resetting the ICD's CPU resets the NDP as well.



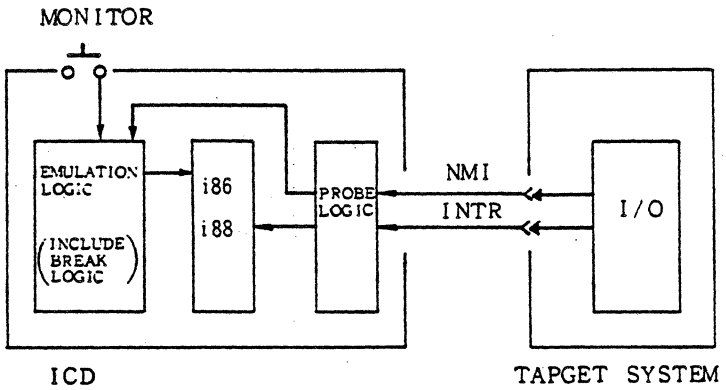
	10		11		12	
	MONITOR	EMULATION	MONITOR	EMULATION	MONITOR	EMULATION
ICD RESET SW	○	×	○	×	○	×
RESET	×	×	△ *1	○	△ *1	○

○: Effective △: Conditionally effective ×: Not effective
*1 Does not work as the hardware reset of the emulation CPU.

INTERRUPT Signal

The INTERRUPT (INT) signal returns control to the ICD monitor during emulation, and is activated by pressing the Monitor switch on the ICD's Indicator/Control panel. A NON-MASKABLE INTERRUPT (NMI) signal is also sent to the ICD's CPU when the Monitor switch is used. This NMI signal is assigned a higher priority than the target system's NMI.

The NMI signal is masked when the ICD is in an emulation break. However, the NMI signal from the target system is latched by an edge-trigger circuit, so that when an NMI occurs during a break, an interrupt sequence is generatedd at the transition from the ICD monitor run to the target system run. The INT signal is also masked during an emulation break.



	10		11		12	
	MONITOR	EMULATION	MONITOR	EMULATION	MONITOR	EMULATION
ICD MONITOR SW	×	○	×	○	×	○
NMI	×	×	△*1, 2	○ *1	△ *2	○
INTR	×	×	× *1	○ *1	×	○

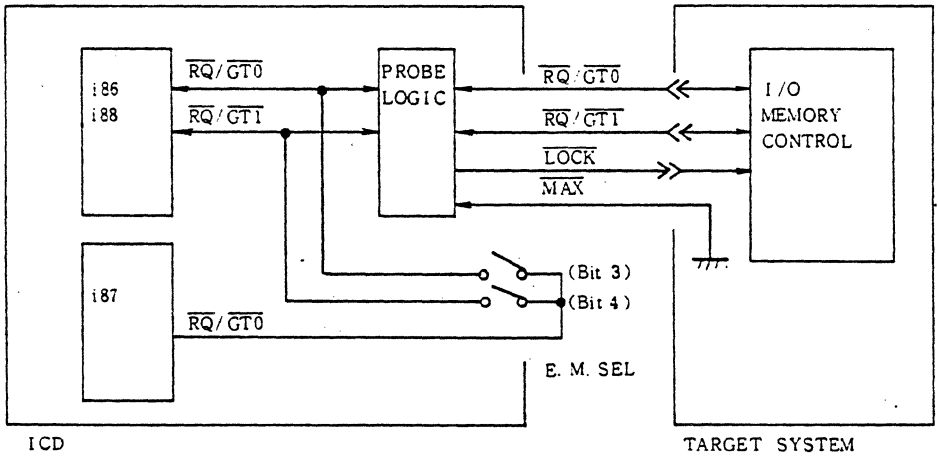
BUS Control

RQ/GT, LOCK Signals (Min Mode)

The ICD accepts the Request/Grant (RQ/GT) signal if the in-circuit mode is I1 or I2, and is enabled and disabled by the PIN command. This permits direct memory access (DMA) during an ICD or target system emulation break. (See the PIN command in Section 2.)

The NDP's RQ/GT0 signal may be connected to the CPU's RQ/GT0 signal by using emulation Method Select switch #1. (See "More About Your ICD," in Section 1, to learn how to use this feature.)

LOCK Signal. The ICD can output the LOCK signal at any time. It remains active during the execution of the "LOCK prefix" instruction when emulation.



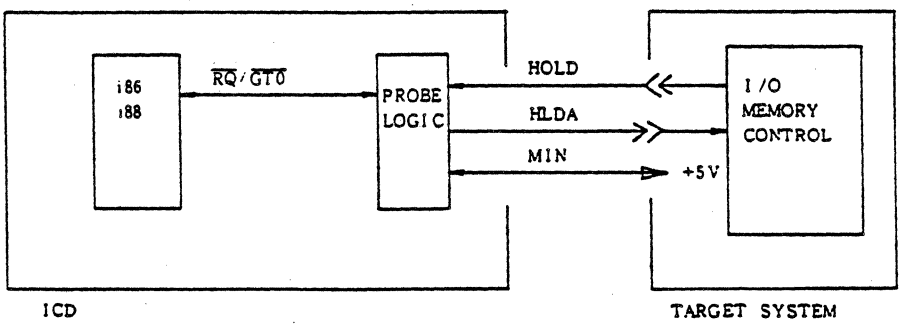
	10		11		12	
	MONITOR	EMULATION	MONITOR	EMULATION	MONITOR	EMULATION
RQ/GT0 RQ/GT1	×	×	O *1	O *1	C	C
LOCK	×	O	×	O	×	O

O: Effective × : Not effective
* Enable/Disable is possible by the "Pin" command.

HOLD/HLDA Signals (Min Mode)

The ICD accepts the HOLD signal whn the in-circuit mode is I1 or I2, to allow direct memory access (DMA) during an emulation break.

The HOLD signal may be enabled or disabled by using the PIN command. (See the PIN command in Section 2.)

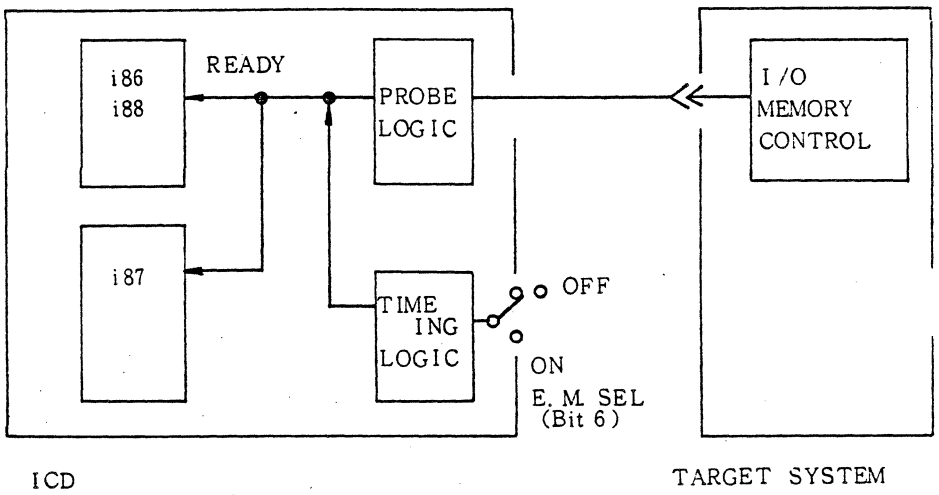


READY Signal (Min and Max Modes)

The READY signal is active when the target system or I/O is accessed. This signal is useful when emulating target systems that operate at high clock speeds; when the target memory or I/O access time is short; or when a small margin of time is left in the READY set-up time of the target system.

Emulation Method Select switch #1 can be used to generate wait states (1, 2, or 3) into the machine cycle operation by controlling the input to the READY signal. (See "More About Your ICD," in Section 1.)

If the ICD is unable to access the target system's memory contents within a certain time period (128 clock cycles), you can make the ICD cause a break in the program by using the "BREAK: Timeout Breakpoint" command. (See the BREAK command in Section 2.)



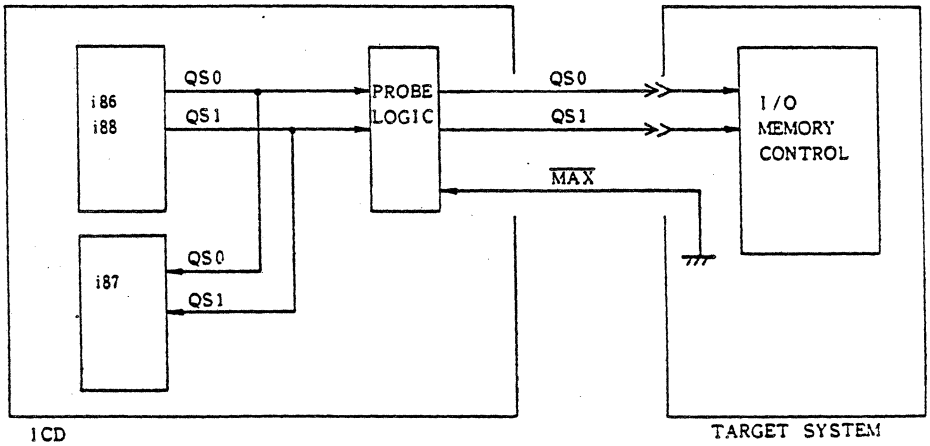
	10		11		12	
	MONITOR	EMULATION	MONITOR	EMULATION	MONITOR	EMULATION
READY	×	×	× *1	○ *1	× *1	○ *1

○: Effective ×: Not effective
*1 READY is considered effective the target system is accessed.

QSO and QS1 Signals (Max Mode)

The QS0 and QS1 (queue) signals can output to the target system at any time. Although these signals are effective during emulation, they are useless during emulation breaks since the ICD's S0, S1, and S2 signals stop in the inactive state.

The ICD shows the status of "queue empty" before starting emulation; therefore, the target system can supervise the instruction queue during emulation.

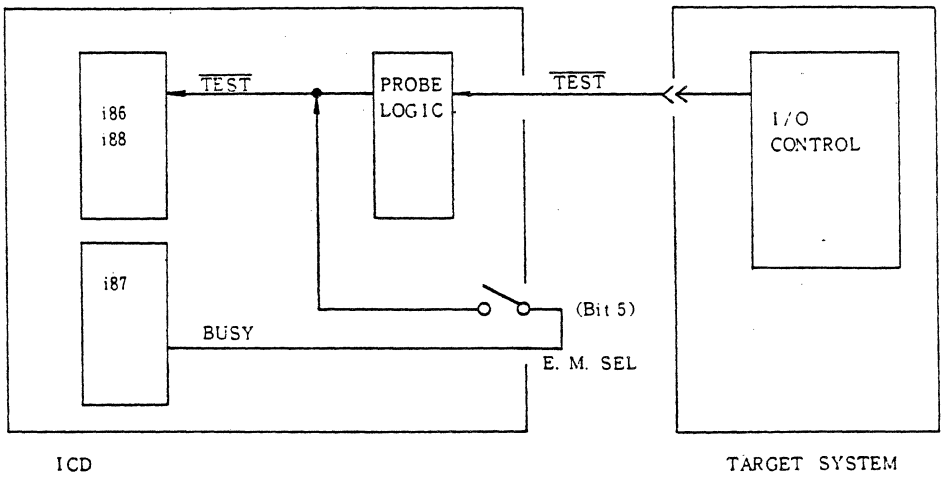


	10		11		12	
	MONITOR	EMULATION	MONITOR	EMULATION	MONITOR	EMULATION
QS0 QS1	△ *1	△ *1	△ *1	○	△ *1	○

TEST Signal

The ICD can accept the TEST signal at any time if the in-circuit mode is I1 or I2. If the TEST signal is inactive when the WAIT instruction is executed during emulation, the processor does not proceed to the next instruction.

The NDP's BUSY signal can input the CPU's TEST signal by using Emulation Method Select switch #1. If this setting is used, the TEST input of the target system is "ORed" with the BUSY signal. (See "More About Your ICD," in Section 1.)



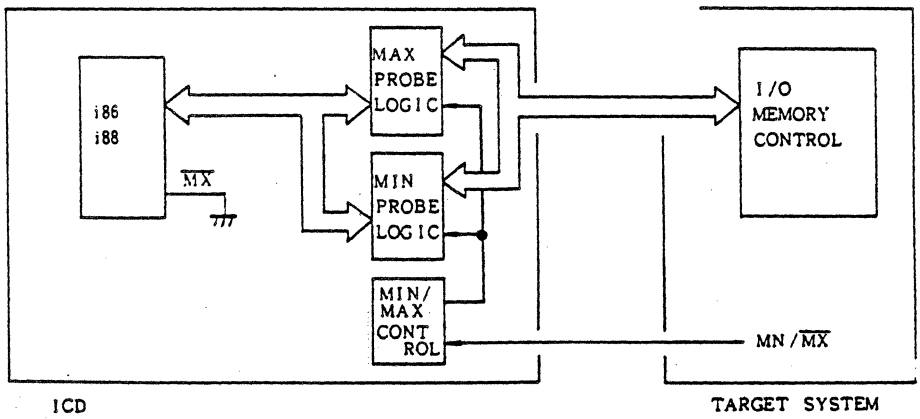
	10		11		12	
	MONITOR	EMULATION	MONITOR	EMULATION	MONITOR	EMULATION
TEST	× *1	△ *1	○ *1	○ *1	○ *1	○ *1

○: Effective ×: Not effective

*1 When Bit 5 of the E.M.SEL is set to ON, BUSY of the emulation NDP is input to the TEST of the emulation CPU.

MN/MX Signals

For proper minimum/maximum (MIN/MX) input, the ICD must be operating with a target system, the in-circuit mode must be either I1 or I2, and the MN/MX signals must not change during emulation.



	I0		I1		I2	
	MONITOR	EMULATION	MONITOR	EMULATION	MONITOR	EMULATION
MN/MX	×	×	○ *1	○ *1	○ *1	○ *1

○: Effective : Not effective

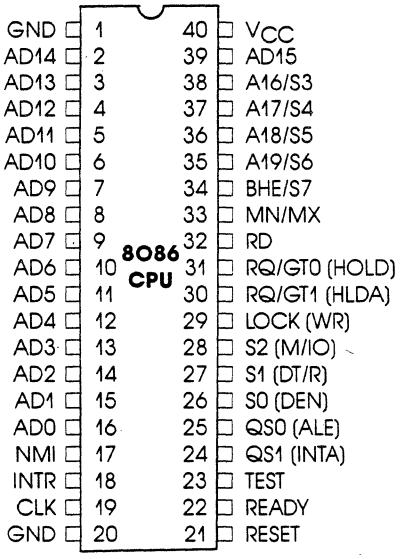
*1 Changing MN/MX in in-circuit mode I1 or I2 is not permitted. The in-circuit mode I1/I2 must be set after deffining the MN/MX in the IO status.

NDP Emulation

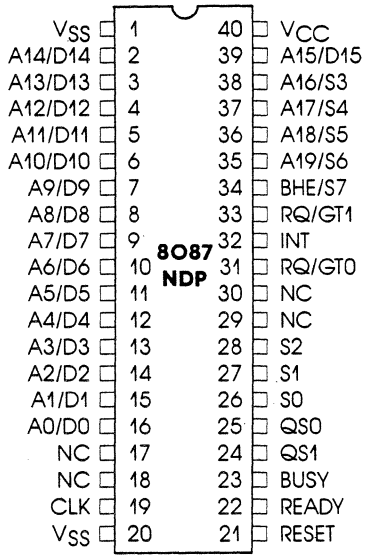
The 8087 Numeric Data Processor (NDP) performs arithmetic and comparative operations on a variety of numeric data types, as well as executing numerous built-in transcendental functions (e.g., tangent and log functions).

The NDP does not operate as an independent device, but rather as an extension processor to the existing CPU (8086/8088) when in the maximum mode. By operating as a co-processor, the NDP effectively extends the register and instruction sets of the host CPU, and adds several new data types as well.

8086/88 Pin Configuration



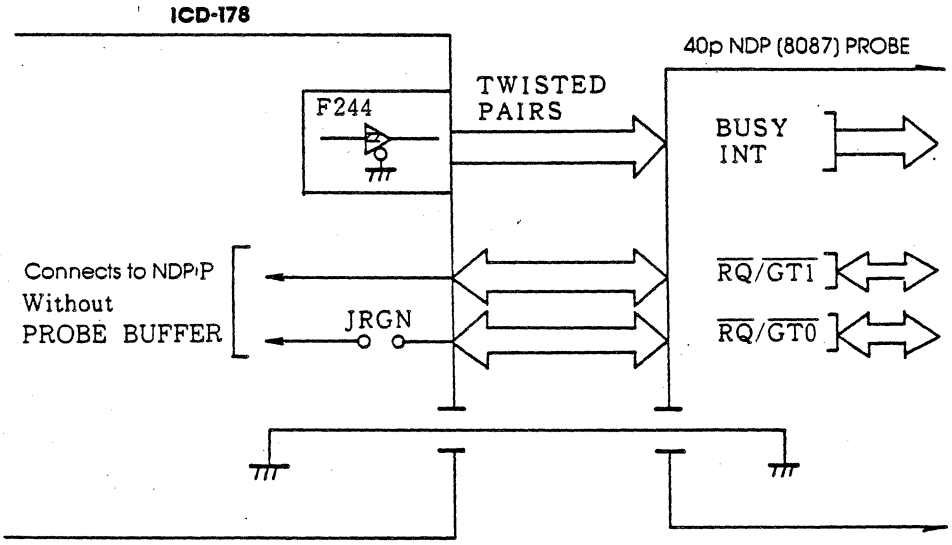
8087 Numeric Data Processor Pin Configuration



NC = NO CONNECT

NDP/CPU Interface

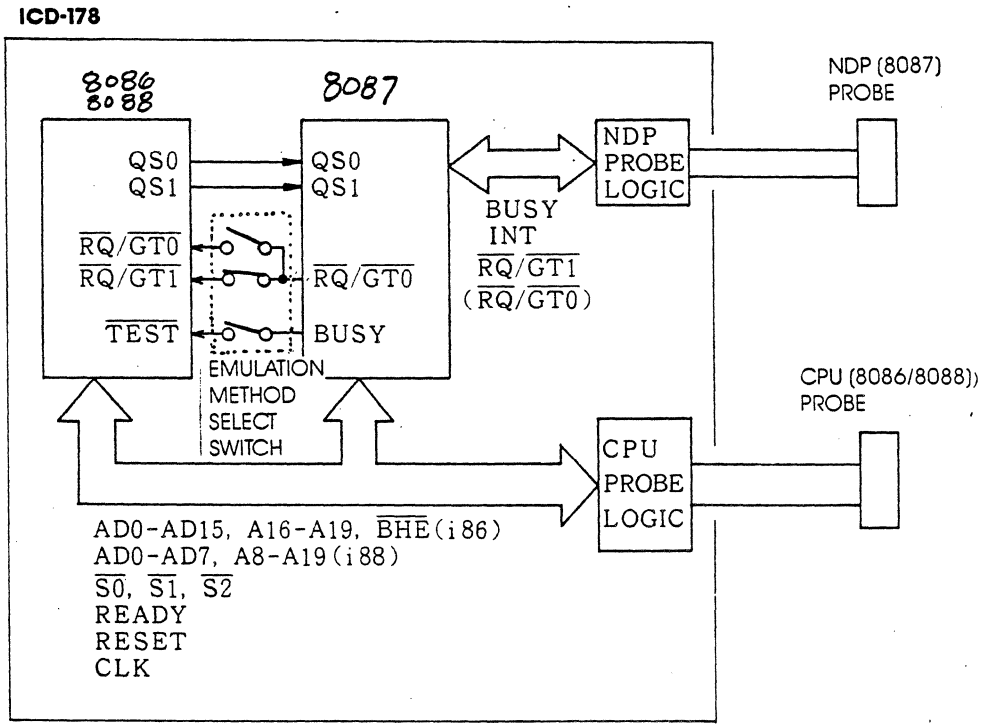
Since the NDP is wired directly to the host CPU, the two processors can be thought of as a single processor. The CPU's queue status lines (QS0 and QS1) enable the NDP to obtain and decode instructions in synchronization with the CPU. The NDP's BUSY signal informs the CPU that the NDP is executing. The WAIT instruction tests this signal to ensure that the NDP is ready to execute instructions. The NDP can interrupt the CPU when it detects an exception. The NDP also uses one of the host CPU's request/grant lines to obtain control of the local bus for data transfers. The CPU and NDP processors all utilize the same clock generator and system bus interface (bus controller, latches, transceivers, bus arbiter); no additional hardware is needed to interface the two processors.



Emulating The NDP

Since both NDP and CPU interface signal lines are internally connected within the ICD, a target system which incorporates an NDP in its design can be emulated, in most cases, using only the ICD's CPU in-circuit probe. However, the RQ/GT0 and BUSY signals of the NDP are connected to the CPU via the Emulation Method Select switch #1, and this switch must be set correctly in order to emulate the target system's NDP without using the NDP in-circuit probe.

For a complete description of how the Emulation Method Select switch #1 affects the signal interface, see "More About Your ICD," in Section 1.



When To Use The NDP In-circuit Probe

There are certain conditions under which the NDP in-circuit probe must be used to emulate an NDP-equipped design. You should use the NDP in-circuit probe when any of the following conditions exist:

1. The NDP's INT signal is used for an interrupt to the CPU.
2. The NDP's BUSY signal is not used to monitor the TEST terminal of the CPU.
3. The NDP's RQ/GT1 signal is used to connect a bus request from another IOP (Input/Output Processor—8089) to the target IOP.
4. The NDP's RQ/GT signal is not used for a bus request to RQ/GT0 or RQ/GT1.

If the NDP in-circuit probe is required for your particular application, see "System Preparation," in Section 1.

NDP Machine Cycles

The QS1, QS2, S0, S1, S2, and CLK signals of the CPU and NDP processors are directly connected within the ICD. This allows the NDP to be synchronized with the CPU, regardless of the in-circuit mode or whether the ICD's NDP is connected to the target system.

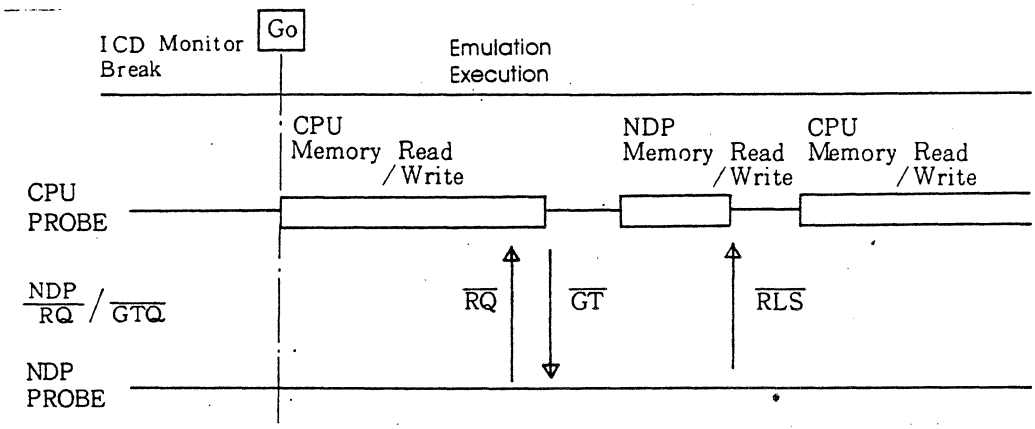
The NDP memory access sequence is executed in the following manner:

1) The RQ signal is generated from the NDP's RQ/GT0 line that is connected to the CPU via the Emulation Method Select switch #1.

2) After a GT response from the CPU, the CPU probe interface sets the bus signal to 3-state.

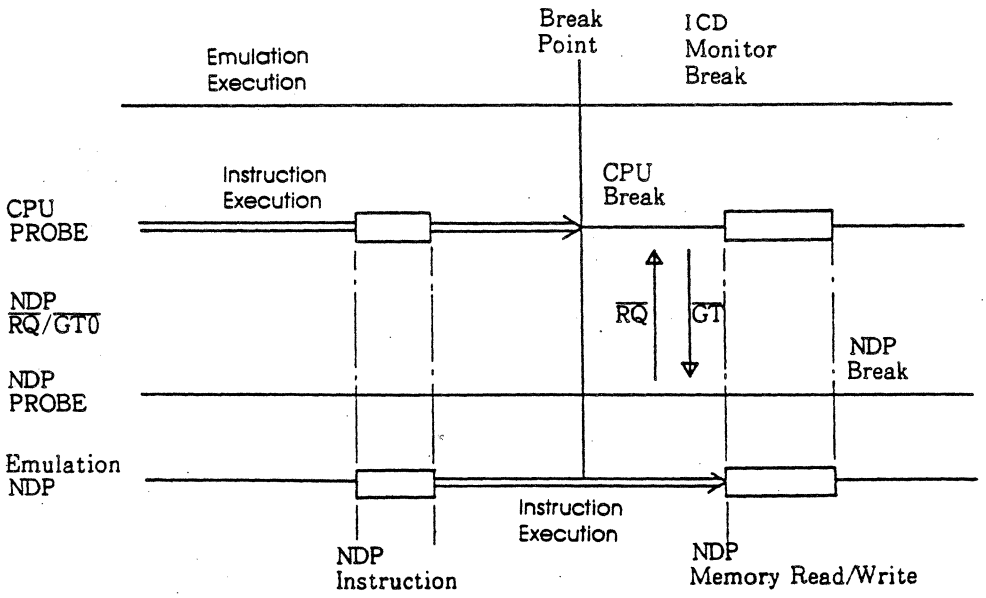
3) When the NDP starts memory access, the bus signal and S0/S1/S2 signals become active and emulate the NDP memory read/write operation.

When the RLS signal of the NDP is generated, the CPU, via the probe, resumes CPU memory access.



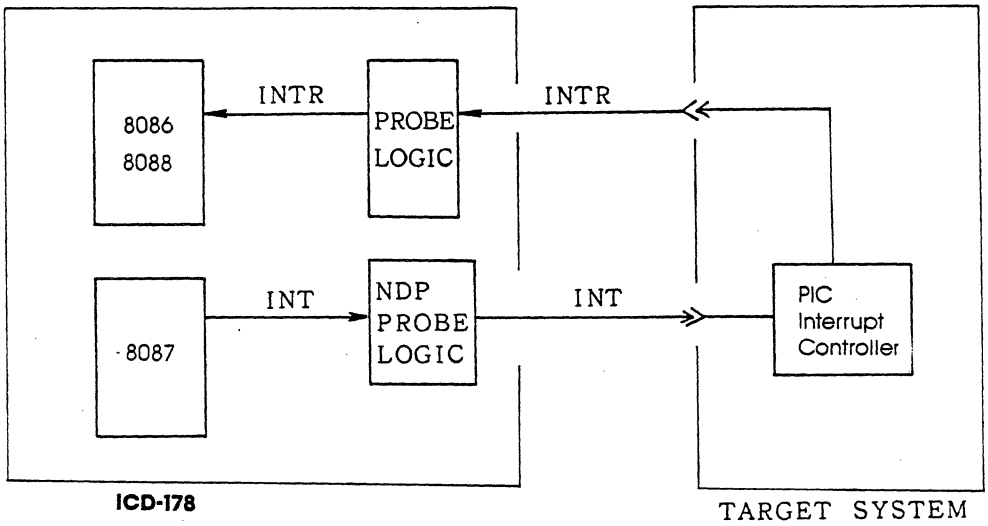
The NDP memory may be accessed after the ICD breaks emulation. The ICD breaks CPU emulation if 500 clock cycles pass, from the time of the NDP instruction fetch (e.g., FST, FIST, EBSTP) to the time of memory data writing. In this case, the ICD can complete a bus request and memory access of the NDP.

The RQ/GT0 signal, connected from the NDP to the CPU via Emulation Method Select switch #1, is received by the CPU even during an emulation break; it cannot be inhibited by the PIN command.



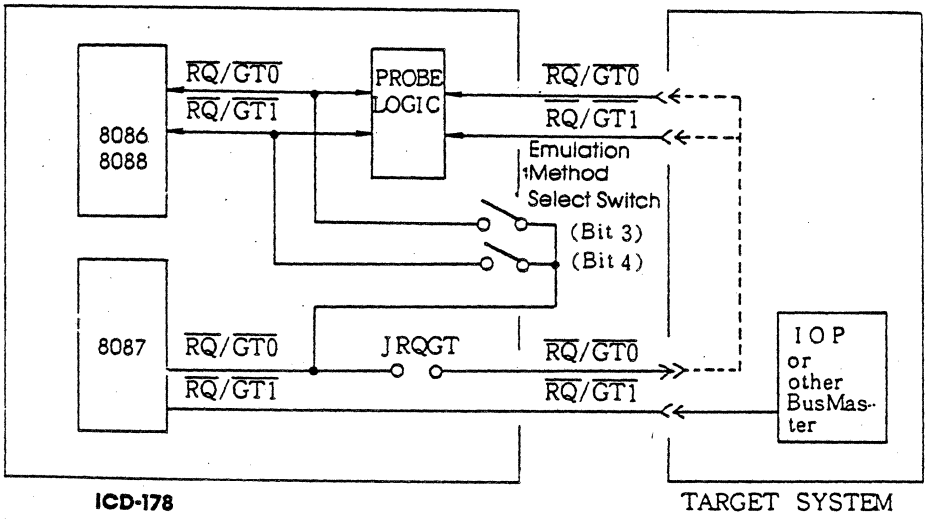
NDP Interrupt Signal

The ICD can output the INT signal of the NDP in all in-circuit modes (I1/2/3). The INT signal is used to generate an interrupt sequence to the CPU by the Programmable Interrupt Controller (PIC). The CPU then transmits the interrupt signal to the target system in the I1 or I2 modes only.



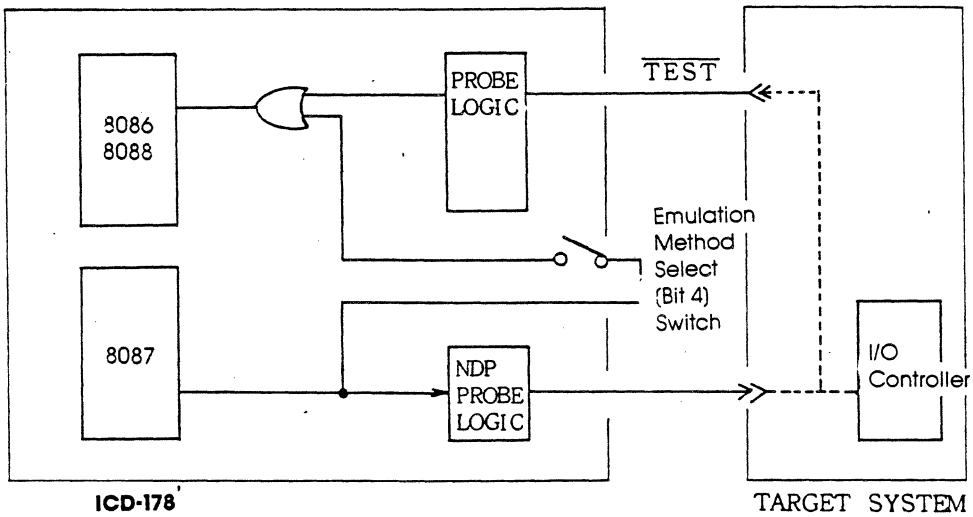
NDP Bus Control

When the ICD's in-circuit mode is I1 or I2, the NDP's RQ/GT1 signal can be used through the NDP probe connected to the target system. In the I0 mode, input of a bus request signal to the NDP's RQ/GT1 is not permitted. A bus request from RQ/GT1 cannot be inhibited with the PIN command because of the direct internal connection between processors.



NDP BUSY Signal

The NDP's BUSY signal can be accessed by the target system in all in-circuit modes (I0/I1/I2). Generally, the BUSY signal connects to the TEST terminal of the CPU, and the BUSY signal between the NDP and CPU is joined via Emulation Method Select switch #1.



Emulator Control Module Description

The Emulator Control module (EMU S-772) controls the emulation and monitor modes of operation. The module also houses the Event Trigger connector and the External Break connector; both are externally accessible. There are no internal user-serviceable controls or components on this module.

(half tone of module)

Real-time Storage Module Description

The Real-time Storage module (RTS S-775) includes the controller, memory, and real-time counter for tracing and storing the user program. By using the HISTORY command, different sections in the program can be traced, stored, and then dumped and displayed.

The HISTORY command is used to control the functions of the real-time trace module; there are no user-serviceable controls or components on this module. (For a complete description of how the real-time trace feature works, see the HISTORY command in Section 2.)

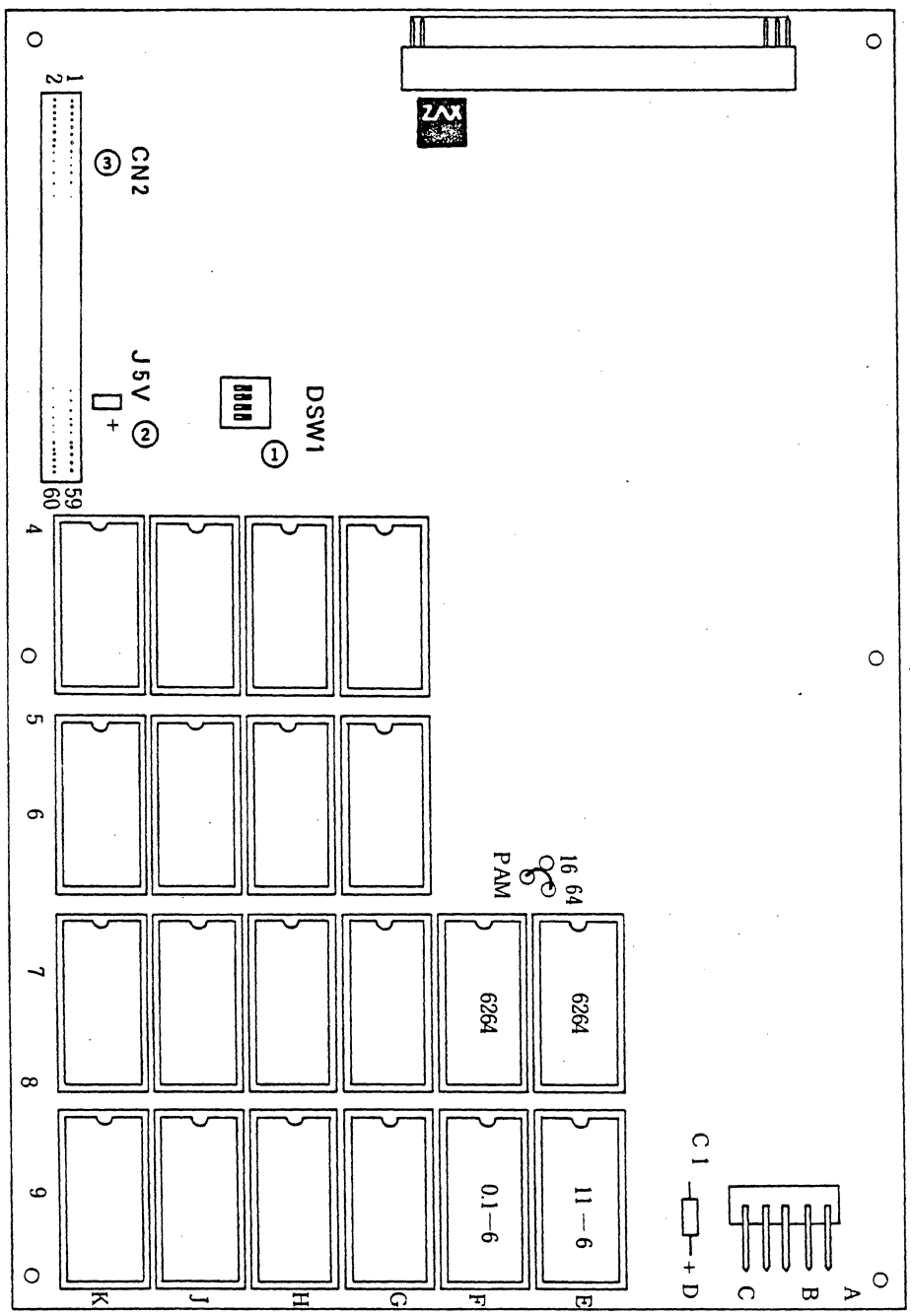
(halftone of real-time storage module)

Memory Mapping Unit Module Description

The Memory Mapping Unit module (MMU S-776) contains 128K bytes of high-speed static RAM (known as "emulation memory"), which can be used for downloading files, altering the memory contents, and loading future memory into the target system.

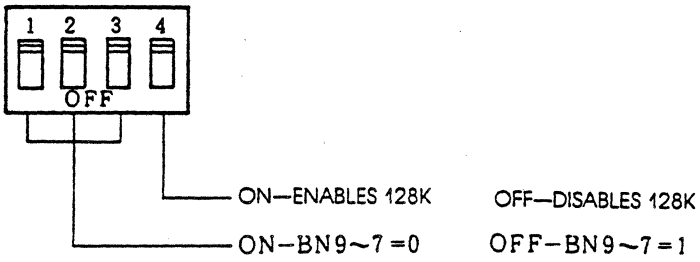
There are a few user-serviceable components on this module. See "How To Disassemble Your ICD," at the end of this section, after reading about the MMU's components on the following pages.

(halftone of module)


CN2
J5V
DSW1
PAM
16 64
6264
6264
0.1—6
11—6
C 1
+ D
A
B
C
K
J
H
G
F
E

MMU Components

1) ICD Program Memory Block Number switch. This 4-bit switch sets the allocation block number of the 128K-byte memory mapping unit in 1K-byte blocks. (For more information on memory allocation, see the ALLOCATION command in Section 2.)



ICD Bus Connector Pin Assignment

SIGNAL NAME	PIN No.	PIN No.	SIGNAL NAME
GND	2	1	GND
A1M	4	3	GND
A3M	6	5	A2M
A5M	8	7	A4M
A7M	10	9	A6M
A9M	12	11	A8M
A11M	14	13	A10M
A13M	16	15	A12M
A15M	18	17	A14M
A17M	20	19	A16M
A19M	22	21	A18M
A21M (GND)	24	23	A20M (GND)
A23M (GND)	26	25	A22M (GND)
A25M (GND)	28	27	A24M (GND)
GND	30	29	GND
D1M	32	31	D0M
D3M	34	33	D2M
D5M	36	35	D4M
D7M	38	37	D6M
D9M	40	39	D8M
D11M	42	41	D10M
D13M	44	43	D12M
D15M	46	45	D14M
GND	48	47	GND
GND	50	49	WRLM
GND	52	51	WRHM
GND	54	53	RDM
GND	56	55	GND
+5V	58	57	+5V
GND	60	59	GND

ICD Bus Connector Pin Assignment

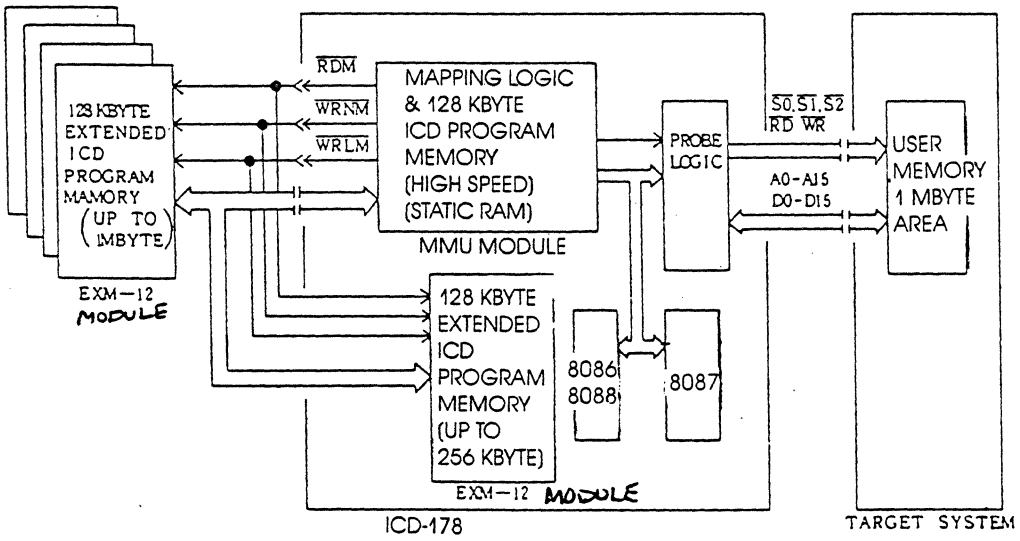
Pin Descriptions

Signal Names	I/O	Description
A1M—A25M	Output	ICD Program Memory Address Bus: A1—A19 are effective, and A20—A25 are set to low level.
D0M—D15M	Output/ Input	ICD Program Memory Data Bus: I/O timing conforms to machine cycle operation of 8086 processor.
RDM	Output	Expansion Memory Read: Acts as data read signal to Expansion Memory module. Output is synchronized with RD signal of 8086 processor.
WRHM	Output	Expansion Memory High Data Write: D8—D15 data write signals are sent to Expansion Memory module. Output is synchronized with WR signal of 8086 processor.
WRLM	Output	Expansion Memory Low Data Write: D0—D7, 8-bit data signal is sent to Expansion Memory module.
+5V		Vcc reference line: Used by Expansion Memory module.
GND		ICD ground signal.

ICD Emulation Memory

The ICD-178 for 8086/8088 features 128K bytes of RAM (emulation memory). This memory can be used for downloading object files, as well as altering or manipulating the target system's memory. (Emulation memory contrasts to user memory in that user memory is contained in the target systemm itself.) Emulation memory can be expanded internally to 256K bytes by the addition of the Expansion Memory module.

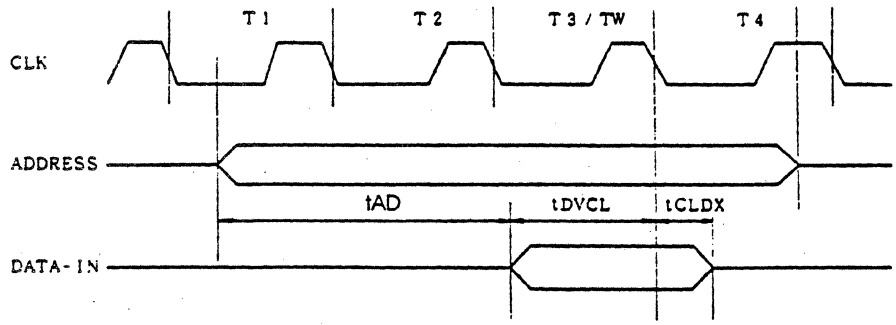
ICD emulation memory is composed of high-speed static RAM, which allows the support of multi-speed target systems. When viewed from the target system, emulation memory is different from a normal memory area in that it is contained within the 8086/8088 processor. Because of the special characteristics of emulation memory, DMA transfer between the target system and the ICD emulation memory is not possible; however, DMA transfer between the address spaces within the target system is permitted.



Target System (User) Memory

The memory contained in the target system is called target system memory or user memory. The ICD can address any area of the 1M-byte target system memory.

The access time required to write to the target system memory from the ICD is identical to that of the processor; however, the access time needed to read from the target system memory is slightly shorter than with the processor. Therefore, certain access conditions must be satisfied for accurate reading. These conditions are shown below:

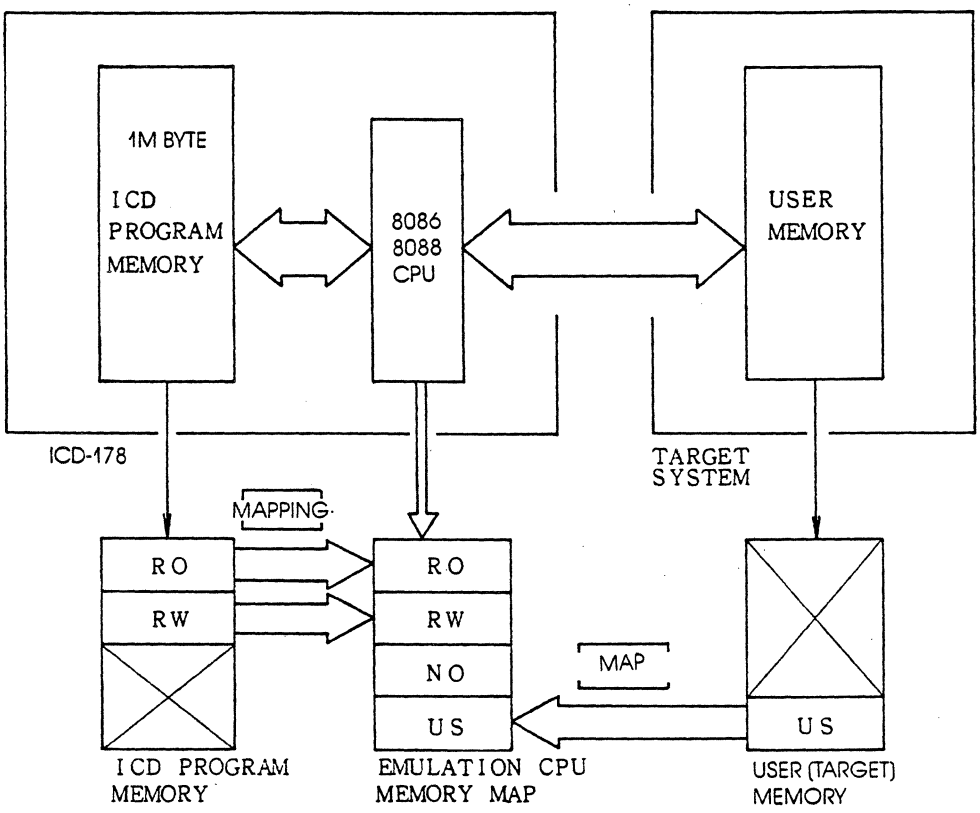


tAD	A0-A19 Valid to Valid Data In	max (3+N) T−95 ns	
tDVCL	Data Setup Time	min	35 ns
tCLDX	Data Hold Time	min	10 ns

N is equal to the total WAIT state T = CLK Cycle Period

Target Memory Timing Diagram

Mapping You can use all or part of the ICD's RAM in place of target system memory by creating a memory map. The emulation memory or target system memory can be mapped in increments of 1K bytes using the MAP command. (For an explanation and example of how this works, see the MAP command in Section 2.)



Power Supply Specifications

Line voltage: 100 to 120 volts AC
200 to 240 volts AC

Frequency: 50 or 60 Hz

Power: 50 watts

Output voltage: +5 volts DC
+12 volts DC
−12 volts DC

The Power Supply provides +5 volts to the control modules and 24 volts to the external cooling fan. The voltage to the control modules is filtered to reduce noise from the power supply line.

How To Disassemble Your ICD

Introduction

The ICD must be partially or fully disassembled in order to modify the components and controls or change certain settings on the control modules. In this chapter, you'll find the procedure for disassembling the ICD and removing (and installing) the control modules.

Important Notice To Users!

Before you begin any disassembly of your ICD, you should be aware of certain guidelines which must be followed in order to preserve the Warranty Policy on this equipment.

1) All adjustments and modifications to the ICD are limited to the Serial Interface Output (SIO) module, CPU control module, Memory Mapping Unit (MMU) module, and Expansion Memory (EXM) module. The adjustments and modifications authorized by ZAX are clearly identified (▲) in each of these chapters. Any other alterations or adjustments on the modules void the Warranty Policy.

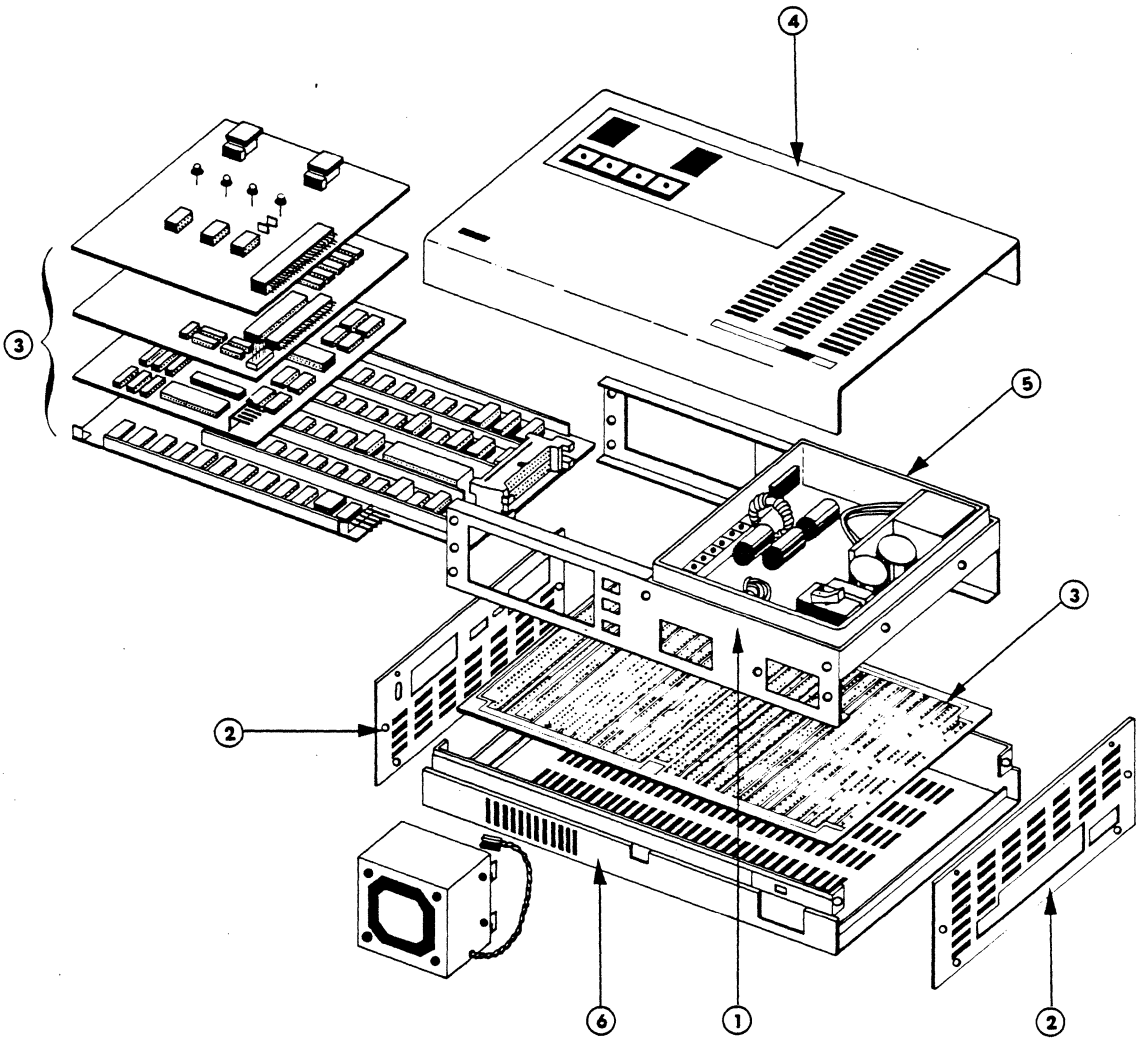
2) Do not adjust, modify, and/or in any way alter the controls or components on any of the four remaining modules (Indicator/Control, Real-time Storage, Emulator Control, Break Matrix) or the power supply.

3) Follow the disassembly procedure described here. Damage may result if the ICD is disassembled, or the modules removed, in a manner other than that described in this chapter.

Basic Parts Of Your ICD

The construction of all **ZAX** ICD-series emulators is very similar. The basic ICD unit includes the mainframe, seven (excluding the optional Expansion Memory module) control modules, power supply, Mother Bus cable, and outside casing. The **mainframe** is a metal chassis that houses the control modules and power supply. The seven **control modules** are circuit boards (sometimes called "cards") that do the actual work of emulating the target system, storing memory, tracing programs, etc. The **power supply** provides voltage for the modules. The **Mother Bus** cable permits the modules to communicate with each other. The ICD **case** consists of a top cover, bottom cover, and two side covers.

① Main Frame
② Side Covers
③ Control Modules
④ Top Cover
⑤ Power Supply
⑥ Bottom Cover


4
3
5
3
2
6
1
2

▲ Procedure For Disassembling The ICD

WARNING HAZARDOUS VOLTAGE IS PRESENT WITHIN THE ICD-178. DISCONNECT THE AC POWER PLUG BEFORE BEGINNING ANY INTERNAL WORK ON THE ICD-178.

Disassembling the ICD requires the following tools:

(1) medium Phillips-type screwdriver: used for removing the outside case screws.

(1) small Phillips-type screwdriver: used for removing the modules from the internal mainframe.

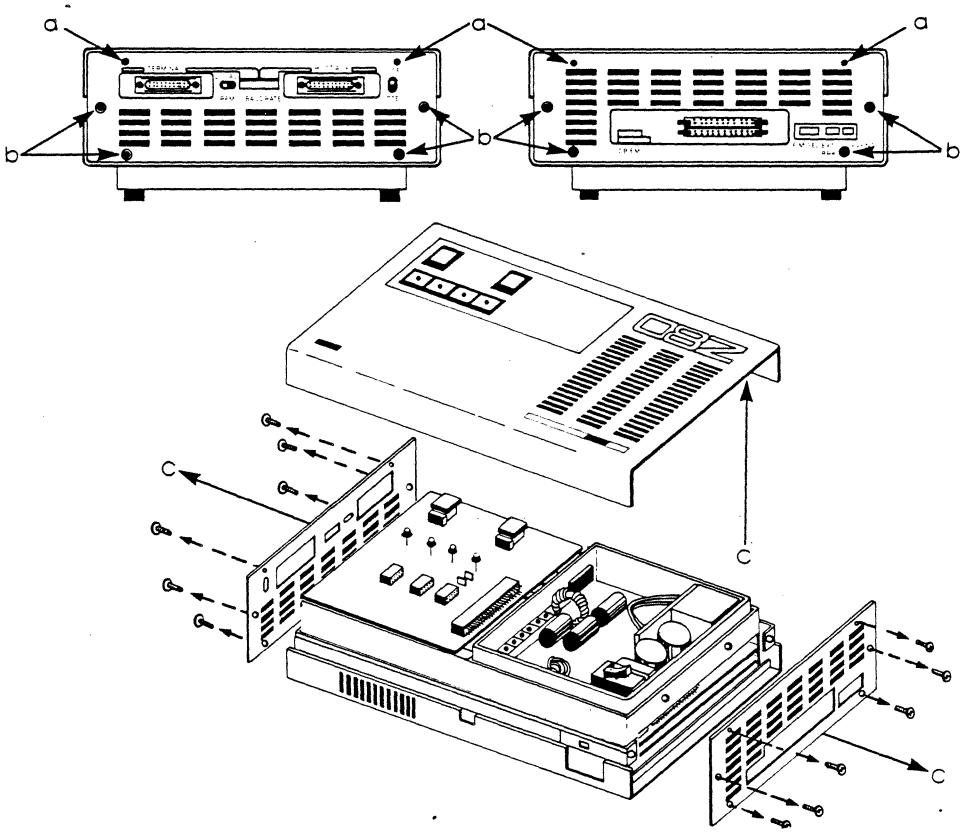
(1) small slot-type screwdriver: used for prying the bus cable sockets away from the pin connectors.

(1) pair of needle-nosed pliers: used for removing and attaching power-supply connectors from the 5-pin plugs.

1. Remove the top cover and two side covers.

   a) Remove the four raised screws that connect the top cover to the side covers. Lift the top cover off the ICD.

   b) Remove the eight countersunk screws that attach the side covers to the ICD and detach each cover.



2. Gently turn the ICD over and remove the bottom cover.

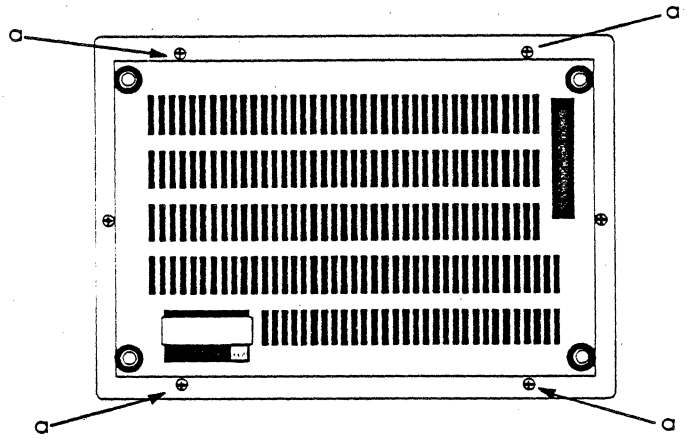
NOTE: Place the ICD on a soft foam-type pad to protect the top components.

a) Remove the four chrome-plated screws that attach the bottom cover to the mainframe (it is not necessary to remove the two black countersunk screws).

b) Lift the bottom cover off the ICD.

c) Turn the ICD back over on the foam pad so the control panel and power supply are facing up.

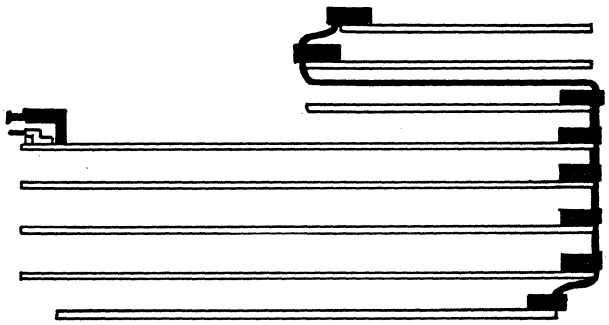
The control modules are now accessible for removal.



How The Modules Are Connected

Each module is linked by the Mother Bus cable. Power is supplied to each module by a socket-type power-supply connector (except for the Indicator/Control Panel, S-730, which receives its power from the Mother Bus cable). The power-supply connector and Mother Bus cable must be detached before removing any of the control modules.

IMPORTANT: Note the position of the power-supply connectors before removing them. Both the socket and plug have a black label on one side to indicate the polarity of the connectors.



Procedure For Removing The Modules

1. Remove the Indicator/Control Panel (S-730).
   a) Detach the Mother Bus cable from the Indicator/Control panel (location CN-1).
   b) Remove the four screws that attach the panel to the mainframe.
   c) Remove the Indicator/Control panel from the mainframe.
2. Remove the Serial Interface Output (SIO-S-771) module.
   a) Remove the two small screws that attach the module to the mainframe.
   b) Detach the Mother Bus cable from the module.
   c) Detach the power-supply connector from the module.
3. Remove the optional Expansion Memory module (EXM-12 S-766), if supplied.
   a) Detach the auxiliary bus cable from the module.
   b) Detach the power-supply connector from the module.
   c) Remove the two small screws that attach the module to the mainframe.
   d) Remove the Expansion Memory module from the mainframe.

NOTE: When installing this module, carefully fold the bus cable to make a 90-degree turn so that it can attach to the 60-pin bus receptacle on the Memory Mapping Unit module (located on the bottom of the ICD).

4. **Remove the Break Matrix (BRX S-788) module.**
   a) Detach the Mother Bus cable from the module. Detach the 30-pin bus cable that connects the BRX module to the EMU module.
   b) Detach the power-supply connector from the module.
   c) Remove the two small screws that attach the module to the mainframe.
   d) Remove the Break Matrix module from the mainframe.

5. **Remove the Central Processing Unit (CPU S-773) module.**
   a) Detach the Mother Bus cable from the module by pushing out the retaining clips from the connector housing.
   b) Detach the power-supply connector from the module.
   c) Remove the two small screws that attach the module to the mainframe.
   d) Slide the module away from the mainframe at the power-supply connector-end of the ICD.

6. **Remove the Emulator Control (EMU S-772) module.**
   a) Detach the Mother Bus cable from the modulee.
   b) Detach the power-supply connector from the module.
   c) Remove the two small screws that attach the module to the mainframe.
   d) Slide the module away from the mainframe at the power-supply connector-end of the ICD.

7. **Remove the Real-time Trace Storage (RTS S-775) module.**
   a) Detach the Mother Bus cable from the module.
   b) Detach the power-supply connector from the module.
   c) Remove the two small screws that attach the module to the mainframe.
   d) Slide the module away from the mainframe at the power-supply connector-end of the ICD.

8. **Remove the Memory Mapping Unit (MMU S-776) module.**
   a) Turn the ICD over so the MMU module is on top.
   b) Detach the Mother Bus cable from the module.
   c) Detach the power-supply connector from the module.
   d) Remove the four small screws that attach the module to the bottom of the mainframe.
   e) Lift the module from the mainframe.

Installing The Modules

To install the modules, reverse the "removing the modules" procedure.

CAUTION: DO NOT REVERSE POWER CONNECTOR POSITION DURING INSTALLATION. CONNECTOR MISPLACEMENT WILL CAUSE DAMAGE TO THE ICD-178.

NOTE: When replacing the side panels, loosely position all the screws in place to allow the panels to align properly before tightening the screws.

Introduction

ZAX ICD-series emulators require the management of a communications utility in order to interface to a host computer. Normally, this process is controlled by ZAX's own communications utility, ZICE (pronounced "Zee ice"). However, when using the ICD with systems not supported by ZICE, it is necessary to write your own communications programs to ensure an orderly information exchange between the ICD and the host computer system.

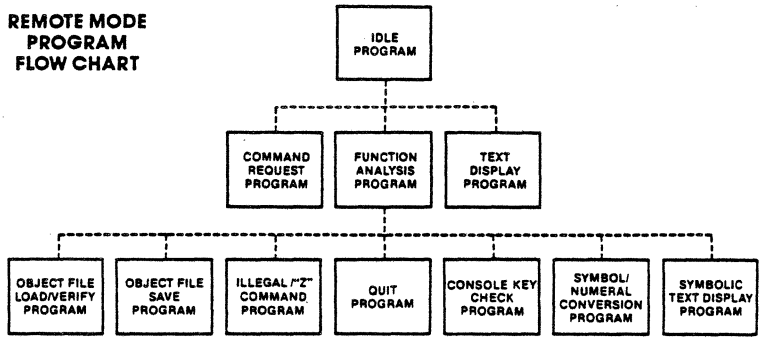
This section shows the communications programs for interfacing a host computer to the ICD when operating in the LOCAL and REMOTE modes.

When interfaced to a host computer, ZAX ICD-series emulators can operate in one of two system configurations. In one configuration, a host computer is used to directly control the ICD via the utility software program ZICE. This configuration is called "Host Computer Control of the ICD." The ICD operates in the REMOTE mode for this configuration.

In the other configuration, the ICD is under the direct control of a console terminal and uses a computer as either a data storage facility or as a conduit to the ZICE commands (i.e., help files, "Z" commands, etc.). This configuration is called "Terminal Control of the ICD—With Host Data Files." The ICD operates in the LOCAL mode for this configuration. (The HOST command activates the LOCAL "host computer assisted" mode.)

NOTE: Although this manual is specifically designed for use with your particular emulator, this section can be used with all ZAX ICD-series emulators which feature the "backslash" (\) protocol format. This format is structured as: \code{text} <CR>. *A Number and Symbol Conversion Code chart, which indicates the correct numbers and symbols to use with your particular emulator, is shown at the end of this section.*

REMOTE MODE PROGRAM FLOW CHART

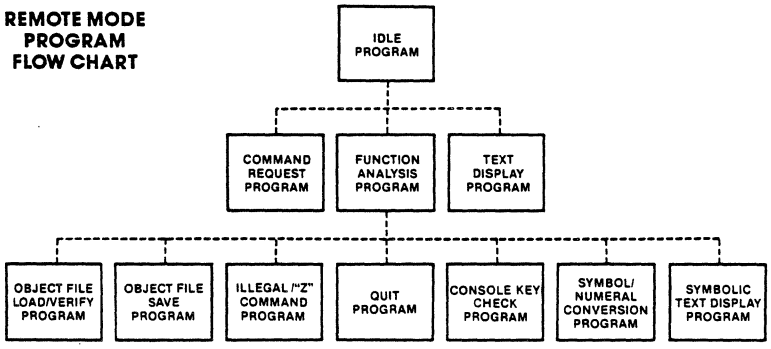


Program:	Mode:	Control:
COMMAND REQUEST	REMOTE	HOST COMPUTER

Action: This program acts as the main intermediary program (transferring instructions and text only) between the ICD and the subprograms (COMMAND REQUEST, TEXT DISPLAY, and FUNCTION ANALYSIS). After being converted to symbols, the parameters sent from the ICD are displayed. The host computer waits for an input from the ICD. (The host system must have an input buffer to hold the input code from the ICD.) The host computer receives one line of data and places it in the input buffer. The host computer then executes one of the following programs depending on the code it receives:

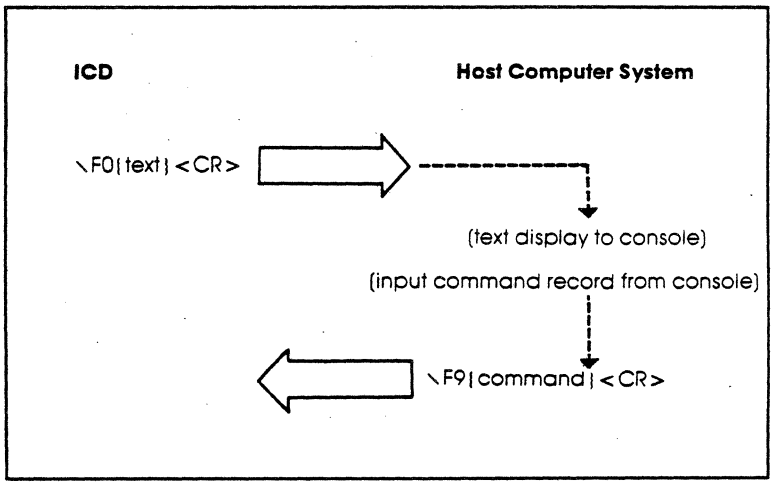
Input Buffer Contents	Program Executed
\FO<CR>	COMMAND REQUEST
\80{text}<CR>	TEXT DISPLAY
any other code	FUNCTION ANALYSIS

REMOTE MODE PROGRAM FLOW CHART

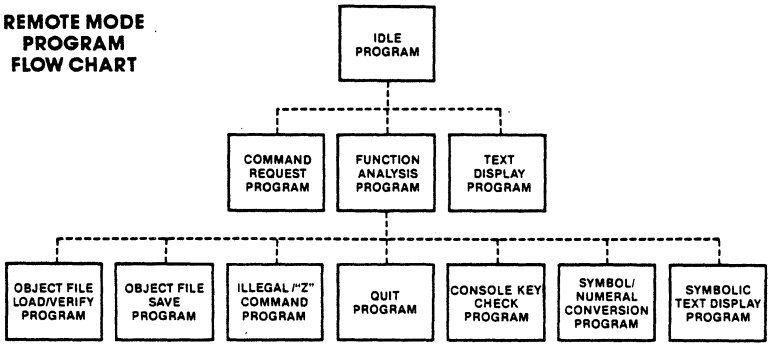


Program: COMMAND REQUEST		**Mode:** REMOTE	**Control:** HOST COMPUTER
Action: Sends a command from the host computer in response to a command request from the ICD.			
Code/Record	**Code/ Record Name**	**ICD⟷HOST**	**Notes**
\F0{text}<CR>	COMMAND REQUEST RECORD	ICD→HOST	Contains ASCII text to be displayed, but does not include <ACK>, <NAK>, or <SOH>. Host computer then displays {text}.
\F9{command}<CR>	COMMAND RECORD	ICD←HOST	ASCII text sent as a command to the ICD. Record cannot contain any control code and must end with <CR>. When record is entered from host computer, the system accepts one line of data, echoing it back to the screen.

	(The cursor stays on the same line after the echo. The ICD sends a code in the text display sequence to move the cursor to the next line.)
Program Description: The ICD first requests a command by sending \F0{text}<CR> to the host computer. Upon receiving the record from the ICD, the host computer waits for an input after displaying the text record. When a command record is entered from the host computer, the system sends it to the ICD and then returns to the IDLE program.	



REMOTE MODE PROGRAM FLOW CHART

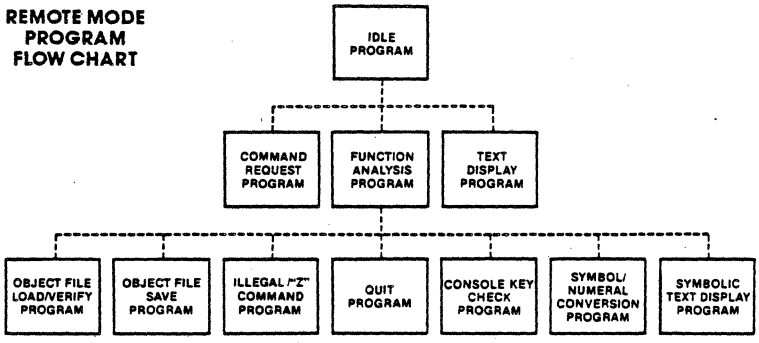


Program:	Mode:	Control:
FUNCTION ANALYSIS	REMOTE	HOST COMPUTER

Action: Host computer places one line of data from the ICD into the input buffer and analyzes the data. The host computer then executes one of the following programs based upon the contents of the input buffer.

Input Buffer Contents	Program Executed
\00{filename}<CR> or \{filename}<CR<	FILE LOAD
\01{filename}<CR> or \03{filename}<CR>	FILE VERIFY
\{filename}<CR> or \12{filename}<CR>	FILE SAVE
\43{parameter}<CR>	"Z" COMMAND
\44<CR>	QUIT
\2X{symbol}<CR>	SYMBOL CONVERSION
\3X{parameter}{text}<CR>	SYMBOLIC TEXT DISPLAY
\88 or \8A<CR>	Checks the console input Checks the console input in the host computer.

REMOTE MODE PROGRAM FLOW CHART



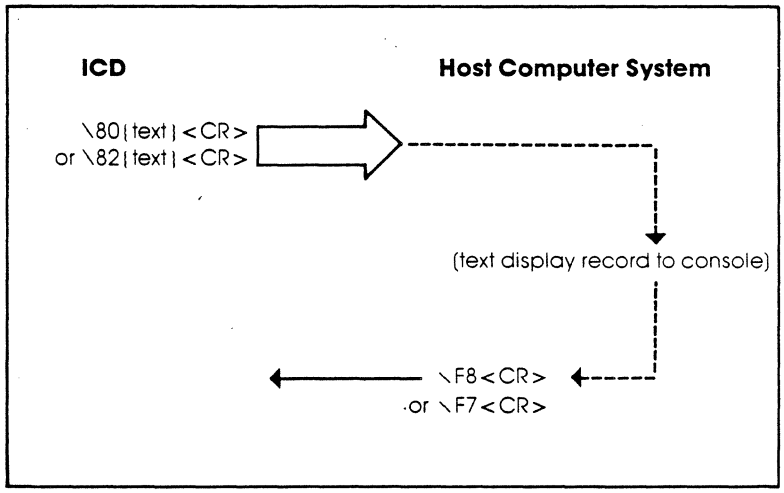
Program: TEXT DISPLAY		Mode: REMOTE	Control: HOST COMPUTER
Action: Text sent from the ICD is displayed.			
Code/Record	**Code/ Record Name**	**ICD←→HOST**	**Notes**
\80{text}<CR> or \82{text}<CR>	TEXT RECORD	ICD→HOST	ASCII text sent to host computer screen from ICD. <ACK>, <NAK>, <ENQ>, or <SOH> cannot be contained in the text record.
\F8<CR># or #	DISPLAY COMPLETE ACKNOWL-EDGE CODE	ICD←HOST	Sent to ICD when display has been completed.
F7<CR>	DISPLAY INTERRUPT CODE	ICD←HOST	Sent to ICD to interrupt it from sending a text record when displaying a "scrolling-type" command (e.g. DUMP, TRACE).

Program Description: When the TEXT DISPLAY program is requested, the ICD sends \80{text}<CR> or \82{text}<CR> to the host computer. The host computer displays the text record, and then checks the console input status and acts on one of the following:

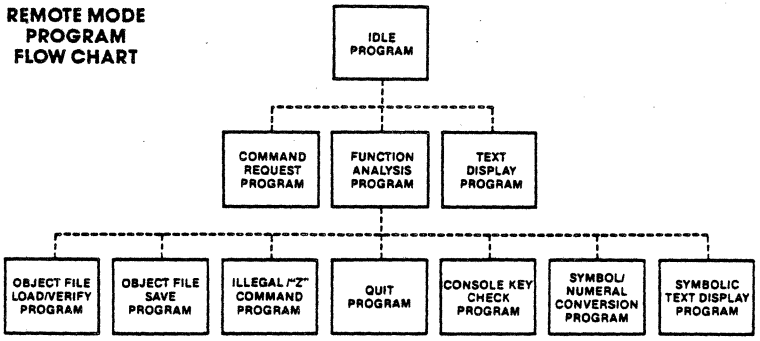
a) If no input is given, the host computer sends \F8<CR>to the ICD and returns to the IDLE program.

b) If the input code is ESC, the host computer sends \F7<CR> to the ICD and returns to the IDLE program, suspending any further text display.

c) If the input is a code other than ESC, the host computer sends \F8<CR> to the ICD and returns to the IDLE program.



REMOTE MODE PROGRAM FLOW CHART



Program:	Mode:	Control:
OBJECT FILE LOAD/VERIFY	REMOTE	HOST COMPUTER

Action: An object file is sent from the host computer in response to a LOAD/VERIFY request from the ICD. NOTE: The transmission of the object file may be interrupted by a TEXT DISPLAY REQUEST or a CONSOLE KEY INPUT CHECK REQUEST.

Code/Record	Code/ Record Name	ICD⟷HOST	Notes
\00{filename} <CR> or \02 {filename} <CR>RECORD or S FORMAT LOAD REQUEST RECORD	INTEL HEX LOAD REQUEST	ICD→HOST	Sent to host computer when ICD loads object file.
\01{filename} <CR> or 03 {filename} <CR>	INTEL HEX VERIFY REQUEST RECORD or S FORMAT VERIFY REQUEST RECORD	ICD→HOST	Sent to host computer when display has been completed.
{record} <CR>	OBJECT FILE RECORD	ICD←HOST	Intel Hex or S format record sent to ICD. Record may not contain any control code and must end with <CR>.

\F8<CR>	OBJECT FILE ACKNOWL-EDGE CODE	ICD→HOST	Sent to host computer to acknowledge successful receipt of an Intel Hex or S format record.
\F6<CR>	OBJECT FILE RE-TRANS-MISSION REQUEST CODE	ICD→HOST	Used when ICD requests host computer to re-transmit object record, usually due to a sum-check error.
\F7<CR>	OBJECT FILE TRANSMIS-SION INTERRUPT CODE	ICD→HOST	Ends LOAD/VERIFY sequence due to irrecoverable error.
\80{text}<CR>	TEXT RECORD	ICD←HOST	This record usually contains a verify error message.
\F8<CR> # or #	DISPLAY COMPLETE CODE	ICD←HOST	Sent to ICD when LOAD or VERIFY sequence is completed.
\F7<CR>	LOAD/ VERIFY SEQUENCE ABORT INDICATION CODE	ICD←HOST	Host computer aborts LOAD/ VERIFY sequence by sending this code.
\8A<CR>	CONSOLE KEY INPUT CHECK REQUEST CODE	ICD→HOST	Generally used to check status of an abort, or interrupt of verify error messages.
\F8<CR>	NO-CONSOLE INPUT CODE	ICD←HOST	

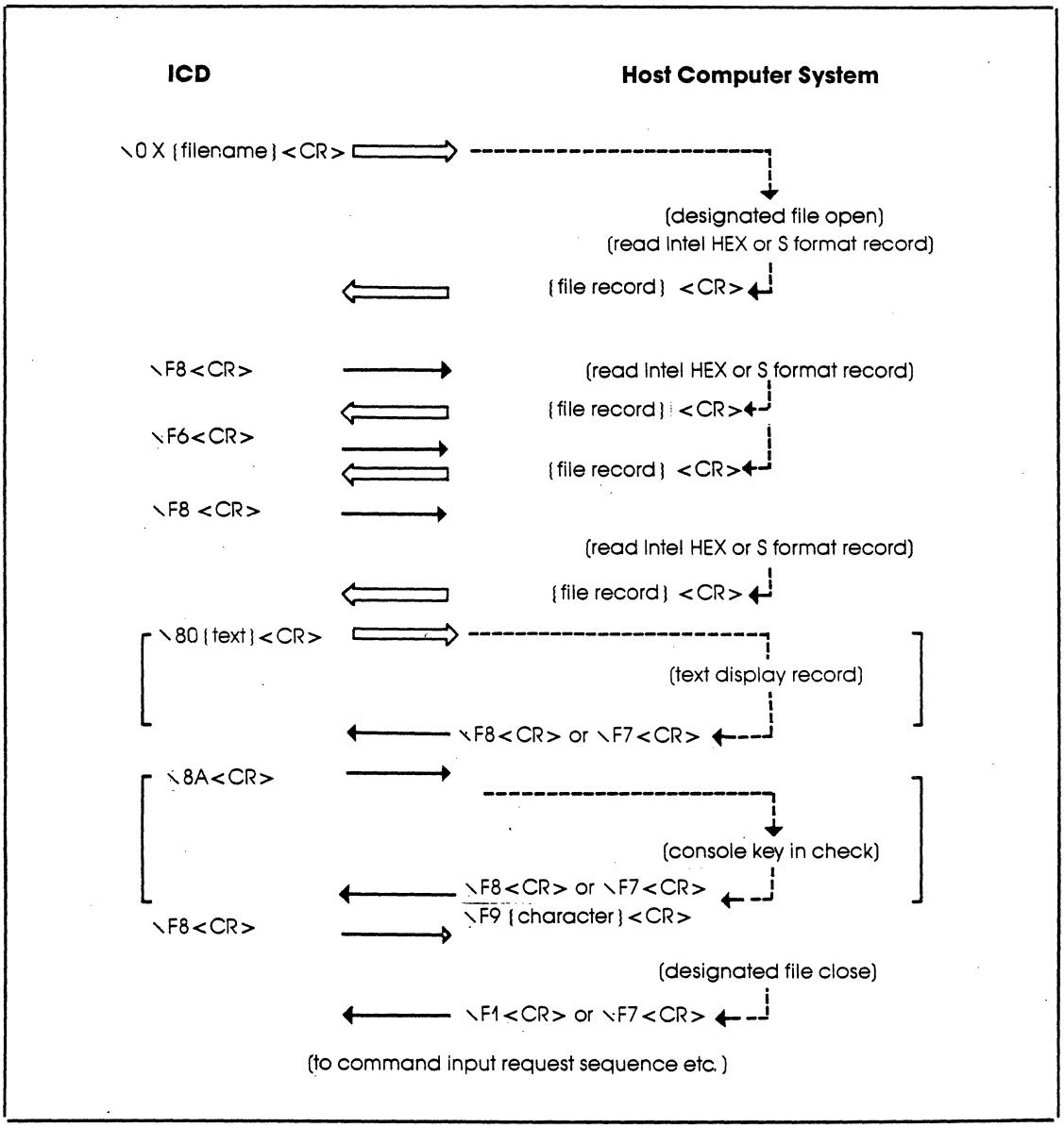
\F9{character}<CR>	CONSOLE INPUT CODE	ICD←HOST	
\F7<CR>	LOAD/ VERIFY SEQUENCE ABORT INDICATION CODE	ICD←HOST	Sent to ICD when host computer aborts object LOAD/VERIFY sequence.
\F1<CR>	LOAD/ VERIFY END CODE	ICD←HOST	Sent to ICD (after closing file) if records are exhausted.
\F7<CR>	LOAD/ VERIFY SEQUENCE ABORT INDICATION CODE	ICD←HOST	Informs ICD it is aborting LOAD/ VERIFY sequence.

Program Description: The ICD sends \00{filename}<CR> or \02{filename}<CR> to the host computer to load or verify a user program. The host computer then opens the requested program file and acts on the following:

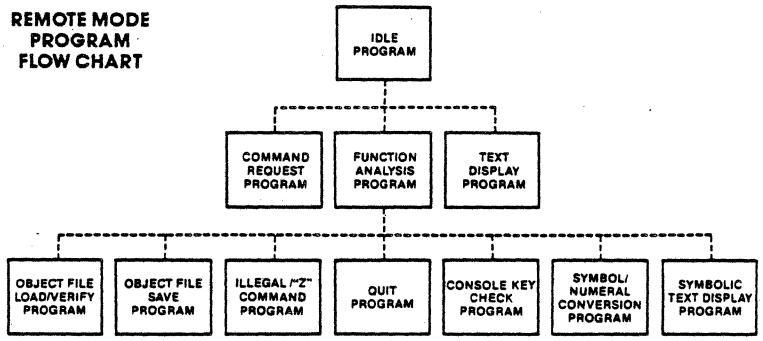
a) If an error occurs when reading the file, the host computer sends \F7<CR> to the ICD and returns to the IDLE program.

b) If no error is detected, the host computer sends the Intel Hex or S format record to the ICD and then waits for \F8<CR> from the ICD. If the host computer receives \F8<CR>, it then reads the Intel Hex or S format record. If the code is \F7<CR>, the host computer sends \F8<CR> after closing the file, and then returns to the IDLE program. If the code is \F6<CR>, the host computer waits for \F8<CR> after re-transmitting the Intel Hex or S format record to the ICD. When the text record is received from the ICD, the host computer displays the text record and then waits for \F8<CR>. If \8A<CR> is received from the ICD, the host computer sends \F8<CR> to the ICD if there is no input, or {character} when there is an input.

c) If there is no record to send when \F8<CR> is received from the ICD, the host computer closes the file, sends \F1<CR> to the ICD, and returns to the IDLE program.


ICD
Host Computer System
\0 X {filename} <CR>
(designated file open)
(read Intel HEX or S format record)
{file record} <CR>
\F8<CR>
(read Intel HEX or S format record)
{file record} <CR>
\F6<CR>
{file record} <CR>
\F8 <CR>
(read Intel HEX or S format record)
{file record} <CR>
\80 {text} <CR>
(text display record)
\F8<CR> or \F7<CR>
\8A<CR>
(console key in check)
\F8<CR> or \F7<CR>
\F9 {character} <CR>
\F8<CR>
(designated file close)
\F1<CR> or \F7<CR>
(to command input request sequence etc.)

REMOTE MODE PROGRAM FLOW CHART



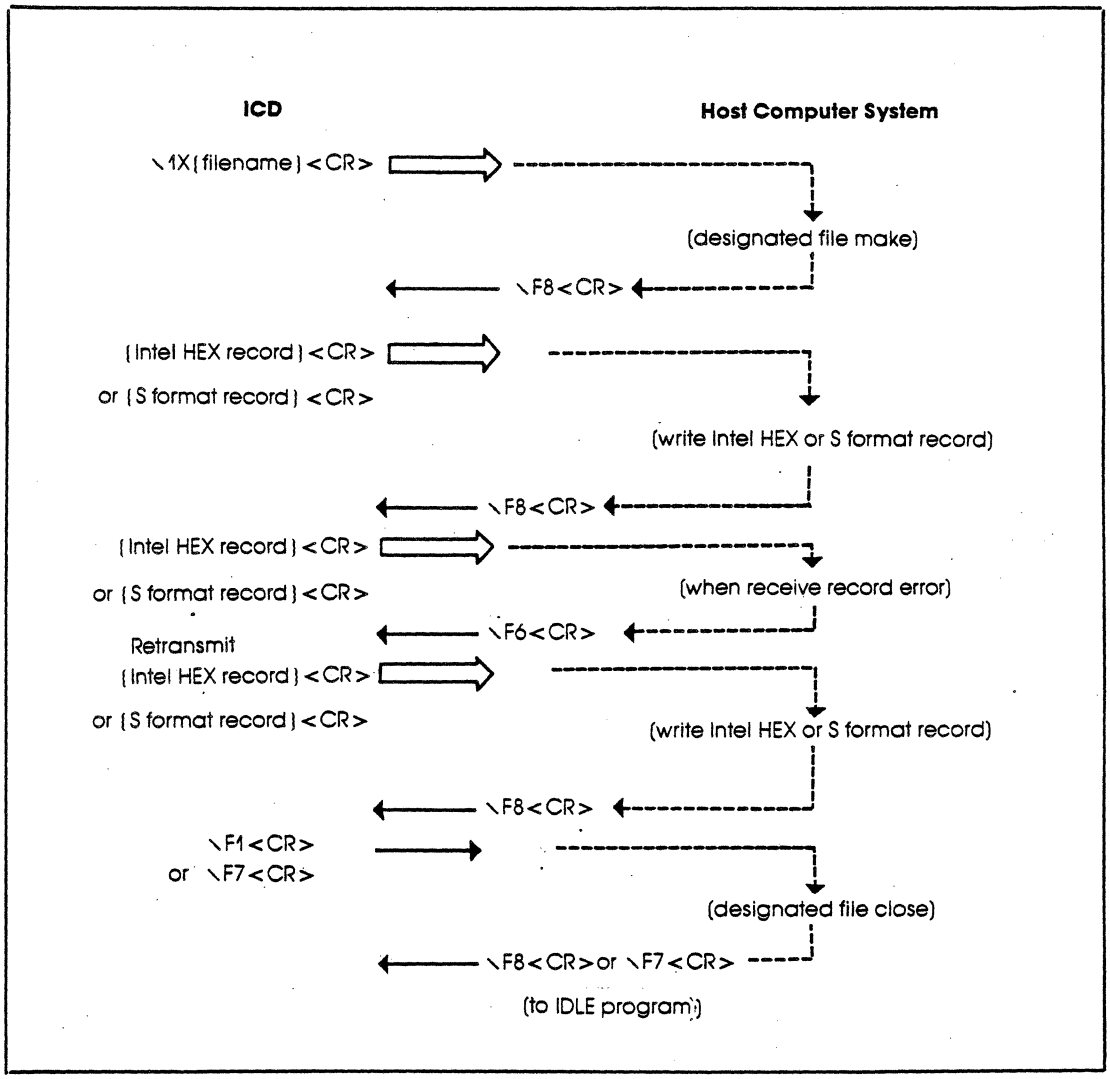
Program: OBJECT FILE SAVE		**Mode:** REMOTE	**Control:** HOST COMPUTER
Action: The host computer receives an object program and creates a file upon receiving a SAVE REQUEST from the ICD.			
Code/Record	**Code/ Record Name**	**ICD⟷HOST**	**Notes**
\10{filename} <CR> or \12 {filename} <CR>	INTEL HEX SAVE REQUEST RECORD or S FORMAT SAVE REQUEST RECORD	ICD→HOST	Sent to host computer to save a file.
\F8<CR>	OBJECT RECORD REQUEST CODE	ICD←HOST	Sent to ICD to request Intel Hex or S format record.
{record} <CR>	OBJECT FILE RECORD	ICD→HOST	
\F6<CR>	OBJECT FILE RE-TRANS-MISSION REQUEST CODE	ICD←HOST	Used when host computer requests ICD to re-transmit object file. (Most re-transmission requests are caused by a sum-check error of an Intel Hex or S format record.)

\F1<CR>	FILE END CODE	ICD→HOST	Sent to host computer when record transmission is exhausted.
\F7<CR>SAVE	ICD→HOST SEQUENCE ABORT REQUEST CODE	Directs host	computer to abort object save sequence.
\F8<CR>	FILE CLOSE END CODE	ICD←HOST	Sent to ICD in response to \F1<CR> if the file has been closed successfully, then returns to IDLE program.
\F7<CR>	SAVE SEQUENCE ABORT INDICATION CODE	ICD←HOST	Informs ICD that it is aborting the object save sequence.

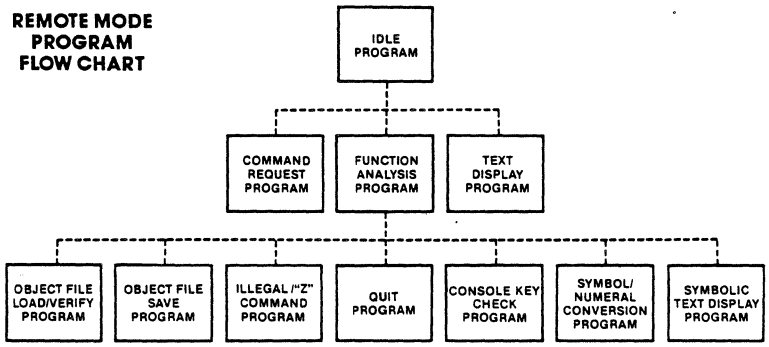
Program Description: The ICD sends \10{filename}<CR> or \12{filename}<CR> to the host computer when saving a user program. The host computer then opens the selected user file. If the file does not open, the host computer sends \F7<CR> to the ICD and returns to the IDLE program. If the file opens, the host computer sends \F8<CR> to the ICD. The host computer then waits for the Intel Hex or S format record from the ICD. If it receives \F1<CR> from the ICD, the host computer sends \F8<CR> after closing the user program file, and returns to the IDLE program.

The host computer then executes a file write of the record received from the ICD. If an error occurs during the file write operation, the host computer closes the user program, sends \F7<CR> to the ICD, and returns to the IDLE program.

If a sum-check error occurs, the host computer waits for the Intel Hex or S format record from the ICD after sending \F6<CR>. The host computer then waits for the next Intel Hex or S format record (sending \F8<CR> to the ICD) if no error occurs during the file write.


ICD
Host Computer System
\1X{filename}<CR>
(designated file make)
\F8<CR>
{Intel HEX record}<CR>
or {S format record}<CR>
(write Intel HEX or S format record)
\F8<CR>
{Intel HEX record}<CR>
or {S format record}<CR>
(when receive record error)
\F6<CR>
Retransmit
{Intel HEX record}<CR>
or {S format record}<CR>
(write Intel HEX or S format record)
\F8<CR>
\F1<CR>
or \F7<CR>
(designated file close)
\F8<CR>or \F7<CR>
(to IDLE program)

REMOTE MODE PROGRAM FLOW CHART



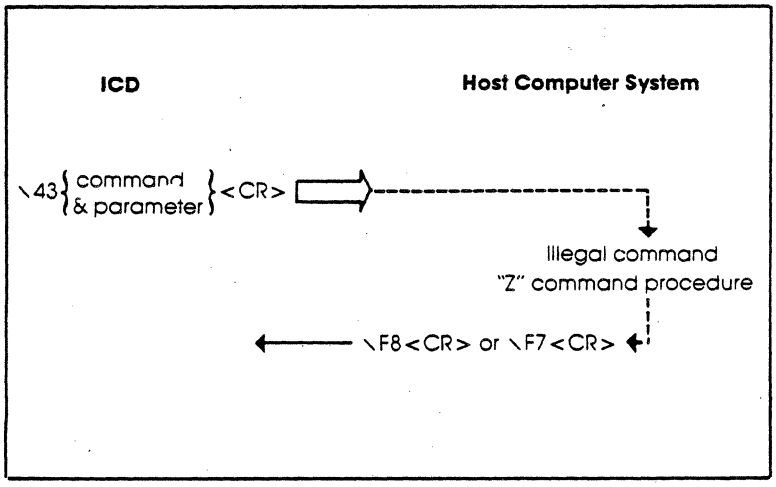
Program: ILLEGAL/"Z" COMMAND		**Mode:** REMOTE	**Control:** HOST COMPUTER
Action: This sequence is used to process an ILLEGAL or "Z" command, according to the parameters sent from the ICD. The ILLEGAL command is a command not defined in the ICD, but it is interpreted and processed by the host computer. The host computer can use the ILLEGAL and "Z" commands to process HELP or macro commands.			
Code/Record	**Code/** Record Name	**ICD⟷HOST**	Notes
\43{parameter} <CR>	ILLEGAL/ "Z" COMMAND RECORD	ICD→HOST	Sent to host computer to process the ILLEGAL/"Z" command.
\F8<CR>	ILLEGAL/ "Z" COMMAND NORMAL END CODE	ICD←HOST	Sent to ICD when ILLEGAL/"Z" command has been processed successfully.
\F7<CR>	ILLEGAL/ "Z" COMMAND ABNORMAL END CODE	ICD←HOST	Sent to ICD when ILLEGAL/"Z" command has not been processed successfully.

Program Description: The ICD sends \43{parameter}<CR> (and the specified "Z" command) to the host computer. The host computer performs the specified "Z" command and then acts on the following:

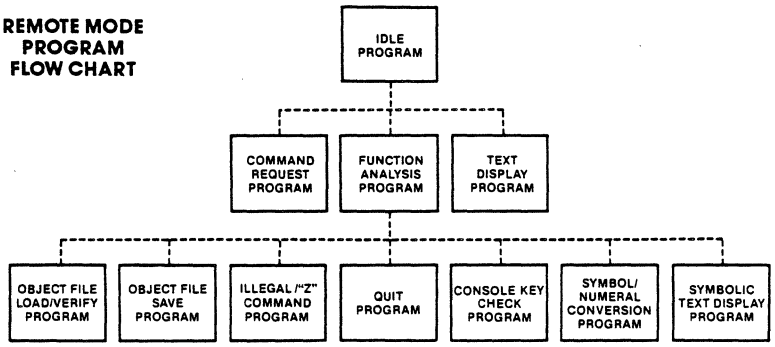
a) If an error is contained in the "Z" command specification, the host computer sends \F7<CR> to the ICD and then returns to the IDLE program.

b) If no error is detected, the host computer sends \F8<CR> to the ICD and then returns to the IDLE program.

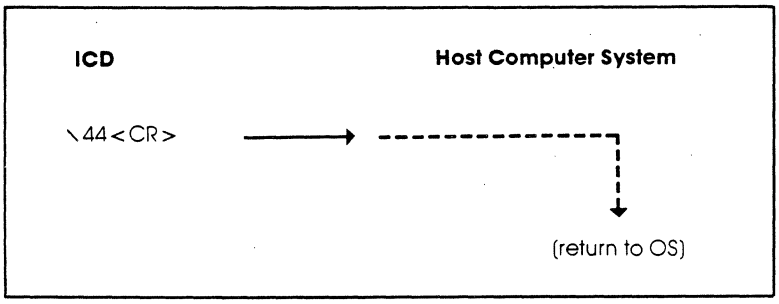
NOTE: The ICD does not react differently to the \F7<CR> code than it does to the \F8<CR> code. The ICD normally assumes that the host program has issued its own error messages if an error has occurred.



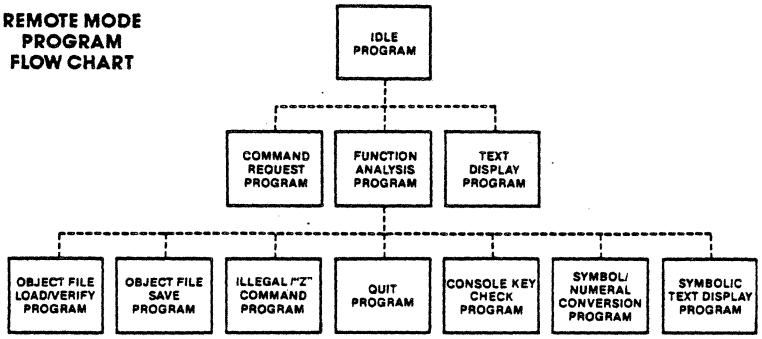
REMOTE MODE PROGRAM FLOW CHART



Program: QUIT		Mode: REMOTE	Control: HOST COMPUTER
Action: The host computer returns to the operating system (OS).			
Code/Record	**Code/ Record Name**	**ICD⟷HOST**	**Notes**
\44<CR>	QUIT CODE	ICD→HOST	Sent to host computer to exit the communications utility ZICE and returns control to the OS.



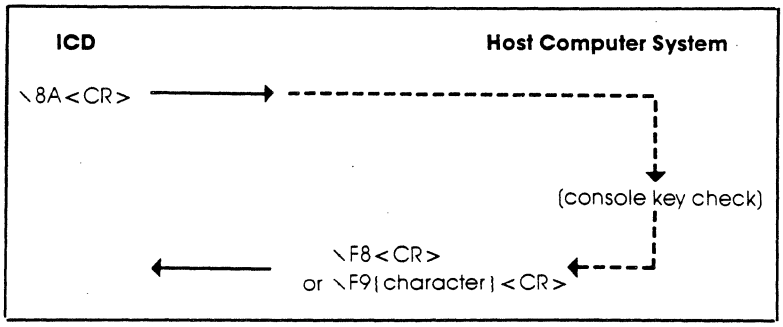
REMOTE MODE PROGRAM FLOW CHART



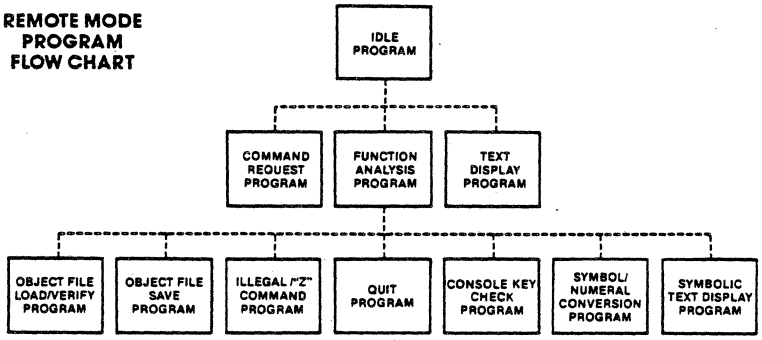
Program:	Mode:	Control:
CONSOLE KEY CHECK	REMOTE	HOST COMPUTER

Action: The ICD uses this sequence to check the console input to the host computer. (The console input could be inquiries about an interruption, restart or abort trace sequence, or dump output).

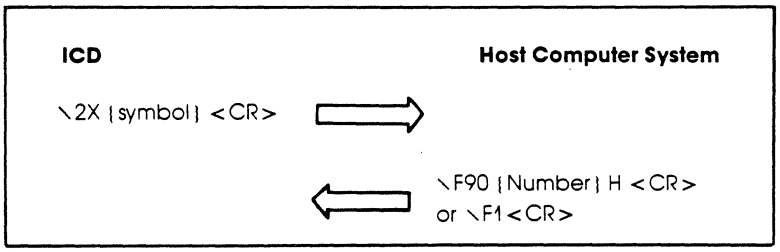
Code/Record	Code/	ICD⟷HOST	Notes
\8A<CR>	CONSOLE KEY INPUT CHECK REQUEST CODE	ICD→HOST	Sent to host computer to request a console key input check.
\F8<CR>	NO-CONSOLE-INPUT CODE	ICD←HOST	Sent to ICD if there is no console key input.
\F9[any ASCII code]<CR>	CONSOLE RECORD	ICD←HOST	Sent to ICD if there is a console key input.


ICD
Host Computer System
\8A<CR>
(console key check)
\F8<CR>
or \F9{character}<CR>

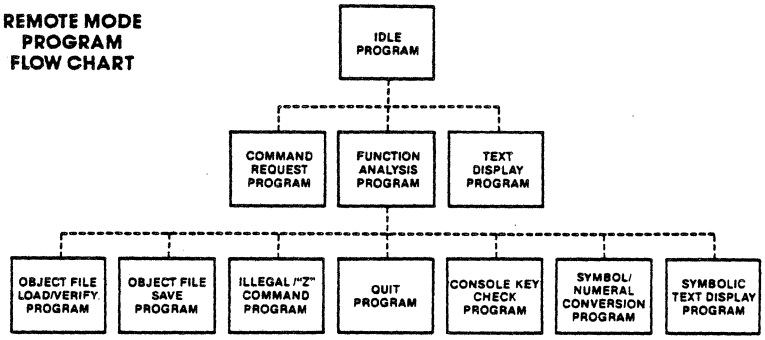
REMOTE MODE PROGRAM FLOW CHART



Program: SYMBOL/NUMERAL CONVERSION		Mode: REMOTE	Control: HOST COMPUTER
Action: This sequence is used when the host computer requests a symbol or numeral conversion.			
Code/Record	**Code/ Record Name**	**ICD⟷HOST**	**Notes**
\2X{symbol} <CR>	SYMBOL/ NUMERAL CONVERSION REQUEST RECORD	ICD→HOST	Sent to host computer to request numeric conversion of a symbol.
\F9"0"{number} (hexadecimal ASCII)"H"<CR>	NUMERIC RECORD	ICD←HOST	Sent to ICD when the symbol has been converted to a numeral. (The host computer attaches "0" to the head of the converted value and "H" at the end.)
\F1<CR>	SYMBOL/ NUMERAL CONVERSION ERROR CODE	ICD←HOST	Sent to ICD when the symbol chosen cannot be converted to a numeral.


ICD
Host Computer System
\2X {symbol} <CR>
\F90 {Number} H <CR>
or \F1 <CR>

REMOTE MODE PROGRAM FLOW CHART

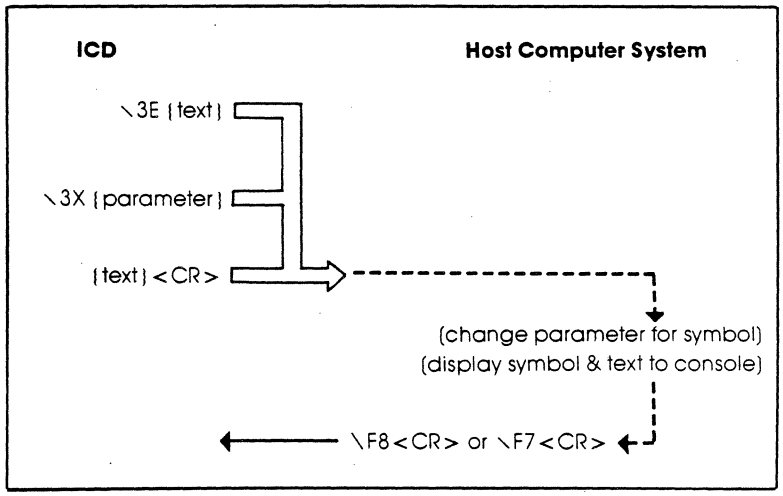


Program: SYMBOLIC TEXT DISPLAY		Mode: REMOTE	Control: HOST COMPUTER
Action: The host computer displays the parameters sent from the ICD after they are converted from numerals to symbols.			
Code/Record	**Code/ Record Name**	**ICD⟷HOST**	**Notes**
\3E{text which includes \3X {parameter}} <CR>	NUMERAL/ SYMBOL CONVERSION RECORD	ICD→HOST	Informs host computer to display the parameters after converting to symbols. (The control codes <ACK>, <NAK>, and <ENQ> are not allowed in the symbolic text record. The header 3X before {parameter} may contain values from 30 to 3F.)
\F8<CR>	DISPLAY COMPLETE CODE	ICD←HOST	Sent to ICD when the symbol display and text in the symbolic text record have been completed.
\3X{parameter} <CR>	SYMBOL CONVERSION RECORD	ICD←HOST	Informs the host computer to display the parameters after converting to symbols.

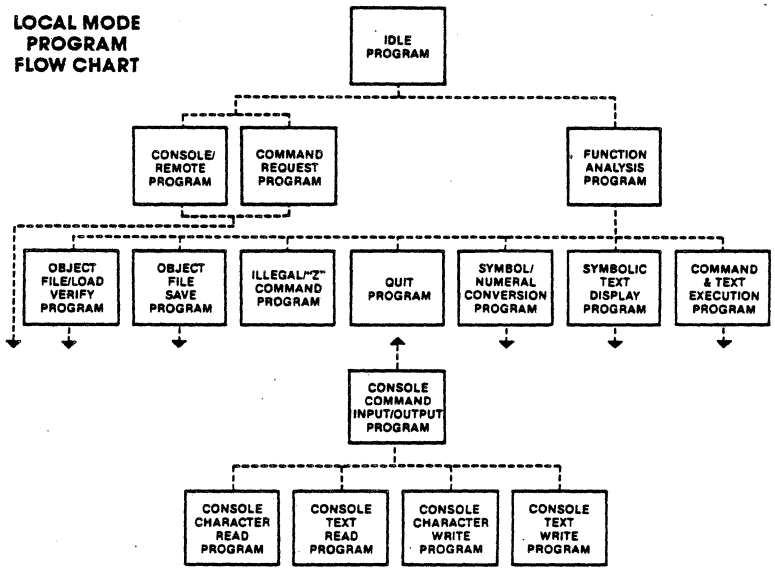
Program Description: The ICD sends \3E{text string}<CR>, which may contain one or more \3X{parameter} within the text line, to the host computer when it displays a parameter by a symbol. The host computer then enters all data before <CR> into the input buffer and acts on the following:

a) If \3X{parameter} cannot be found in the input buffer, the host computer displays the contents of the input buffer already converted to symbols, sends \F8<CR> to the ICD, and then returns to the IDLE program.

b) If \3X{parameter} is found, the host computer searches the symbol table for {parameter}. If {parameter} cannot be found in the symbol table, the system converts \3X{parameter} to {parameter}, and searches the input again. If {parameter} is found in the symbol table, the system converts \3X{parameter} to a symbol.



LOCAL MODE PROGRAM FLOW CHART

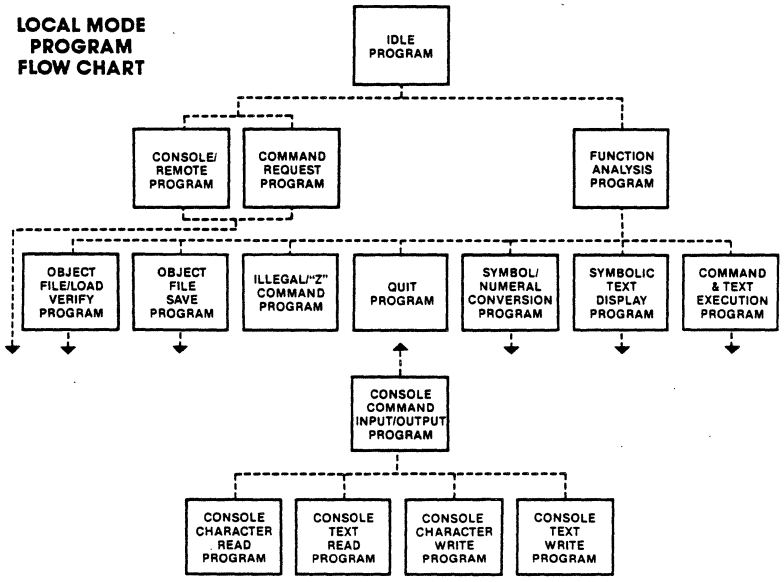


Program:	Mode:	Control:
IDLE	LOCAL	TERMINAL

Action: This program acts as the main intermediary program (transferring instructions and text only) between the ICD and the subprograms (COMMAND REQUEST and FUNCTION ANALYSIS). The host computer waits for an input from the ICD. (The host system must have an input buffer to hold the input code from the ICD.) The host computer receives one line of data and places it in the input buffer. The host computer then executes one of the following programs depending on the code it receives:

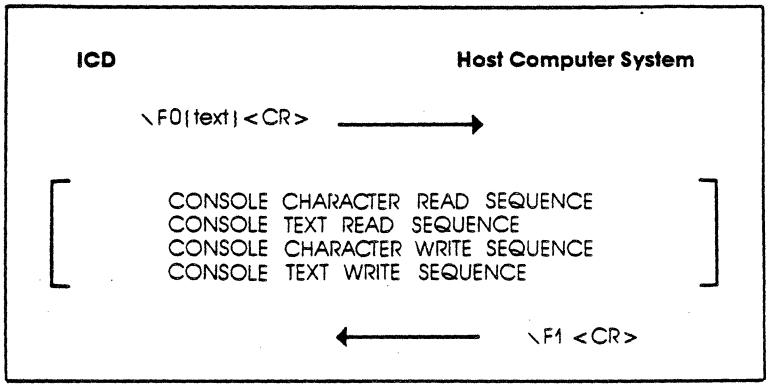
Input Buffer Contents	Program Executed
\FO{text}<CR>	COMMAND REQUEST
any other code	FUNCTION ANALYSIS

LOCAL MODE PROGRAM FLOW CHART

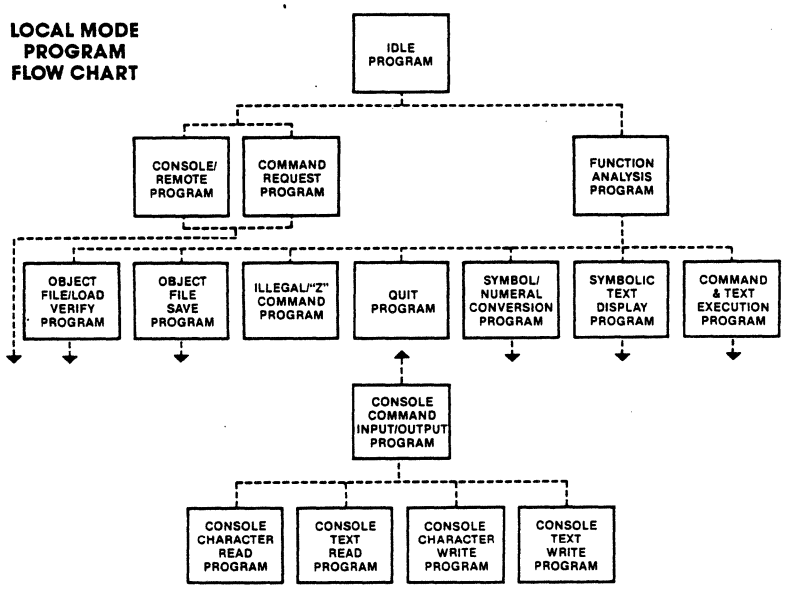


Program: COMMAND REQUEST —CONSOLE		Mode: LOCAL	Control: TERMINAL
Action: These sequences allow commands to input to the ICD through a console terminal in the LOCAL mode.			
Code/Record	**Code/ Record Name**	**ICD⟷HOST**	**Notes**
\F0{text}<CR>	COMMAND INPUT STATUS WAIT RECORD CONSOLE CHARACTER READ/WRITE SEQUENCE or CONSOLE TEXT READ/WRITE SEQUENCE	ICD→HOST	Sent to host computer before ICD displays a prompt (>). See the individual programs for a description.

\F1<CR>	CONSOLE COMMAND INPUT REQUEST CODE	ICD←HOST	ICD outputs a prompt to the console screen after receiving this code.



LOCAL MODE PROGRAM FLOW CHART

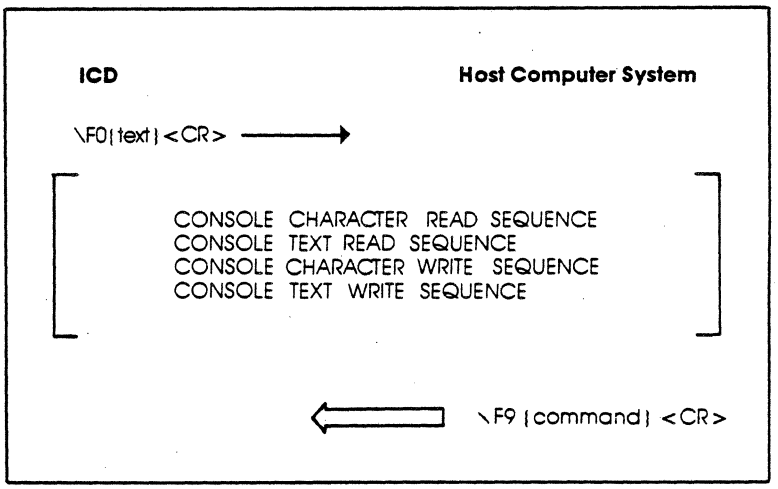


Program:		Mode:	Control:
COMMAND REQUEST —REMOTE		LOCAL	TERMINAL
Action: These sequences enable the ICD to directly execute commands in the LOCAL mode.			
Code/Record	**Code/ Record Name**	**ICD⟷HOST**	**Notes**
\F0{text}<CR>	COMMAND INPUT STATUS WAIT RECORD	ICD→HOST	Sent to host computer before ICD displays a prompt (>).
	CONSOLE CHARACTER READ/WRITE SEQUENCE or CONSOLE TEXT READ/WRITE SEQUENCE		Optional sequences. See the individual programs for a description.

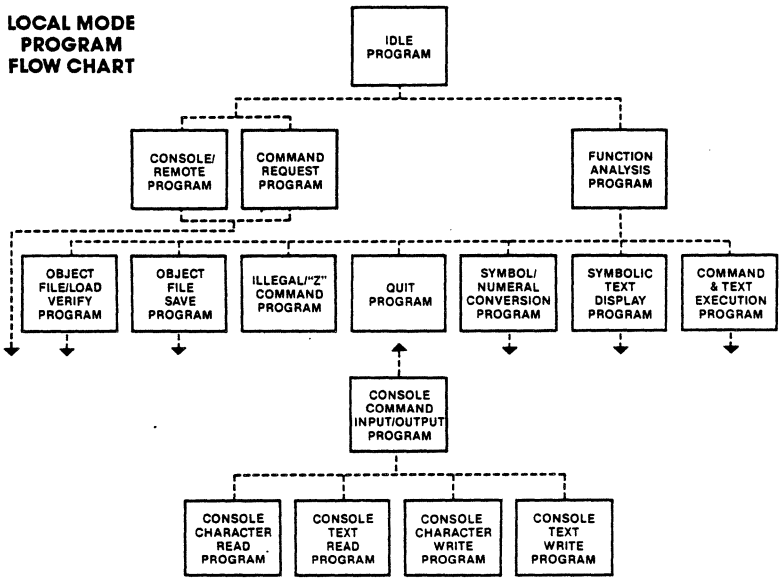
\F9{ICD command} <CR>	REMOTE COMMAND REQUEST RECORD	ICD←HOST	Sent to ICD to display a prompt and the {ICD command}.

Program Description: The ICD requests a command by sending \FO{text}<CR> to the host computer. All of the CONSOLE or REMOTE commands can be executed when the host computer receives \FO{text}<CR> from the ICD. Additionally, any of the following four console input/output sequences can be executed: 1) CONSOLE CHARACTER READ PROGRAM, 2) CONSOLE TEXT READ PROGRAM, 3) CONSOLE CHARACTER WRITE PROGRAM, and 4) CONSOLE TEXT WRITE PROGRAM.

If CONSOLE commands are used, the sequence ends with \F1<CR>, and the host computer returns to the IDLE program. If REMOTE commands are used, the sequence ends with \F9{ICD command}<CR>, and the host computer returns to the IDLE program.



LOCAL MODE PROGRAM FLOW CHART

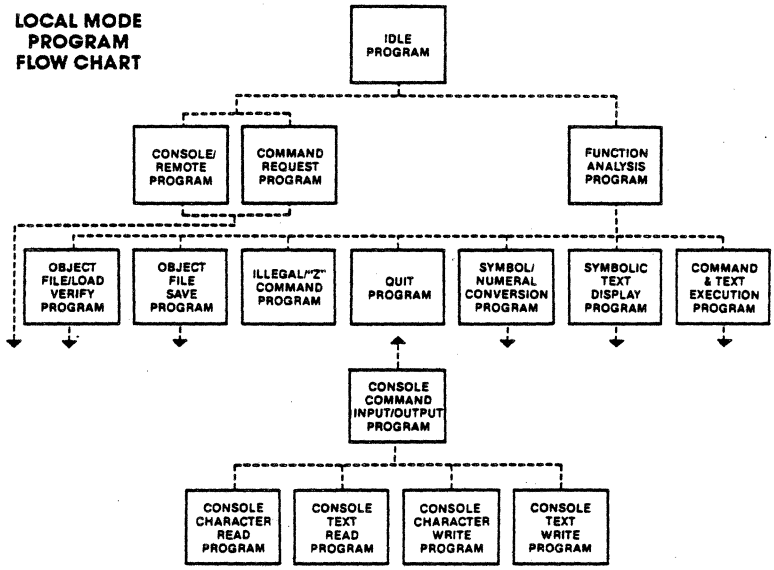


Program:	Control:
FUNCTION ANALYSIS	LOCAL TERMINAL

Action: The host computer places one line of data (received from the ICD) into the input buffer and then analyzes the data. The host computer then executes one of the following programs based on the contents of the input buffer:

Input Buffer Contents	Program Executed
\{filename}<CR> or \02{filename}<CR>	FILE LOAD
\01{filename}<CR> or \03{filename}<CR>	FILE VERIFY
\10{filename}<CR> or \12{filename}<CR>	FILE SAVE
\43{parameter}<CR>	"Z" COMMAND
\44<CR>	QUIT
\2X{symbol}<CR>	SYMBOL CONVERSION
\3X{parameter}{text}<CR>	SYMBOLIC TEXT DISPLAY

LOCAL MODE PROGRAM FLOW CHART



Program: OBJECT FILE LOAD/VERIFY		**Mode:** LOCAL	**Control:** TERMINAL
Action: An object file is sent from the host computer in response to a LOAD/VERIFY request from the ICD.			
Code/Record	**Code/ Record Name**	**ICD⟷HOST**	**Notes**
\00{filename} <CR> or \02 {filename} <CR>	INTEL HEX or S FORMAT LOAD REQUEST RECORD	ICD→HOST	Sent to host computer when ICD loads an object file. (The {filename} field may be used for the user-defined load message.)
\01{filename} or \03{filename} <CR>	INTEL HEX or S FORMAT VERIFY REQUEST RECORD	ICD→HOST	Sent to host computer when ICD verifies an object file with the memory. The {filename} field may be used for the user-defined verify message.

{record} <CR>	OBJECT FILE RECORD	ICD←HOST	Intel Hex or S format record sent to ICD from host computer. Record may not contain any control code, and must end with <CR>.
\F8<CR>	OBJECT FILE REQUEST CODE	ICD→HOST	Sent to host computer to request an Intel Hex or S format record.
\F6<CR>	OBJECT FILE RE-TRANS-MISSION REQUEST CODE	ICD→HOST	Used when ICD requests host computer to re-transmit object file. (Most re-transmission requests are caused by an error occurring in the sum check of the Intel Hex or S format record.)
\F7<CR>	OBJECT FILE TRANSMIS-SION REQUEST CODE	ICD→HOST	Sent to host computer to stop the LOAD/VERIFY sequence.
\80{text}<CR>	TEXT RECORD	ICD→HOST	Usually contains a verify error message.
	CONSOLE CHARACTER READ/WRITE SEQUENCE OR CONSOLE TEXT READ/WRITE SEQUENCE		Optional sequences. See individual programs for description.

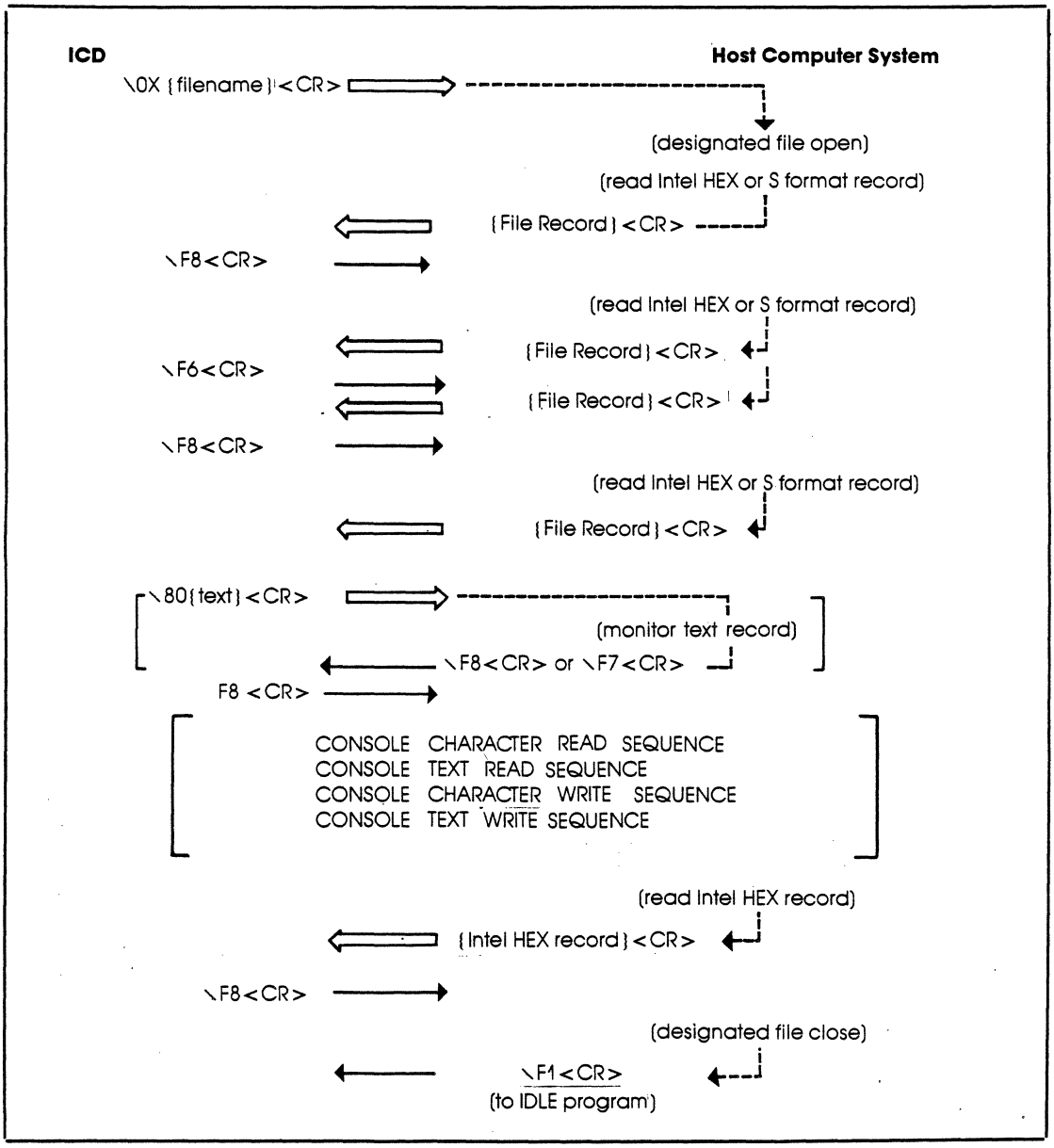
\F1<CR>	OBJECT FILE ICD←HOST LOAD/ VERIFY END CODE	Sent to ICD (after closing the file) if the records are exhausted.
\F7<CR>	OBJECT FILE ICD←HOST LOAD/ VERIFY ABORT INDICATION CODE	Sent to ICD when host computer aborts the Object Load/ Verify sequence.

Program Description: The ICD sends \0X{filename}<CR> too the host computer to load or verify a user program. The host computer opens the requested program file, reads the Intel Hex or S format records from the file, and acts on the following:

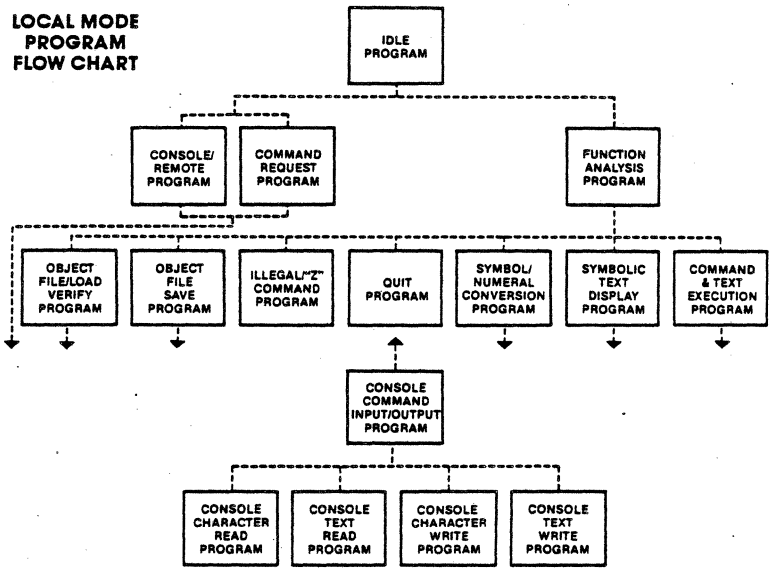
a) If an error occurs when reading the file, the host computer sends \F7<CR> to the ICD, and returns to the IDLE program.

b) If no error occurs, the host computer sends the Intel Hex or S format record to the ICD and thenwaits for \F8<CR> from the ICD. If the host computer receives \F8<CR>, it reads the Intel Hex or S format record. If the code is \F7<CR>, the host computer sends \F8<CR> after closing the file, and then returns to the IDLE program. If the code is \F6<CR>, the host computer waits for \F8<CR>, after re-transmitting the Intel Hex or S format record to the ICD.

c) If there is no record to send when \F8<CR> is received from the ICD, the host computer closes the file, sends \F1<CR> to the ICD, and then returns to the IDLE program.


ICD
Host Computer System
\0X {filename}<CR>
(designated file open)
(read Intel HEX or S format record)
{File Record}<CR>
\F8<CR>
(read Intel HEX or S format record)
{File Record}<CR>
\F6<CR>
{File Record}<CR>
\F8<CR>
(read Intel HEX or S format record)
{File Record}<CR>
\80{text}<CR>
(monitor text record)
\F8<CR> or \F7<CR>
F8 <CR>
CONSOLE CHARACTER READ SEQUENCE
CONSOLE TEXT READ SEQUENCE
CONSOLE CHARACTER WRITE SEQUENCE
CONSOLE TEXT WRITE SEQUENCE
(read Intel HEX record)
{Intel HEX record}<CR>
\F8<CR>
(designated file close)
\F1<CR>
(to IDLE program)

LOCAL MODE PROGRAM FLOW CHART



Program: OBJECT FILE SAVE		**Mode:** LOCAL	**Control:** TERMINAL
Action: The host computer receives an object program and creates a file.			
Code/Record	**Code/ Record Name**	**ICD⟷HOST**	**Notes**
\10{filename} <CR> or \12 {filename} <CR>	INTEL HEX OR S FORMAT REQUEST RECORD	ICD→HOST	Sent to host computer to request a file save. The {filename} field may be used for a user-defined save message.
\F8<CR> (at file write)	OBJECT FILE REQUEST CODE	ICD←HOST	Sent to ICD to request an Intel Hex or S format record.

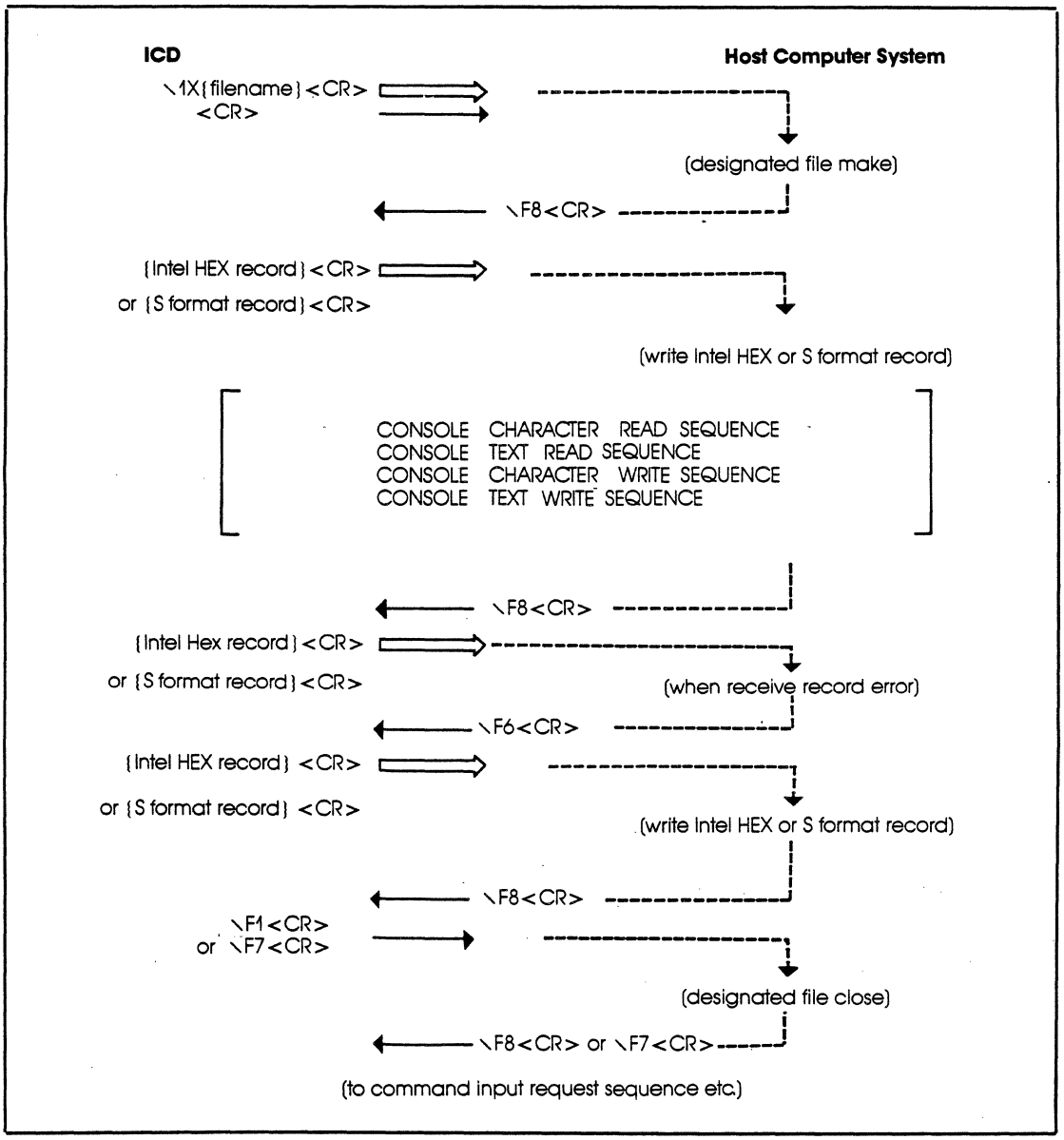
{record}<CR>	OBJECT FILE RECORD	ICD→HOST	Intel Hex or S format record sent to host computer.
	CONSOLE CHARACTER READ/WRITE PROGRAM OR CONSOLE TEXT READ/WRITE PROGRAM		Optional programs. See the individual programs for a description.
\F8<CR>	OBJECT FILE REQUEST CODE	ICD←HOST	Sent to ICD to request a record.
\F6<CR>	OBJECT FILE RE-TRANS-MISSION REQUEST CODE	ICD←HOST	Requests ICD to re-transmit an object code.
\F1<CR>	OBJECT FILE END CODE	ICD→HOST	Sent to host computer when the transmission of the records have been exhausted. Host computer ends the object save sequence by sending \F8<CR> after closing the file.
\F7<CR>	SAVE SEQUENCE ABORT INDICATION CODE	ICD←HOST	Host computer uses this code to inform ICD it is aborting the object save sequence.

\F8<CR> (at file close)	FILE CLOSE END CODE	ICD←HOST	Sent to ICD when the file close is successful.
\F7<CR>	SAVE SEQUENCE ABORT INDICATION CODE	ICD←HOST	Indicates that host computer has stopped the object save sequence.

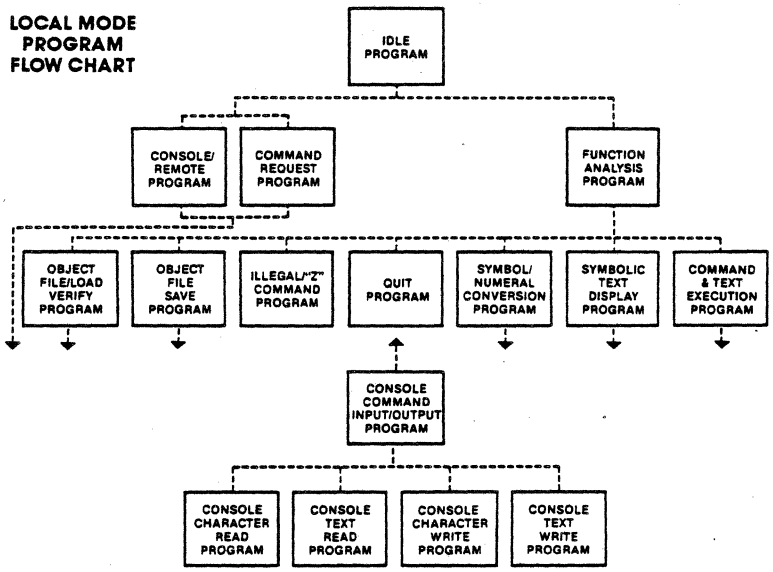
Program Description: The ICD sends \10{filename}<CR> or \12{filename}<CR> to the host computer when saving a user program. The host computer then opens the selected user file. If the file does not open, the host computer sends \F7<CR> to the ICD and returns to the IDLE program. If the file opens, the host computer sends \F8<CR> to the ICD.

The host computer then waits for an Intel Hex or S format record from the ICD. If it receives \F1<CR> from the ICD, the host computer sends \F8<CR>, after closing the user program file, and returns to the IDLE program. The host computer then executes a file write of the record received from the ICD. If an error occurs during the file write operation, the host computer closes the user program, sends \F7<CR> to the ICD, and returns to the IDLE program.

If an error occurs in a sum check, the host computer sends \F6<CR> and waits for the Intel Hex or S format record from the ICD. The host computer then waits for the next Intel Hex or S format record (sending \F8<CR> to the ICD) if no error occurs during the file write.


ICD
Host Computer System
\1X{filename}<CR>
<CR>
(designated file make)
\F8<CR>
{Intel HEX record}<CR>
or {S format record}<CR>
(write Intel HEX or S format record)
CONSOLE CHARACTER READ SEQUENCE
CONSOLE TEXT READ SEQUENCE
CONSOLE CHARACTER WRITE SEQUENCE
CONSOLE TEXT WRITE SEQUENCE
\F8<CR>
{Intel Hex record}<CR>
or {S format record}<CR>
(when receive record error)
\F6<CR>
{Intel HEX record} <CR>
or {S format record} <CR>
(write Intel HEX or S format record)
\F8<CR>
\F1<CR>
or \F7<CR>
(designated file close)
\F8<CR> or \F7<CR>
(to command input request sequence etc.)

LOCAL MODE PROGRAM FLOW CHART



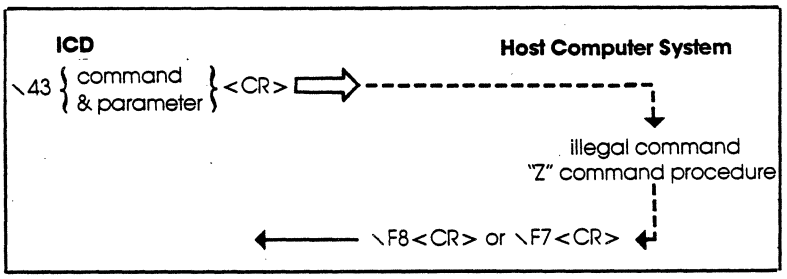
Program: ILLEGAL/"Z" COMMAND		**Mode:** LOCAL	**Control:** TERMINAL
Action: This sequence is used to process an ILLEGAL command or "Z" command according to the parameters sent from the ICD. The ILLEGAL command is a command not defined in the ICD, but is interpreted and processed by the host computer.			
Code/Record	**Code/ Record Name**	**ICD⟷HOST**	**Notes**
\43{parameter} <CR>	ILLEGAL COMMAND/ "Z" COMMAND RECORD	ICD→HOST	Sent to host computer to process the ILLEGAL/"Z" command.

\F8<CR>	ILLEGAL COMMAND/ "Z" COMMAND NORMAL END CODE	ICD←HOST	Sent to ICD when the ILLEGAL/"Z" command has been processed successfully.
\F7<CR>	ILLEGAL COMMAND/ "Z" COMMAND ABNORMAL END CODE	ICD←HOST	Sent to ICD when the ILLEGAL/"Z" command has not been processed successfully.

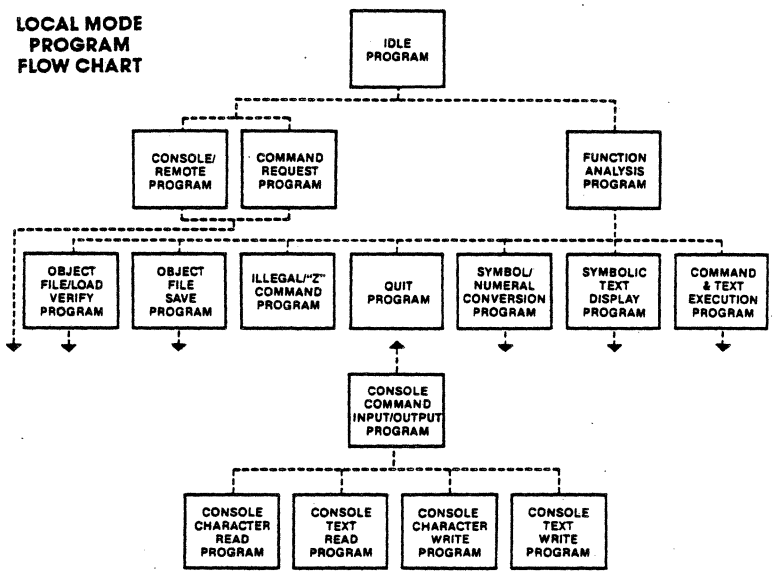
Program Description: The ICD sends \43{parameter}<CR> (and the specified "Z" command) to the host computer. The host computer then performs the specified "Z" command and acts on the following:

a) If an error is contained in the "Z" command specification, the host computer sends \F7<CR> to the ICD and then returns to the IDLE program.

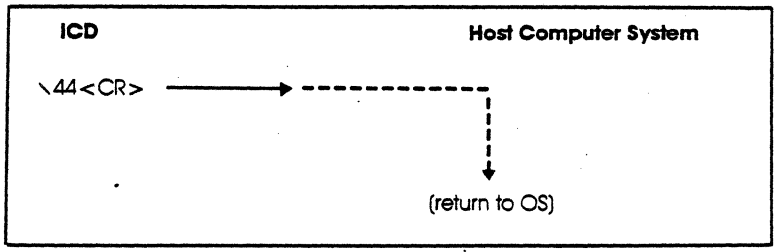
b) If no error is detected, the host computer sends \F8<CR> to the ICD and then returns to the IDLE program.



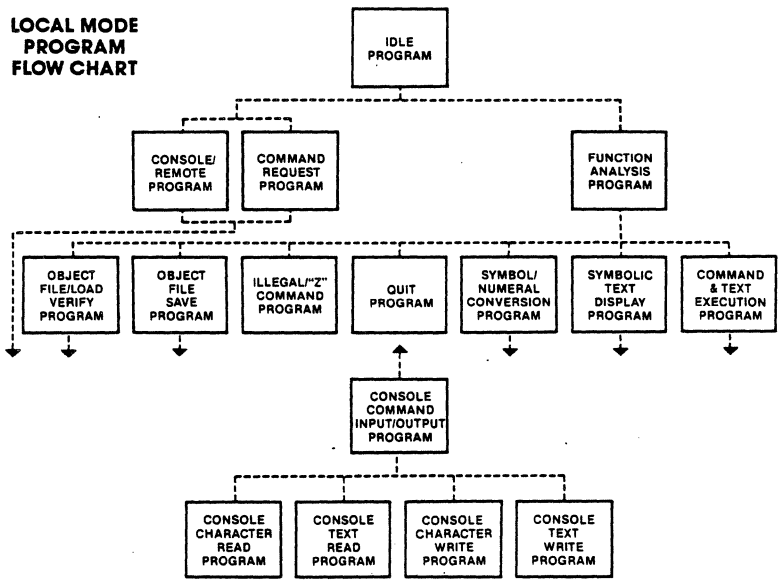
LOCAL MODE PROGRAM FLOW CHART



Program:	Mode:	Control:
QUIT	LOCAL	TERMINAL
Action: The host computer returns to the operating system (OS). (The ICD HOST ON mode is also cancelled.)		
Code/Record	**Code/ Record Name**	**ICD⟷HOST** **Notes**
\44<CR>	QUIT CODE	ICD→HOST



LOCAL MODE PROGRAM FLOW CHART



Program: SYMBOL/NUMERAL CONVERSION		**Mode:** LOCAL	**Control:** TERMINAL
Action: This sequence is used when the host computer requests a symbol/numeral conversion.			
Code/Record	**Code/ Record Name**	**ICD←→HOST**	**Notes**
\2X{symbol} <CR>	SYMBOL/ NUMERAL CONVERSION REQUEST RECORD	ICD→HOST	Sent to host computer requesting numeric conversion of a symbol.
	CONSOLE CHARACTER READ/WRITE SEQUENCE OR CONSOLE TEXT READ/WRITE SEQUENCE		Optional sequences. See the individual programs for a description.

\F9"0"{number} (hexadecimal ASCII)}"H" <CR>	NUMERIC RECORD	ICD←HOST	Sent to ICD when symbol received has been successfully converted to a numeral. (The host computer attaches "0" to the head of the converted value and "H"<CR> at the end.)
\F1<CR>	SYMBOL/ NUMERAL CONVERSION ERROR CODE	ICD←HOST	Sent to ICD when symbol chosen cannot be converted to a numeral.

SYMBOL CONVERSION Program Description: The ICD sends \20{symbol}<CR> to the host computer when the symbol/number conversion is executed. The host computer then searches the symbol table for the {symbol} received from the ICD to convert to a numeral, and acts on the following:

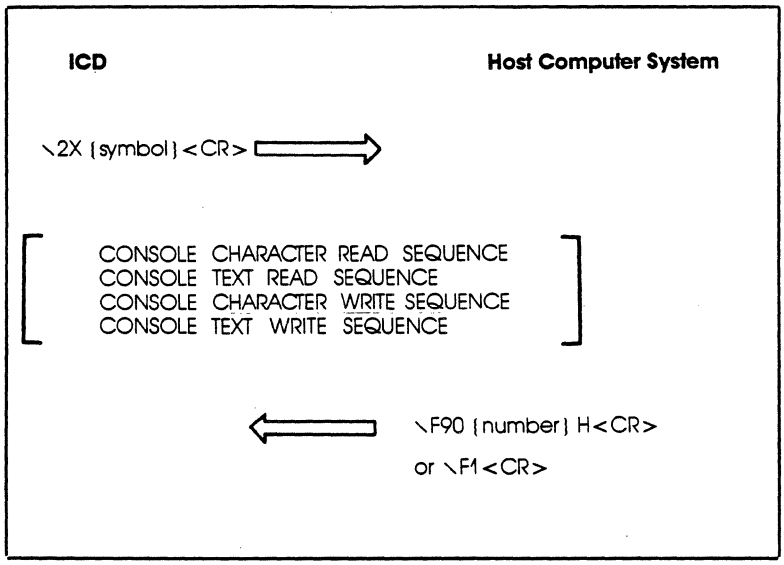
a) If the conversion is successful, the host computer sends the numeral to the ICD and then returns to the IDLE program.

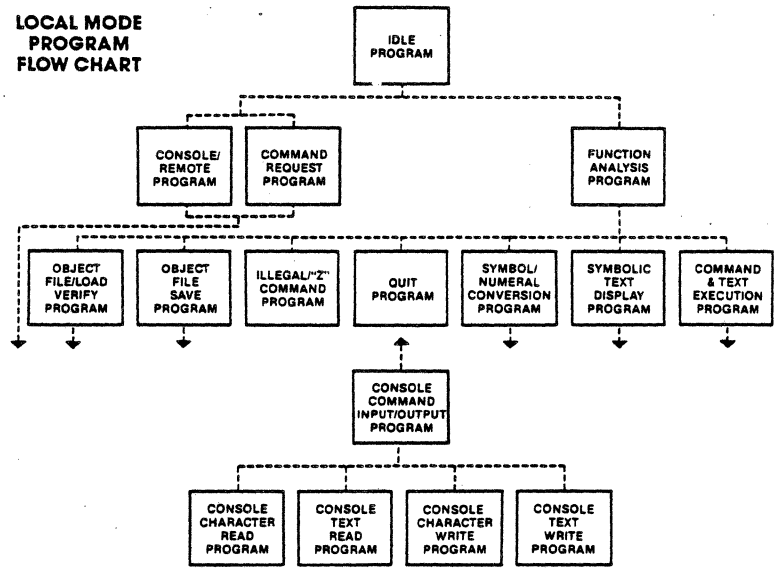
b) If the conversion is unsuccessful, the host computer sends \F1<CR> to the ICD and returns to the IDLE program.

NUMERAL CONVERSION Program Description: The ICD sends \3E{text which includes \3X {parameter}}<CR> to the host computer when the numeral/symbol conversion program is executed. The host computer enters all data before <CR> into the ICD input buffer. The host computer then searches the input buffer for \3X{parameter} and executes one of the following:

a) If \3X{parameter} is not found, the host computer sends out the text, attaching \80 to the front and <CR> at the end of the text, and then waits for \F8<CR> from the ICD. When \F8<CR> is received from the ICD, the host computer sends \F1 to the ICD and returns to the IDLE program.

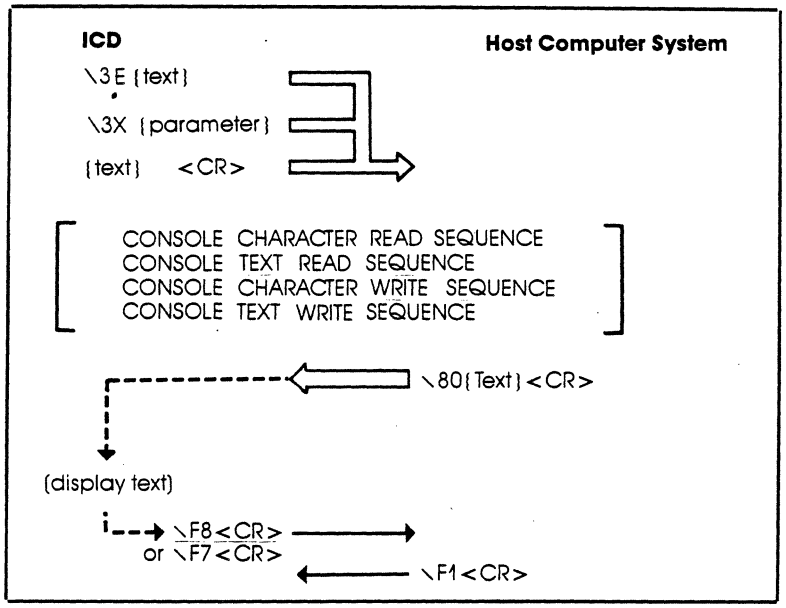
b) If \3X{parameter} is found, the host computer searches the symbol table for {parameter}. If {parameter} is not found in the symbol table, the system converts \3X{parameter} to {parameter}, and searches the input again. If {parameter} is found in the symbol table, the system converts \3X{parameter} to a symbol.


ICD
Host Computer System
\2X {symbol} <CR>
CONSOLE CHARACTER READ SEQUENCE
CONSOLE TEXT READ SEQUENCE
CONSOLE CHARACTER WRITE SEQUENCE
CONSOLE TEXT WRITE SEQUENCE
\F90 {number} H<CR>
or \F1<CR>

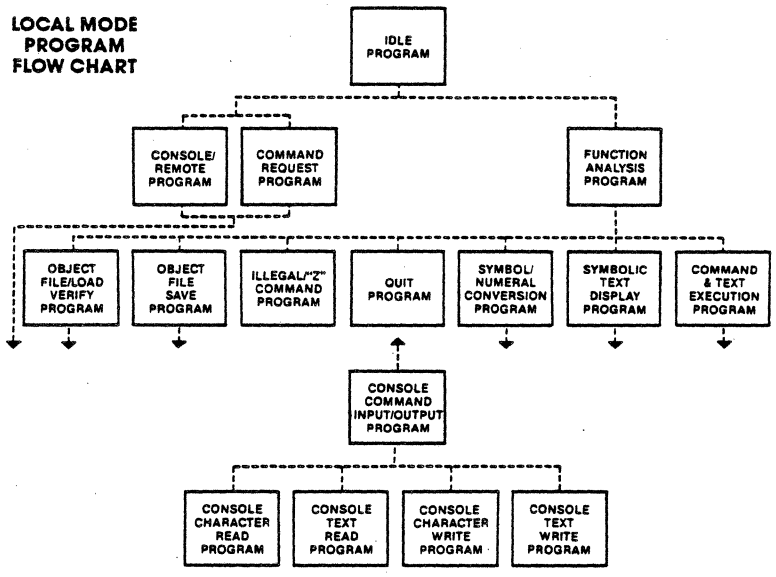


Program: SYMBOLIC TEXT DISPLAY		**Mode:** LOCAL	**Control:** TERMINAL
Action: The host computer displays the parameters sent from the ICD after being converted from numerals to symbols.			
Code/Record	**Code/ Record Name**	**ICD⟷HOST**	**Notes**
\3E{text which includes \3X} <CR>	NUMERAL/ SYMBOL CONVERSION RECORD	ICD→HOST	Informs host computer to convert a numeral to a symbol. (\3X is a header.)
	CONSOLE CHARACTER READ/WRITE SEQUENCE OR CONSOLE TEXT READ/WRITE SEQUENCE		Optional sequences. See the individual programs for a description.

\80{text}{change to symbol}<CR>	NUMERAL/ SYMBOL RECORD	ICD←HOST	Sent to ICD if the numeral is successfully converted to a symbol.
\F8<CR>	DISPLAY END CODE	ICD→HOST	Sent to host computer when the symbol display and text in the symbolic text record have been completed.
\F1<CR>	NUMERAL/ SYMBOL CONVERSION END CODE	ICD←HOST	Sent to ICD to end the sequence.



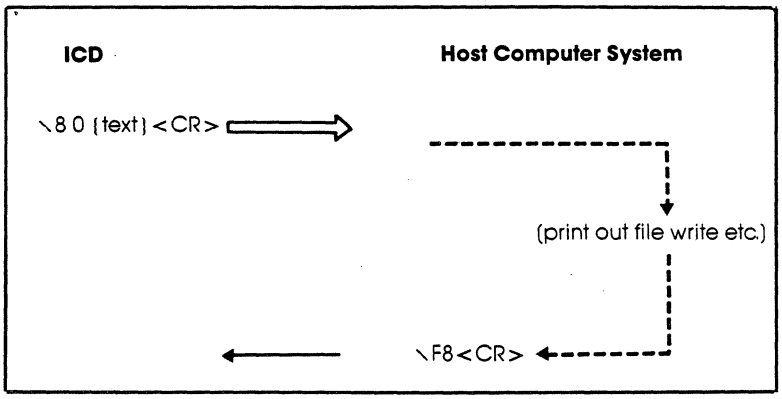
LOCAL MODE PROGRAM FLOW CHART



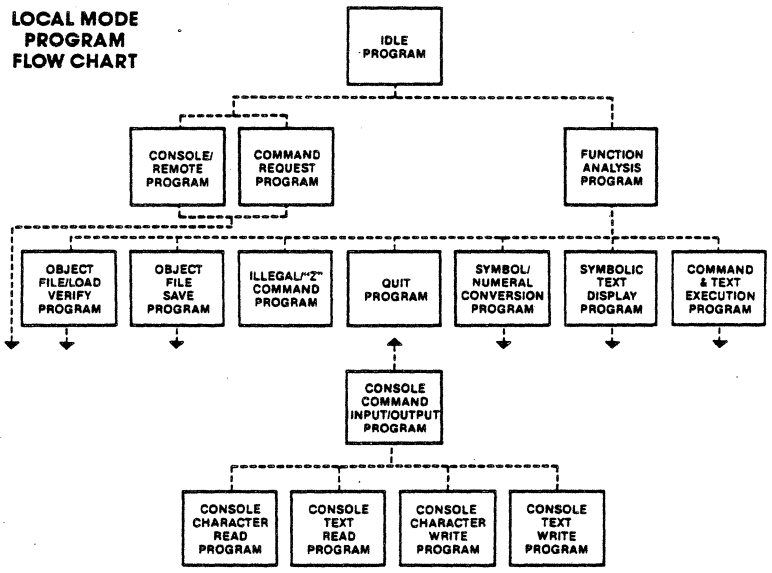
Program: COMMAND & TEXT EXECUTION	Mode: LOCAL	Control: TERMINAL

Action: The ICD outputs the command and the result of its execution to the host computer. The host computer can then output the text to a printer or onto a file. NOTE: In the LOCAL mode, the PRINT ON command is ignored after the HOST ON command is issued.

Code/Record	Code/ Record Name	ICD⟷HOST	Notes
\80{text}<CR>	COMMAND EXECUTION TEXT	ICD→HOST	Outputs one line of text after the command execution.
\F8<CR>	TEXT RECEPTION COMPLETE CODE	ICD←HOST	Transmitted to ICD when host computer has received the text and completed the output execution.


ICD
Host Computer System
\8 0 {text} <CR>
(print out file write etc.)
\F8<CR>

LOCAL MODE PROGRAM FLOW CHART



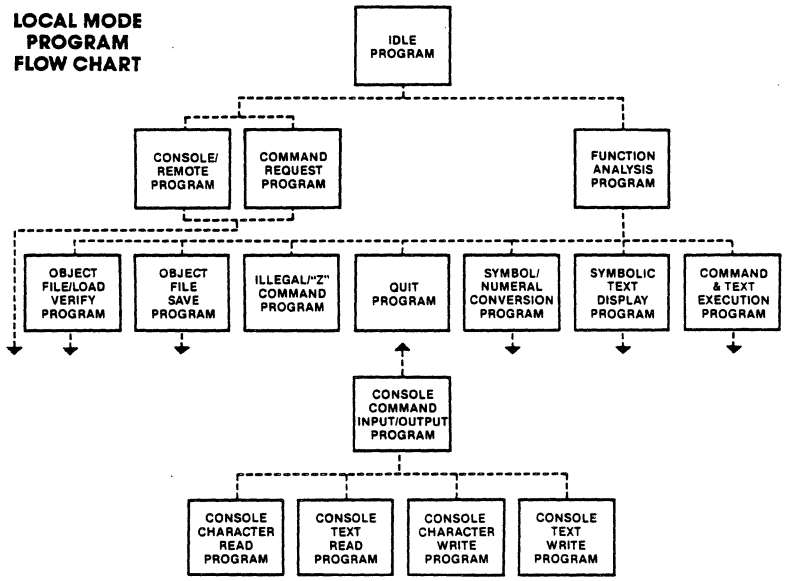
Program:	Mode:	Control:
CONSOLE COMMAND INPUT/OUTPUT	LOCAL	TERMINAL

Action: There are four input/output sequences available when the ICD operates in the LOCAL mode:

1) CONSOLE CHARACTER READ
2) CONSOLE TEXT READ
3) CONSOLE CHARACTER WRITE
4) CONSOLE TEXT WRITE

These sequences can only be used in combination with the CONSOLE COMMAND SEQUENCE, REMOTE COMMAND SEQUENCE, or OBJECT FILE SAVE SEQUENCE.

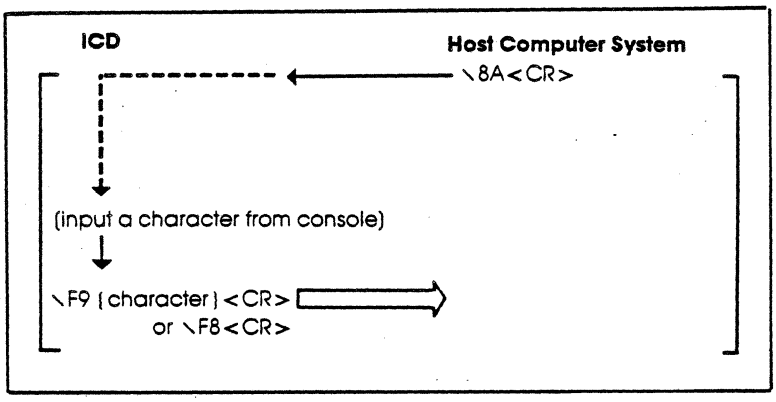
LOCAL MODE PROGRAM FLOW CHART



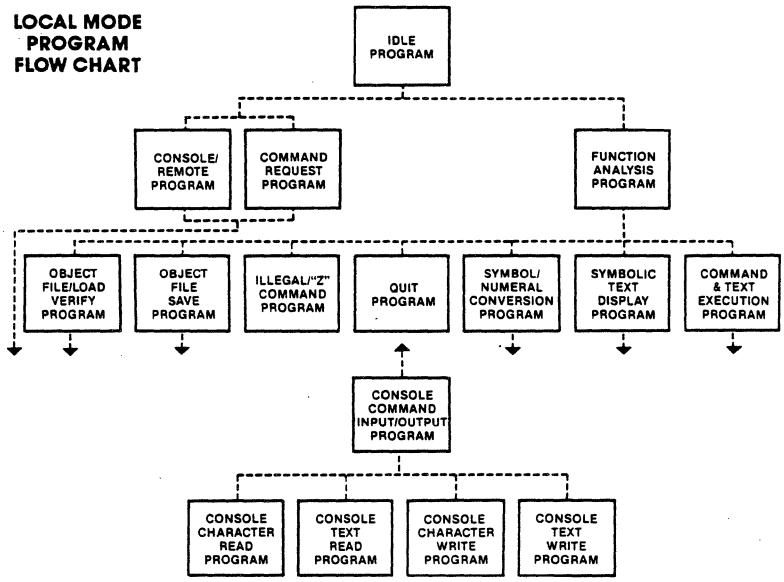
Program:	Mode:	Control:
CONSOLE CHARACTER READ	LOCAL	TERMINAL
Action: The host computer uses this sequence to request a single character from the console through the ICD.		

Code/Record	Code/ Record Name	ICD⟷HOST	Notes
\8A<CR>	CONSOLE KEY INPUT REQUEST CODE	ICD←HOST	

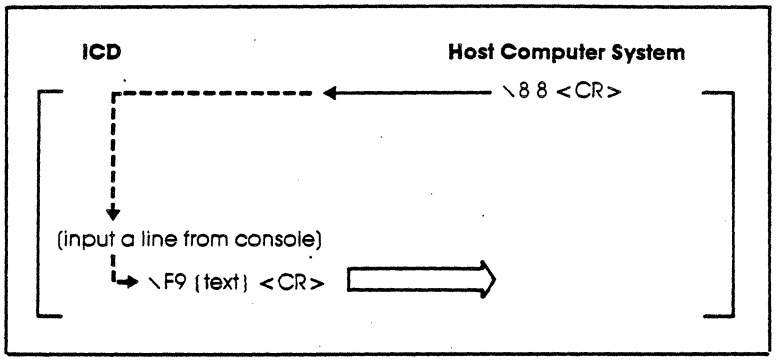
\F9{input character} <CR>	CONSOLE INPUT RECORD	ICD→HOST	Sent to host computer if there is an input character. The input character and <CR> are then sent to the host computer. (The ICD does not echo the console input.)
\F8<CR>	NO CONSOLE KEY INPUT CODE	ICD→HOST	Sent to host computer if there is no console input.



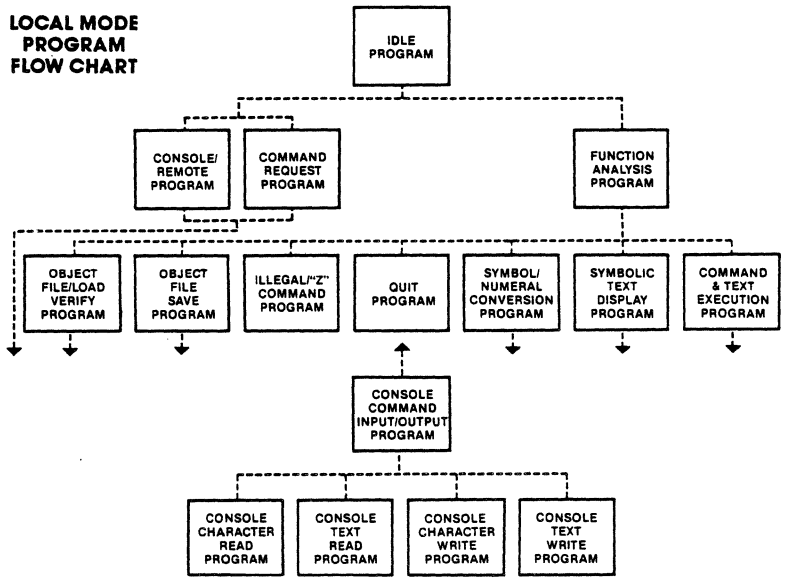
LOCAL MODE PROGRAM FLOW CHART



Program: CONSOLE TEXT READ		Mode: LOCAL	Control: TERMINAL
Action: This sequence is used when the host computer requests the ICD to input one line of data.			
Code/Record	**Code/ Record Name**	**ICD⟷HOST**	**Notes**
\88<CR>	DATA INPUT REQUEST CODE	ICD←HOST	Sent to ICD to request one line of data.
\F9{line of data}<CR>	DATA INPUT RECORD	ICD→HOST	Sent to host computer along with the line of data entered from the console terminal. The maximum number of input characters is 255; subsequent characters are discarded.


ICD
Host Computer System
\8 8 <CR>
(input a line from console)
\F9 {text} <CR>

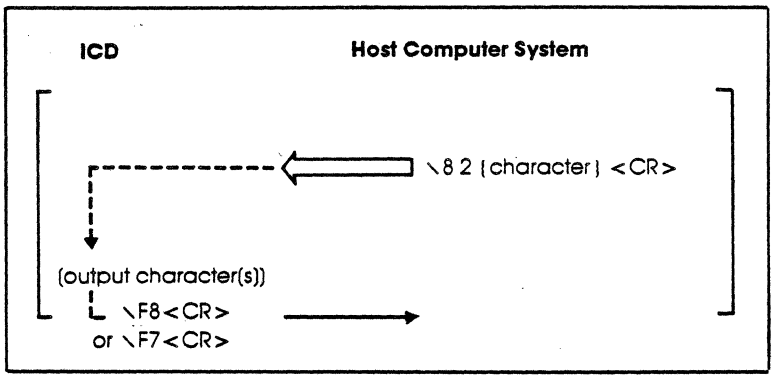
LOCAL MODE PROGRAM FLOW CHART



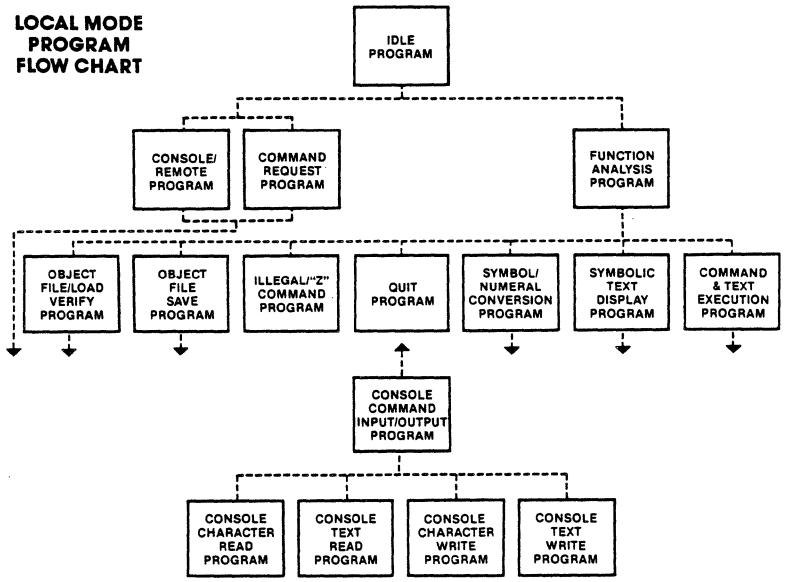
Program:	Mode:	Control:
CONSOLE CHARACTER WRITE	LOCAL	TERMINAL

Action: This sequence is used when the host computer requests the ICD to output n characters to the console.

Code/Record	Code/ Record Name	ICD←→HOST	Notes
\82{characters} <CR>	N CHARACTERS OUTPUT REQUEST RECORD	ICD←HOST	Sent to ICD to request the output of n characters to the console. The ICD then sends {characters} to the console, without feeding a line of data.
\F8<CR> or \F7<CR>	N CHARACTERS OUTPUT END CODE	ICD→HOST	Sent to host computer when the n-character output to the console is completed.


ICD
Host Computer System
\8 2 {character} <CR>
{output character(s)}
\F8<CR>
or \F7<CR>

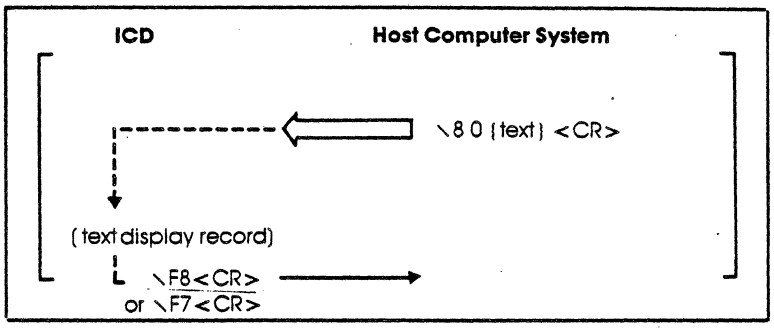
LOCAL MODE PROGRAM FLOW CHART



Program:	Mode:	Control:
CONSOLE TEXT WRITE	LOCAL	TERMINAL

Action: This sequence is used when the host computer requests the ICD to output one line of data to the console.

Code/Record	Code/ Record Name	ICD←→HOST	Notes
\80{text}<CR>	DATA OUTPUT REQUEST RECORD	ICD←HOST	Requests ICD to output one line of data to the console. The ICD outputs {text}<CR> to the console, and then follows it with a line feed.
\F8<CR>	CONSOLE OUTPUT END CODE	ICD→HOST	Sent to host computer when the sequence is completed.


ICD
Host Computer System
\80{text}<CR>
(text display record)
\F8<CR>
or \F7<CR>

NUMBER CONVERSION CODES
ICD278/Z8O

number change code	format	description
\20	\20{.symbol}<CR>	address symbol
\21—\2F	not used	

NUMBER CONVERSION CODES
ICD278/I8O85

number change code	format	description
\20	\20{.symbol}<CR>	address symbol
\21—\2F	not used	

NUMBER CONVERSION CODES
ICD178/I8O86, I8O88

number change code	format	description
\20	\20{.symbol}<CR>	physical address symbol
\21	not used	
\22	\22'{.symbol}<CR>	segment address symbol
\23	not used	
\24	\24xxxx:{.symbol}<CR>	offset address symbol (XXXX is current segment)
\25—\2F	not used	

NUMBER CONVERSION CODES
ICD178/I8O48

number change code	format	description
\20	\20{.symbol}<CR>	address symbol
\21—\2F	not used	

NUMBER CONVERSION CODES
ICD178/6800, 6801O, 6800S

number change code	format	description
\20	\20 { .symbol } <CR>	address symbol
\21—\2F	not used	

SYMBOL CONVERSION CODES
ICD278/Z8O

symbol change code	description	example
\30	header	\300000 00 NOP header address symbol change code
\31	not used	
\32	branch displace-ment	JR \32$—10H branch displacement symbol change code
\33—\35	not used	
\36	label	JP \368000H label symbol change code
\37—\3F	not used	

SYMBOL CONVERSION CODES
ICD278/i8085

symbol change code	description	example
\30	header	\300000 00 NOP header address symbol change code
\31—\35	not used	
\36	label	JMP \368000H label symbol change code
\37—\3F	not used	

SYMBOL CONVERSION CODES
ICD178/i8086, i8088, i80186, i80188

symbol change code	description	example
\30	physical header	\3000000 90 NOP physical header address symbol change code
\31	logical header	\310000:0000 90 NOP logical header address symbol change code
\32	branch displace-ment	JMP \32$-10H branch displacement symbol change code
\33	not used	
\34	number	MOV AL, \3455H number symbol change code
\35 — \36	not used	
\37	label	JMP \378000H (or\370100H: 8000H) label symbol change code
\38	not used	
\39	variable	MOV AL, BYTE PTR \39 0000H: 1000H variable logical address symbol change code
\3A—\3F	not used	

SYMBOL CONVERSION CODES
ICD178/I8048

symbol change code	description	example
\30	header	\30000 00 NOP header address symbol change code
\31—\35	not used	
\36	label	JMP \36100H label symbol change code
\37—\3F	not used	

SYMBOL CONVERSION CODES
ICD178/68000, 68010, 68008

symbol change code	description	example
\30	header	\30000000 header address symbol change code
\31—\33	not used	
\34	number	MOVE.B \34$00,D0 number symbol change code
\35	not used	
\36	label	JMP \36$00001000.W label symbol change code
\37	not used	
\38	variable	MOVE.B D0,\38$00001000,W variable symbol change code
\39—\3F	not used	

INTEL HEX OBJECT FORMAT:

All object files are represented by ASCII codes. This example shows one byte of data being converted to an ASCII hexadecimal number ("0"—"9" and "A"—"F") of two digits:

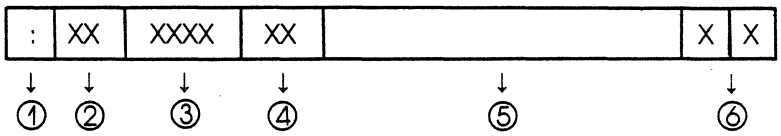
00_H	"00"(3030_H)
$9B_H$	"9B"(3942_H)

An object file is divided into units of records which include four types:

(1) Data Record
(2) End of File Record
(3) Extended Address Record
(4) Start Address Record

ICD/Z80, ICD/i8085, and ICD/8048 use Data and End Record only.

One record is formatted as shown below:



① Record mark
":" ($3A_H$)
Shows the beginning of an Intel Hex object record. The information preceding this mark is treated as a comment.

② Load address
"00"—"FF" (3030_H—4646_H)
Shows the number of data bytes contained in field ⑤.

③ Code address
"0000"—"FFFF" (30303030_H—46464646_H)
Shows the location address where a program or data is intended to be loaded. Normally contains "0000" as a dummy record.

④ Record type
Shows type of record:
"00" (3030_H) Data record
"01" (3031_H) End of File record

⑤ Data
Contains data bytes equal to the record length. (This field void if the record length is "00.")

⑥ Check Sum
2's complement of the value (one byte: carry ignored) of the total starting with the record length and the last data. *NOTE: Addition is made after the ASCII hexadecimal number of two digits has been converted to a 1-byte binary number.*

Example:

```
:020000020100FB
:20000000081000D00525A58608900040000CA00BAD95FFF4DF9AE52DA725FFD4FF2F808384
:0E00200008180010085A5A58B040000B0000490
:020000020200FA
:20000000800000052EA1050A5401D0000B000FAF1DFFB0FFBBA50DAF35DFF5FF4FE008E1
:20002000081000D005A1A506000100D000000007AD05FFC0DFFAE125A585FFF2FF0F3009FD
:04000030010000000F8
:00000001FF
```

DATA RECORD: This record is used to show a program or data.

Example:

```
: 10 0000 00 004992D B246D B6F F4891D A236C B5F E47 B8
↓ ↓  ↓   ↓                ↓                       ↓
① ②  ③   ④                ⑤                       ⑥
```

① Record mark
":" ($3A_H$)

② Record length
"10" (3130_H)
Shows data of 16 bytes contained in the data field ⑤.

③ Load address
"0000" (30303030_H)
Indicates that data in field ⑤ is loaded starting at address 0000_H.

④ Record type
"00" (3030_H)
Shows that record is a data record.

⑤ Data
"0049 . . . " (30303439_H . . .)
Data in this case: 00_H,49_H,92_H . . .

⑥ Check sum
"B8" (4238_H)

END OF FILE RECORD: This record shows the end of an object file.

Example:

: 00 0000 01 F F
↓ ↓ ↓ ↓ ↓
① ② ③ ④ ⑤

① Record mark
":" ($3A_H$)

② Record length
"00" (3030_H)
Shows the data field does not exist.

③ Load address
"0000" (30303030_H)
Normally, "0000" is entered as a dummy address (though this address may be used as a start address if no start address record is found).

④ Check sum
"FF" (4646_H)

NOTE: When using the LOAD or VERIFY commands, the end of the object file is determined by the end of record.

EXTENDED ADDRESS RECORD: This record shows the segment address where data is loaded in the data record subsequent to this record.

Example:

:	02	0000	02	0020	DC
↓	↓	↓	↓	↓	↓
①	②	③	④	⑤	⑥

① Record mark
":" ($3A_H$)

② Record length
"02" (3032_H)
Shows that two bytes of data are contained in the data field in ⑤.

③ Load address
"0000" (30303030_H)
Contains "0000" as a dummy, though this field is ignored in this record. (It is still required.)

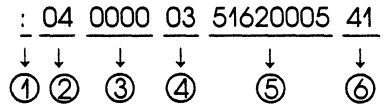
④ Record type
"02" (3032_H)
Shows that this record is an extended address record.

⑤ Segment base address
"0020" (30303230_H)
Base address in this case is 0020_H.

⑥ Check sum
"DC" (4443_H)
02H+00H+00H+02H+00H+20H=24H
24H Two's Complement DC_H

START ADDRESS RECORD: This record shows the object file start address.

Example:



① Record mark
":" ($3A_H$)

② Record length
"04" (3034_H)

Indicates that the data field in ⑤ contains data of four bytes.

③ Load address
"0000" (30303030_H)

Contains "0000" as a dummy, though this field is not necessary for this record.

④ Record type
"03" (3033_H)

Shows this record is a start address record.

⑤ Start address
"51620005" (3531363230303035_H)

Start address in this case:

Segment = 5162_H

Offset = 0005_H

⑥ Check sum
"41" (3431_H)

S FORMAT OBJECT FILE:

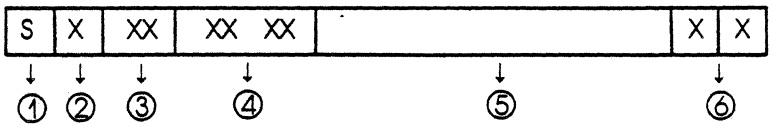
All object files are described by ASCII codes. In the example, one byte of data is shown converted to a hexadecimal number ("0"—"9," and "A"—"F") of two digits:

00_H "00" (3030_H)
$9B_H$ "9B" (3942_H)

An object file is composed of the records listed below:

(1) Data set name record
(2) 16-bit address data record
(3) 24-bit address data record
(4) 32-bit address data record
(5) Send data record count record
(6) 16-bit address end record
(7) 24-bit address end record
(8) 32-bit address end record

ICD/68000.68008.68010 uses the data records (2) and (3) and the end records (6) and (7) only. The record format is shown below:



① Record mark "S" (53_H)

Indicates the start point of an object record in S format. Information before this mark is treated as a comment.

② Record type

Shows the type of this record.

(1) "0" (30_H) Data set name record
(2) "1" (31_H) 16-bit address data record
(3) "2" (32_H) 24-bit address data record
(4) "3" (33_H) 32-bit address data record
(5) "5" (35_H) Send data record count record
(6) "7" (37_H) 32-bit address end record
(7) "8" (38_H) 24-bit address end record
(8) "9" (39_H) 16-bit address end record

③ Record length
"00"—"FF" (3030_H—4646_H)

Shows how many bytes of data are contained in fields ④ , ⑤ , and ⑥.

④ Load address
"0000"—"FFFF" (30303030_H—46464646_H)
or "000000"—"FFFFFF" (303030303030_H—464646464646_H)
or "00000000"—"FFFFFFFF"
(3030303030303030_H—4646464646464646_H)

When used with data records, this address shows the address to load a program or data. When used with end records, it shows the restart address of the program. When used with data set name records (Record type "0"), the address normally contains "0000" as a dummy data. 16-bit address, 24-bit address, and 32-bit address are identified by the record type.

⑤ Data

Data is equal to the record length minus the load address and check sum. (When the number of record bytes is 00, this field does not exist.)

⑥ Check sum

1's complement of the total value of the bytes up to the last data beginning with the record length (one byte and carry are ignored).

NOTE: Addition is made after converting an ASCII hexadecimal number of two digits to a binary number of one byte.

Example:
S006000041424333
S214010000A14E0A405ADF02E067D00410EC1F013A05
S21401001085C906905AFB0490E5580C0042BE00E2E2
S214010020A1060C41D22F00F2A14B8E00C4E300B210
S214010030D14B04A0784E4090AB470940808E10D03B
S214010040A15D0B08721F4C504FCC4A10A41D006ACC
S214010050E9400F005B9B0AF2F5158F1120EF0CF8B3
S214010060A5890B10DADF08E28548060020D708BA0C
S214010070A1C041017ADF0050A15E280406FF005AA4
S804010000FA

DATA SET NAME RECORD:
(Record type "0")

A record to show the record name of an object file.

Example:

S 0 06 0000 414243 33
↓ ↓ ↓ ↓ ↓ ↓
① ② ③ ④ ⑤ ⑥

① Record mark
"S" (53_H)

② Record type
"0" (30_H)
Indicates that this record is a data set name record.

③ Record length
"06" (3036_H)
Shows that the total of the load address, data, and check sum is six bytes.

④ Load address
"0000" (30303030_H)
This record contains "0000" as a dummy, though this field is ignored in this record.

⑤ Data set name
"414243" (343134323433_H)
The record name is interpreted as ASCII codes 41_H, 42_H, and 43_H, producing "ABC."

⑥ Check sum
"33" (3030_H)

DATA RECORD:
(Record type "1"—"3")

Shows a program or data.

Example:

S 2 14 010000 A14E0A405ADF02E067D00410EC1F013A 05

① ② ③ ④ ⑤ ⑥

① Record mark
"S" (53_H)

② Record type
"2" (32_H)

③ Record length
"14" (3134_H)

Indicates that the total of the load address and check sum is 20 bytes.

④ Load address
"010000" (303130303030_H)

Indicates data in field ⑤ is loaded starting at address 01000_H. (The number of address bits will be 16, 24, or 32 depending upon the record type in field ②.)

⑤ Data
"A14E . . . 3A" (41313445_H . . . 3341_H)

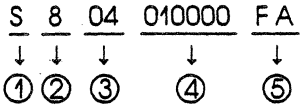
In this case, data is $A1_H$, $4E_H$, ... $3A_H$.

⑥ Check sum
"05" (3035_H)

END RECORD:
(Record type "7"–"9")

Shows the end of an object file.

Example:



① Record mark
"S" (53_H)

② Record type
"8" (38_H)

Indicates this record is an end record with the 24-bit start address.

③ Record length
"04" (3034_H)

Shows that the total of the start address and check sum is four bytes. (Normally, an end record does not contain the data field.)

④ Start address
"010000" (303130303030_H)

In this case, the start address is 010000_H.

⑤ Check sum
"FA" (4641_H)

NOTE: When using LOAD and VERIFY commands, the end of an object file is determined by the end record.

ZAX

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