ICD-278

FOR

Z80

USER'S MANUAL

Part No. 20-101-00, Rev. A. Printed: June, 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 materials 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 instructions contained in the User's Manual; damage resulting from the performance of repairs by someone not authorized by ZAX; and 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 D. 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 20-101-00
Reorder Command Reference Guide 20-602-01

Contents

SECTION 2 — MASTER COMMAND GUIDE

SECTION 3 — TECHNICAL REFERENCES

SECTION 4 — COMMUNICATION PROTOCOL

ICD-278 for Z80 FEATURES

Transparency

- All memory available
- All I/O ports (256) available
- Dynamic RAM refresh
- DMA supported
- Drive data bus on emulation reads
- Support systems with data bus buffer
- No conflict between emulation memory and user memory when overlaying

User Interface

- You control all functions from computer or terminal
- Mnemonic command names
- Setup emulation controls from batch file on host computer

Emulation Controls

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

Memory Mapping

- 1K mapping resolution
- Read-only emulation memory
- "No memory" 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 interrupt acknowledge
- Break on Nth occurrence
- Break on wait state timeout
- External break input (triggers from HI or LO edge)
- External break output
- Unlimited breakpoints on address

Non-Real-Time Trace

- Single step
- Step n steps
- Trace jumps only

Real-Time Trace

- Stores addresses, data, and status
- 2K x 32 bits trace memory size
- Trace control modes include:
    - Begin Monitor
    - End Monitor
    - Begin Event
    - End Event
    - Center Event
    - Multiple Event
- Adjustable delay

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 memry for pattern
- Search trace memory for pattern


Cooling Fan
AC Power Cord
Data Bus Emulation Connector and Test Clip
Emulation Quality / Event Trigger Probes
CPU In-circuit Probe
Map Control / External Break Probes

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 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 and accessories) of your emulator and 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 came 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 communication between the emulator and 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 your 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 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-278 for Z80 (ICD stands for IN-CIRCUIT DEBUGGER; 278 is the model number). That's quite a mouthful though, so to shorten things up we'll use the initials **ICD** whenever we mean the ICD-278, in-circuit debugger, emulator, or in-circuit emulator.

Now turn to Section 1 and get started.


Cooling Fan
AC Power Cord
Data Bus Emulation Connector and Test Clip
Emulation Qualify/ Event Trigger Probes
CPU In-circuit Probe
Map Control/External Break Probes

Contents

SECTION 1 — ICD DESCRIPTION & OPERATION

Contents

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 about 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." 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 microprocessor-based systems. 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 register's 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."

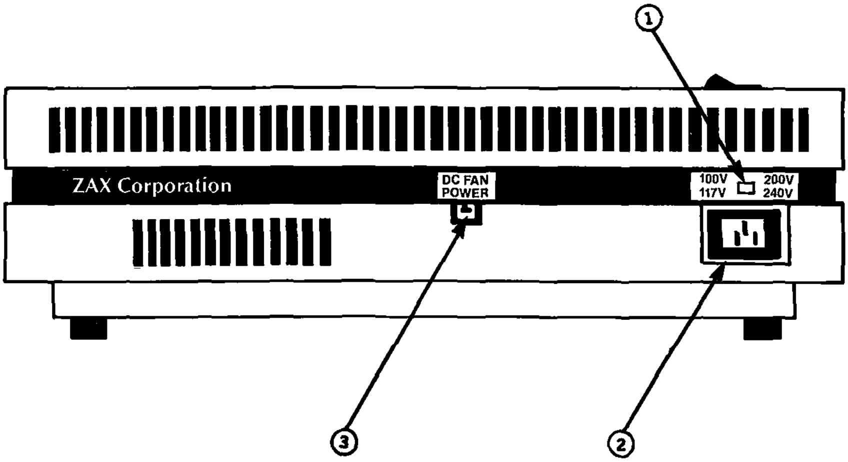
Now turn the page to learn about the parts of your ICD.

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 110-120VAC, or select 200V/240V to run on a power supply of 200-240.

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 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 in a stand-alone (LOCAL mode) configuration. When using the ICD in the REMOTE 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 operating in the REMOTE mode. ICD commands can then be entered using the computer's keyboard. When using the ICD in a stand-alone (LOCAL mode) configuration, 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 **BAUDRATE Switches (TERMINAL and HOST/AUX).** 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 of the ICD, see "Technical References"; SIO module.

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 **DB. EMUL (Data Bus Emulation) Connector.** Accepts female end of the Data Bus Emulation Cable. (See "More About Your ICD" for details on how to use this cable.)

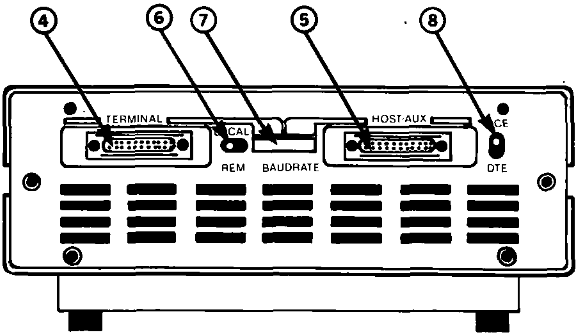
⑩ **Top In-circuit Probe Receptacle.** Accepts female end of the Top In-circuit Probe.

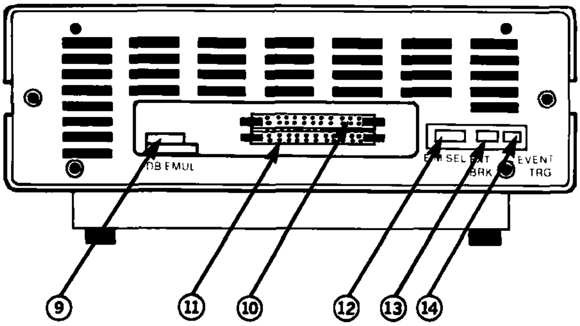
⑪ **Bottom In-circuit Probe Receptacle.** Accepts female end of the Bottom In-circuit Probe.

⑫ **E.M. SEL (Emulation Select) Switch.** This switch is used to set the machine cycle operation to the target system. (See "More About Your ICD" 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" 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" 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=external).

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.

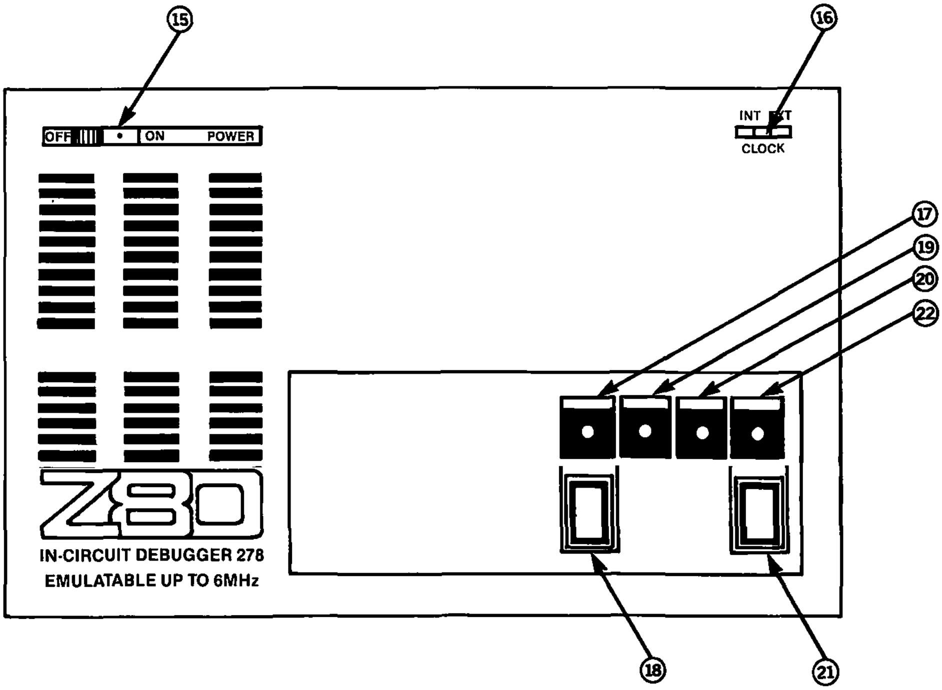
⑱ **RESET Switch.** This switch is used to reset the ICD monitor. You can push it any time the MONITOR lamp is lit. After you push the RESET switch, you'll see the ICD's identification message on your terminal's monitor.

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 in-circuit mode I1 or I2.

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 your ICD to your system.

How to Connect Your ICD To Other Devices

Your Goal: A Microprocessor Development System

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 have completed the procedures outlined in this chapter, 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 Second, will you control the system with a **terminal** or a **host computer**? And third, if a terminal is used to control the ICD, will a host computer be used as a **source** for data files?

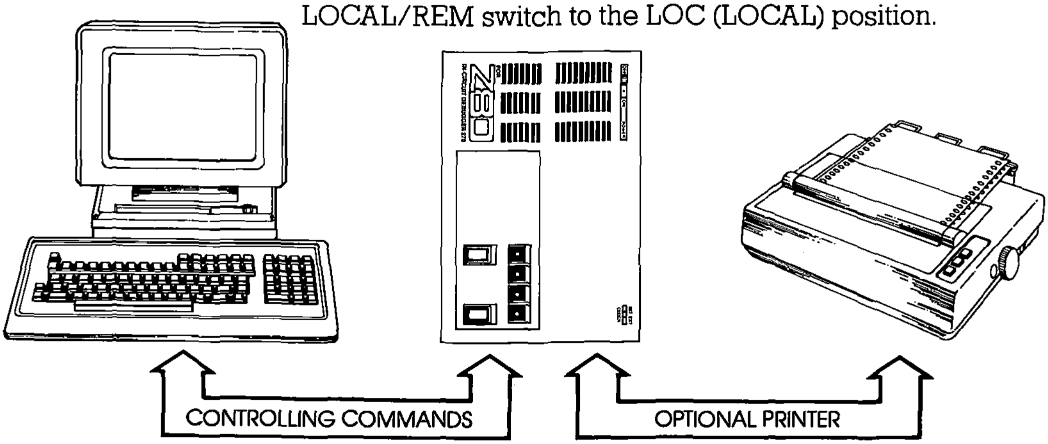
Hardware or Software?

Your **hardware** is called a "target system." By physically removing the CPU 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. If this is the mode you'll be operating in, look at the three system configurations in **USING THE ICD WITH A TARGET SYSTEM**.

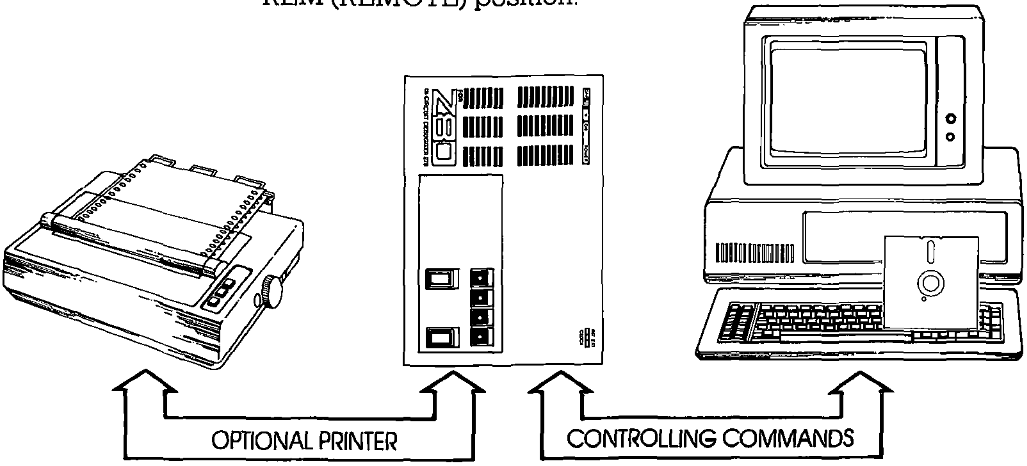
Can you use your ICD without a target system? Of course! Whenever you develop and debug **software** you'll be doing it without the use of a target system. This mode is also an effective way to demonstrate some of your ICD's features. If this is the mode you'll be operating in, look at the three system configurations in **USING THE ICD WITHOUT A TARGET SYSTEM**. *(In fact, if this is the first time you are using a* ***ZAX*** *emulator, you should construct this system configuration and then turn to Appendix B at the back of the manual. There you will find a demonstration of the functions and features of your ICD.)*

Terminal or Host Computer Controlled?

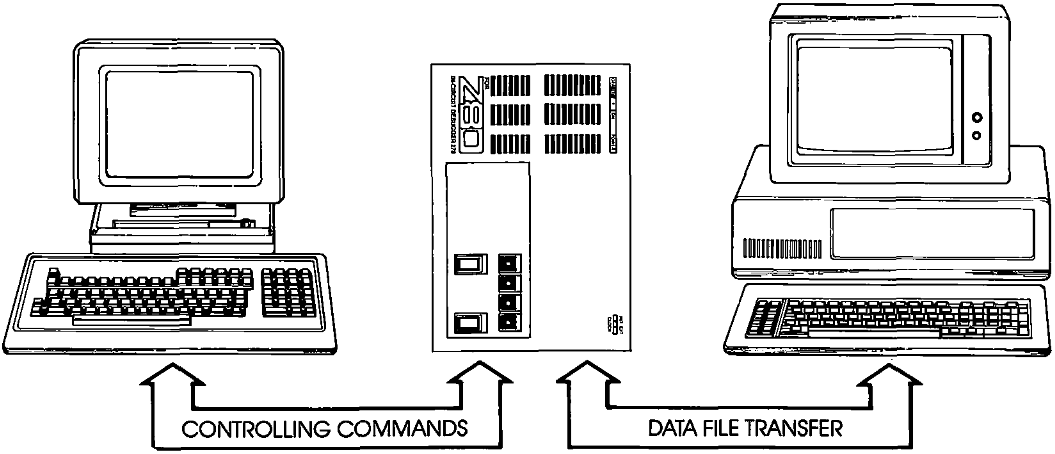
If you'll be controlling the ICD by a console terminal, it's called **TERMINAL CONTROL OF THE ICD.** In this mode, the ICD "stands alone" (hence the name, stand-alone emulator) or apart from the auxiliary control of a host computer system. The ICD assumes a stand-alone mode of operation when you place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



Finally, you may choose to control the ICD with a terminal and use a separate host computer to store data files, or connect a printer to the host computer to dump data for hard copies. This mode of operation is called **TERMINAL CONTROL OF THE ICD (WITH HOST DATA FILES)**. In this mode, the ICD is still under direct control of the terminal while 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.

Reviewing The Operation Modes

Now let's review the six different operation modes of your ICD.

USING THE ICD *WITHOUT* A TARGET SYSTEM	USING THE ICD *WITH* A TARGET SYSTEM
■ Terminal Control of the ICD—LOCAL mode of operation	■ Terminal Control of the ICD—LOCAL mode of operation
■ Terminal Control of the ICD (With Host Data Files) —LOCAL mode of operation	■ Terminal Control of the ICD (With Host Data Files) —LOCAL mode of operation
■ Host Computer Control of the ICD—REMOTE mode of operation	■ Host Computer Control of the ICD—REMOTE mode of operation

Summing It All Up . . .

- Your ICD can function in any of six different modes.
- 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 then use a separate host computer for storing data files or outputting data to a printer.
- Your ICD can be controlled by a terminal and then 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 Before You Connect Anything!

Grounds

Your ICD is equipped with a three-wire polarized receptacle that accepts a three-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 three-prong plug wth an adaptor (3- into 2-prong adaptor).

THE GROUND TERMINAL OF THE PLUG IS USED TO PREVENT SHOCK HAZARDS—DO NOT BYPASS IT!

Power

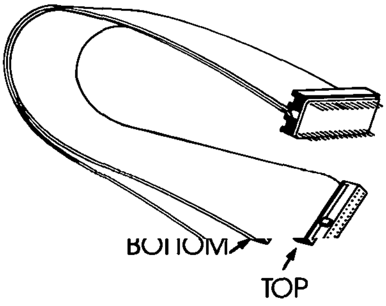
Your ICD is normally set to operate on a voltage supply of 110-120 VAC, but this can be changed to 200-240 VAC by setting the Power Select Switch to the 200V/240V position.

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 switch.

Important Facts About The CPU In-Circuit Probe

The CPU In-circuit Probe is used to connect the ICD to your target system. 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 topp and bottom receptacles.



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

Now turn to the appropriate heading on the next few pages to construct *your* microprocessor development system.

USING THE ICD WITHOUT A TARGET SYSTEM (TERMINAL CONTROLLED)

System Configuration: Terminal control of the ICD
Operation Mode: LOCAL
Facilities needed for this system configuration: ICD, Console Terminal, (1) RS-232 cable.

To use the ICD in this mode, construct the system configuration shown on the opposite page using the information below.

First . . .
Make sure that the power to the ICD and all externally attached devices (terminal, printer) is OFF, then proceed as follows:

1) Attach the COOLING FAN to the ICD and then plug the fan's connector to the receptacle labeled DC FAN POWER.

2) 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 up your terminal to these specifications.

3) (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.

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

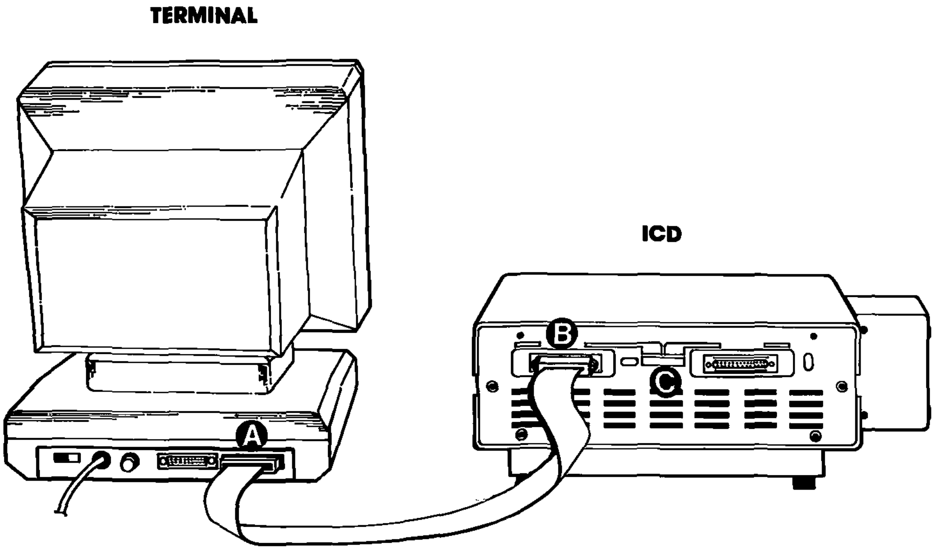
Now Set This:		To This:
100V/117V 200V/240V	>	110V/117V
LOCAL/REM	>	LOCAL
INT/EXT	>	INT
DCE/DTE	>	DCE
Baud Rates	>	9600 bps *(NOTE: To change the ICD's baud rates, see the chart on the opposite page.)*
POWER ON/OFF Switch	>	ON

The following message should now appear on your monitor's screen (you may have to press the

```
ICD-278 for Z80        V2.0
```

Now turn to page 1-26.

Ⓐ Terminal's EIA RS-232 port
Ⓑ ICD's TERMINAL port
Ⓒ Use chart below to change baud rate for ICD's TERMINAL port



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	—	—

USING THE ICD WITHOUT A TARGET SYSTEM (TERMINAL CONTROLLED/HOST STORAGE)

System Configuration: Terminal control of the ICD (with host data files)
Operation Mode: LOCAL
Facilities needed for this system configuration: ICD, Console Terminal, Host Computer, (2) RS-232 cables.

To use the ICD in this mode, construct the system configuration shown on the opposite page using the information below.

First . . .
Make sure that the power to the ICD and all externally attached devices (terminal, printer, host computer) is OFF, then proceed as follows:

1) Attach the COOLING FAN to the ICD and then plug the fan's connector to the receptacle labeled DC FAN POWER.

2) 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 up your terminal to these specifications.

3) 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.

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

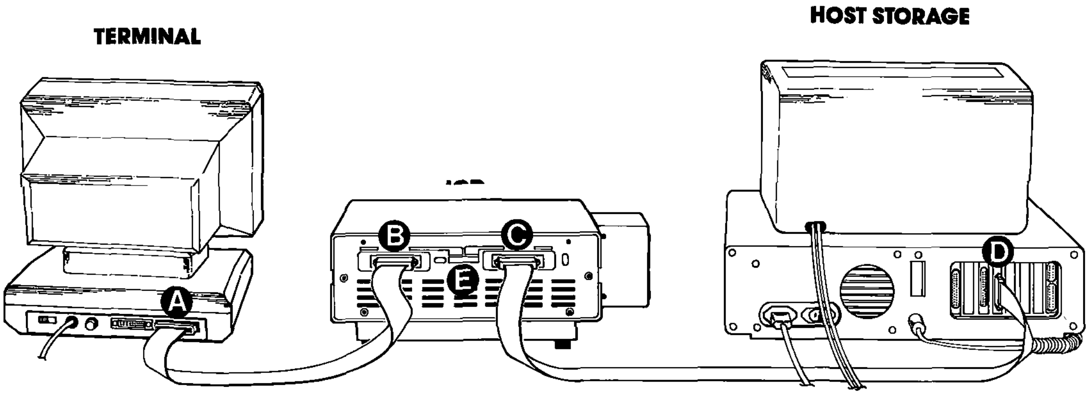
Now Set This:		To This:
100V/117V 200V/240V	>	110V/117V
LOCAL/REM	>	LOCAL
INT/EXT	>	INT
DCE/DTE	>	DTE if you're using **ZAX**'s BOX microcomputer, DCE for other personal computers.
Baud Rates	>	9600 bps *(NOTE: To change the ICD's baud rates, see the chart on the opposite page.)*
POWER ON/OFF Switch	>	ON

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 Z80     V2.0
```

Now turn to page 1-26.

- Ⓐ **Terminal's EIA RS-232 port**
- Ⓑ **ICD's TERMINAL port**
- Ⓒ **ICD's HOST/AUX port**
- Ⓓ **Computer's SIO port**
- Ⓔ **Use chart below to change baud rates for ICD's TERMINAL and HOST/AUX ports**



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	—	—

USING THE ICD WITHOUT A TARGET SYSTEM (HOST COMPUTER CONTROLLED)

System Configuration: Host computer control of the ICD
Operation Mode: REMOTE
Facilities needed for this system configuration: ICD, Host Computer, ZICE software, (1) RS-232 cable.

To use the ICD in this mode, construct the system configuration shown on the opposite page using the information below.

First . . .
Make sure that the power to the ICD and all externally attached devices (host computer, printer) is OFF, then proceed as follows:

1) Attach the COOLING FAN to the ICD and then plug the fan's connector to the receptacle labeled DC FAN POWER.

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.

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

100V/117V 200V/240V > 110V/117V
LOCAL/REM > REM
INT/EXT > INT
DCE/DTE > DTE if you're using **ZAX**'s BOX microcomputer; DCE for other personal computers.
Baud Rates > 9600 bps *(NOTE: To change the ICD's baud rates, see the chart on the opposite page.)*
POWER ON/OFF Switch > ON

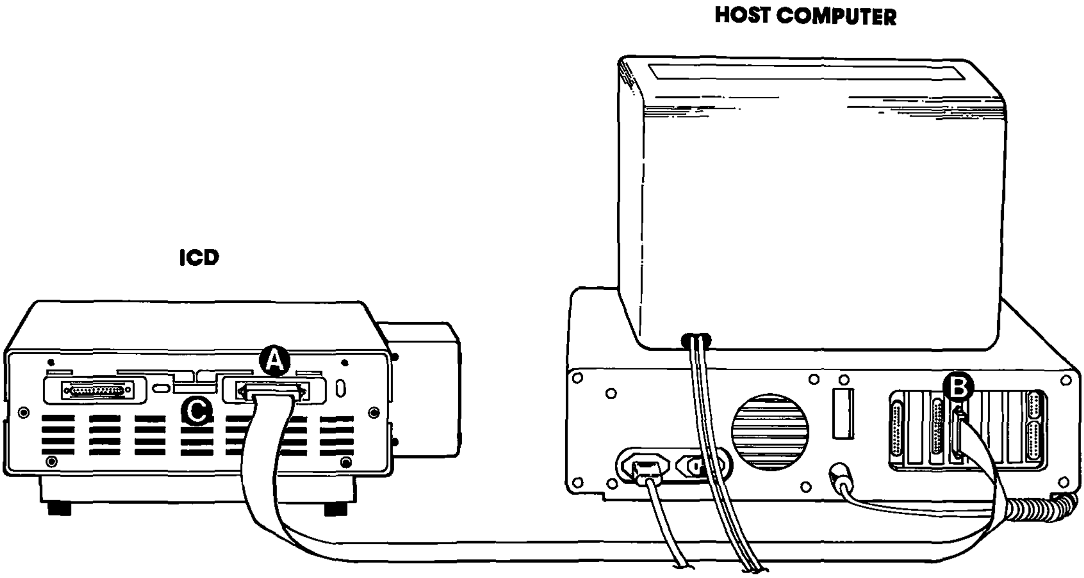
At this point, you will have to load the ZICE software program necessary for interfacing the ICD to your host computer. Execute the program loading commands as outlined in the ZICE software documentation.

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

```
ICD-278 for Z80        V2.0
```

Now turn to page 1-26.

- Ⓐ **ICD's HOST/AUX port**
- Ⓑ **Host computer's SIO port**
- Ⓒ **Use chart below to change baud rate for ICD's Host/AUX port**



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	—	—

USING THE ICD WITH A TARGET SYSTEM (TERMINAL CONTROLLED)

System Configuration: Terminal control of the ICD
Operation Mode: LOCAL
Facilities needed for this system configuration: ICD, Console Terminal, Target System, CPU In-circuit Probe, (1) RS-232 cable.

To use the ICD in this mode, construct the system configuration shown on the opposite page using the information below.

First . . .
Make sure that the power to the ICD and all externally attached devices (terminal, printer, target system) is OFF, then proceed as follows:

1) Attach the COOLING FAN to the ICD and then plug the fan's connector to the receptacle labeled DC FAN POWER.

2) 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 up your terminal to these specifications.

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

4) Remove the existing (Z80) CPU from your target system and insert the IN-CIRCUIT PROBE (40-pin end) socket into the target system's CPU socket (pin 1 of the ICD's In-circuit probe socket goes into pin 1 of the target system's CPU socket). Connect the other end of the IN-CIRCUIT PROBE to the ICD's TOP and BOTTOM In-circuit Probe Receptacles. THE LONGEST CABLE MUST BE CONNECTED TO THE TOP IN-CIRCUIT PROBE RECEPTACLE.

5) Plug the AC POWER CORD into the ICD's power receptacle, then connect the other end of the cable to the same power source that is used by your target system.

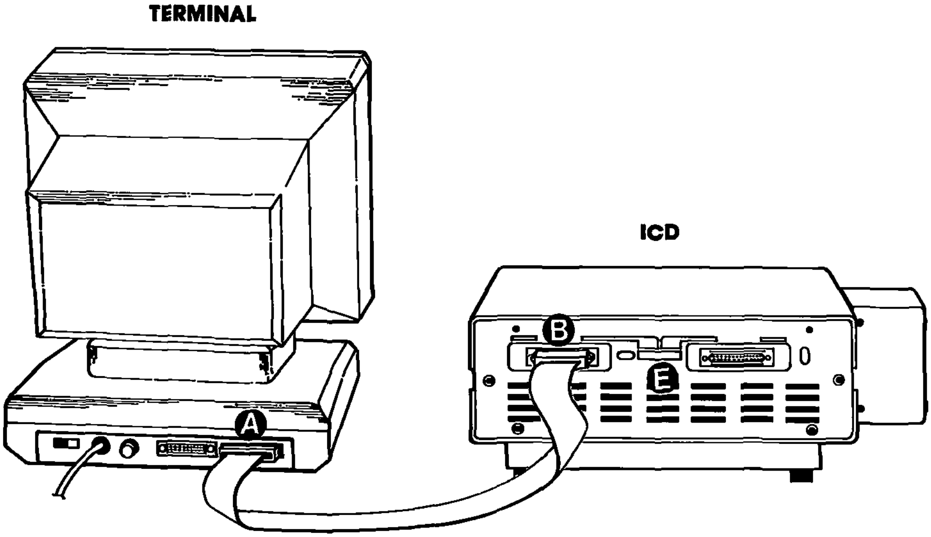
Now Set This:		To This:
100V/117V 200V/240V	>	110V/117V
LOCAL/REM	>	LOCAL
INT/EXT	>	EXT
DCE/DTE	>	DCE
Baud Rates	>	9600 bps *(NOTE: To change the ICD's baud rates, see the chart on the opposite page.)*
POWER ON/OFF Switch	>	ON

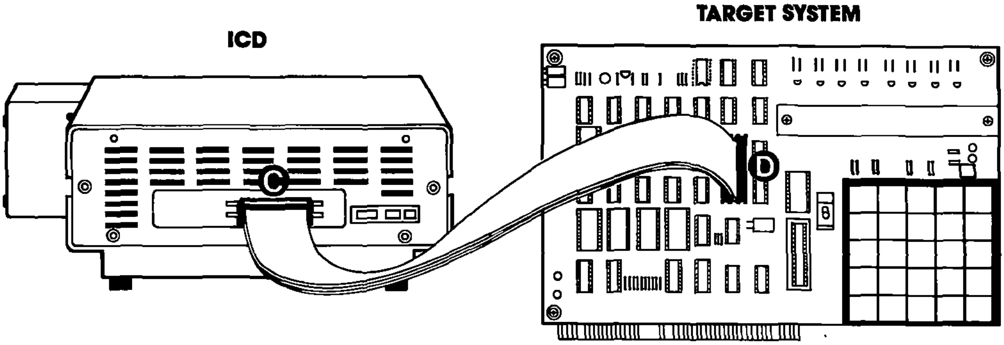
The following message should now appear on your monitor's screen (you may have to press the RESET switch on the ICD):

ICD-278 for Z80 V2.0

Now turn to page 1-26.

Ⓐ Terminal's EIA RS-232 port
Ⓑ ICD's TERMINAL port
Ⓒ ICD's In-circuit probe receptacle
Ⓓ Target system's CPU socket
Ⓔ Use chart below to change baud rate for ICD's TERMINAL port





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	—	—

USING THE ICD WITH A TARGET SYSTEM
(TERMINAL CONTROLLED/HOST STORAGE)

System Configuration: Terminal control of the ICD (with host data files)
Operation Mode: LOCAL
Facilities needed for this system configuration: ICD, Console Terminal, Host Computer, Target System, CPU In-circuit Probe, (2) RS-232 cables.

To use the ICD in this mode, construct the system configuration shown on the opposite page using the information below.

First . . .
Make sure that the power to the ICD and all externally attached devices (terminal, printer, host computer, target system) is OFF, then proceed as follows:

1) Attach the COOLING FAN to the ICD and then plug the fan's connector to the receptacle labeled DC FAN POWER.
2) 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 up your terminal to these specifications.
3) Connect your terminal 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.
4) Remove the existing (Z80) CPU from your target system and insert the IN-CIRCUIT PROBE (40-pin end) into the target system's CPU socket (pin 1 of the ICD's In-circuit probe socket goes into pin 1 of the target system's CPU socket). Connect the other end of the IN-CIRCUIT PROBE to the ICD's TOP and BOTTOM In-circuit Probe Receptacles. THE LONGEST CABLE MUST BE CONNECTED TO THE TOP IN-CIRCUIT PROBE RECEPTACLE.
5) Plug the AC POWER CORD into the ICD's power receptacle, then connect the other end of the cable to the same power source that is used by the target system.

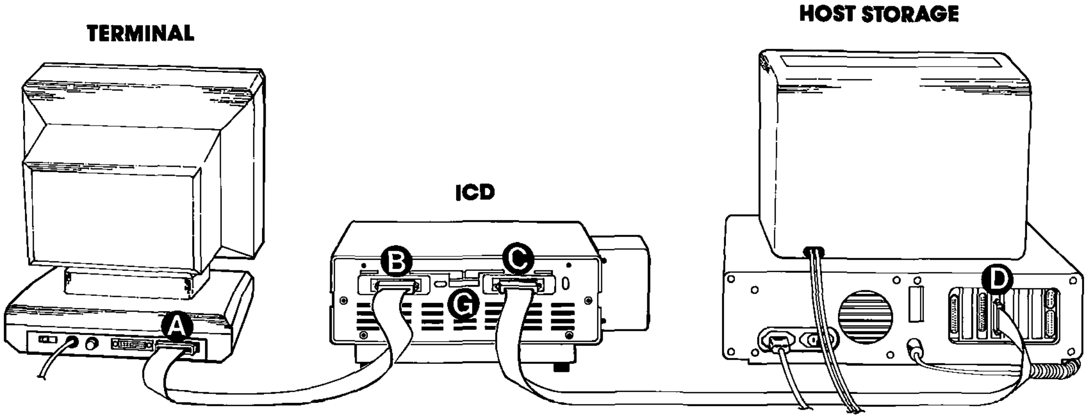
Now Set This:		To This:
100V/117V 200V/240V	>	110V/117V
LOCAL/REM	>	LOCAL
INT/EXT	>	EXT
DCE/DTE	>	DTE if you're using **ZAX**'s BOX microcomputer; DCE for other personal computers.
Baud Rates	>	9600 bps *(NOTE: To change the ICD's baud rates, see the chart on the opposite page.)*
POWER ON/OFF Switch	>	ON

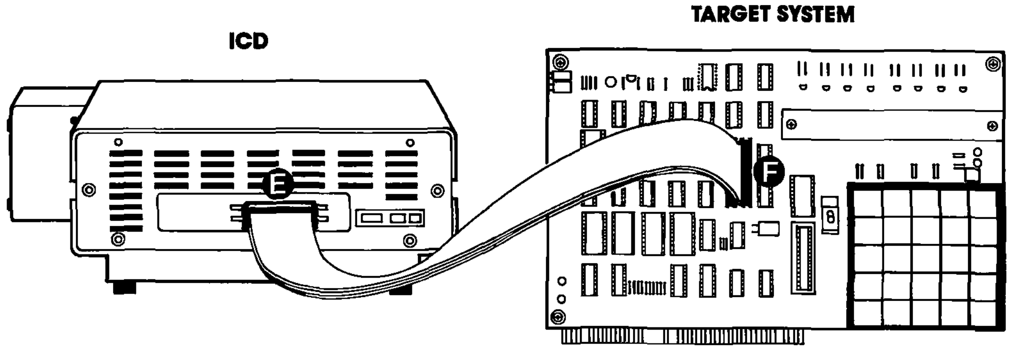
The following message should now appear on your monitor's screen (you may have to press the RESET switch on the ICD):

```
ICD-278 for Z80    V2.0
```

Now turn to page 1-26.

Ⓐ **Terminal's EIA RS-232 port**
Ⓑ **ICD's TERMINAL port**
Ⓒ **ICD's HOST/AUX port**
Ⓓ **Computer's SIO port**
Ⓔ **ICD's In-circuit probe receptacle**
Ⓕ **Target system's CPU socket**
Ⓖ **Use chart below to change baud rates for ICD's TERMINAL and HOST/AUX ports**





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	—	—

USING THE ICD WITH A TARGET SYSTEM (HOST COMPUTER CONTROLLED)

System Configuration: Host computer control of the ICD
Operation Mode: REMOTE
Facilities needed for this system configuration: ICD, Host Computer, ZICE Software, Target System, CPU In-circuit Probe, (1) RS-232 cable.

To use the ICD in this mode, construct the system configuration shown on the opposite page using the information below.

First . . .
Make sure that the power to the ICD and all externally attached devices (host computer, printer, target system) is OFF, then proceed as follows:

1) Attach the COOLING FAN to the ICD and then plug the fan's connector to the receptacle labeled DC FAN POWER.

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) Remove the existing (Z80) CPU from your target system and insert the IN-CIRCUIT PROBE (40-pin end) into the target system's CPU socket (pin 1 of the ICD's In-circuit probe socket goes into pin 1 of the target system's CPU socket). Connect the other end of the IN-CIRCUIT PROBE to the ICD's TOP and BOTTOM In-circuit Probe Receptacles. THE LONGEST CABLE MUST BE CONNECTED TO THE TOP IN-CIRCUIT PROBE RECEPTACLE.

4) Plug the AC POWER CORD into the ICD's power receptacle, then connect the other end of the cable to the same power source that is used by the target system.

Now Set This:		To This:
100V/117V 200V/240V	>	110V/117V
LOCAL/REM	>	REM
INT/EXT	>	EXT
DCE/DTE	>	DTE if you're using **ZAX**'s BOX microcomputer; DCE for other personal computers.
Baud Rates	>	9600 bps *(NOTE: To change the ICD's buad rates, see the chart on the opposite page.)*
POWER ON/OFF Switch	>	ON

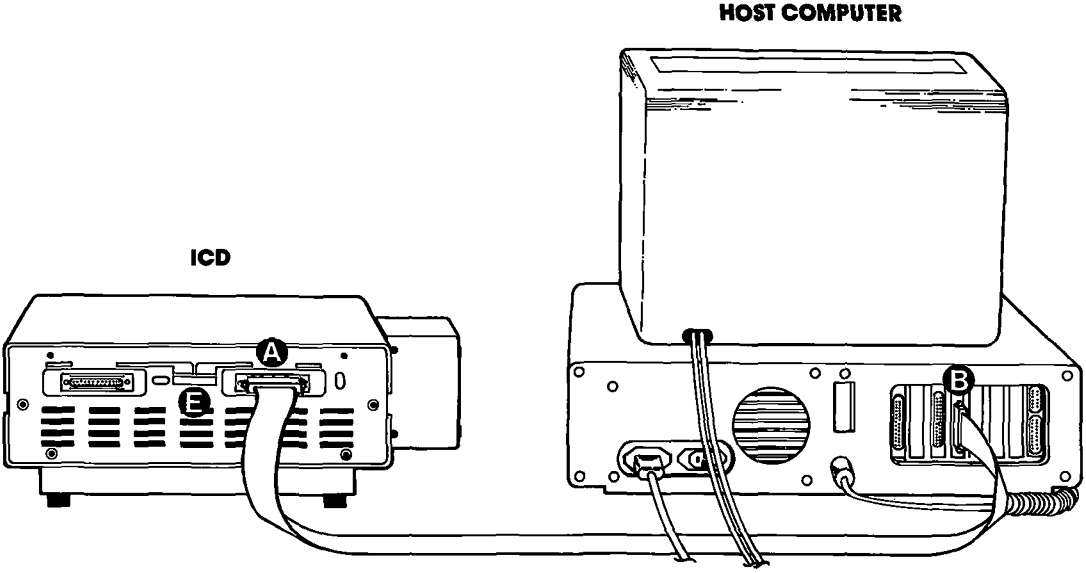
At this point, you will have to load the ZICE software program necessary for interfacing the ICD to your host computer. Execute the program loading commands as outlined in the ZICE software documentation.

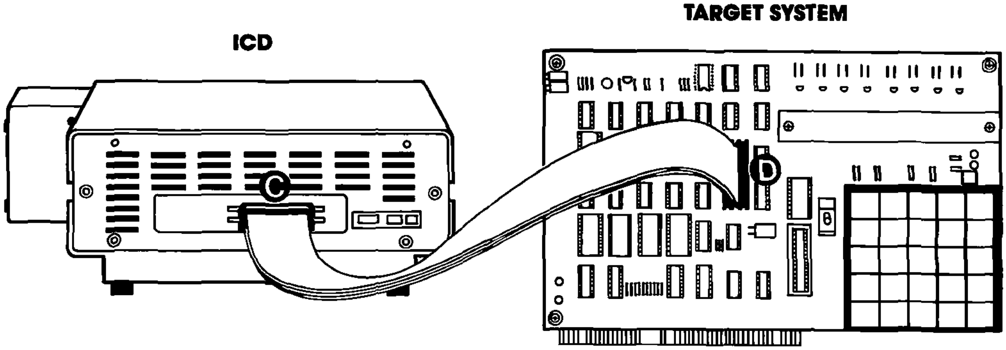
The following message should now appear on your monitor's screen (you may have to press the RESET switch on the ICD):

```
ICD-278 for Z80     V2.0
```

Now turn to page 1-26.

- **Ⓐ ICD's HOST/AUX port**
- **Ⓑ Host computer's SIO port**
- **Ⓒ ICD's In-circuit probe receptacle**
- **Ⓓ Target system's CPU socket**
- **Ⓔ Use chart below to change baud rate for ICD's HOST/AUX port**





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	—	—

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 appeared on your monitor's screen, you can now:

- Turn to the "Master Command Guide" 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 "Command Reference Guide" as a source for 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" which starts 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.

⇦ COMMAND REFERENCE GUIDE

SECTION 2 MASTER COMMAND GUIDE ⇨

APPENDIX B ⇨

TROUBLE SHOOTING ⇨

Trouble Shooting (contains "Checking Electrical Connections" and "Diagnosing ICD Interface Problems")

Introduction: The Problem . . .

Your ICD must be controlled by either a separate terminal or a host computer's keyboard. And, 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 might have encountered in "How To Connect Your ICD To Other Devices," and begins with a typical example of what the ICD should do if the system is operating correctly. Then the ICD by itself is tested, followed by testing the ICD and terminal. ICD, terminal, and target system configuration is then examined.

What Should Happen

When the ICD is connected to a terminal (keyboard and monitor), the following should happen:

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

```
ICD-278 for Z80        V2.0
```

(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 that 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 follow the procedures 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. Then you'll check the ICD by itself (just connect its power cord), then attach a terminal. If that configuration works properly, you'll 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:	SOLUTION:
The external cooling fan doesn't work.	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 Operators 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.

If this checks out, the ICD is probably working correctly. Now connect a terminal (no target system yet) to the ICD and carry out the next procedure.

ICD and Terminal

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

PROBLEM	SOLUTION
The terminal does not respond at all when the RESET switch is pressed.	What's Probably Wrong: There is either an interface problem or a defect with a 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 the terminal have been set 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
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 button 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 then read through the next checkout 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 properly.

SOLUTION
What's Probably Wrong:
There is either an interface problem or a defect with a 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 set up correctly. Select the EXTERNAL (EXT) clock, and press the RESET switch on the ICD. The ICD's identification message and prompt should appear. If no prompt appears on EXTERNAL clock setting, switch to 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.

NOTE: In this next checkout procedure, you will need to enter certain commands in order to test the system. See "Master Command Guide" for an explanation of how to enter these commands.

Terminal works all right but the ICD still doesn't emulate properly.

What's Probably Wrong:
There is a problem with data bus loading, interrupt processing, or the target system being disturbed during an emulation break.

What To Do:
Step 1. Select in-circuit mode 2 (I2) (see IN-CIRCUIT command) and start your program by entering the GO command. (This assumes there are ROMs in your target system. If there aren't any, then mode 2 will not work; proceed to Step 2.) Now test your target system. If it still doesn't work, then there is probably a data bus loading problem. Adding pull-up resistors to the data bus may help.

Step 2. If the in-circuit mode 2 works, try mode I1. If there are ROMs in the system, copy the ROMs to emulation memory (use the MOVE command). The start address is 0, and the end address is 07FF for 2K bytes, 0FFF for 4K bytes, and 1FFF for 8K bytes. If the ROMs are not all at adjacent addresses, then additional move commands will be needed. If there are no ROMs in the system, you will need to download the program from the host computer. Map all memory except the program memory to your target system (mapping code US). Select in-circuit mode 1 (I1), and start your program (GO command). Check to see if your target system is working properly now. If not, the problem could be related to interrupt processing (see next page).

Step 3. If the ICD works in the in-circuit mode 1 (I1), check for problems during an emulation break. If your target system works at the start of emulation, but fails when it is stopped and restarted, then the target system is probably being disturbed during an emulation break. This may be because your target system's design uses RD or M1 without gating them with MREQ. If this is the problem and you cannot modify your system, then the ICD can probably be modified by **ZAX**.

Interrupt Processing Problems:

Is the target system data bus buffered between the microprocessor and the peripheral chips? Are Z80 family peripheral chips (PIO, SIO, CTC) used? If the answer to either question is no, then the ICD should not cause any problems with interrupt processing.

If the data bus is buffered and Z80 peripheral chips are used, then the problem occurs when MREQ is not decoded by the buffer direction control logic. The easiest solution is to remove the data bus buffer and replace it with jumpers. If this is not possible, then the Emulation Data Bus connector (the connector labeled DB.EMUL on the ICD) can be connected to the buffered data bus. (See "More About Your ICD"—Data Bus Emulation Connector.)

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 or your target system is still not functioning correctly. If this is the case, you should consult directly with **ZAX** Corporation.

More About Your ICD

Introduction

Here you'll learn how to use the accessories that come with your ICD and what the Emulation Select switch does. By using the accessories and adjusting the settings on the switch, you'll be able to further expand your ICD's debugging capabilities.

From the following information, you will learn how to: 1) use the two accessory cables, 2) use the Data Bus Emulation connector, and 3) adjust the settings on the Emulation Select switch.

Accessory Cables

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.

Data Bus Emulation Connector

The Data Bus Emulation connector bypasses the Bi-directional Bus transceiver and forcibly outputs a RETI instruction to various Z80 peripheral chips (CTC, PIO, etc.) after an interrupt occurs.

Emulation Select Switch

The Emulation Select switch lets you: 1) use the Data Bus Emulation connector (by disabling the ICD's data bus from the target system's data bus), 2) send or suppress the RD signal, and 3) insert 1, 2, or 3 wait states into a machine cycle.

Accessory Cables & Probes

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 External 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 bootstrap 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.

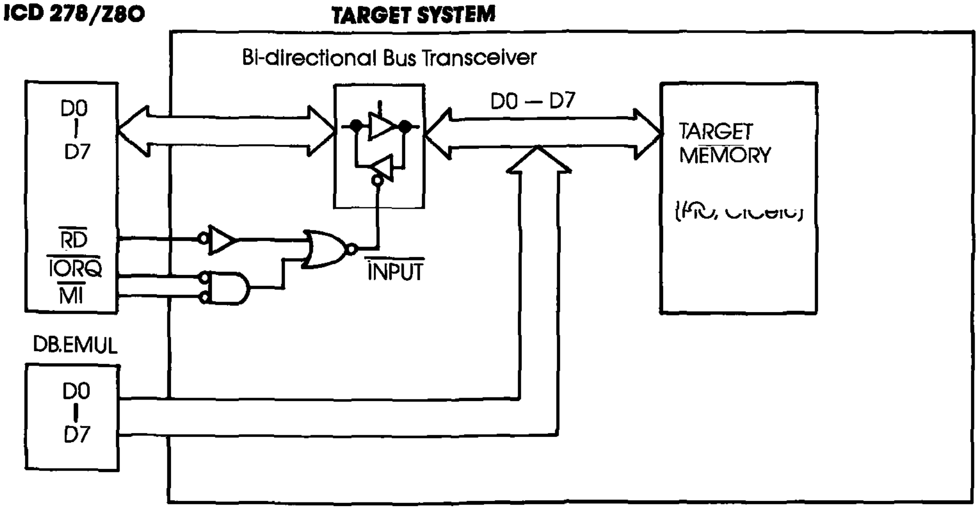
Data Bus Emulation Connector

Description The Data Bus Emulation Connector is an eight-pin socket connector with eight plug-in leads on the end of the connector.

Location Plugs into the DB.EMUL connector on the side of the ICD. (See "The Controls And Component Functions Of Your ICD.")

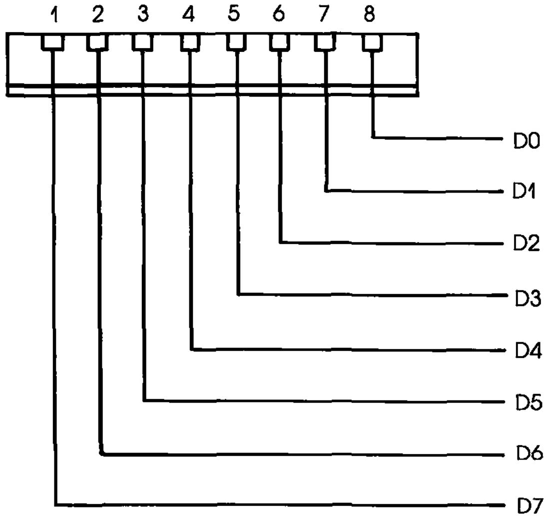
Function The Data Bus Emulation Connector is used to forcibly output a RETI instruction (from the ICD) to Z80 peripheral chips (PIO, CTC, SIO, etc.).

Application The Z80 uses a Bi-directional Bus Transceiver which is capable of transmitting and receiving signals through the same lines. If this data bus buffer is not pointed in the proper direction after an interrupt instruction, the Z80 peripheral chips will not recognize the RETI instruction. The easiest way to correct this problem is to bypass the data bus buffer and forcibly output the RETI instruction directly to the Z80 peripheral chips.



Using The Data Bus Emulation Connector

Connect the Data Bus Emulation Connector (socket side) to the pin connector labeled "DB.EMUL" on the end-panel of the ICD. Connect the eight leads directly to the dip-clip (included with the ICD) and then to the buffered data bus.



Emulation Select Switch

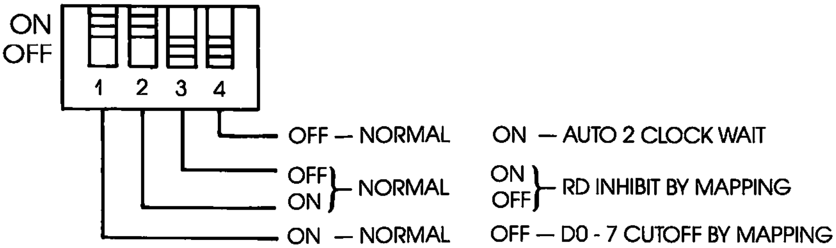
Description The Emulation Select Switch is a 4-bit, ON/OFF type switch.

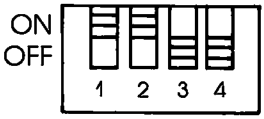
Location The E.M.SEL Switch end of the ICD. (See "The Controls And Component Functions Of Your ICD.")

Function The Emulation Select Switch disables the ICD's data bus from the target system's data bus (Bit 1), sends or suppresses the RD signal (Bits 2 & 3), and inserts 1, 2, or 3 wait states into the machine cycle (Bit 4).

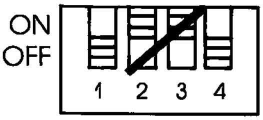
Application See the individual bit settings that follow.

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





NOTE: FACTORY BIT SETTINGS

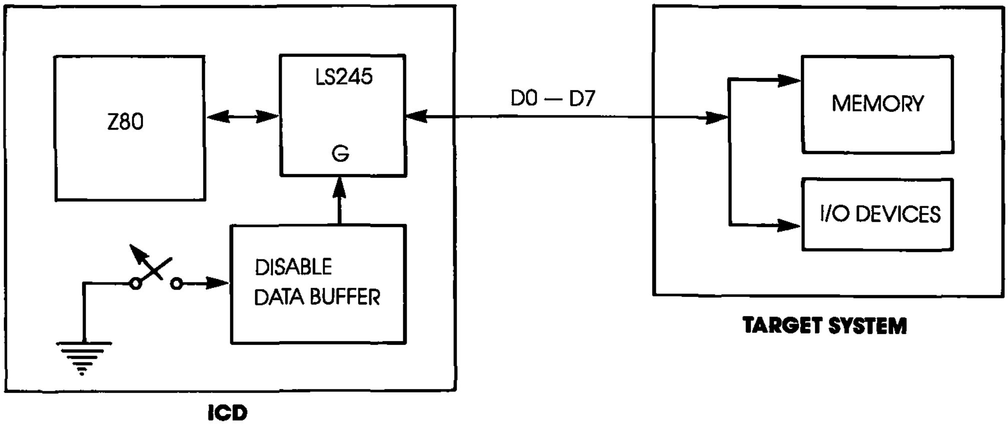


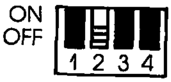
NOTE: DO NOT SET BITS 2 & 3 TO THE "ON" POSITION AT THE SAME TIME.

Bit ONE

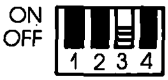
OFF—Disables the ICD's data bus (pins D0-D7) from the target system's data bus.

ON—D0-D7 output to the target system from the ICD's data bus (Normal setting).



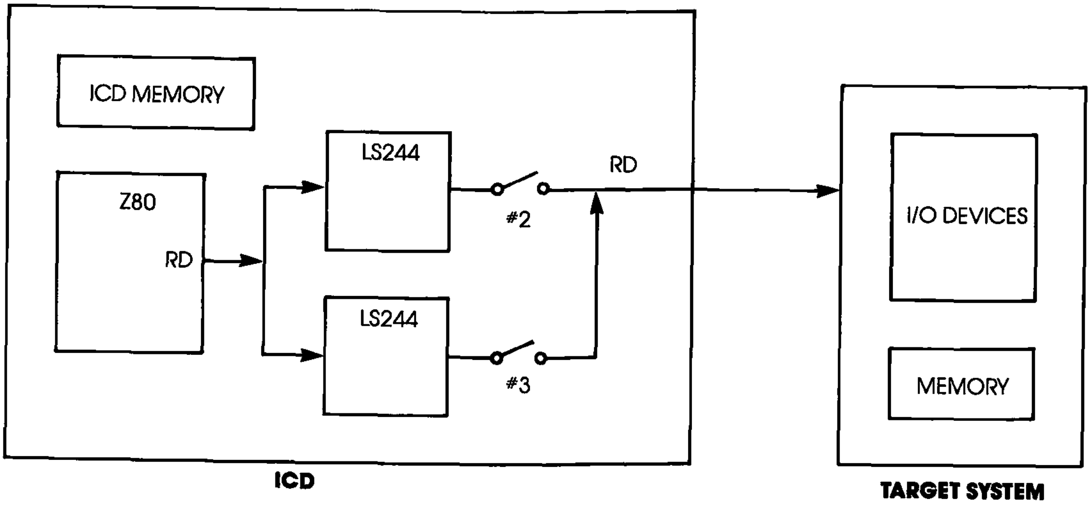


Bit TWO



Bit THREE

2 ON 3 OFF	Outputs the RD signal to the target system independently of the Mapping command (Normal setting).
2 OFF 3 ON	RD signal does not output to the target system when executing out of the ICD memory. Used in the in-circuit mode I1 only.



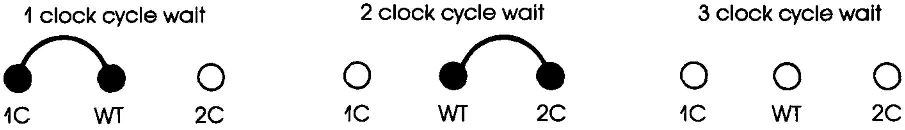
Bit FOUR

ON A 1, 2, or 3 clock wait is inserted in each machine cycle.

OFF No clock wait is inserted in machine cycle.

The wait state produced by the ICD-278 can hold for a period of two (optional one or three) clocks (wait states) by connecting the WT, 1C and 2C points on the S-793 CPU module.

Setting the wait state:



(Factory Setting)

Contents

SECTION 2 — MASTER COMMAND GUIDE

Contents

Section 2 MASTER COMMAND GUIDE

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

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 and changes the memory contents

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 serial input/output ports

PRINT—Sends the display to a printer

PIN—Enables or disables selected 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

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 there 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 it 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/0 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 must 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). These parameters specify a memory address with 16-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:

> **1A3X—Don't care condition in hexadecimal notation. May be specified in 4-bit units (0-F, or X).**
>
> **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), 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 8/16 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 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 D[/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 . . .

① **Command.** 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/development 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 information 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 "Example Of 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.

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 lets you quickly write your own routines, develop small programs, write hardware/software test routines, etc.

Syntax

```
A mem_addr <cr>
xxxx  (Z80 assembly code) <cr>
xxxx  <cr>
```

Terms *mem_addr* = The beginning memory address where assembled code is stored.

xxxx = The current storage location.

Z80 assembly code = The mnemonic instruction to be assembled and stored.

<cr> = Exits the assemble mode.

Syntax Example

```
>A  100
```

ASSEMBLE

Notes

The ICD will not accommodate the keyboard entry: EX AF,AF' (AF prime) as would normally be entered by a programmer. Instead, enter EX AF,AF (non-prime). The ICD interprets this correctly and will display EX AF,AF' on disassembly.

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).

Command Example

Execute this sequence:

```
>A 100       ←STARTS ASSEMBLING THE PROGRAM INTO ADDRESS 100H
0100 LD HL,0A000H
0103 PUSH DE
0104 LD DE,0B000H
0107 EX DE,HL
0108 POP DE
0109 INC HL
010A INC DE
010B JP 5000H
010E         ←PRESS THE RETURN KEY HERE TO END THE PROGRAM INPUT
>
>DI 100,110   ←DISPLAYS THE PROGRAM JUST ENTERED
```

BREAK

Command BREAK

Introduction The best way to safely stop a moving car is by using 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, whereas event points are used to trigger various external events, including stopping execution, but 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. Read about how to use the accessory cables and probes.)

There are 15 different BREAK command formats. See each format for an explanation and an example.

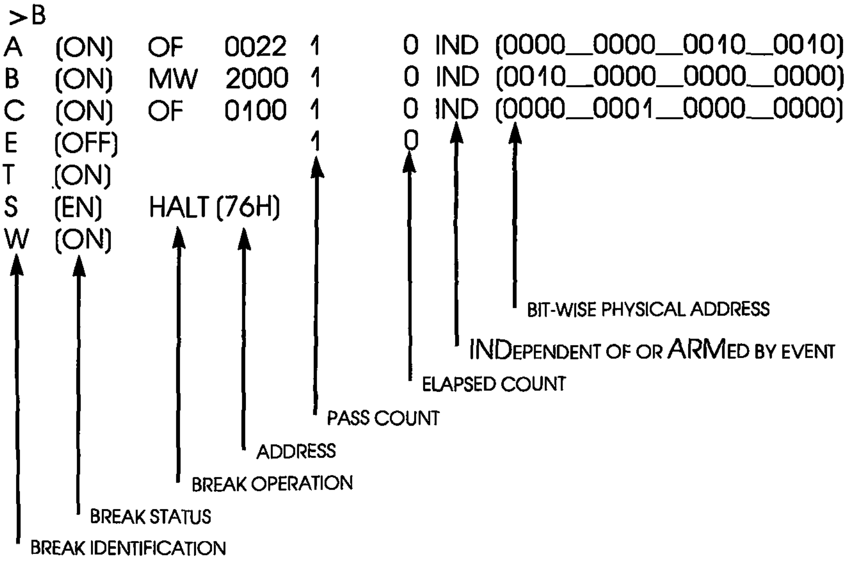
BREAK

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



```
>B
A  (ON)  OF   0022  1     0  IND  (0000_0000_0010_0010)
B  (ON)  MW   2000  1     0  IND  (0010_0000_0000_0000)
C  (ON)  OF   0100  1     0  IND  (0000_0001_0000_0000)
E  (OFF)              1     0
T  (ON)
S  (EN)  HALT (76H)
W  (ON)
```

NOTE: A,B,C = hardware break names, E = event break, T = ready timeout break, S = software break opcode, W = write-protect break.

BREAK

Command

BREAK: Hardware Breakpoint Qualification

Operation

Enables, disables, or clears the setting of the hardware breakpoints.

Applications Note: This command can be used to temporarily disable hardware breakpoints (set by the "BREAK: Hardware Breakpoint" command) 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.

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 above, 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)

addr = The address to break on.

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

Syntax Example B/C M,000X_111X_XXXX_0000

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

If the *name* is omitted, the next available breakpoint is used.

With this command, *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.

BREAK

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*. No spaces are permitted where commas are used as separators.

Command Example

Execute this sequence:

```
>B/B  OF,200      ←SPECIFIES HARDWARE BREAKPOINT
>B          ← CHECKS BREAKPOINT STATUS
B  (ON)  OF 0200     1      0  IND (0000__0010__0000__0000)
E  (OFF)
T  (ON)
S  (DI)  LD  A,A (7FH)
W  (ON)
>B/B OFF       ← DISABLES HARDWARE BREAKPOINT B
>B         ← CHECKS THE BREAKPOINT STATUS AGAIN
B  (OFF)  OF 0200     1      0  IND (0000__0010__0000__0000)
E  (OFF)
T  (ON)
S  (DI)  LD A,A (7FH)
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 op code 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 show that the change has been made.

BREAK

Command BREAK: Event then Hardware Breakpoint

Operation Causes a break in the program at a hardware breakpoint (A, B, and 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, and then to stop the program at the breakpoint.

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 *status*. If *name* is omitted, a space is required between B and *status.*

Command Example See Syntax Example above.

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 Spacing: A space is required betweeen B and INI.

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

Command Example Execute this sequence:

```
>A 100              ← ASSEMBLE PROGRAM WITH NESTED LOOPS THAT PERFORM
0100 LD HL,1000H      PERIODIC  WRITES TO MEMORY OF DECREMENTING DATA
0103 LD BC,0          VALUE
0106 DJNZ $
0108 DEC C
0109 LD (HL),C
010A JR NZ,-5
010C
>B/A OF,106          ← BEGIN EXECUTION
>G 100            ← SET BREAKPOINT FOR EXECUTION OF INNER LOOP
PC    MC         OP          SP   AF   BC   DE   HL   IX   IY   I  IF (SP)
0106  10FE       DJNZ $      0000 0000 FF00 0000 1000 0000 0000 00 0  FCFF
<Break Hardware A>      ← BREAK OCCURS IMMEDIATELY
>EV  ST=MW,A=10XX,D=33  ← SET EVENT TO OCCUR AFTER C REGISTER
>B/A ARM                   HAS BEEN DECREMENTED  TO 33H, AND
>B                         ARM THE BREAKPOINT WITH EVENT
A (ON)  OF 0106   1   0 ARM (0000__0001__0000__0110)
E (OFF)           1   0
T (ON)
S (DI)  LD A,A (7FH)
W (ON)
>G 100        ← BEGIN EXECUTION
```

BREAK

```
  PC   MC             OP          SP   AF   BC   DE   HL   IX   IY   I  IF (SP)
0106  10FE          DJNZ $        0000 0022 FF33 0000 1000 0000 0000 00 0  FCFF
                      <Break Hardware A>   ← NOTICE C REGISTER HAS BEEN DECREMENTED TO 33H
                      >B
                      Event Done    ← SHOWS EVENT HAS OCCURRED
                      A  (ON)  OF 0106          1    0 ARM (0000__0001__0000__0110)
                      E  (OFF)                  1    0
                      T  (ON)
                      S  (DI)           LD  A,A (7FH)
                      W  (ON)
                      >G  100     ← BEGIN EXECUTION AGAIN
PC    MC              OP          SP   AF   BC   DE   HL   IX   IY   I  IF (SP)
0106  10FE          DJNZ $        0000 0022 FF00 0000 1000 0000 0000 00 0  FCFF
                     <Break Hardware A>  ← ARM CONDITION STILL IN EFFECT FROM PREVIOUS EXECUTION
                     >B  INI                SO BREAK OCCURS IMMEDIATELY
                     >B            ← INITIALIZE THE ARM CONDITION, AND NOTICE THE CONDITION REMOVED
                     A  (ON)   OF 0106          1    0 ARM (0000__0001__0000__0110)
                     E  (OFF)                   1    0
                     T  (ON)
                     S  (DI)   LD  A,A (7FH)
                     W  (ON)
                     >G  100    ← BEGIN EXECUTION AGAIN
PC    MC              OP          SP   AF   BC   DE   HL   IX   IY   I  IF (SP)
0106  10FE          DJNZ $        0000 0022 FF33 0000 1000 0000 0000 00 0  FCFF
<Break Hardware A>          ← AGAIN, BREAK WAITS FOR EVENT ARM CONDITION
>
```

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 an LD A,A or a HALT 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 code.

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
```

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

If the *name* is omitted, the next available breakpoint is used.

With this command, *addr* can be specified by 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.

BREAK

A software breakpoint is not allowed to be specified in a USER-ROM area since a software breakpoint requires changing the memory contents at the specified location to an LD A,A or HALT 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.

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

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)          1000      1    0   ← SHOWS THAT SOFTWARE
E  (OFF)                   1    1     BREAKPOINT #5 IS
T  (ON)                               ACTIVE AT ADDR 1000
S  (EN)  LD  A,A (7FH)
W  (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 show that the change has been made.

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 above, 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. The ICD can use either HALT (76H) or LD A,A (7FH) to cause a software break within the user program. Either code may be selected by the user to conveniently cause a break in the program without having to continuously specify the breakpoint parameters.

Syntax B S=*op_code*

Terms *op_code* = HALT or LD A,A

Syntax Example B S=HALT

Notes The ICD defaults to LD A,A 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
E  (OFF)                    1     0
T  (ON)
S  (DI)    LD  A,A (7FH)     ← SHOWS SOFTWARE BREAK CODE
W  (ON)                         IS CURRENTLY LD  A,A
>B  S=HALT       ← CHANGES SOFTWARE BREAK CODE TO HALT
>B  S=EN         ← ENABLES ALL SOFTWARE BREAKPOINTS
>B          ← CHECKS THE BREAKPOINT STATUS AGAIN
E  (OFF)                    1     0
T  (ON)
S  (EN)    HALT (76H)      ← SHOWS THE SOFTWARE BREAK
W  (ON)                       CODE IS NOW HALT
>
```

This example shows how the software break code is changed from LD A,A to HALT and then enabled. The "BREAK: Status" command shows that the change has been made.

BREAK

Command BREAK: Software Breakpoint Qualification

Operation Enables, disables, or clears the software breakpoints.

Applications Note: This command can be used to temporarily disable software breakpoints (or all the breakpoints) without affecting their address locations within the program, or their parameter specifications.

Syntax B[/*name*] *switch*

Terms *name* = 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
E  (OFF)                 1    0
T  (ON)
S  (DI)    LD  A,A (7FH)
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)      07FF        1    0     ← SHOWS THE SOFTWARE
E  (OFF)                 1    0       BREAKPOINT IS ACTIVE
T  (ON)                               AT ADDR 7FF
S  (EN)    LD  A,A (7FH)
W  (ON)
>
>B/2  OFF      ← DISABLES SOFTWARE BREAKPOINT #2
>B        ← CHECKS THE STATUS AGAIN
2  (OFF)     07FF        1    0     ← SHOWS SOFTWARE
E  (OFF)                 1    0       BREAKPOINT #2 IS INACTIVE
T  (ON)
S  (EN)    LD  A,A (7FH)
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: 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; either from 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 External Breakpoint Qualification is always enabled.

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 the 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.

Command Example Execute this sequence:

```
>B       ← SHOWS THE BREAKPOINT STATUS
E  (OFF)                  1    0
T  (ON)
S  (DI)    LD  A,A (7FH)
W  (ON)
>B/X  HI     ← SETS SIGNAL RECOGNITION TO HIGH EDGE OF SIGNAL
>B
X  (ON)    HI             1    0    ← SHOWS EXTERNAL BREAK
E  (OFF)                  1    0      FEATURE IS ACTIVE
T  (ON)
S  (DI)    LD  A,A (7FH)
W  (ON)
>B/X  OFF     ← DISABLES EXTERNAL BREAK FEATURE
```

BREAK

```
>B       ← CHECKS BREAKPOINT STATUS AGAIN
X  (OFF)     HI          1    0  ←   SHOWS EXTERNAL BREAK
E  (OFF)                 1    0      FEATURE IS INACTIVE
T  (ON)
S  (DI)    LD  A,A (7FH)
W  (ON)
>B/X  CLR      ← CLEARS THE EXTERNAL BREAKPOINT FEATURE
>B
E  (OFF)                 1    0
T  (ON)
S  (DI)    LD  A,A (7FH)
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 disabled temporarily, 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 disables that feature.

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.

```
>EV        ← SHOWS EVENT STATUS
Event is Clear     ← SHOWS ABSENCE OF EVENT POINTS
>EV  ST=OF  A=7FF      ← SETS AN EVENT POINT IN PROGRAM
>EV        ← SHOWS NEW EVENT POINT SETTING
(ON)
Status       = OF
Address      = 07FF      (0000_0111_1111_1111)
Data         = XX        (XXXX_XXXX)
>B/E  ON       ← MAKES THE EVENT POINT ACTIVE IN PROGRAM
>B
E  (ON)                        1     0      ← SHOWS EVENT POINT SETTING
T  (ON)                                        IS ACTIVE
S  (DI)    LD  A,A (7FH)
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 point is enabled by using a breakpoint command. The "BREAK: Status" command is used again to verify that the event point is enabled (ON).

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

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 a 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. (The write protect 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
>MA                     FROM ADDRESS 0 TO FFF
In-Circuit Mode 0  (US= >RW)
0000-0FFF = RO       ←  SHOWS STATUS OF MEMORY IS READ-ONLY
1000-FFFF = RW          FROM ADDR 0 TO FFFF
>B/W  ON      ← ENABLES THE WRITE PROTECT FEATURE
>B
E  (OFF)                    1     0
T  (ON)
S  (DI)    LD  A,A (7FH)
W  (ON)
```

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 0 (debugging using the ICD's memory only), any area mapped as user (target system) memory is now remapped as read/write memory in the ICD. This causes all remaining memory areas to act as read/write memory. Next, the write protect feature is enabled (ON) 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. If the wait signal is activated 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 Time-out.

Applications Note: This break command can be used to flag an un-negated wait condition caused by the target system. This could be caused by a problem in the hardware, or it could be inherent in the design. If the problem lies in the design, this 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 activated wait 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*

Terms *switch* = ON or OFF

Syntax Example B/T OFF

Notes ON enables the timeout feature and OFF disables the timeout feature. (The timeout feature is automatically 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 above.

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

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 above. 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 100,200
DI 20
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). No spaces are permitted where a comma is used as the separator.

Command Example See Syntax Example above. The first example shows that the memory contents in the ICD are disassembled beginning from address 100 to address 200. In the second example, the ending address is omitted, which causes the memory contents to be disassembled from address 20 to address 002B (11 lines). In the third example, 11 instruction lines are displayed from the current PC. In the fourth example, the display is from the current PC to PC + 3FH.

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.*

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*. No space is permitted where a comma is used as the separator.

Command Example 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 1FFF.

EVENT

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. In one operation, the event point in the program can be used to externally trigger a peripheral device, such as a logic analyzer. The event point can also be used to internally trigger the real-time trace feature, which is defined by the HISTORY command. The event can also arm a hardware breakpoint in an A then B type sequence. And lastly, an event point can be used to stop the program in a manner similar to the BREAK command. The event, however, has the advantage of letting you specify a certain data pattern on the data bus, in addition to the normal address parameters, memory accesses, and I/O access conditions.

The event can also 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 To see how to use an event point as a breakpoint, first read about the EVENT command format here (for all four functions, the event point must be specified using the "EVENT: Specification" command), and then see the "BREAK: Event Breakpoint" command. To arm a hardware breakpoint, see "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. Read the chapter on using the accessory cables and probes.

EVENT

Command EVENT: Status

Operation Displays the current event point settings. When changes are made to the event point setting by using the "EVENT: Specification" command, this command can be 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:

```
>EV
(ON)   ← SHOWS EVENT SETTING IS ACTIVE
Status    = PR        ← PORT WRITE STATUS
Address   = 34        (0010_0100)    ← EVENT AT ADDRESS 34
Data      = 55        (0101_0101)    ← DATA VALUE TO MATCH
                                       FOR EVENT
```

This status display shows that the EVENT command has been enabled (ON), that the status of the event point is port read (PR), that the port is located at address 34, and that the matching data value for the event point is 55.

EVENT

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.

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

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

EVENT

Command EVENT: Specification

Operation Sets the condition parameters for an event point trigger.

Syntax EV [ST=*status*][,A=*addr*][,D=*data*]

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)
ANY	(don't care)

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

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

Syntax Example

```
EV ST=MR,A=100,D=55
EV A=250
```

Notes All parameters for this command are optional, and 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. Also, any undefined parameter defaults as "don't care."

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

EVENT

Spacing: A space is required between EV and any of the parameters. Spaces are not permitted between the parameters; commas are used to separate the parameters.

Command Example

Execute this sequence:

```
>EV          ← SHOWS EVENT STATUS
Event is Clear       ← SHOWS ABSENCE OF EVENT POINTS
>EV  ST=OF,A=7FF,D=41       ← SETS AN EVENT POINT IN THE PROGRAM
>EV
(ON)
Status    = OF
Address   = 07FF     (0000_0111_1111_1111)
Data      = 41       (0100_0001)
>EV  OFF        ← DISABLES THE EVENT POINT SETTING
>EV
(OFF)      ← SHOWS EVENT POINT SETTING IS DISABLED
Status    = OF
Address   = 07FF     (0000_0111_1111_1111)
Data      = 41       (0100_0001)
>
```

In this example, the event point status is first checked, an event point is set in the program, and the status is checked again. The event point is then disabled temporarily, as a check of the status shows.

EXAMINE

Command EXAMINE Only or EXAMINE and Modify

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 = New data for this location.

Notes If *mod_data* is omitted, the command enters a repeat mode which allows several locations to be changed. When /W option is selected, the word will be displayed or entered in LSB/MSB (Least Significant Bit/Most Significant Bit) order (bytes swapped).

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.

EXAMINE

Command Example

```
>E 0
0000  54=74,      ← CHANGE VALUE TO 74H AND RE-EXAMINE
0000  74=      ← LEAVE VALUE UNCHANGED, GO TO NEXT ADDRESS
0001  68=      ← LEAVE VALUE UNCHANGED, GO TO NEXT ADDRESS
0002  69='a'      ← CHANGE VALUE AND GO TO NEXT ADDRESS
0003  73=74^      ← CHANGE VALUE AND GO TO PREVIOUS ADDRESS
0002  61=^      ← LEAVE VALUE UNCHANGED, GO TO PREVIOUS ADDRESS
0001  68=,      ← LEAVE VALUE UNCHANGED, RE-EXAMINE ADDRESS
0001  68=^      ← LEAVE VALUE UNCHANGED, GO TO PREVIOUS ADDRESS
0000  74=/      ← LEAVE VALUE UNCHANGED, EXIT COMMAND
>E/W 20
0020  BFOA=4455,      ← CHANGE WORD VALUE, RE-EXAMINE
0020  4455=      ← LEAVE VALUE, GO TO NEXT LOCATION
0022  6DFF='HI',      ← CHANGE VALUE (ASCII), RE-EXAMINE
0022  4948=      ← LEAVE VALUE, GO TO NEXT LOCATION
0024  FFFE=      ← LEAVE VALUE, GO TO NEXT LOCATION
0026  EB29=^      ← LEAVE VALUE, GO TO PREVIOUS LOCATION
0024  FFFE=0/      ← CHANGE VALUE AND EXIT COMMAND
>E  20
0020  55=      ← EXAMINE ONLY
0021  44=
0022  48=
0023  49=
0024  00=
0025  00=/
>
```

FILL

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

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

Command Example See Syntax Example above. This example fills memory from address 100 to address 3FF with a data value of 55.

GO

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  0,800
```

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. One hardware breakpoint each must be available to activate both the *end_addr* or *end_addr#2* parameters.

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

Example Command

See Syntax Example above. 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 800.

HISTORY

Command HISTORY (Real-time Tracing)

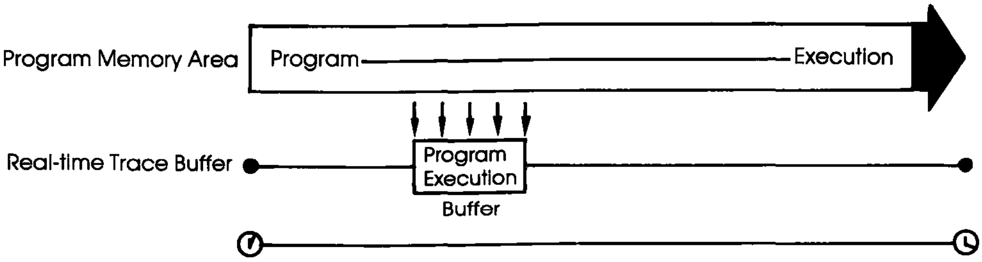
Introduction The real-time trace is one of the most powerful and useful features of your ICD. It allows you to record (hence the name "History" command) and then analyze a specific section of program execution, rather than sift through the entire program looking for a problem. Event points (which you set in the program) can trigger the real-time trace buffer to start or stop the data storage process when program execution begins, or continues until a break occurs.

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 (see the PRINT command). In this way, if a problem develops during the program execution, the real-time trace provides a record that can be reviewed to determine what the problem is.

Trace Width and Depth An emulator's trace memory should be wide enough to accommodate the processor's address and data lines. With the ICD-278 for Z80, the trace memory is 32 bits wide (8 bits data/16 bits address/8 bits status). But 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. However, if the trace memory depth is insufficient, the chances of recording the trace section where the problem exists are greatly diminished. Your ICD has a maximum trace memory depth of 2K (2047) machine cycles, but 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 size permits very specific tracing.

HISTORY

Real-time Trace Buffer The data that is recorded from the program execution is stored in the real-time trace buffer. The real-time trace buffer can be thought of as a data storage facility that moves along parallel to the user program, storing the same data that is being executed by the user program. The storage capacity of the real-time trace buffer is 2K machine cycles, and, in certain modes, when the buffer is full, it begins storing new data on a "First In/First Out" (FIFO) basis—writing over the oldest data it has stored. In this way, the buffer always displays the latest data it has stored.



HISTORY

Trigger Modes

Triggers are the real workhorses of the real-time trace. They determine where (and when) the trace section is recorded within the user program. Your ICD features six different trigger modes, and specifications about each of the trigger modes are shown below.

BEGIN MONITOR
Specified by: H BM,trace_range
Activated by: GO command
Terminated when: Buffer full
FIFO when buffer full? No
Range affects: Storage size
End result in buffer: First 2K cycles executed

END MONITOR
Specified by: H EM
Activated by: GO command
Terminated when: Break in execution
FIFO when buffer full? Yes
Range affects: Nothing (ignored)
End result in buffer: Last 2K cycles executed

BEGIN EVENT
Specified by: H BE,trace_range
Activated by: An event point
Terminated when: Buffer full
FIFO when buffer full? No
Range affects: Storage size
End result in buffer: 2K cycles following event

CENTER EVENT
Specified by: H CE,trace_range
Activated by: GO command
Terminated when: An event point + range # of cycles
FIFO when buffer full? Yes
Range affects: Offset of event from center
End result in buffer: 2K surrounding event

HISTORY

END EVENT
Specified by: H EE
Activated by: GO command
Terminated when: An event point occurs
FIFO when buffer full? Yes
Range affects: Nothing (ignored)
End result in buffer: Event point + previous 2K cycles

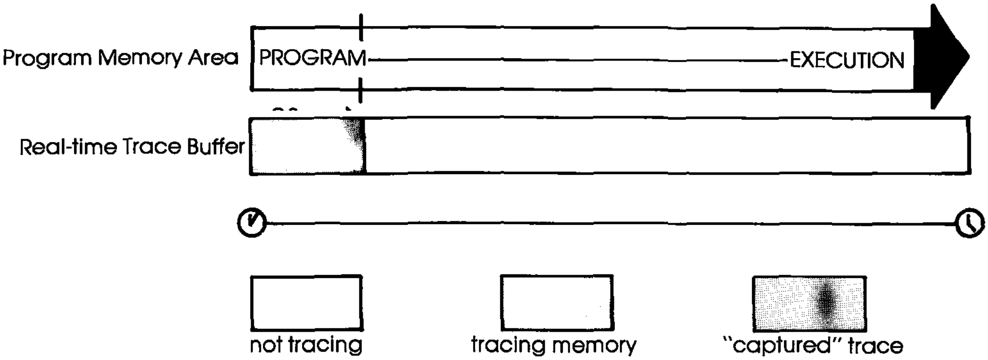
MULTIPLE EVENT
Specified by: H ME,trace_range
Activated by: An event point
Terminated when: Buffer full
FIFO when buffer full? No
Range affects: Temporary storage termination until another event point
End result in buffer: Several "snapshots" triggered by event points

HISTORY

Simplest case: Begin Monitor Mode

An easy way to understand how the real-time trace works is to examine the Begin Monitor mode. With 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.

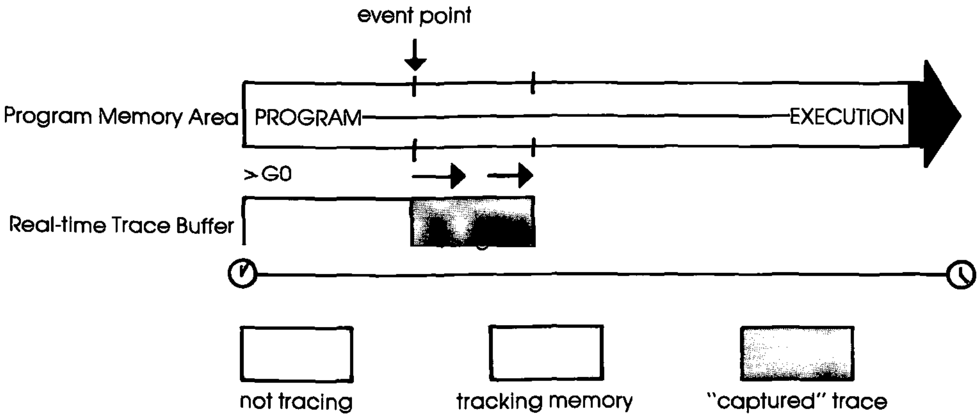
After the user program executes (and the buffer stores) the data equivalent of the range, the trace buffer fills to that point and then stops. The data that is now stored in the buffer is the "captured" trace section (the section that the ICD displays). The real-time trace then enters a non-trace mode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



HISTORY

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 2K) and then stops.

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

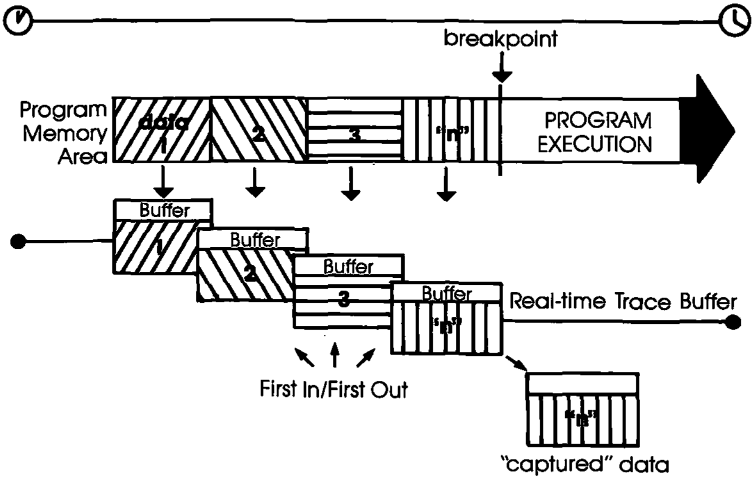


HISTORY

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 2K before the breakpoint or MONITOR break.

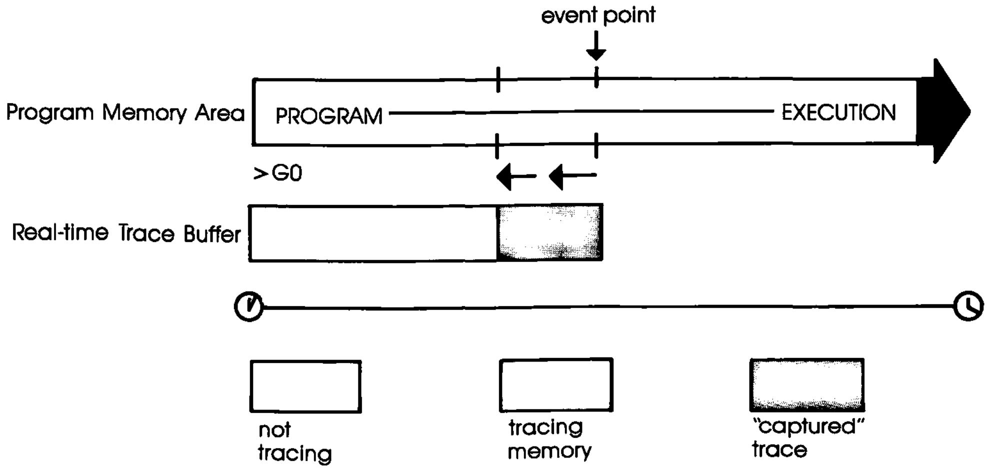
The ICD accomplishes this type of tracing by recording and storing data on a First-In/First-Out (FIFO) basis after the buffer is filled. By using this technique, the ICD always displays the latest data in the trace buffer.

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



HISTORY

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 stop storing data. The captured trace section is the last 2K before and including the event point.

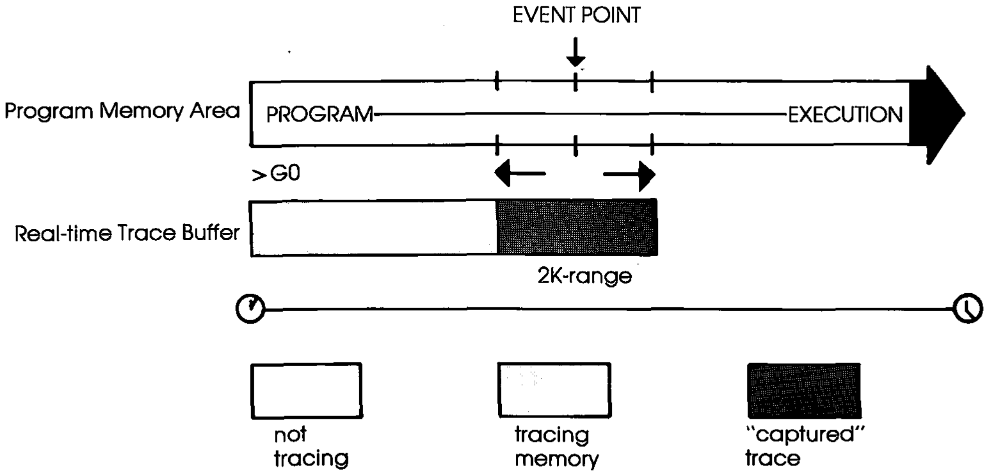


HISTORY

Center Event Mode

The Center Event mode is used when you desire the trace to surround a single event point in the program. The Center Event mode takes the range specification and records that number of cycles after the event point occurs. The remainder of the 2K buffer contains cycles just prior to and including the event point. For example, if 1K is specified as the range, 1K of data is captured before the event point, and 1K is captured after the event point. If the specified range is 2000, 45 cycles would be captured before the event, and 2000 after.

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

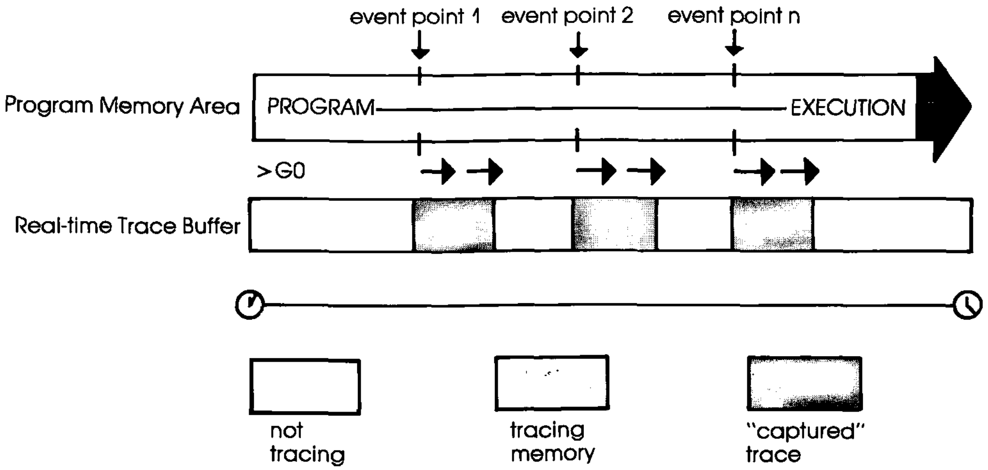


HISTORY

Multiple Event Mode

The Multiple Event mode is identical to the Begin Event mode, with the exception that when the trace range is filled, the tracing is only temporarily stopped until another event point occurs. Then the buffer is re-opened to continue storing another trace range number of cycles. When the 2K buffer is filled, the event points are then ignored, and the buffer remains in a non-storage mode. This allows several occurrences of the event to trigger the History buffer, giving several "snapshots" of a particular routine.

NOTE: The smaller the trace range, the more times the event can retrigger the buffer before it fills and begins to ignore event points.



HISTORY

Command HISTORY: Real-time Trace Status

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

Applications Note: Use the real-time trace status to analyze the condition of the real-time trace buffer, i.e., storage mode name, size of the trace range, the number of cycles executed, and the 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

```
>H
Clock Counts = 00000000/0     ← NUMBER OF CLOCK CYCLES
Storage Mode = BE  2045      ← MODE AND TRACE RANGE
Storage Size = 0/0     ← NUMBER OF CYCLES PASSED
```

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 "Begin event" mode has been specified and that the trace range is 2045. "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). "Full" indicates a full buffer, or 2045 cycles.

HISTORY

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 the examples that begin on page 2-63.

HISTORY

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 or D

int_point = Initial point of display, from 1 to 2047.

term_point = Point at which display terminates, from 1 to 2047.

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. With this command, *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*. No spaces are permitted where commas are used as separators.

Command Example See the examples that begin on page 2-63.

HISTORY

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 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 2045.

Syntax Example H ME,800

Notes The range specified for the EM and EE modes will be ignored; it defaults to the maximum 2K size.

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

Command Example See the examples that begin on page 2-63.

HISTORY

HISTORY: Real-time Trace Command Examples

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

Example trace mode: End Monitor
Command format: H EM
Trace range: 2K

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

Execute the following:

```
>I  0             ← SPECIFY IN-CIRCUIT MODE 0
>F  0,1FFF,0         ← FILLS 0 TO 1FFF WITH NOPs (IT TAKES A FEW SECONDS
                       FOR THE ICD TO DO THIS)
>B/A  OF,1770          ← SETS A HARDWARE BREAKPOINT TO TERMINATE
                         EMULATION
>G  0             ← STARTS EMULATION AND INITIATES REAL-TIME TRACE STORAGE.
                    ICD RUNS PROGRAM, STOPS AT BREAKPOINT A, AND DISPLAYS:
```

```
0000 0000 0000 0000 0000 0000 0000 00  0  0000
```

```
1770  00  NOP
<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   DT      ST      OP
†2047     0F73   00              NOP
 2046     OF74   00              NOP
  ↓        ↓      ↓               ↓
 0003     176F   00              NOP
 0002     1770   00              NOP
 0001     PAUSE
```

NOTE: POINT = *Address in* HISTORY *Buffer,* T = *Event point indicator,* ADDR = *Cycle address,* DT = *Cycle data,* OP = *Op code.*

†*NOTE: Displays memory contents from beginning of storage pointer.*

HISTORY

Example trace mode: Begin Monitor
Command format: H BM
Trace range: 2K

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

Execute the following:

```
>H   CLR          ← RESETS THE CLOCK COUNTER
>H   BM          ← SETS THE REAL-TIME TRACE TO THE BEGIN MONITOR MODE
>B/A   OF,0FA0        ← SETS A HARDWARE BREAKPOINT TO TERMINATE
                        EMULATION
>G   0          ← STARTS EMULATION AND INITIATES THE REAL-TIME TRACE STORAGE.
                  ICD RUNS PROGRAM, STOPS AT BREAKPOINT A, AND DISPLAYS:
```

```
  PC     MC     OP     SP     AP     BC     DE     HL     IX     IY     I    IF   (SP)
<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  DT      ST      OP
2047   *  0000  00              NOP
2046      0001  00              NOP
  ↓        ↓     ↓               ↓
0002      07FD  00              NOP
0001      PAUSE
```

**NOTE: Indicates trigger point.*

HISTORY

Example trace mode: Begin Event
Command format: H BE
Trace range: 2K

This example continues from the Begin Monitor example.

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

```
 PC   MC   OP   SP   AF   BC   DE   HL   IX   IY   I   IF  (SP)
IF40  00  NOP  0000 0000 0000 0000 0000 0000 0000  00   0  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  DT     ST     OP
2002   *  1770  00            NOP
2001      1771  00            NOP
 ↓         ↓     ↓             ↓
0002      1F40  00            NOP
0001      PAUSE
```

**NOTE: Indicates event point.*

HISTORY

Example trace mode: Center Event
Command format: H CE
Trace range: 2K

This example continues from the Begin Event example.

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

```
 PC    ME    OP    SP    AF    BC    DE    HL    IX    IY    I    IF   (SP)
 IF40  00   NOP  0000  0000  0000  0000  0000  0000  0000   00    0   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  DT      ST      OP
2047      1371  0               NOP
2046      1372   00             NOP
 ↓         ↓     ↓               ↓
1025      176F   00             NOP
1024   *  1770   00             NOP
 ↓         ↓     ↓               ↓
0002      1B6E   00             NOP
0001      PAUSE
```

NOTE: *Indicates event point.*

HISTORY

Example trace mode: End Event
Command format: H EE
Trace range: 2K

This example continues from the Center Event example.

Execute the following:

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

```
 PC    ME   OP    SP   AF   BC   DE   HL   IX   IY    I   IF  (SP)
1770   00   NOP  0000 0000 0000 0000 0000 0000 0000  00   0   0000
<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  DT      ST      OP
2047      0F73  00              NOP
2046      0F74  00              NOP
 ↓         ↓     ↓               ↓
0005      176D  00              NOP
0004      176E  00              NOP
0003      176F  00              NOP
0002   *  1770  00              NOP
0001      PAUSE
```

**NOTE: Indicates event point.*

HISTORY

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

ICD Displays:     Your Response:
    0FFE            JP   0H <cr>
    1001            <cr>
>H   CLR        ← RESETS THE CLOCK COUNTER
>H   ME,100        ← SETS THE REAL-TIME TRACE TO THE MULTIPLE EVENT MODE
                     AND THE STORAGE SIZE AS 100 INSTRUCTIONS PER LOOP
>B/A   0F,F00,50      ← SETS A HARDWARE BREAKPOINT TO TERMINATE
                         EMULATION
>EV   ST=0F,A=0700      ← SETS AN EVENT POINT
>B/E   OFF        ← DISABLES PROGRAM BREAK BY AN EVENT POINT
>G   0        ← STARTS EMULATION
```

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

```
 PC   MC   OP    SP    AF    BC    DE    HL    IX    IY    I    IF   (SP)
0F00  00   NOP  0000  0000  0000  0000  0000  0000  0000  00    0   0000
<Break Hardware A>
>
```

HISTORY

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   DT      ST      OP
2047      075F   00              NOP
2046      0760   00              NOP
 ↓         ↓     ↓               ↓
1922   *  0700   00              NOP
 ↓         ↓     ↓               ↓
0002      0764   00              NOP
0001      PAUSE
```

**NOTE: Indicates event point.*

HISTORY

Command HISTORY: Real-time Trace Search

Operation Allows you to search 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,/[*data*]/[*cycle*][,*int_point*][,*term_point*]

Terms *addr* = Value to search for.

data = Data to search for.

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

MR	(memory read)
MW	(memory write)
PR	(port read)
PW	(port write)
M1	(opcode fetch)
IA	(interrupt acknowledge)
HA	(halt acknowledge)

int_point = Initial point of display, from 1 to 2047.

term_point = Point at which display terminates, from 1 to 2047.

Syntax Example H S,/100/55/MR,200,100

HISTORY

Notes

If *data* is specified, *addr* specification is also required. The *int_point* defaults to 2047, 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. If *addr*, *data*, or *cycle* is excluded, slashes must still be present (e.g., H S,///MR).

Command Example

See Syntax Example.

IDENTIFICATION

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-278 for Z80             V2.0
```

This example shows that the ICD emulates the Z80 processor and that the firmware version within the ICD is 2.0. Your firmware version may be different, depending on your purchase date.

IN-CIRCUIT

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.

IN-CIRCUIT

Command IN-CIRCUIT: Specification

Operation Sets the ICD mapping mode. 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 enabled. Any area mapped as NO (non-memory) will act as NO memory regardless of the in-circuit mode.

IN-CIRCUIT

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

In-circuit	Memory Type				PIN Functions	
Mode/Description	RO	RW	US	NO	EN	DI
I0/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.

MAP

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  0      ← USES ICD'S MEMORY RESOURCES
>MA        ← SHOWS HOW MEMORY IS CATEGORIZED
In-Circuit Mode 0     (US= >RW)
0000-FFFF = RW
```

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 in-circuit mode is indeed 0, that user (target system) memory now acts as read/write memory (US= >RW), and that the entire memory area is categorized as read/write. 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)
0000-FFFF = RW
```

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 0000 to FFFF) now functions as user (target system) memory.

MAP

Command MAP: Specification

Operation Categorizes your 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 mainframe 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=RW
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*.

1K block areas are predefined as addresses X000-X3FF, X400-X7FF, X800-XBFF, X000-XFFF.

MAP

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 breakpoint is disabled) will also cause a break.

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 causes a break in the 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'S AND TARGET SYSTEM'S MEMORY RESOURCES
>MA  0000,0FFF = RO
>MA  1000,1FFF = US     ← CATEGORIZES MEMORY BLOCKS
>MA  2000,FFFF = RW
>
>MA        ← SHOWS HOW THE MEMORY IS CATEGORIZED
In-Circuit Mode 1
0000-0FFF = RO
1000-1FFF = US
2000-FFFF = RW
```

MAP

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 FFFF). 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'S MEMORY RESOURCES
>MA        ← SHOWS HOW THE MEMORY IS CATEGORIZED
In-Circuit Mode 2 (RW,RO= >US)
0000-0FFF = RO
1000-1FFF = US
2000-FFFF = RW
```

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

```
>I  0      ← USES ICD'S MEMORY RESOURCES
>MA        ← SHOWS HOW THE MEMORY IS CATEGORIZED
In-Circuit Mode 0 (US= >RW)
0000-FFFF = RW
```

In this example, the I 0 (debugging using ICD memory only) is selected, which is the default condition for the ICD. When the status of the MAP command is examined, it shows that user (target system) memory acts as read/write memory. Read-only memory can still be specified in this mode.

MOVE

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 100,200,100,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 above. In this example, a block of memory in the target system, beginning at address 100 and ending at address 200, is moved to the ICD beginning at address 100.

NEXT

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 program counter.

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.*

NEXT

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

```
>F  0,FFF,0      ← FILLS MEMORY WITH NOPS
>G  0,2FF       ← STARTS THE PROGRAM RUNNING FROM ADDRESS 0
                  AND STOPS AT ADDRESS 2FF, THEN DISPLAYS:

 PC   MC   OP    SP   AF   BC   DE   HL   IX   IY    I    IF   (SP)
02FF  00  NOP  0000 0000 0000 0000 0000 0000 0000  00    0   0000
 <Break Hardware A>      ← PROGRAM BREAKS AT ADDR 2FF
 >N  3     ← SHOWS THE NEXT THREE INSTRUCTION LINES

 PC   MC    OP    SP   AF   BC   DE   HL   IX   IY    I    IF   (SP)
0300  00   NOP  0000 0000 0000 0000 0000 0000 0000  00    0   0000
0301  00   NOP  0000 0000 0000 0000 0000 0000 0000  00    0   0000
0302  00   NOP  0000 0000 0000 0000 0000 0000 0000  00    0   0000
>
```

This example illustrates how the NEXT command is used with the program execution. When the program stops at address 2FF, entering N 3 causes the next three instruction lines to be displayed.

OFFSET

Command OFFSET: Status

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

Syntax O

Command Example

```
>O      ← 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 conditions of the OFFSET command. Changing the address of any one of the four offset values (0-3) causes a change in the 0000 display.

OFFSET

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 `O &number[=addr]`

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

addr = Offset to place in the register.

Syntax Example `O &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 O and &. No spaces are permitted between &*number*=*addr*; the equal sign (=) acts as the separator.

OFFSET

Command Example Execute this sequence:

```
>O  &1=2240       ← SETS #1 VALUE TO OFFSET OF 2240
>O      ← SHOWS CURRENT OFFSET VALUES
&0 = 0000
&1 = 2240
&2 = 0000
&3 = 0000
>DI  212&1       ← DISASSEMBLES FROM ADDRESS 212 + THE OFFSET VALUE
2452  ....      ....     P,OF92FH    ← DISPLAY BEGINS
                                       AT 2240 + 212 = 2452
2455  ....      ....
2456  ....      ....     A,B
2457  ....      ....
....  ....      ....
etc.  ....      ....
```

NOTE: . . . . = Example display.

PIN

Command PIN: Status

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

Syntax PI

Command Example

```
>PI        ← SHOWS STATUS OF INTERRUPT SIGNALS
In-Circuit Mode 1
NMI/     (EN) = H
INT/     (EN) = H
BUSRQ/   (EN)
```

This example shows that the in-circuit mode is 1, which allows the interrupt signals to be individually enabled and disabled. With this example, all interrupt signals are enabled (EN). If the be enabled, and if the float was 0, all interrupt signals would automatically be disabled. H shows that the current logic levels of the signal are high. The slash (/) after the signal name signifies an "active-low" signal.

PIN

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)
INT (Maskable Interrupt Request)
BUSRQ (Bus Request)

switch = EN or DI

Syntax Example PI BUSRQ=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*.

PIN

Command Example Execute this sequence:

```
>I  1       ← SETS MODE TO PERMIT INTERRUPT FEATURE
>PI      ← SHOWS STATUS OF INTERRUPTS
In-Circuit Mode 1
NMI/      (EN) = H
INT/      (EN) = H
BUSRQ/  (EN)
>PI  INT=DI      ← DISABLES THE INT SIGNAL
>PI      ← SHOWS THE STATUS OF INTERRUPTS AGAIN
In-Circuit Mode 1
NMI/      (EN) = H
INT/      (DI) = H      ← VERIFIES THE CHANGE
BUSRQ/  (EN)
>
```

In this example, the in-circuit mode 1 is selected (ICD and target system memory resources) to manipulate the various interrupt signals. The PIN status then shows that all the interrupts are active (*EN*abled). Next, the Interrupt Request (INT) signal is disabled, and the PIN status used again to verify the change.

PORT

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. P *port_addr*[=*mod_data*]
Terms	*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 (/) 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. See EXAMINE command examples for additional information.

PRINT

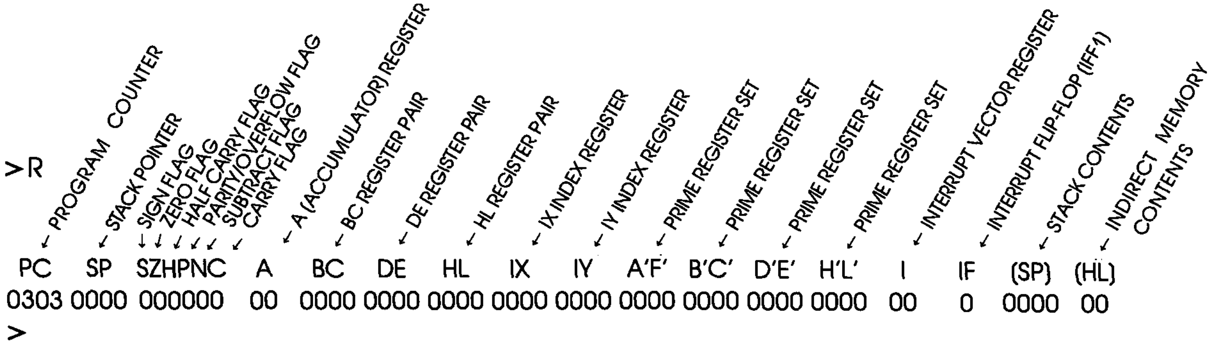
Command	PRINT
Operation	Controls logging of ICD commands by sending the terminal display to an external serial printer.
Syntax	PR *switch*
Terms	*switch* = ON or OFF
Syntax Example	PR ON
Notes	ON enables the printing feature and OFF disables the printing feature. The printing is routed to the HOST/AUX 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 above.

REGISTER

Command REGISTER: Status

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

Syntax R



```
>R
 PC   SP   SZHPNC  A   BC   DE   HL   IX   IY   A'F' B'C' D'E' H'L'  I    IF  (SP) (HL)
0303 0000 000000  00  0000 0000 0000 0000 0000 0000 0000 0000 0000  00    0  0000  00
>
```

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

REGISTER

Command REGISTER: Reset

Operation Sets all the registers to zero.

Syntax R RESET

Notes Spacing: A space is required between R and RESET.

Command Example Execute this sequence:

```
>R        ← SHOWS THE STATUS OF THE REGISTERS
>R  A=AF     ← SETS REGISTER A TO A VALUE OF AF
>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 A is changed from 00 to AF, and then set back to 00 using the "REGISTER: Reset" command.

REGISTER

Command REGISTER: Examine and Change

Operation Examines and changes the contents of the Z80 internal registers.

Syntax R *reg_name*[=*data*]

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

A A' PC SP IX IY
B C BC B' C' BC'
D E DE D' E' DE'
H L HL H' L' HL'
I IF F F' S Z P N CY

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_name*(s) IF,S,Z,P,N, and CY, 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.

REGISTER

Command Example Execute this sequence:

```
>R  DE=2FFF     ← SETS THE DE REGISTER TO 2FFF
>R  DE       ← SHOWS THE VALUE OF THE DE REGISTER
2FFF      ← VALUE OF DE REGISTER
>R      ← SHOWS THE VALUES OF ALL THE REGISTERS
```

```
>R
 PC   SP   SZHPNC  A   BC   DE    HL   IX   IY   A'F' B'C' D'E' H'L'  I  IF  (SP) (HL)
0000 0000 000000  00  0000 2FFF  0000 0000 0000 0000 0000 0000 0000 00  0  0000  00
```

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

SEARCH

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 above. 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 55H.

SUPERVISOR

Command SUPERVISOR

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

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

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 E register, and the I/O data is transferred via the A register.

Omitting all parameters will display the current supervisor call settings.

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

SUPERVISOR

Command Example Execute this sequence:

```
>A  100        ← STARTS ASSEMBLING THE SAMPLE PROGRAM
                 FROM ADDRESS 100
0100  LD   HL,120H
0103  LD   E,2
0105  LD   A,(HL)
0106  INC  HL
0107  OR   A
0108  JR   NZ,-3
010A  LD   A,A
010B       ← A <cr> HERE TERMINATES THE INPUT
>
>B  S=EN       ← ENABLES ALL SOFTWARE BREAKPOINTS
>B/C  OF  106     ← SETS HARDWARE BREAKPOINT C
>SU/C  ON      ← USES BREAKPOINT C AS A SUPERVISOR CALL
>F  120,139,'This is a SUPERVISOR call'    ← MESSAGE
>F  13A,143,'message',0D,0A,00    ← MESSAGE
>G  100      ← RUNS PROGRAM FROM ADDRESS 100
This is a SUPERVISOR call message    ← DISPLAY SHOWS MESSAGE
                                       THEN STOPS AT USER
                                       BREAKPOINT

 PC    MC   OP          SP    AF    BC    DE    HL    IX    IY    I    IF  (SP)
010A   7F   LD   A,A  0000  0044  0000  0012  0144  0000  0000  00    0   0FFB
<Break User>
>
```

SUPERVISOR

Supervisor Function Code Key for ICD-278 for Z80

Port Input Status Fetch

Entry Conditions:

Register E = 01H	Get input status from TERMINAL Port
Register E = 11H	Get input status from HOST/AUX Port

Exit Conditions:

Register E	Unchanged
Register A = 00H	No data is available at specified Port
Register A = FFH	Data has been received at specified Port

Input Character from Port

Entry Conditions:

Register E = 00H	Input character from TERMINAL Port
Register E = 10H	Input character from HOST/AUX Port

Exit Conditions:

Register E	Unchanged
Register A =	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 E = 03H	Get output status from TERMINAL Port
Register E = 13H	Get output status from HOST/AUX Port

Exit Conditions:

Register E	Unchanged
Register A = 00H	Port transmit buffer is busy (not ready)
Register A = FFH	Port transmit buffer is empty (ready)

Output Character to Port

Entry Conditions:

Register E = 02H	Output character to TERMINAL Port
Register E = 12H	Output character to HOST/AUX Port
Register A =	Character to be output to specified Port

Exit Conditions:

Register E	Unchanged
Register A	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 CODE	DATA OUT	DATA IN
FUNCTION	E-reg	A-reg	
TERMINAL Port data in	0 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

TRACE

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-FFFF     ← SHOWS ALL SPECIFICATION
>T   J        ← SETS TRACE TO JUMP ONLY DISPLAY
>T          ← DISPLAYS NEW TRACE SETTING
(ON)         JUMP 0000-FFFF     ← SHOWS JUMP SPECIFICATION
```

TRACE

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 its parameter specifications.

Syntax T *switch*

Terms *switch* = ON, OFF, or CLR

Syntax Example T ON

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

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

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

TRACE

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 = A or J

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

end__addr = Ending address of memory to trace (default = FFFF).

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 (JP & JR) instructions are displayed.

If *beg__addr* is omitted, the trace starts from address 0. If *end__addr* is omitted, the trace ends at address FFFF. 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.

TRACE

Command Example Execute this sequence:

```
>A  1000      ← STARTS ASSEMBLING PROGRAM AT ADDR 1000
1000  LD  BC,0
1003  XOR  A
1004  ADD  A,C           SIMPLE PROGRAM THAT
1005  LD  C,A            FINDS THE SUMATION
1006  INC  B             OF 0+1+2+3+ . . . +n,
1007  LD  A,B            WHERE N = OFH. THE
1008  CP  10H            RESULT (78H) CAN BE
100A  JP  NZ, 1004H      FOUND IN MEMORY
100D  LD  A,C            LOCATION 1020H.
100E  LD  (1020H),A
1011      ←<CR> HERE TO TERMINATE ENTRY
>
>DI  1000     ← DISPLAYS THE PROGRAM JUST ENTERED
>T  A 1000,1100     ← TRACES ALL INSTRUCTIONS FROM ADDRESS 1000
                      TO ADDRESS 1100
>G  1000      ← STARTS THE PROGRAM FROM ADDRESS 1000 AND
                DISPLAYS IT AS IT RUNS; PRESS Esc TO EXIT
>T/S  A  1000,1100  ← TRACES ALL INSTRUCTIONS AND DISPLAYS
                      ONE LINE AT A TIME
>G 1000      ← STARTS PROGRAM AND DISPLAYS ONE LINE AT A TIME;
               PRESS SPACE BAR TO SCROLL; PRESS Esc TO EXIT
>T  J     ← DISPLAYS ONLY JUMP (JP) INSTRUCTIONS
>G 1000      ← STARTS PROGRAM AND DISPLAYS ONLY JUMP INSTRUCTIONS;
               PRESS SPACE BAR TO SCROLL; PRESS Esc TO EXIT
>T  CLR      ← CLEARS THE TRACE FEATURE
```

This example is illustrated by first entering a simple program so that a trace can be performed on the program. After the program is entered, it is disassembled for inspection, and then the trace parameters are specified. This example shows that a trace is made of all the instruction lines from address 1000 to address 1100. The program is then run from address 1000 , and the trace is displayed. The next command shows how the Jump (JP) instruction command is specified. Finally, the trace feature is cleared from the ICD memory.

USER

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, when the system is operating in the LOCAL (terminal control of the ICD) mode. 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 [illegible] 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.

Spacing: A space is required between U and *code*.

Syntax Example

```
U !
U ~
```

LOAD

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 and using a computer for storage) and REMOTE (ICD controlled by a host computer running ZICE software) modes.

Syntax L[/*source*] *filename*[,*bias*][,*message*]

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

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

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

message = Any ASCII message (in 'single' quotes) or hex data, or any combination separated by commas.

Syntax Example

```
L/H    TEST.ABS,100
L/A    ,,'TYPE TEST.HEX',0D
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.)

The *message* is sent out the *source* port at the beginning of the load operation to provide a way of prompting the host computer to begin transmitting a file.

LOAD

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), this 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 above. 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 100H 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 message is sent to the host computer, followed by a carriage return (specified by 0D—which is its ASCII code) prompting the host computer to transmit the file TEST.HEX to the ICD.

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.

SAVE

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,beg_addr,end_add,entry_addr* [,*message*]

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

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

beg_addr = First address to save.

end_addr = Last address to save.

entry_addr = Starting address of the user program.

message = Any ASCII message (in 'single' quotes) or hex data, or any combination separated by commas.

Syntax Example

```
SA/H   TEST.HEX,0,3FF,0
SA/A   TEXT.HEX,0,1FFF,0,'CREATE TEXT.HEX',0D
```

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.)

SAVE

The *message* will be sent out the destination port at the beginning of the save operation to provide a way of prompting the host computer to receive a file. (Remember to use the USER command to access the host and to terminate the file input.)

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.

VERIFY

Command

VERIFY

Operation

Compares an Intel Hex format file on the host computer to the ICD memory (or through the ICD to the user 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*[,*bias*][,*message*]

Terms

source = T, P, A, or H.

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

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

message = Any ASCII message (in 'single' quotes) or hex data, or any combination separated by commas.

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 sofware 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.)

The *message* is sent out the *source* port at the beginning of the load operation to provide a way of prompting the host computer to begin transmitting a file.

VERIFY

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.

HOST

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 ZICE. 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 version 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 above.

QUIT

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

Command Syntax Summary

ASSEMBLE	A mem_addr <cr> xxxx (Z80 assembly code) <cr> xxxx <cr>
BREAK	B [INI] B[/name] switch B[/name] status, addr[,passcount] B[/name] addr[,passcount] B S=switch B S=op_code B/X edge[,passcount] B/X switch B/E switch B/W switch B/T switch
COMPARE	CO beg_addr,end_addr,comp_addr[,direction]
DISASSEMBLE	DI [beg_addr] [,end_addr]
DUMP	D[/W] beg_addr[,end_addr]
EVENT	EV [switch] EV [ST=status] [,A=addr] [,D=data]
EXAMINE	E[/W] [/N] beg_addr[=mod_data]
FILL	F[/W] [/N] beg_addr,end_addr,data
GO	G [beg_addr] [,end_addr] [,end_addr#2]
HISTORY	H [CLR] H mode[,int_point] [,term_point] H mode[,range] H S,/[addr]/[data]/[cycle] [,int_point] [,term_point]
IDENTIFICATION	ID

IN-CIRCUIT	I [mode]
MAP	MA [beg_addr[,end_addr]=area]
MOVE	M beg_addr,end_addr,mov_addr[,direction]
NEXT	N [steps]
OFFSET	O [&number=addr]
PIN	PI [signal=switch]
PORT	P port_addr[=mod_data]
PRINT	PR switch
REGISTER	R [RESET] R reg_name[=data]
SEARCH	S[/W] [/D] beg_addr,end_addr,search_data
SUPERVISOR	SU[/break switch]
TRACE	T [switch] T[/S] mode[,beg_addr] [,end_addr]
USER	U code
LOAD	L[/source] filename[,bias] [,message]
SAVE	S[/destination] filename,beg_addr,end_add,entry_addr [,message]
VERIFY	V[/source] filename[,bias] [,message]
HOST	H switch
QUIT	Q

NOTE: Items in brackets ([ ]) *are optional.*

Contents

SECTION 3 — TECHNICAL REFERENCES

Contents

Introduction

In this section you'll learn about the five internal control modules which, with the power supply, make up your ICD. These modules are used to control the various processes that are required for emulation, including such things as electronically substituting your target system's microprocessor with the ICD's own processor, controlling communication betweeen the ICD and host computer or terminal, and tracing (and storing) a portion of the program memory contents for analysis.

Special Environments

Although it is not necessary to read this section to use your ICD, you may find the information in this section helpful if you require an examination of how the ICD operates under certain conditions and in particular environments. In certain instances, some modules may need to be modified to permit the ICD to operate at peak performance with some systems. All possible modifications are detailed in each module description.

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 shows you how to disassemble your ICD and remove (and replace) the five control modules.

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 which are not marked by this symbol. (See page 3-37 for a complete description on this subject.)

What Are The Five Control Modules?

Indicator/Control Module

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

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," located at the end of this section, after reading about the module's components.]

Real-time Trace Module

This module allows tracing of the program memory contents and storage of the captured data in the real-time trace buffer. [There are no user-serviceable controls on this module.]

CPU Control Module

This module contains the connectors, circuitry, and Z80 microprocessor. which allow the ICD to emulate the target system's this module—see "How To Disassemble Your ICD," located at the end of this section, after reading about the module's components.]

Emulation Memory (Unit) Module

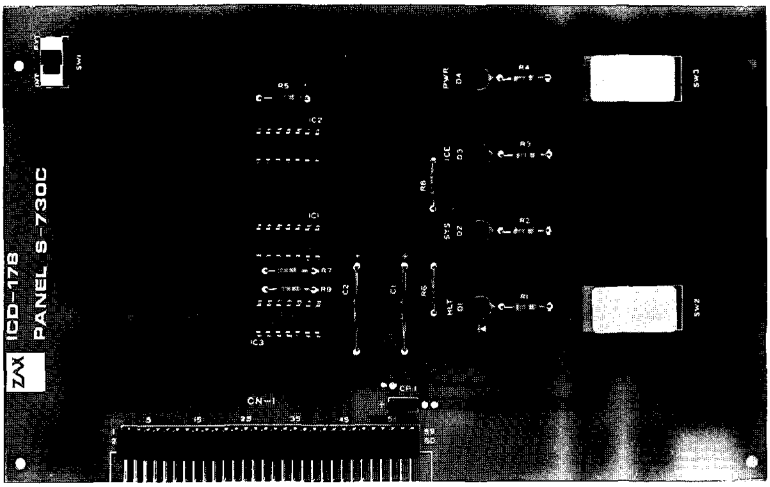
This module contains 64K bytes of high-speed static RAM (known as "emulation memory") which can be used for downloading files, altering the memory contents, and loading future memory into the target system. [There are no user-serviceable components on this module.]

Indicator/Control Module

Description

The Indicator/Control module (S-730) contains three switches, four indicator lamps, one 60-pin connector, 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 the outside of the ICD, so there are no user-serviceable controls or components on this module.



Serial Interface Output Module

Description

The Serial Interface Output (SIO) module (S-791) controls the communication between the ICD and various external devices (host computer, terminal, or 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) which are used to set the data format and stop bits for the TERMINAL and HOST/AUX ports, and a special socket which allows any key on the console keyboard to activate the MONITOR break switch in the ICD.

These components are all user-serviceable, which means the ICD must be disassembled before they can be adjusted or modified (see "How To Disassemble Your ICD" located at the end of this section).

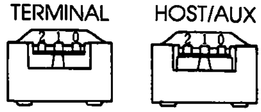
The module's remaining components are all externally accessible. These include the DCE/DTE and LOCAL/REMOTE switches, the TERMINAL and HOST/AUX port connectors, and two rotary switches which set the communication baud rates for the ports.

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.



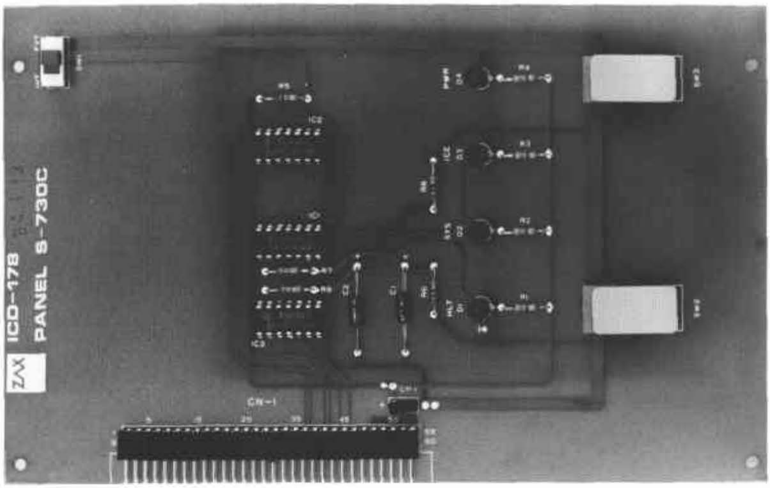
Baud Rate Switch No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Baud Rate (bps)	19.2K	9.6K	4.8K	2.4K	1.2K	600	300	150	75	110	134.5	200	1.8K	2K	—	—

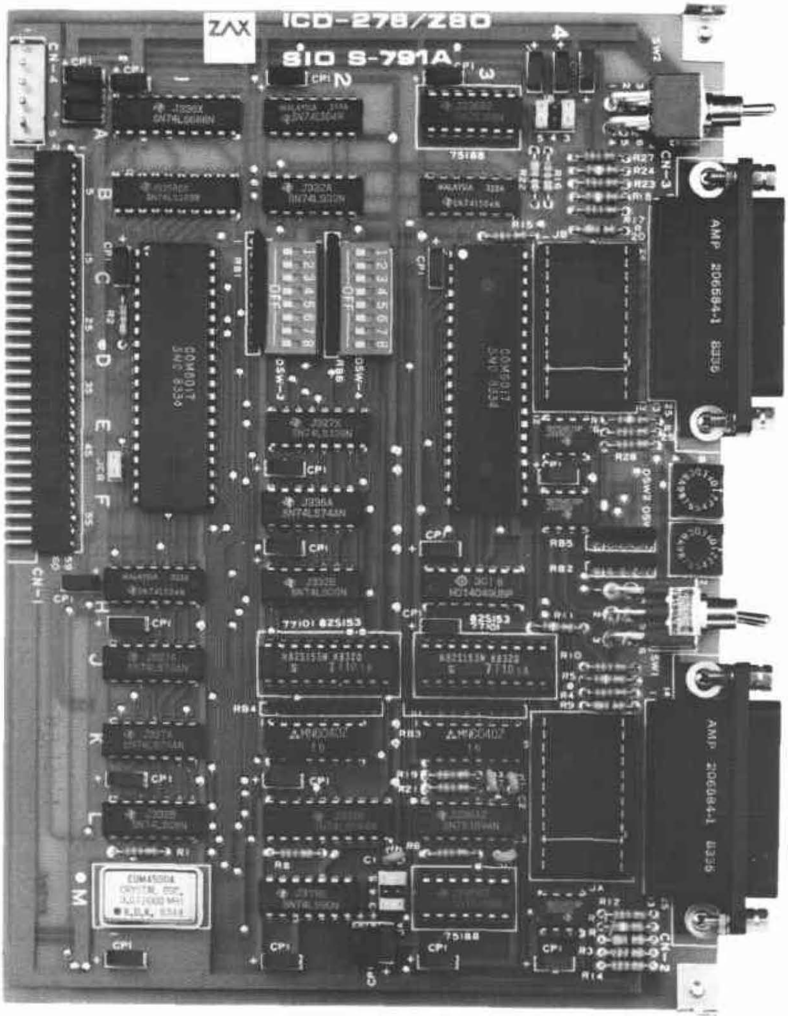
Indicator/Control Module

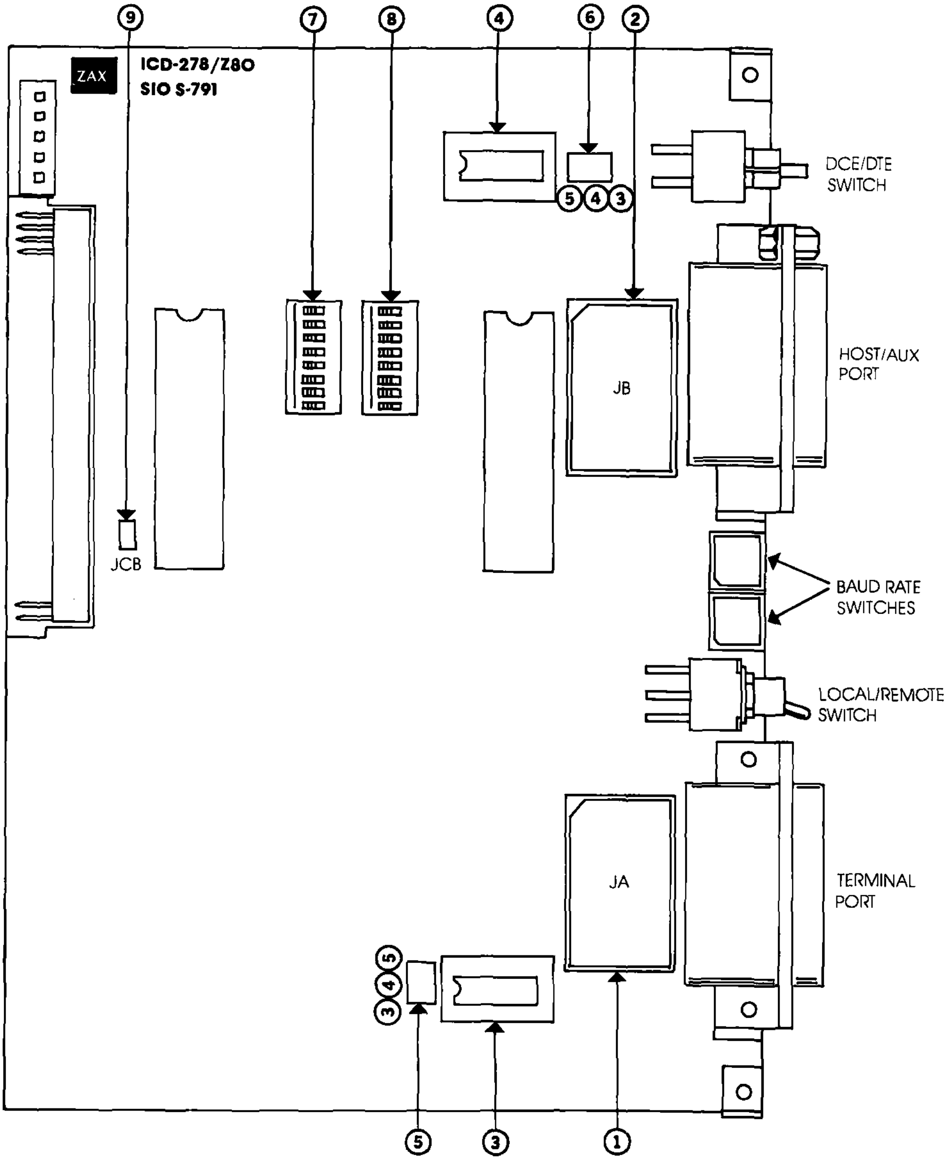
Description

The Indicator/Control module (S-730) contains three switches, four indicator lamps, one 60-pin connector, 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 the outside of the ICD, so there are no user-serviceable controls or components on this module.




ZAX
ICD-278/Z80
SIO S-791A
CN-1
CN-2
CN-3
CN-4
AMP 206584-1 8335
75188



SIO S-791 SERIAL INTERFACE OUTPUT MODULE

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. (See "How To Change The Interface Settings.")

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 5/4/3, 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.")

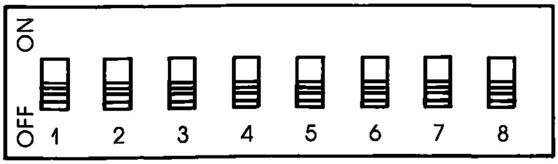
⑧ **DSW4 Transmission Format Switch.** Sets the data format and stop bits for the HOST/AUX port. (See "How To Set The Transmission Format Switches.")

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.)

▲ 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

Factory settings of the transmission format switches for the TERMINAL and HOST/AUX ports are shown below.

TERMINAL Port	HOST/AUX Port
8 data bits	8 data bits
2 stop bits	2 stop bits
no parity bit	no parity bit

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

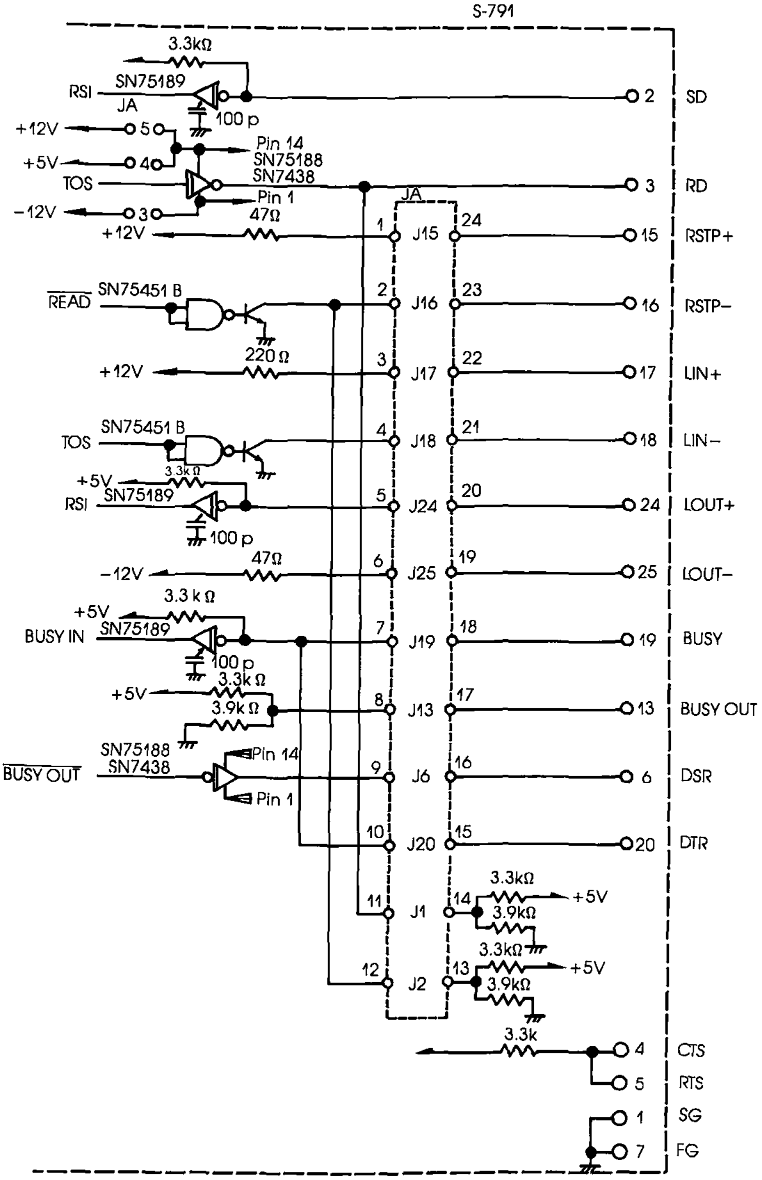
Signals for multiple ICDs can I/O through the HOST/AUX port by setting bits 6 and 7. When this feature is enabled, the External Break, Emulation Qualify, and Event Trigger signals can be monitored by more than one ICD. (I/O level is EIA.)

To activate this feature, set the following bits:

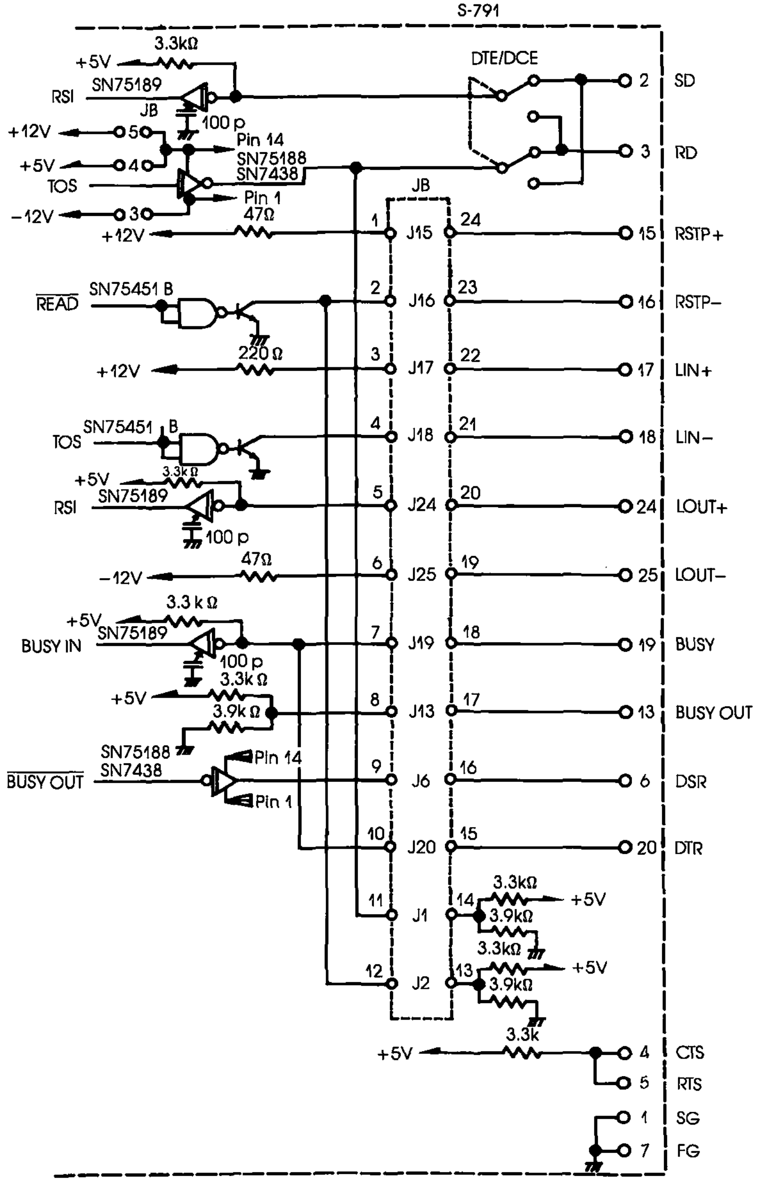
DSW3 bit 6 = ON   DSW4 bits 6 & 7 = ON

This feature affects 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



SIO S-791 DIAGRAM (TERMINAL PORT)

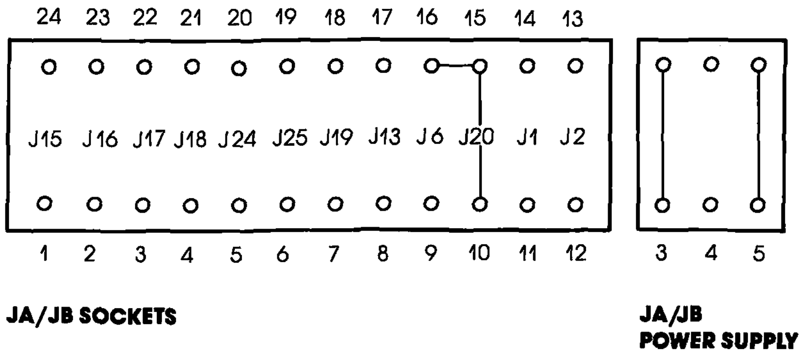


SIO S-791 DIAGRAM (HOST/AUX PORT)

RS-232 Interface

The RS-232 interface is the normal configuration for the ICD. The diagram below shows how the pins on the JA and JB sockets are arranged for the RS-232 setting. 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.

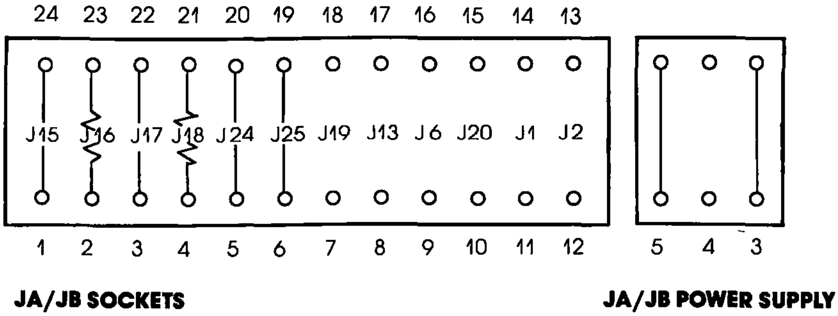
NOTE 2: CTS and RTS signals are looped back (null modem) within the ICD and pulled up to +5V.

NOTE 3: Connecting pins 15 and 16 (JA and JB socket) together causes the DTR and DSR signals to be looped back (null modem) within the ICD. Connecting pins 10 and 15 (JA and JB socket) causes the DTR signal to be used as the BUSY signal to the terminal. Connecting 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 and 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 pins 4 and 21 (JA18/JB18) together with a 220-ohm, 1/4-watt resistor, or adjust the resistance to the associated circuit.

b) Connect pins 2 and 23 (JA16/JB16) together with a 47-ohm, 1/4-watt resistor.

c) Connect the other pins as shown in the Current Loop Interface diagram.

d) Set the DCE/DTE select switch on the ICD 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	OUT	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

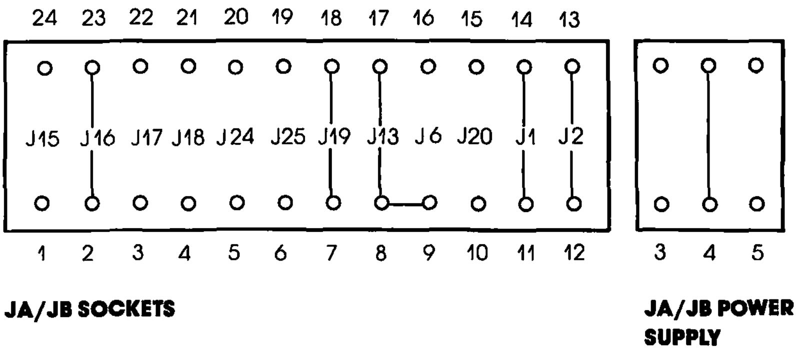
NOTE 1: Pin 25 is the current source pin for current loop input signals pulled down to −12V.

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 and JB sockets are modified. The diagram below shows how the pins on the JA and 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.

TTL Interface
(Modified connection is shown)



▲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/JB 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 and JB socket) causes the DTR signal to be used as the BUSY signal to the terminal.

XON and XOFF Protocol

XON/OFF 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:

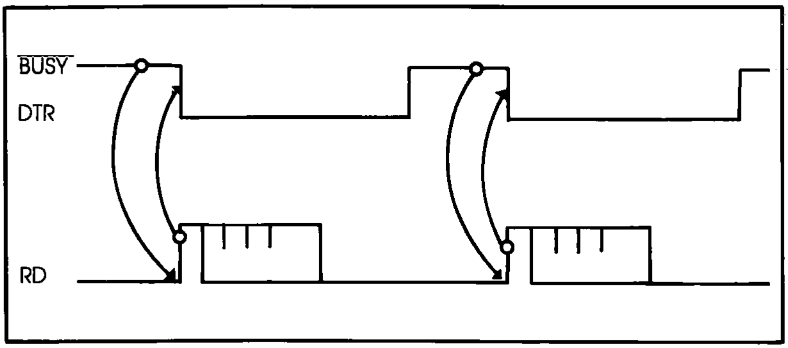
1. The host computer or terminal sends XOFF to the ICD before the reception buffer overruns.
2. When the reception buffer is ready, the host computer or terminal sends XON to the ICD and resumes reception.

The control codes for XON/OFF signals are:

XON —DC3 (CTR-S; 13H)
XOFF—DC1 (CTR-Q; 11H)

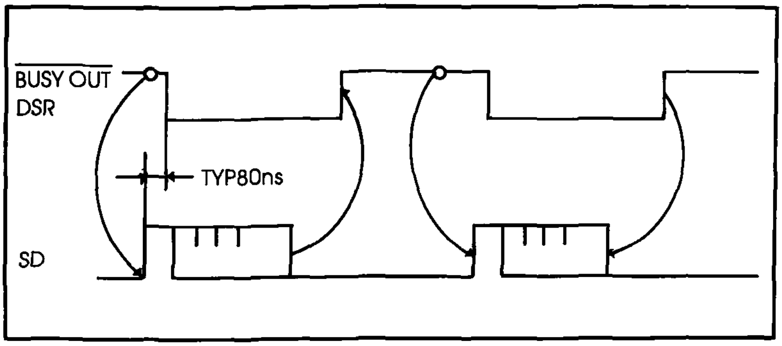
BUSY and DTR Input Signals

The BUSY signal sent from a low-speed terminal can be used to stop the ICD from transmitting data. Normally, 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.



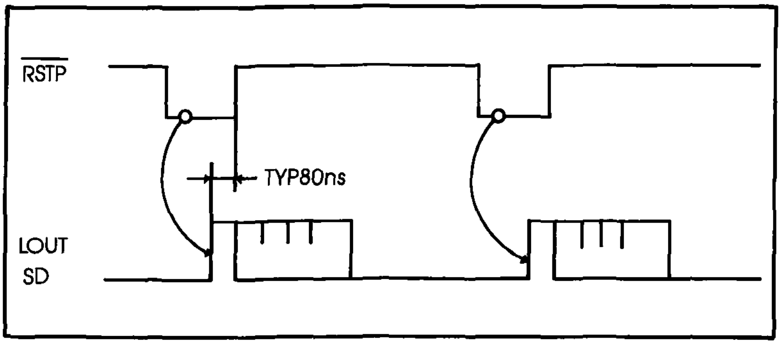
BUSYOUT and DSR Output Signals

When a host computer sends data at a higher speed than the ICD's internal monitor processor can accept, the BUSYOUT signal of 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.

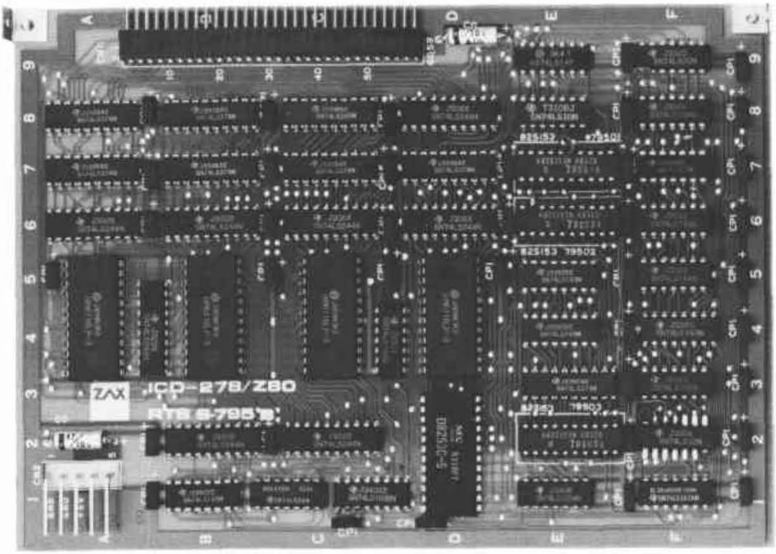


Real-time Trace Module

Description

The Real-time Trace module (S-795) allows you to record a portion of the program memory area and store the data in the real-time trace buffer. 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, so there are no user-serviceable controls or components. (For a complete description of how the real-time trace feature works, see the HISTORY command in the Master Command Guide.)

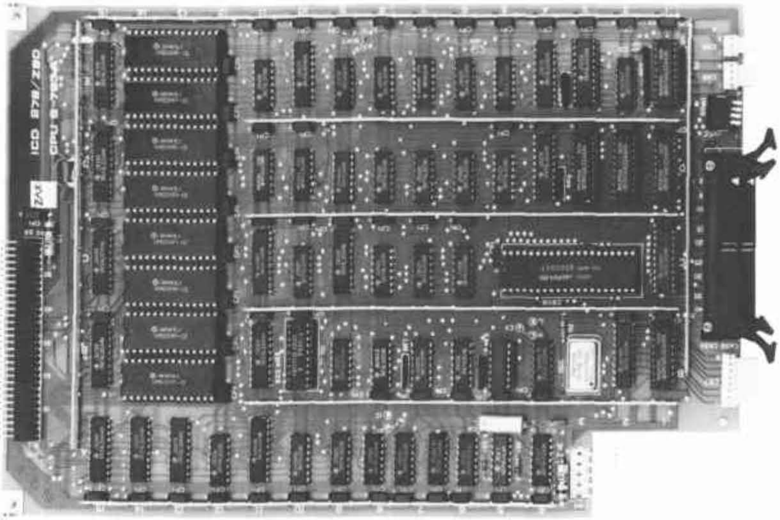


CPU Control Module

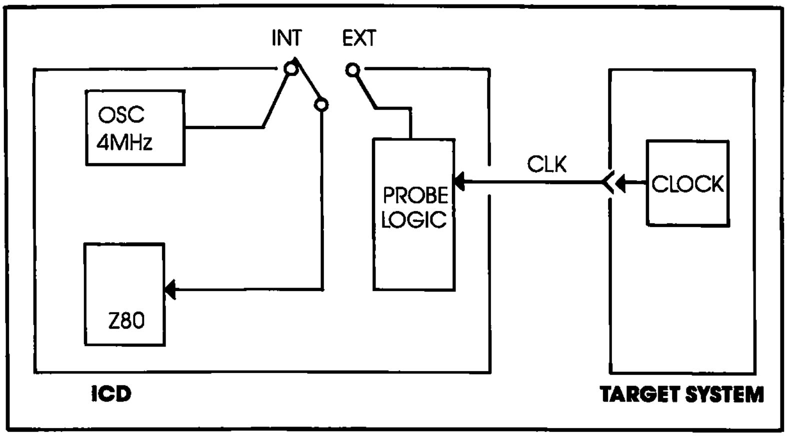
Description

The CPU Control module (S-793) contains the connectors, circuitry, and Z80 microprocessor, which allow the ICD to emulate the target system's processor.

The only user-serviceable components on this module are the H, CX, and L jumpers, which allow you to set the ICD's internal clock speed to either 2 MHz or 4 MHz. The remaining components are all externally accessible. These include the CPU probe connectors (which connect the ICD to the target system), the Data Bus Emulation connector, the External Break connector, the Event Trigger connector, and the Emulation Select switch.



Internal and External Clock



▲ How to Change the Internal Clock

Selecting the INT setting on the INT/EXT switch enables the ICD's internal clock. The internal clock normally runs at a speed of 4 MHz with a 50% duty cycle, but can be changed to 2 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 = 4 MHz) or low (L = 2 MHz) clock speed.

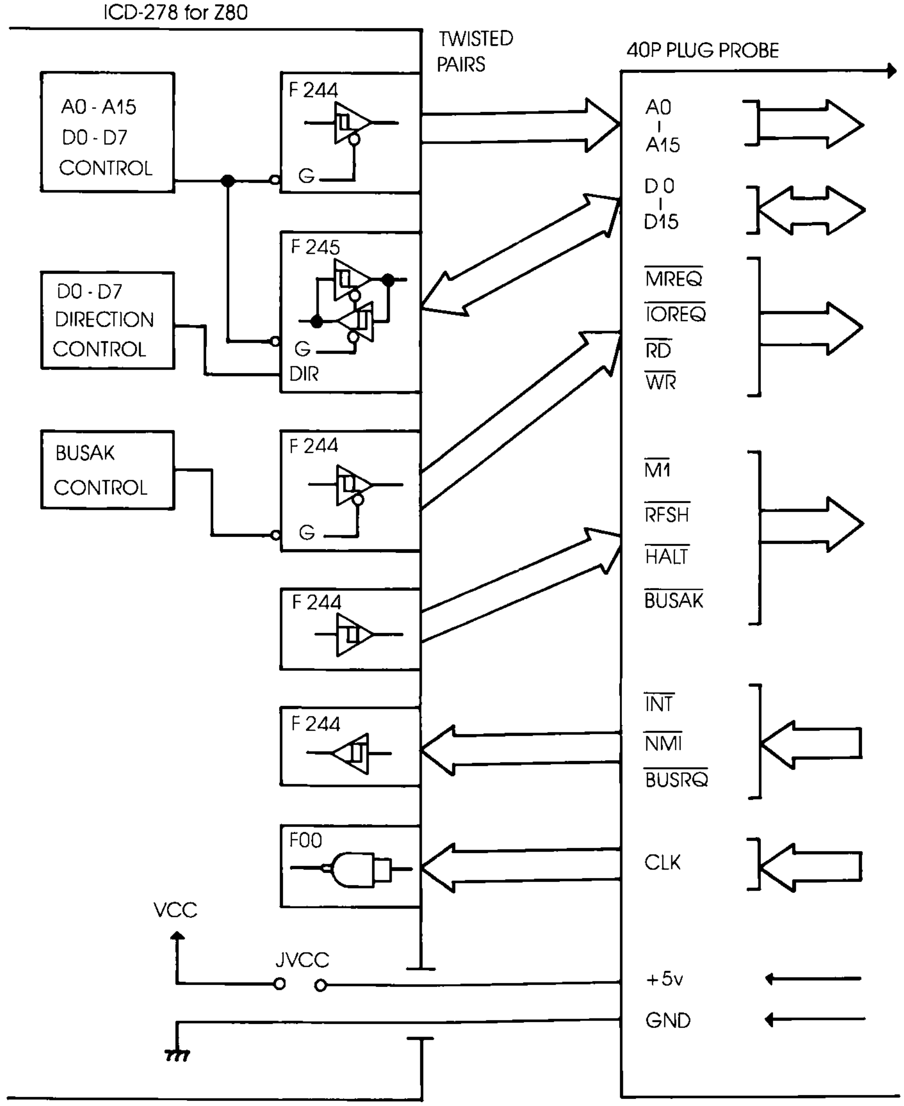
To change the clock speed to 2 MHz, remove the jumper from the H pin and connect the L and CX pins together.

External Clock

Selecting the EXT setting on the INT/EXT switch enables the ICD to use an external clock of up to 6 MHz. 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 50% duty cycle is required for high-speed clocks greater than 2.5 MHz.*

ICD/Target System Interface

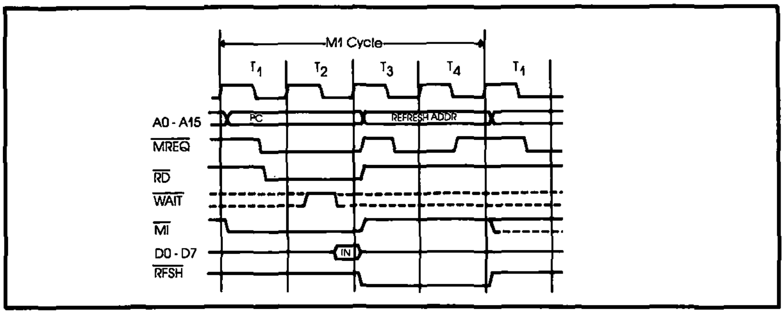
Z80 Pin Functions & Pin Assignments



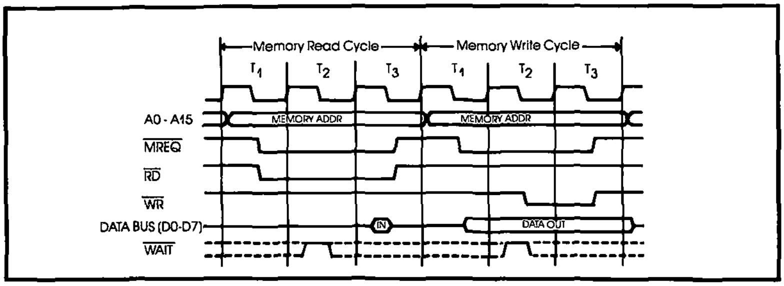
EMULATOR/TARGET SYSTEM INTERFACE

CPU Timing

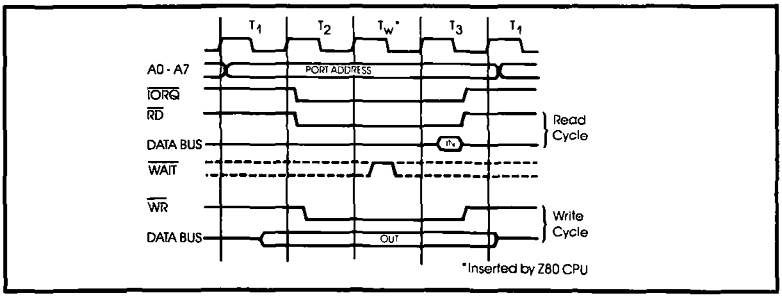
Instruction Opcode Fetch Cycle



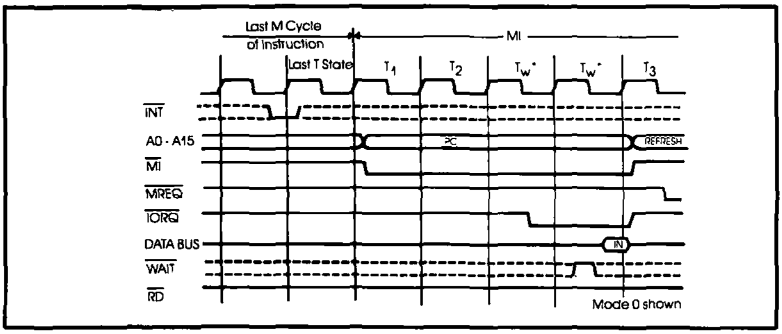
Memory Read or Write Cycle



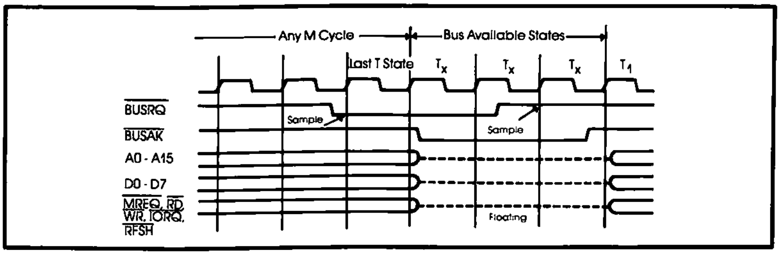
Input or Output Cycles



Interrupt Request/ Acknowledge Cycle



Bus Request/ Acknowledge Cycle



MACHINE CYCLES

MACHINE CYCLE	CONTROL					D0 - D7
	$\overline{\text{MREQ}}$	$\overline{\text{IORQ}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{M1}}$	
Instruction OP code	0	1	0	1	0	IN
Memory Read	0	1	0	1	1	IN
Memory Write	0	1	1	0	1	OUT
Input	1	0	0	1	1	IN
Output	1	0	1	0	1	OUT
Interrupt Acknowledge	1	0	1	1	0	IN
Bus Acknowledge	TS	TS	TS	TS	1	TS
Reset	1	1	1	1	1	TS
ICD Program Memory Read ^{*1}	1 ^{*2}	1	0 ^{*3}	1	x	OUT ^{*4}
ICD Program Memory Write ^{*1}	1	1	1	1	1	OUT
I O (In-circuit Mode 0)	1 ^{*2}	1	1	1	1	TS

Signal level: 0=L, 1=H, TS=tristate, X=undefined (depends on CPU machine cycle)

*1. Cycles do not access target system during memory mapping or in an emulation break. The ICD program memory read cycle can be either a Memory Read (where M1=1) or an Instruction Opcode Fetch (where M1=0).

*2. The MREQ signal (synchronized with Z80 RFSH signal) outputs continuously.

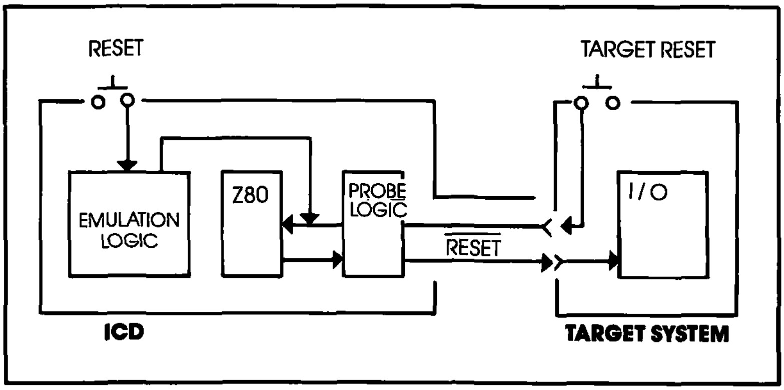
*3. RD signal suppressed when the Emulation Select switch's bit 2=OFF and bit 3=ON.

*4. D0-D7 become tristate when the Emulation Select switch's bit 1=OFF.

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.



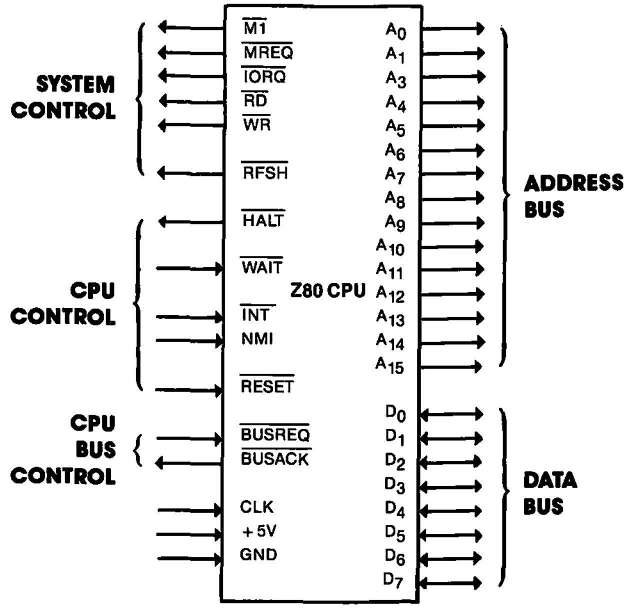
IN-CIRCUIT MODE 0,1,2

	I 0		I 1		I 2	
	MONITOR	EMULATION	MONITOR	EMULATION	MONITOR	EMULATION
ICD RESET SW	○	x	○	x	○	x
$\overline{\text{RESET}}$	x	x	△*	○	△*	○

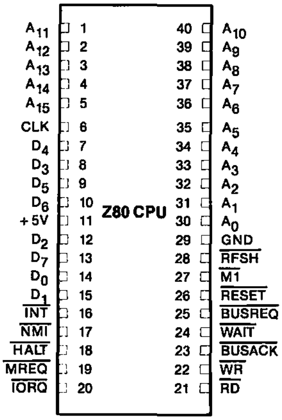
○: Valid △: Conditional valid x: Invalid

* Does not reset hardware in CPU.

Pin Functions



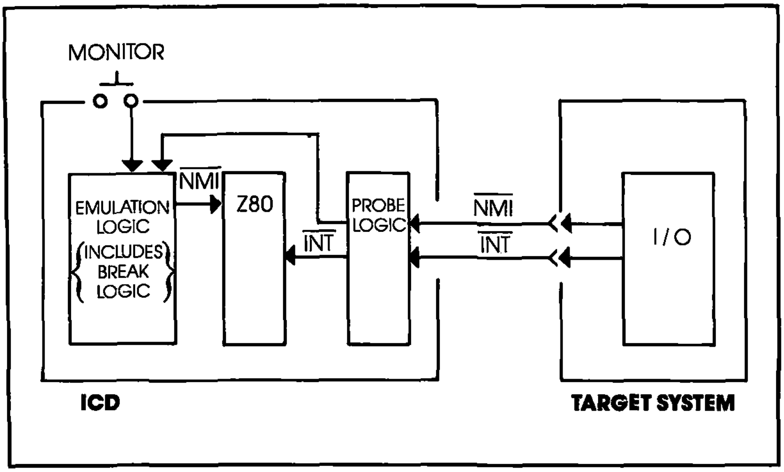
PIN ASSIGNMENTS



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 the break, an interrupt sequence is generated at the transition from the ICD monitor run to the target system run. The INT signal is also masked during an emulation break.



IN-CIRCUIT MODE 0,1,2

	I 0		I 1		I 2	
	MONITOR	EMULATION	MONITOR	EMULATION	MONITOR	EMULATION
ICD MONITOR SW	x	○	x	○	x	○
$\overline{NMI}$	x	x	△*2	○*1	△*2	○
$\overline{INT}$	x	x	x	○*1	x	○

○: Valid △: Conditional valid x: Invalid

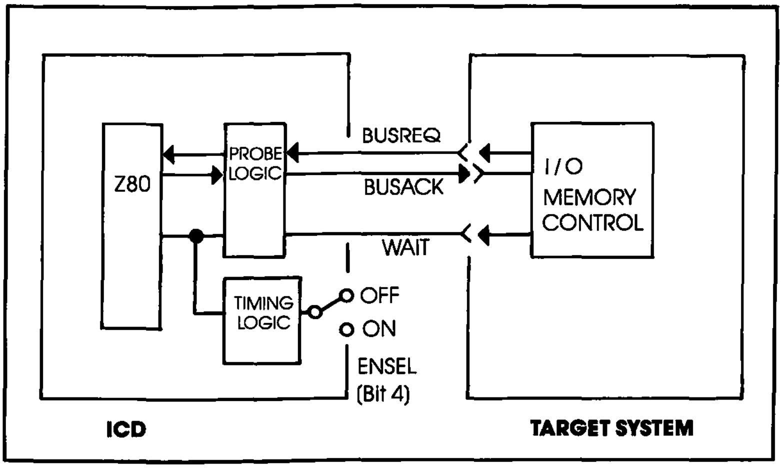
*1 Enable/disable can be set with a PIN command.

*2 NML is sensed at the edge level, and if it occurs on ICD monitor run, an NMI sequence will occur at the transition from the ICD monitor run to target system run.

BUS Control

The ICD accepts the BUSREQ 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.

The WAIT signal is active when the target memory or I/O is accessed. This action allows the target system to operate at higher speeds and permits emulation when the system's access time is short.



IN-CIRCUIT MODE O,1,2

	I 0		I 1		I 2	
	MONITOR	EMULATION	MONITOR	EMULATION	MONITOR	EMULATION
$\overline{\text{BUSAK}}$	x	x	O*	O*	O	O
$\overline{\text{WAIT}}$	x	x	△	O*	△	O

O: Valid △: Conditional valid x: Invalid

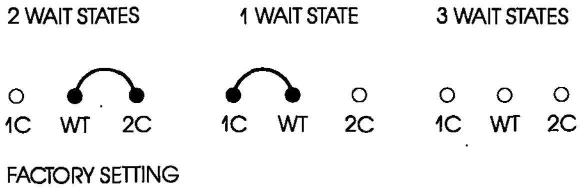
*$\overline{\text{WAIT}}$ signal is valid if the ICD monitor accesses the target system.

▲ Setting Different Wait States

The ICD can insert 1, 2, or 3 wait states into a machine cycle by setting the jumpers on the CPU Module. The normal setting is 2 wait states per machine cycle. To change to 1 or 3 wait states, carry out the procedure below.

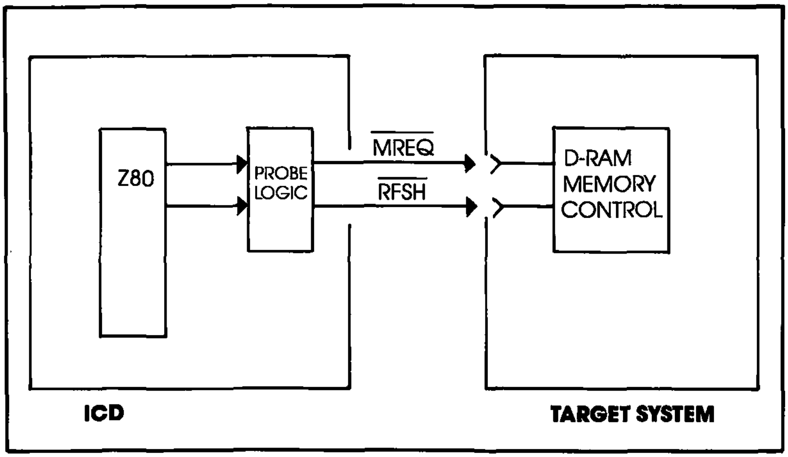
For 3 wait states—remove the jumper from WT and 2C pins.

For 1 wait state—connect the jumper to WT and 1C pins as shown below.



REFRESH Signal

The RFSH signal outputs to the target system during all (0, 1, or 2) in-circuit modes. The memory request (MREQ) signal for refresh is then synchronized with the RFSH signal (independent of the in-circuit mode). This procedure allows the refresh timing of the target system D-RAM to be synchronized with the CPU.



ICD-278 Signal Timing Diagram

			Z-80		Z-80A		Z-80B		ICD 278 for Z80		
Signal	**Symbol**	**Parameter**	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	**Unit**
ϕ	t_C	Check Period	4	30	25		165		165		μsec
	t_{W} (ϕH)	Check Pulse Width, Clock High	180		110		65		65		ns
	t_{W} (ϕL)	Clock Pulse Width, Clock Low	180	2000	110	2000	65	2000	65	2000	ns
	$t_{r,\,f}$	Clock Rise and Fall Time		30		30		20		20	ns
A_{0-15}	t_d (AD)	Address Output Delay		145		110		90		105	ns
	t_F (AD)	Delay to Float		110		90		80		120	ns
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	125		65		35		35		ns
	t_{aci}	Address Stable Prior to $\overline{IORQ}$ (IO Cycle)	320		180		100		110		ns
	t_{ca}	Address Stable from RD, WR, IORQ or MREQ	160		80		35		35		ns
D_{0-7}	t_D (D)	Data Output Delay		230		150		130		145	ns
	t_F (D)	Delay Float During Write Cycle		90		90		80		120	ns
	$t_{S\phi}$ (D)	Data Setup Time to Rising Edge of Clock During M Cycle	50		35		30		45		ns
	$t_{S\phi}$ (D)	Data Setup Time to Falling Edge of Clock During M2 to M5	60		50		40		55		ns
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	190		80		25		25		ns
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	20		10		55		55		ns
	t_{cdi}	Data Stable From $\overline{WR}$	120		60		30		30		ns
	t_H	Any Hold Time for Setup Time	0			0		0		0	ns
$\overline{MREQ}$	$t_{DL\bar{\phi}}$ (MR)	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		100		85		70		85	ns
	$t_{DH\phi}$ (MR)	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		100		85		70		85	ns
	$t_{DH\bar{\phi}}$ (MR)	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		100		85		70		85	ns
	t_W (MRL)	Pulse Width, $\overline{MREQ}$ Low	360		220		135		135		ns
	t_W (MRH)	Pulse Width, $\overline{MREQ}$ High	170		110		65		65		ns
$\overline{IORQ}$	$t_{DL\phi}$ (IR)	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		90		75		65		80	ns
	$t_{DL\phi}$ (IR)	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		110		85		70		85	ns
	$t_{DH\phi}$ (IR)	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		100		85		70		85	ns
	$t_{DH\phi}$ (IR)	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		110		85		70		85	ns

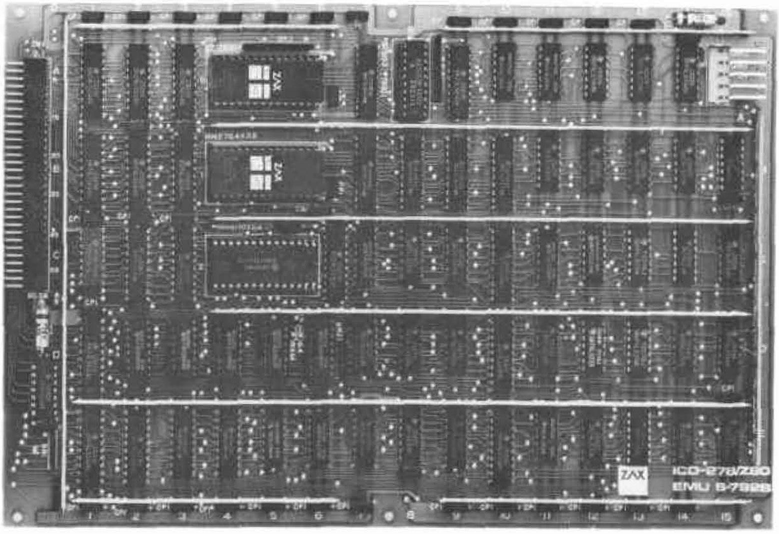
ICD-278 Signal Timing Diagram

			Z-80		Z-80A		Z-80B		ICD 278 for Z80		
Signal	**Symbol**	**Parameter**	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	**Unit**
	$t_{DL\phi}$ (RD)	$\overline{\text{RD}}$ Delay From Rising Edge of Clock, $\overline{\text{RD}}$ Low		100		85		70		85	ns
	$t_{DL\phi}$ (RD)	$\overline{\text{RD}}$ Delay From Falling Edge of Clock, $\overline{\text{RD}}$ Low		130		95		80		95	ns
$\overline{\text{RD}}$	$t_{DH\phi}$ (RD)	$\overline{\text{RD}}$ Delay From Rising Edge of Clock, $\overline{\text{RD}}$ High		100		85		70		85	ns
	$t_{DH\phi}$ (RD)	$\overline{\text{RD}}$ Delay From Falling Edge of Clock, $\overline{\text{RD}}$ High		110		85		70		85	ns
	$t_{DL\phi}$ (WR)	$\overline{\text{WR}}$ Delay From Rising Edge of Clock, $\overline{\text{WR}}$ Low		80		65		60		75	ns
	$t_{DL\overline{\phi}}$ (WR)	$\overline{\text{WR}}$ Delay From Falling Edge of Clock, $\overline{\text{WR}}$ Low		90		80		70		85	ns
$\overline{\text{WR}}$	$t_{DH\phi}$ (WR)	$\overline{\text{WR}}$ Delay From Falling Edge of Clock, $\overline{\text{WR}}$ High		100		80		70		85	ns
	t_W (WRL)	Pulse Width, WR Low	360		220		135		135		ns
	t_{DL} (M1)	$\overline{\text{M1}}$ Delay From Rising Edge of Clock, $\overline{\text{M1}}$ Low		130		100		80		95	ns
$\overline{\text{M1}}$	t_{DH} (M1)	$\overline{\text{M1}}$ Delay From Rising Edge of Clock, $\overline{\text{M1}}$ High		130		100		80		95	ns
	t_{DL} (RF)	$\overline{\text{RFSH}}$ Delay From Rising Edge of Clock, $\overline{\text{RFSH}}$ Low		180		130		110		125	ns
$\overline{\text{RFSH}}$	t_{DH} (RF)	$\overline{\text{RFSH}}$ Delay From Rising Edge of Clock, $\overline{\text{RFSH}}$ High		150		120		100		115	ns
$\overline{\text{WAIT}}$	t_S (WT)	$\overline{\text{WAIT}}$ Setup Time to Falling Edge of Clock	70		70		60		80		ns
$\overline{\text{HALT}}$	t_D (HT)	$\overline{\text{HALT}}$ Delay Time From Falling Edge of Clock		300		300		260		275	ns
$\overline{\text{INT}}$	t_S (IT)	$\overline{\text{INT}}$ Setup Time to Rising Edge of Clock	80		80		70		100		ns
$\overline{\text{NMI}}$	t_W (NML)	Pulse Width, $\overline{\text{NMI}}$ Low	80		80		70		30		ns
$\overline{\text{BUSRQ}}$	t_S (BQ)	$\overline{\text{BUSRQ}}$ Setup Time to Rising Edge of Clock	80		50		50		65		ns
	t_{DL} (BA)	$\overline{\text{BUSAK}}$ Delay From Rising Edge of Clock, $\overline{\text{BUSAK}}$ Low		120		100		90		105	ns
$\overline{\text{BUSAK}}$	t_{DH} (BA)	$\overline{\text{BUSAK}}$ Delay From Falling Edge of Clock, $\overline{\text{BUSAK}}$ High		110		100		90		105	ns
$\overline{\text{RESET}}$	t_S (RS)	$\overline{\text{RESET}}$ Setup Time to Rising Edge of Clock	90		60		60		75		ns
	t_F (C)	Delay to Float ($\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$)		100		80		70		120	ns
	$t_{mt}{}^{\circ}$	$\overline{\text{MI}}$ Stable Prior to $\overline{\text{IORQ}}$ (Interrupt Ack.)	920		565		365		365		ns

Emulation Memory (Unit) Module

Description

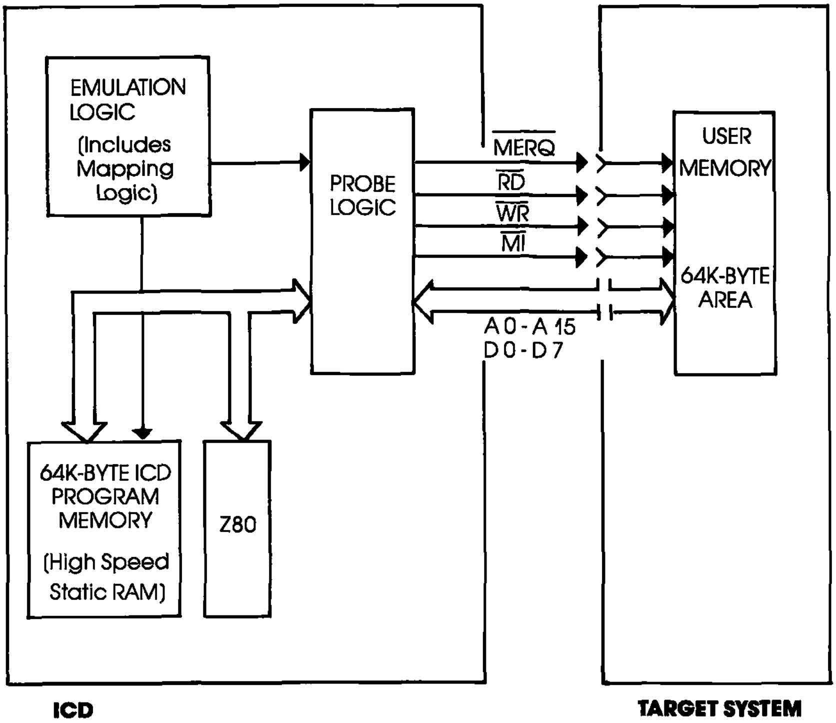
The Emulation Memory (EMU) module (S-792) manages the ICD emulation memory and the mapping of the target system's memory. The EMU module contains no user-serviceable controls or components; all functions are activated using the debugger commands.



ICD Emulation Memory

The ICD-278 for Z80 features 64K bytes of RAM which is called ICD emulation memory. This memory can be used for downloading object files, and altering or manipulating the target system's memory.

The 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, ICD emulation memory is different from a normal memory area in that it is contained within the Z80 processor. And, because of the special characteristics of the ICD 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.

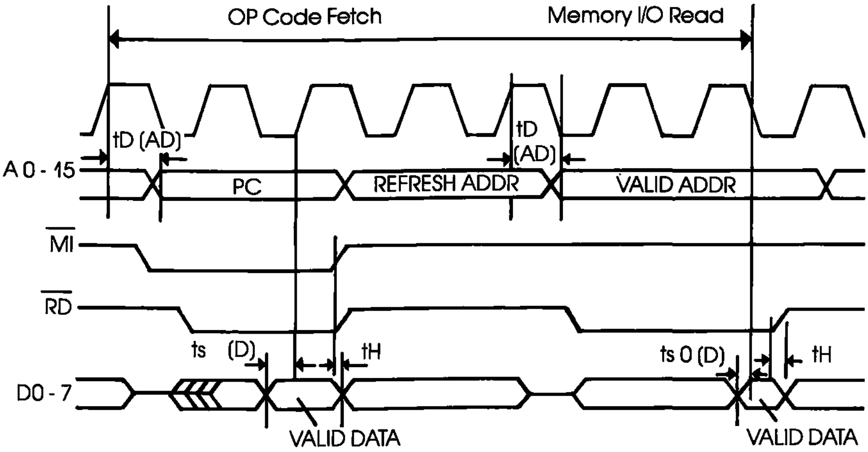


ICD Program Memory Access Time	min. 120 ns

Target System Memory

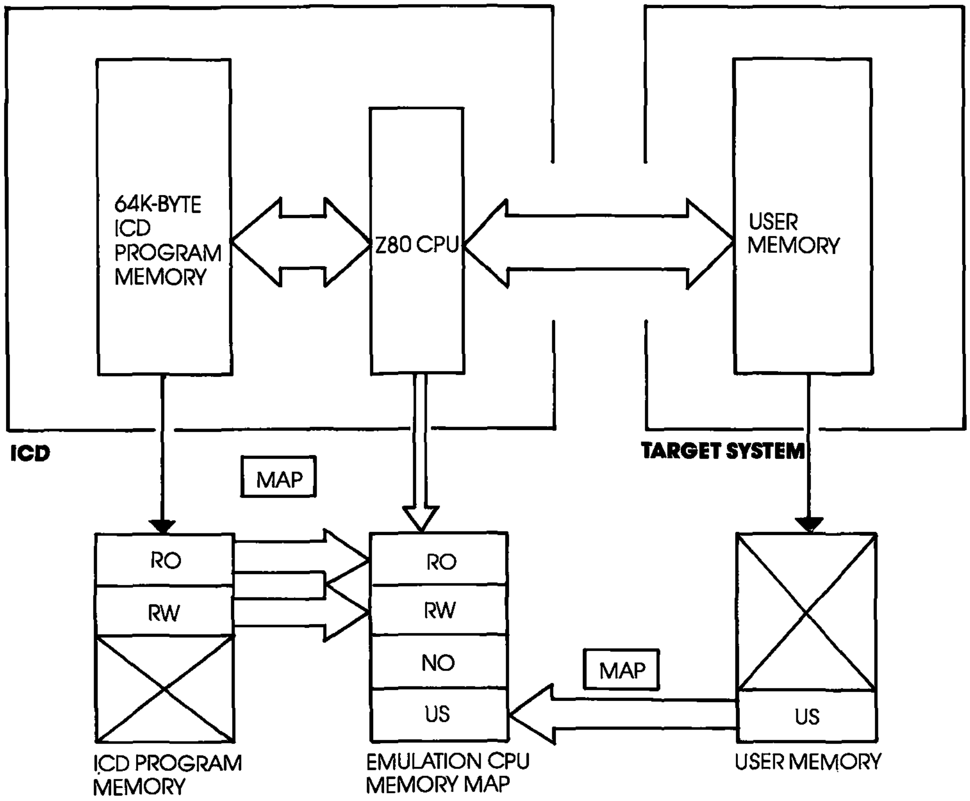
The memory contained in the target system is called target memory or user memory. The ICD can address up to 64K bytes of target memory.

The access time required to write to the target 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 that available with the processor. Therefore, certain access time conditions must be satisfied for accurate reading. These conditions are shown below:



		OP code Fetch	Memory I/O Read
t D (AD)	Address Output Delay	Max.	Max. 90 ns
t S ϕ (D)	Data Setup Time to Rising Edge of clock during M1 Cycle	min. 30 ns	min. 40 ns
t H	Any Hold Time for Setup time	in. 0 ns	min. 0 ns

Mapping You can use all or part of the ICD's RAM in place of target 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 examples of how this works, see the MAP command in Section 2.)

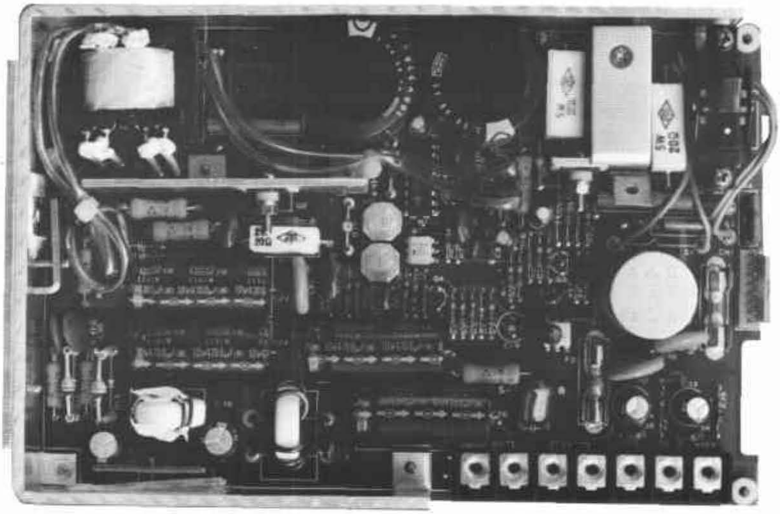


Power Supply Specifications

Line voltage: 100 to 120 volts AC
200 to 240 volts AC
Frequency: 50 or 60 Hz
Power: 40 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 to change certain settings on the control modules. Here you will find the procedure for disassembling the ICD and removing (and installing) the five 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 SIO and CPU control modules. The adjustments and modifications which are authorized by ZAX are clearly identified (▲) in each of these chapters. Any other alterations or adjustments on the SIO and CPU control modules void the Warranty Policy.**

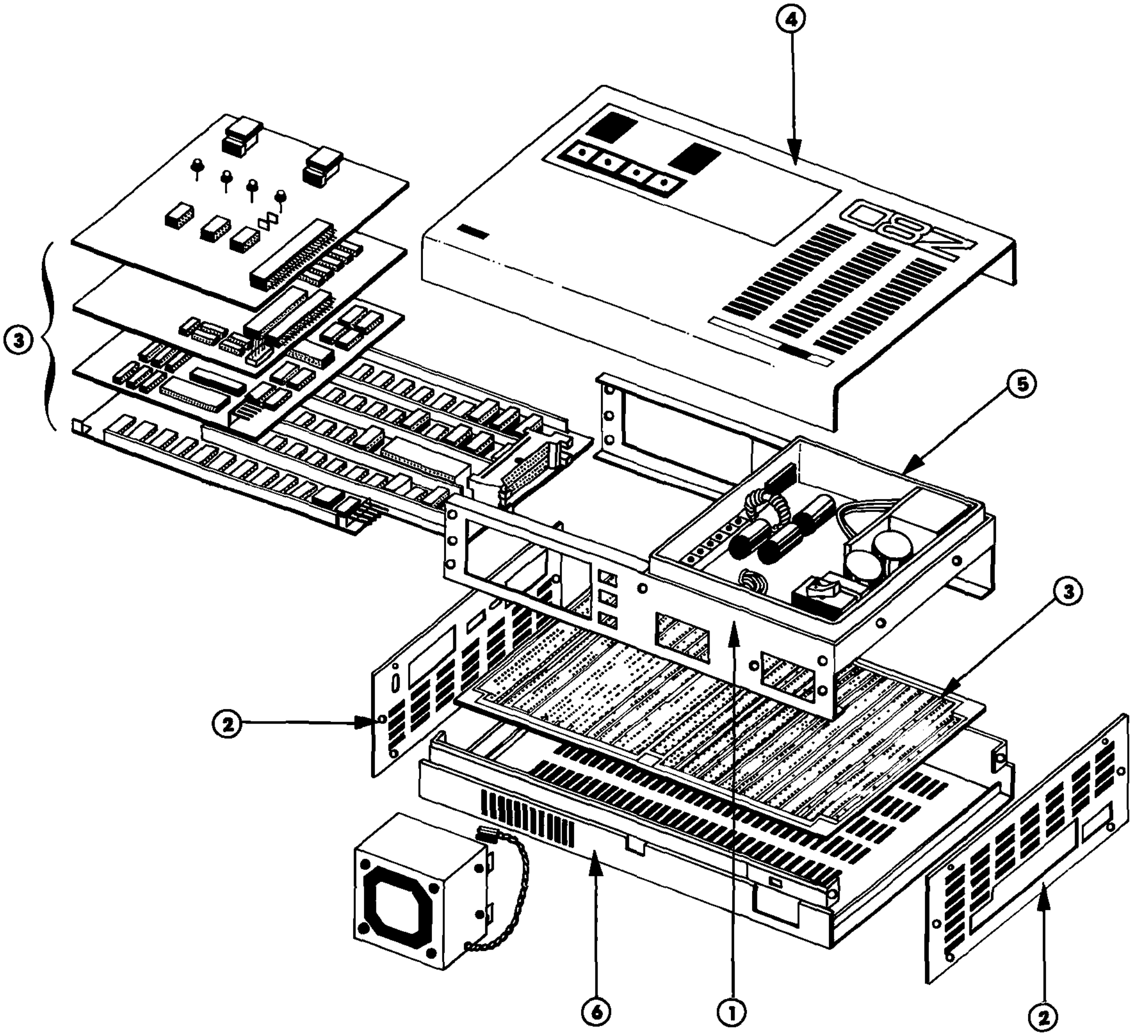
2. **Do not adjust, modify, and/or in any way alter the controls or components on any of the three remaining modules (Indicator/Control, Real-time Trace, or Emulation Memory Unit) or the power supply.**

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.**

The Basic Parts Of Your ICD

The construction of all **ZAX** ICD-series emulators is very similar. The basic ICD unit includes the mainframe, the five control modules, the power supply, the Mother Bus cable, and the outside casing. The **mainframe** is a metal chassis that houses the control modules and the power supply. The five **control modules** are circuit boards (sometimes called "cards") which do the actual work of emulating the target system. 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 Cover**
③ **Control Modules**
④ **Top Cover**
⑤ **Power Supply**
⑥ **Bottom Cover**


④
③
⑤
③
②
⑥
①
②
Z80

Procedure For Disassembling The ICD

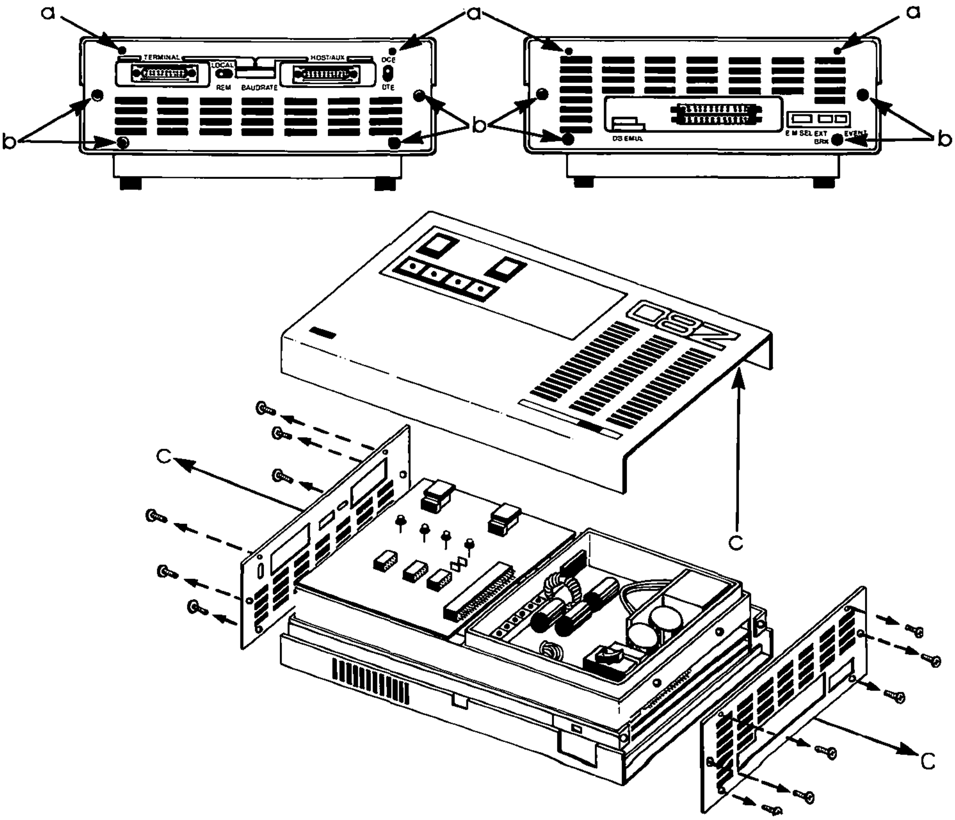
WARNING HAZARDOUS VOLTAGE IS PRESENT WITHIN THE ICD-278. DISCONNECT THE AC POWER PLUG BEFORE BEGINNING ANY INTERNAL WORK ON THE ICD-278.

1. **Remove the two side covers and the top cover.**

   a) Remove the four raised screws that connect the top cover to the mainframe.

   b) Remove the eight countersunk screws on the side covers.

   c) Detach the side covers and the top cover.



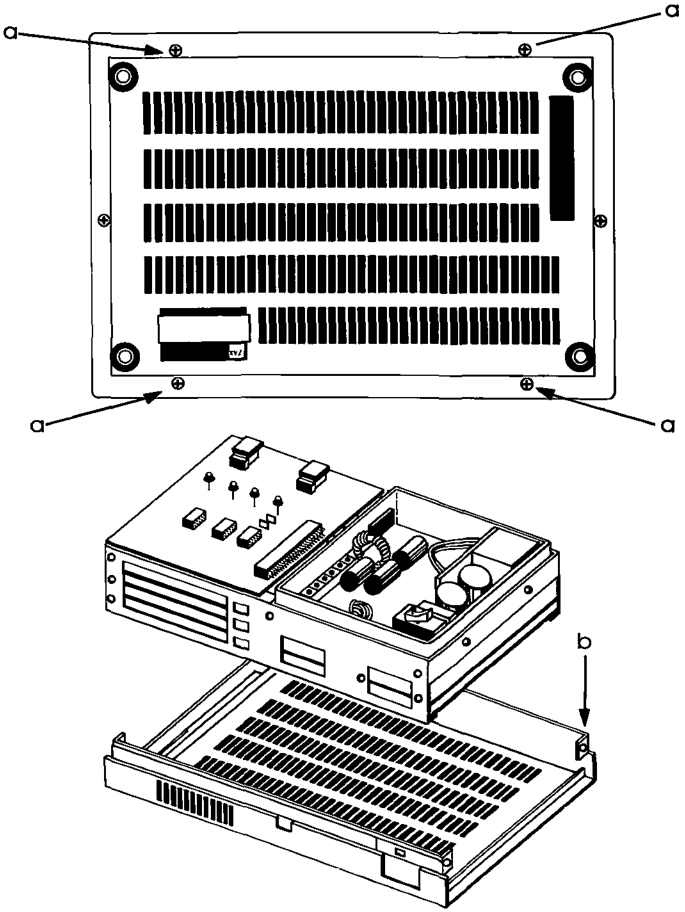
2. **Gently turn over the ICD and remove the bottom cover.**

NOTE: Place the ICD on a soft foam-type pad to protect the case and components.

a) Remove the four screws that attach the bottom cover to the mainframe (it is not necessary to remove the two countersunk screws).

b) Remove the bottom cover completely.

The control modules are now accessible.

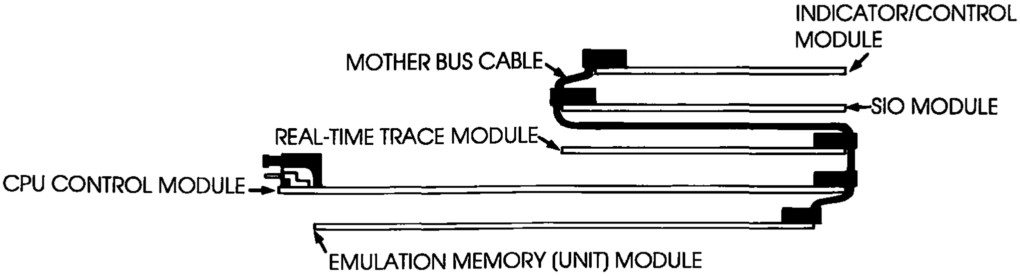


How The Modules Are Connected

Each module is linked by the 60-pin Mother Bus cable. Power is supplied to each module by a 5-pin, plug-type power connector cable (except for the S-730 module which receives its power from the Mother Bus cable). The power and Mother Bus cables must be detached before removing any of the control modules.

IMPORTANT: Note the position of the power connectors before removing them. Both the socket and plug have a black label on one side which marks the polarity of the connectors.

MODULE CONNECTION



Procedure For Removing The Modules

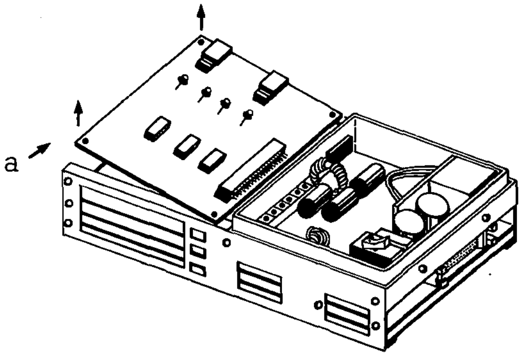
1. **Free the power cable by disconnecting the five-pin socket (CN4) from the module.**

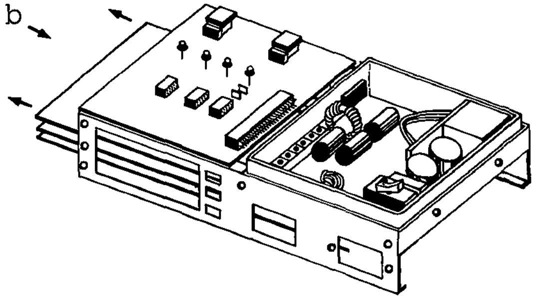
2. **Detach the Mother Bus cable from the modules (location CN1).**

3. **Remove the screws which hold the modules to the mainframe.**

   a) The top and bottom (S-730 and S-792) modules are mounted directly to the mainframe with four and six screws, respectively. Remove these screws to detach the modules.

   b) The three remaining modules (S-791, S-795, and S-793) are connected to brackets which are attached to the ICD's mainframe at one end and slide into holders at the other end. To detach these modules, remove the screws and then carefully slide the modules away 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-278.

NOTE: When replacing the side panels, loosely position all the screws in place to allow the panels to align properly before tightening the screws.

Contents

SECTION 4 — COMMUNICATION PROTOCOL

Introduction

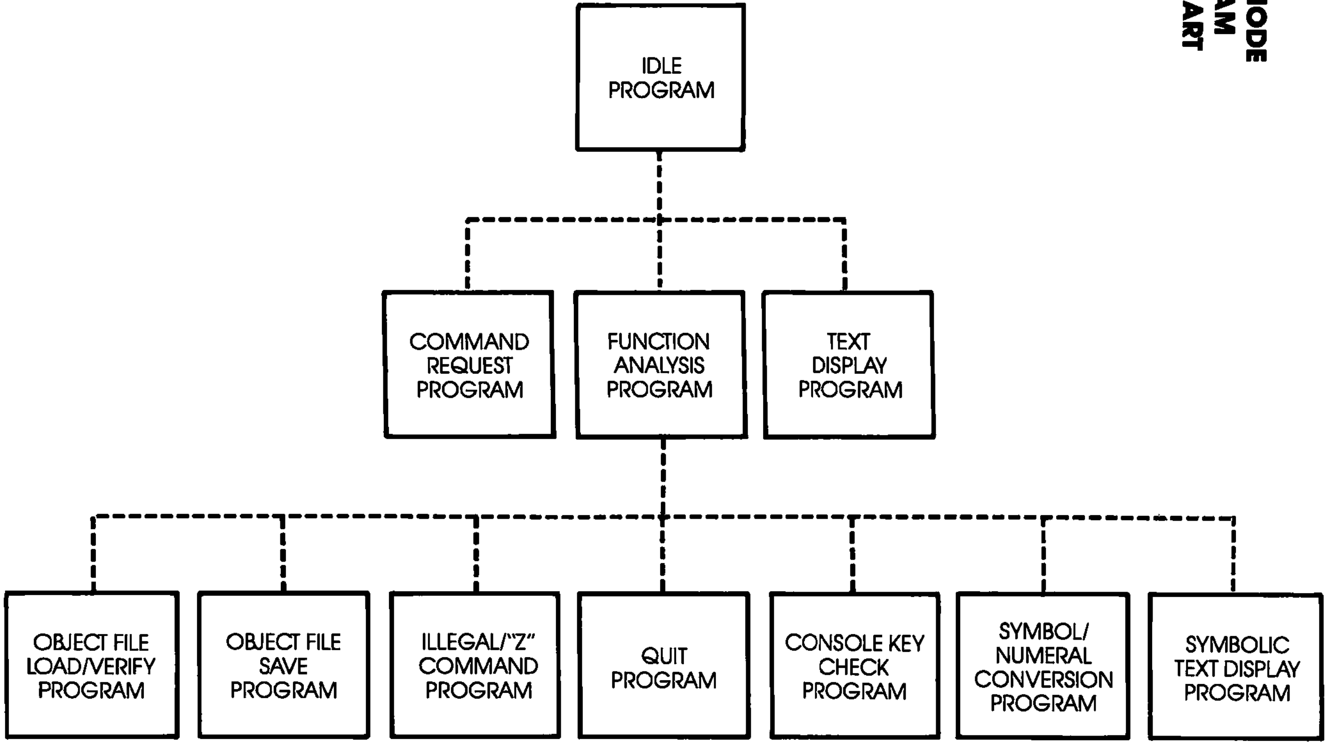
Your ICD can operate in one of two different system configurations with a host computer. In one configuration, a host computer is used to directly control the ICD via ZICE software; this is called the REMOTE mode. In the other configuration, the ICD is under the direct control of a console terminal and 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 is called the LOCAL mode. (The HOST command activates the LOCAL "host computer assisted" mode.)

In either configuration, when the ICD is used with a host computer supported by ZICE software, certain communication rules are observed to ensure an orderly information exchange between the ICD and the host computer system. This is called communication protocol.

In this section you'll be shown (using diagrams) the proper communications protocol (for both the REMOTE and LOCAL modes. The diagrams show the contents of the communication messages from both the ICD and host computer, and the sequence in which they are executed. You can use the communication programs to write your own support software for use with your particular host computer system.

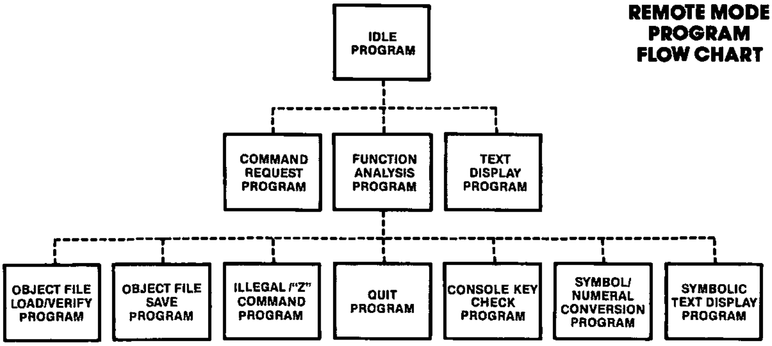
NOTE: Although this manual is specifically designed for the ICD-278's Z80 CPU, this section can be used with ALL ***ZAX*** *ICD-series emulators that feature "backslash" (\) protocol format. This format is structured as:* \code(text)<CR>. *A Number and Symbol Conversion Code chart which shows the correct numbers and symbols to use with your particular emulator is shown at the end of this section.*

REMOTE MODE PROGRAM FLOW CHART



REMOTE MODE: HOST COMPUTER CONTROL OF THE ICD

PROGRAM: IDLE



REMOTE MODE PROGRAM FLOW CHART

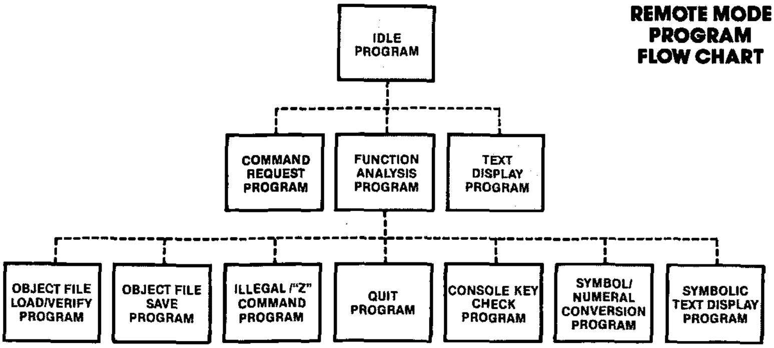
PROGRAM DESCRIPTION: This program acts as the main intermediary program (transferring instructions and text only) between the ICD and the subprograms (Command Request, Text Display, and Function Analysis).

ACTION:

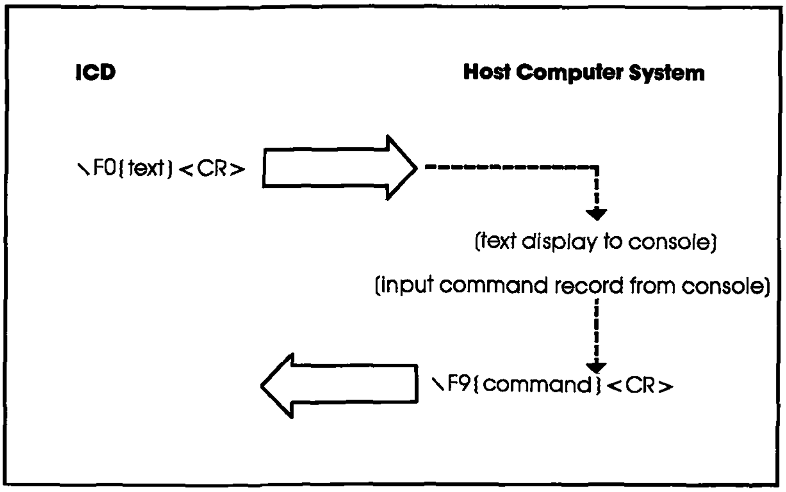
1. The host computer waits for an input from the ICD. (The host system must have an input buffer to hold the input code from the ICD.)
2. The host computer receives one line of data and places it in the input buffer.
3. The host computer then executes one of the following programs depending on the code it receives:

Code Received	Program Executed
\F0<CR>	COMMAND REQUEST
\80{text}<CR>	TEXT DISPLAY
any other	FUNCTION ANALYSIS

PROGRAM: COMMAND REQUEST



REMOTE MODE PROGRAM FLOW CHART



ORIGIN	**ACTION:**	Sends a command from the host computer in response to a command request from the ICD.
ICD	\F0{text}<CR>	COMMAND REQUEST RECORD. This record is a command request sent from the ICD to the host computer. This record also contains the ASCII text to be displayed on the host computer's terminal screen, but does not include <ACK>, <NAK>, <CR>, or <SOH>. The host computer then displays the <text> on the console screen.
HOST	\F9{command} <CR>	COMMAND RECORD. ASCII text is sent as a command from the host computer to the ICD. This record cannot contain any control code and must end with <CR>. When the record is entered through the host computer's console, the system accepts one line of data echoing it back to the console screen. *NOTE: The cursor stays on the same line after the echo. To move the cursor to the next line, the ICD sends a code in the text display sequence.*

PROGRAM DESCRIPTION: **COMMAND REQUEST**

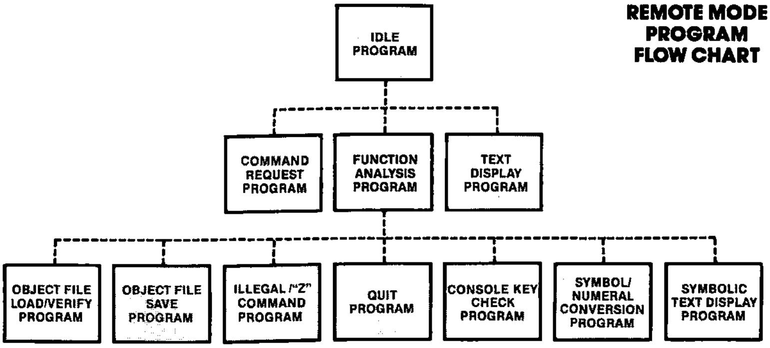
ACTION:

1. The ICD requests a command by sending \F0{text}<CR> to the host computer.

2. Upon receiving \F0{text}<CR> from the ICD, the host computer waits for an input after displaying the text record on the console screen.

3. If a command record is entered from the host computer, the system sends \F9{text}<CR> to the ICD and then returns to the IDLE program.

PROGRAM: **FUNCTION ANALYSIS**



REMOTE MODE PROGRAM FLOW CHART

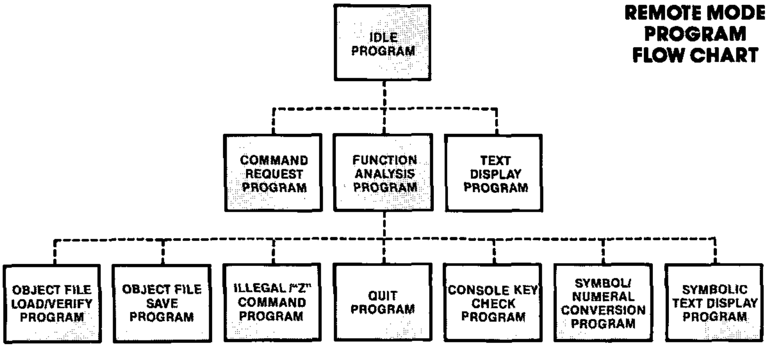
ACTION:

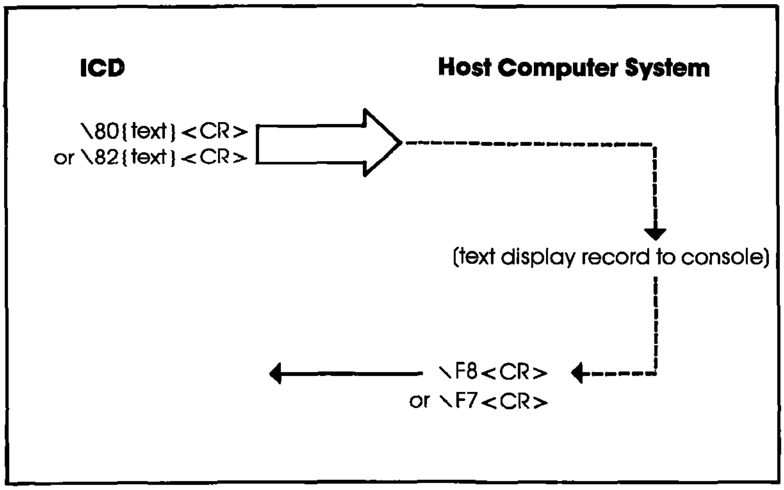
1. The host computer places one line of data (received from the ICD) into the input buffer and analyzes the data.

2. 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 \ 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
\ 88 or \ 8A<CR>	Checks the console input in the host computer.
\ 3X{parameter}{text}<CR>	SYMBOLIC TEXT DISPLAY

PROGRAM: **TEXT DISPLAY**


REMOTE MODE PROGRAM FLOW CHART



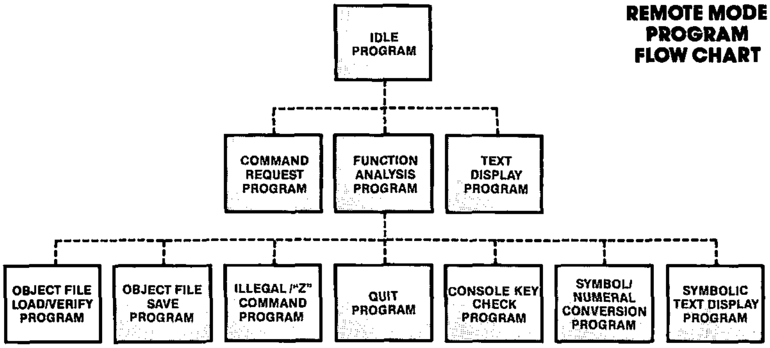
ORIGIN	**ACTION:**	Text sent from the ICD is displayed on the console screen of the host computer.
ICD	\80<CR>	TEXT RECORD. This record is ASCII text sent to the host computer's console screen from the ICD. *(NOTE: <ACK>, <NAK>, <ENQ>, or <SOH> cannot be contained in the text record.)* The host computer displays one line of the text record up to <CR>, and then moves the cursor to the start of the next line.
HOST	\F8<CR>	DISPLAY COMPLETE ACKNOWLEDGE. This code is sent to the ICD when the display has been completed.
HOST	\F7<CR>	DISPLAY INTERRUPT CODE. This code is sent to the ICD to interrupt it from sending a text record during the display of a "scrolling" display command (e.g., DUMP, TRACE, etc.).

PROGRAM DESCRIPTION: **TEXT DISPLAY**

ACTION:

1. The TEXT DISPLAY program is requested when the ICD sends \80{text}<CR> to the host computer.
2. The host computer displays the text record on the console screen.
3. The host computer then checks the console input status and executes 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.

PROGRAM: **OBJECT FILE LOAD/VERIFY**



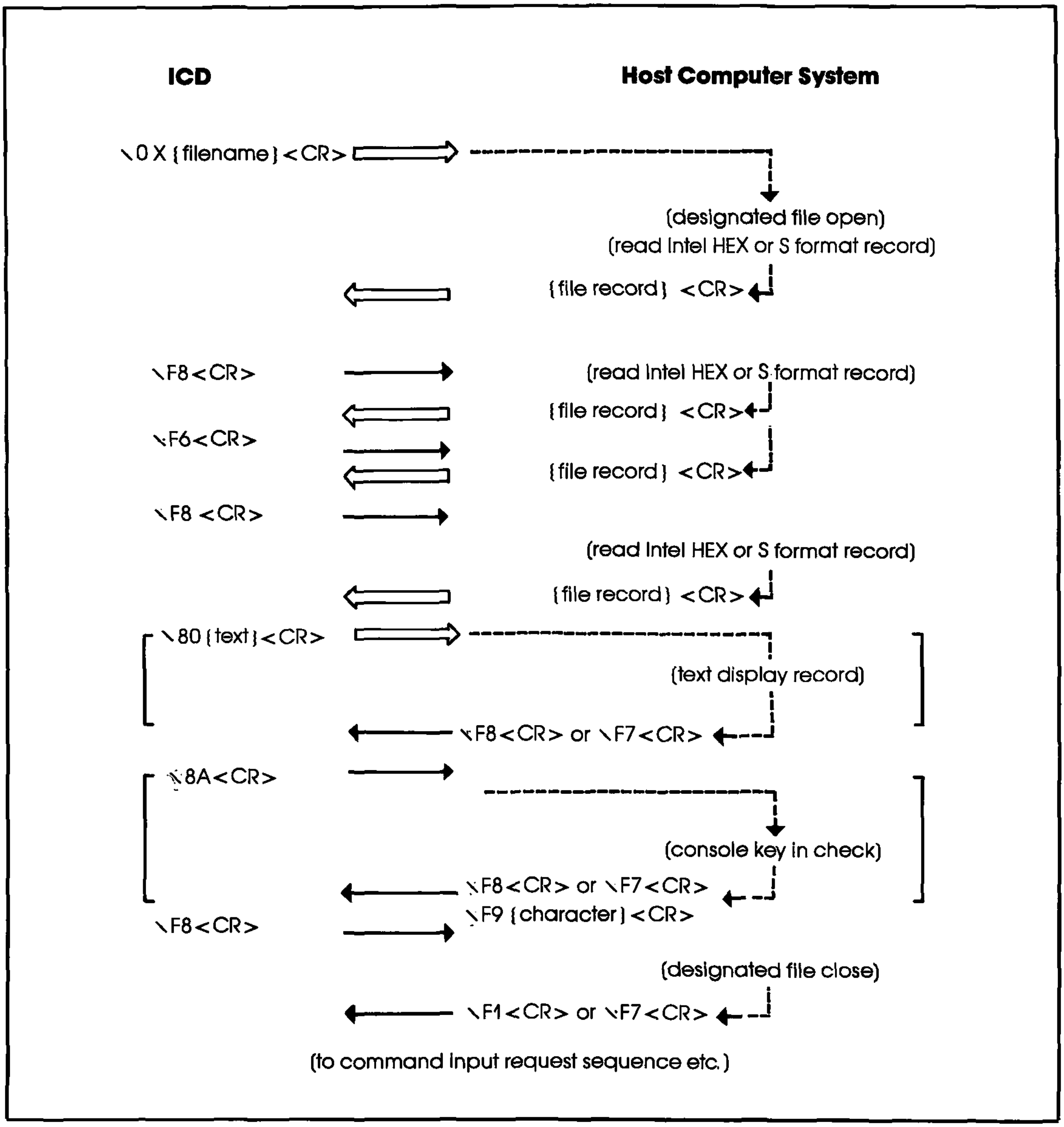
REMOTE MODE
PROGRAM
FLOW CHART

ORIGIN **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 check sequence.

ICD	\00{filename}<CR> or \02{filename}<CR>	INTEL HEX LOAD REQUEST RECORD or S FORMAT LOAD REQUEST RECORD. This record is sent to the host computer when the ICD loads an object file.
ICD	\01{filename}<CR> or \03{filename}<CR>	INTEL HEX VERIFY REQUEST RECORD or S FORMAT VERIFY REQUEST RECORD. This record is sent to the host computer when the ICD verifies an object file with the memory.
HOST	{record}<CR>	OBJECT FILE RECORD. This Intel Hex or S format record is sent to the ICD from the host computer. This record may not contain any control code, and must end with <CR>.
ICD	\F8<CR>	OBJECT RECORD ACKNOWLEDGE CODE. This code is sent by the ICD to acknowledge successful receipt of an Intel Hex or S format record.

ICD	\F6<CR>	OBJECT RECORD RE-TRANSMISSION REQUEST CODE. This code is used when the ICD requests the host computer to retransmit the object record, usually due to a check sum error.
ICD	\F7<CR>	OBJECT FILE TRANSMISSION INTERRUPT CODE. This code ends the LOAD/VERIFY sequence due to irrecoverable error.
ICD	\80{text}<CR> (at text display)	TEXT RECORD. This record contains a verify error message.
HOST	\F8<CR> (at text display)	DISPLAY COMPLETE CODE.
HOST	\F7<CR> (at text display)	LOAD/VERIFY SEQUENCE ABORT INDICATION CODE. The host computer may abort the LOAD/VERIFY sequence by sending this code to the ICD.
ICD	\8A<CR> (at console key input check)	CONSOLE KEY INPUT CHECK REQUEST CODE. This request is generally used to check the status of an abort, or interrupt of the verify error messages.
HOST	\F8<CR> (at console key input check)	NO-CONSOLE-INPUT CODE.
HOST	\F9{any ASCII code}<CR> (at console key input check)	CONSOLE INPUT CODE
HOST	\F7<CR> (at console key input check)	LOAD/VERIFY SEQUENCE ABORT INDICATION CODE. The host computer can abort the object LOAD/VERIFY sequence by sending this code to the ICD.
HOST	\F1<CR>	LOAD/VERIFY END CODE. The host computer sends this code to the ICD (after closing the file) if the file records are exhausted. An object LOAD/VERIFY sequence ends this code.
HOST	\F7<CR>	LOAD/VERIFY SEQUENCE ABORT INDICATION CODE. The host computer uses this code to inform the ICD it is aborting the LOAD/VERIFY sequence.


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.)

PROGRAM DESCRIPTION: **OBJECT FILE LOAD/VERIFY**

ACTION:

1. The ICD sends \0X{filename}<CR> to the host computer to load or verify a user program.

2. The host computer then opens the requested program file, and acts on the following:

   a) If an error occurs when opening or 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.

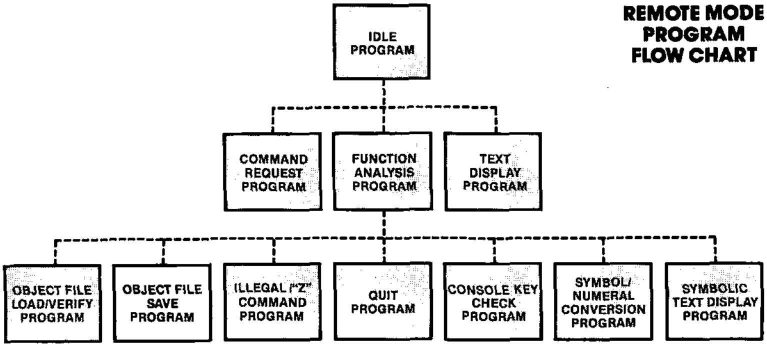
   If the code is \F6<CR>, the host computer waits for \F8<CR> after re-transmitting the Intel Hex or S format record to the ICD.

   When the text record is received from the ICD, the host computer displays the text record on the console screen 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 "\F9<any ASCII code>" 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.

PROGRAM: OBJECT FILE SAVE



REMOTE MODE PROGRAM FLOW CHART

ORIGIN	ACTION:	The host computer receives an object program and creates a file upon receiving a save request from the ICD.
ICD	\10{filename}<CR> or \12{filename}<CR>	INTEL HEX SAVE REQUEST RECORD or S FORMAT SAVE REQUEST RECORD. This record is sent by the ICD to request the host computer to save a file.
HOST	\F8<CR> (at file write)	OBJECT RECORD REQUEST CODE. This code is sent to the ICD from the host computer to request an Intel Hex or S format record.
ICD	{record}<CR>	OBJECT FILE RECORD. This Intel Hex or S format record is sent to the host computer.
HOST	\F6<CR>	OBJECT FILE RE-TRANSMISSION REQUEST CODE. This code is used when the host computer requests the ICD to re-transmit the object file. *NOTE: Most re-transmission requests are caused by a sum check error of an Intel Hex or S format record.*
ICD	\F1<CR>	FILE END CODE. This code is sent to the host computer when the file record transmission is exhausted.

ICD	\F7<CR>	SAVE SEQUENCE ABORT REQUEST CODE. This code directs the host computer to abort the object save sequence.
HOST	\F8<CR> (at file close)	FILE CLOSE END CODE. The host computer sends this code to the ICD, in response to \F1<CR>, if the file has been closed successfully, then returns to the IDLE program.
HOST	\F7<CR>	SAVE SEQUENCE ABORT INDICATION CODE. The host computer uses this code to inform the ICD that it is aborting the object save sequence.

PROGRAM DESCRIPTION: **OBJECT FILE SAVE**

ACTION:

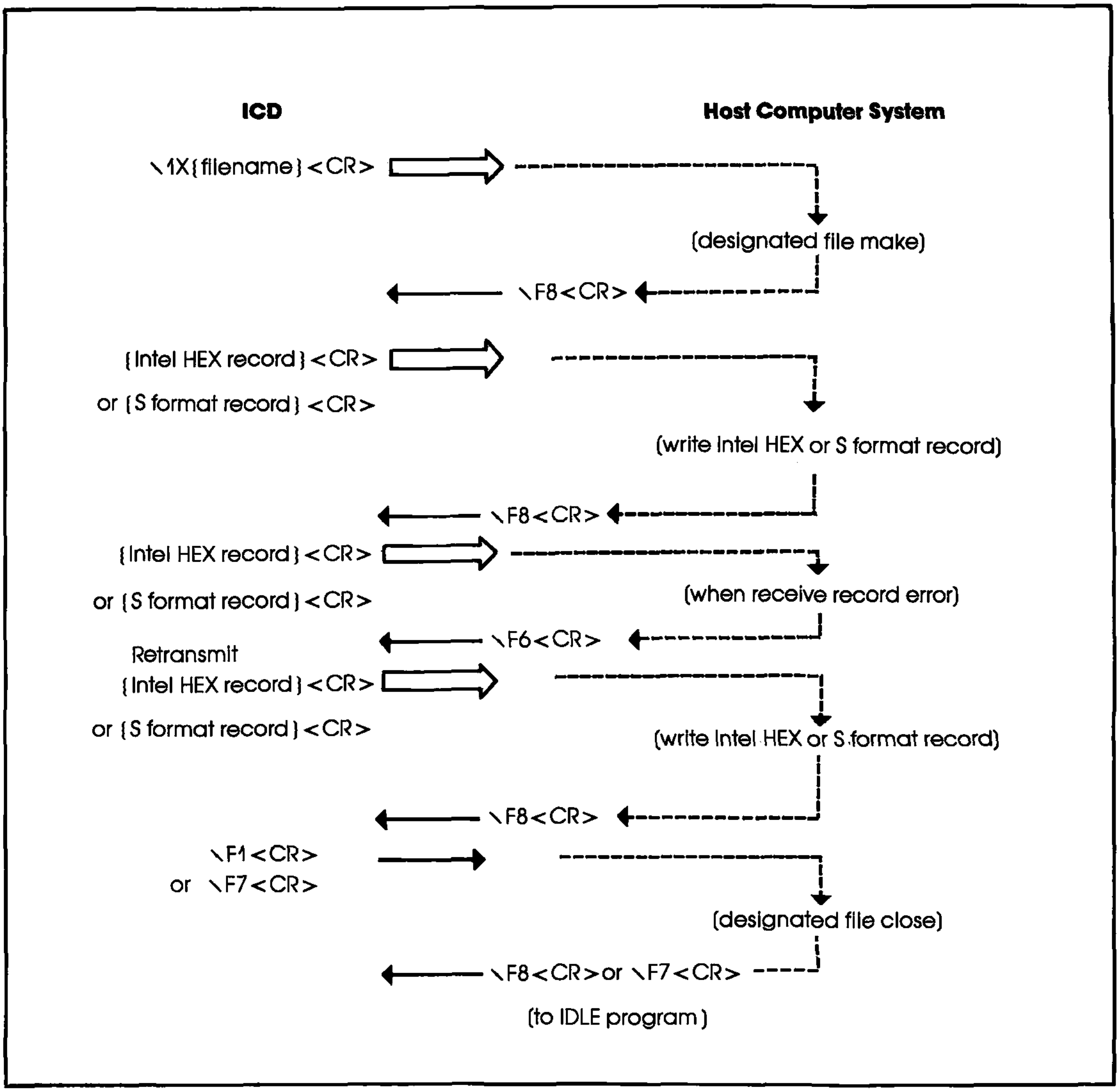
1. The ICD sends \1X{filename}<CR> to the host computer when saving a user program.

2. The host computer opens the selected user file when it receives the \1X{filename}<CR> code.

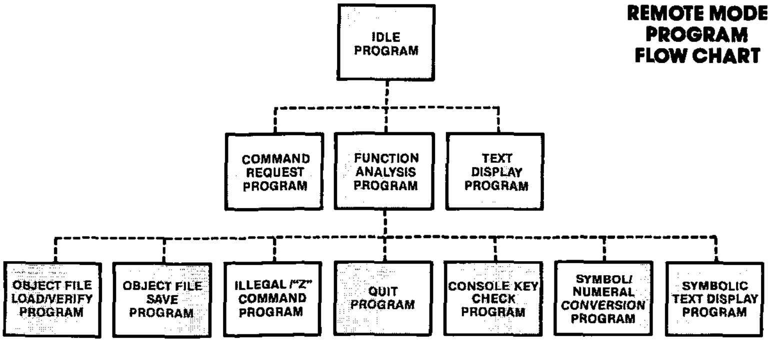
   If the file does not open, the host computer sends \F7<CR> to the ICD and returns to the IDLE program. If the file opens, the host computer sends \F8<CR> to the ICD.

3. The host computer waits for the Intel Hex or S format record from the ICD. 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 file, sends \F7<CR> to the ICD, and returns to the IDLE program.

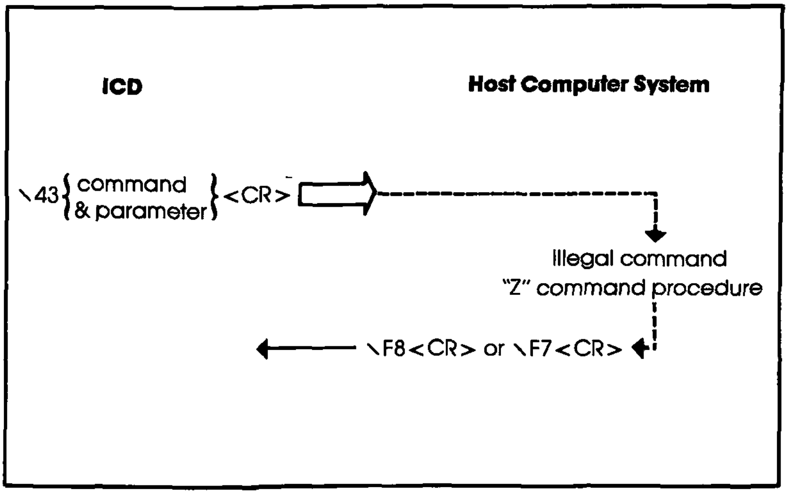
   If an error occurs in a check sum, the host computer waits for a retransmission of the Intel Hex or S format record from the ICD after sending \F6<CR>. The host computer then sends \F8<CR> to the ICD if no error occurs during the file write.


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)

PROGRAM: ILLEGAL/"Z" COMMAND



REMOTE MODE PROGRAM FLOW CHART



ORIGIN	**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 is interpreted and processed by the host computer. The host computer can use the ILLEGAL and "Z" commands to process a HELP command or the macro commands.
ICD	\43{parameter}<CR>	ILLEGAL/"Z" COMMAND RECORD. This record is sent to the host computer to process the ILLEGAL/"Z" command.
HOST	\F8<CR>	ILLEGAL/"Z" COMMAND NORMAL END CODE. This code is sent to the ICD when the ILLEGAL/"Z" command has been processed successfully.
HOST	\F7<CR>	ILLEGAL/"Z" COMMAND ABNORMAL END CODE. This code is sent to the ICD when the ILLEGAL/"Z" command has not been processed successfully.

PROGRAM DESCRIPTION: **ILLEGAL/"Z" COMMAND**

ACTION:

1. The ICD sends \43{parameter}<CR> (and the specified "Z" command) to the host computer.

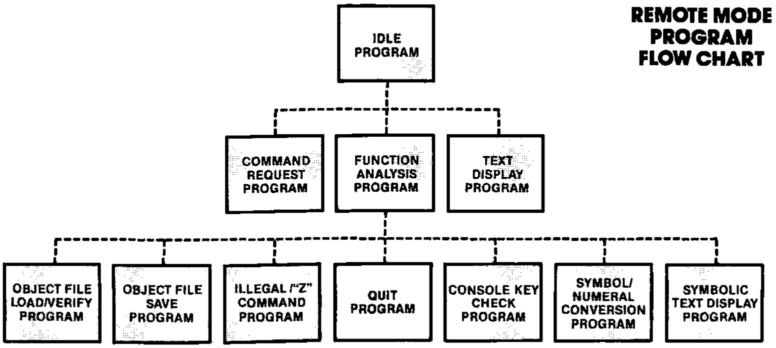
2. The host computer performs the specified "Z" command and then acts on the following:

   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.

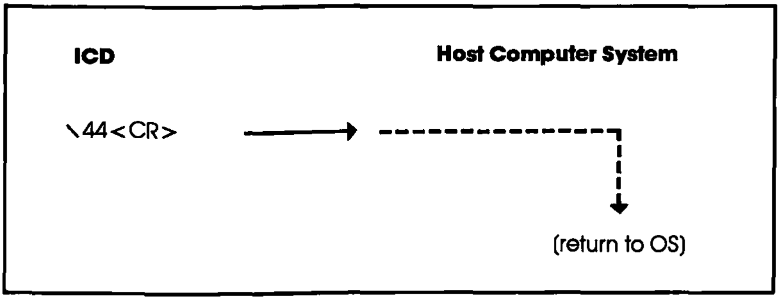
   If no error is detected, the host computer sends \F8<CR> to the ICD and then returns to the IDLE program.

NOTE: The ICD does not react differently to the \F7<CR> than it does to the \F8<CR> code. It normally assumes that the host program has issued its own error messages if an error has occurred.

PROGRAM: **QUIT**

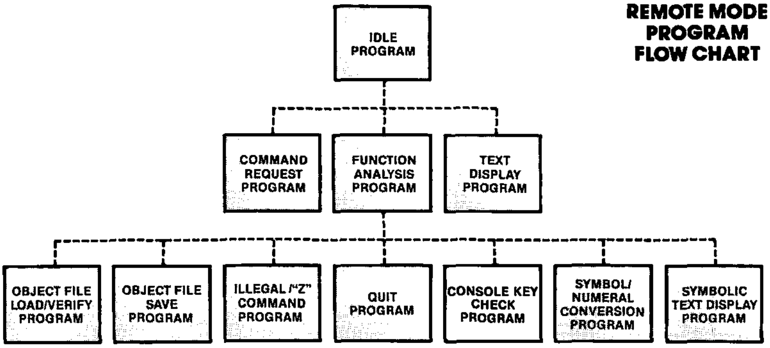


REMOTE MODE PROGRAM FLOW CHART

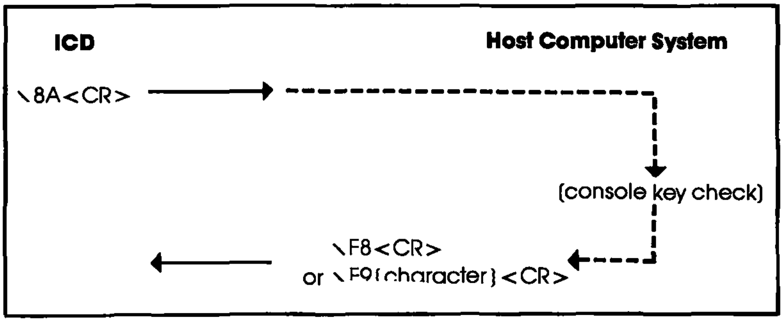


ORIGIN	ACTION:	
		The host computer returns to the operating system (OS) indicated by the ICD's code.
ICD	\44<CR>	QUIT RECORD.

PROGRAM: **CONSOLE KEY CHECK**

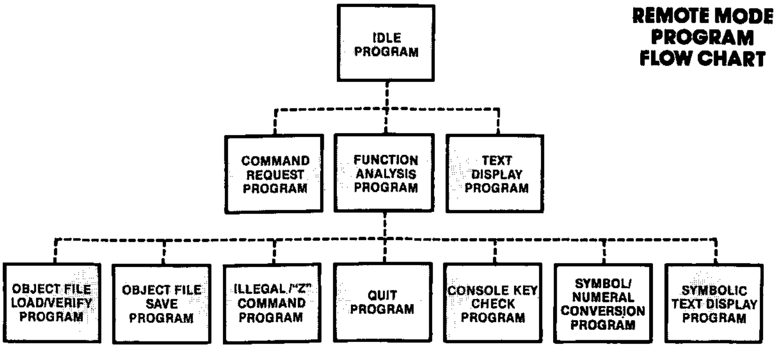


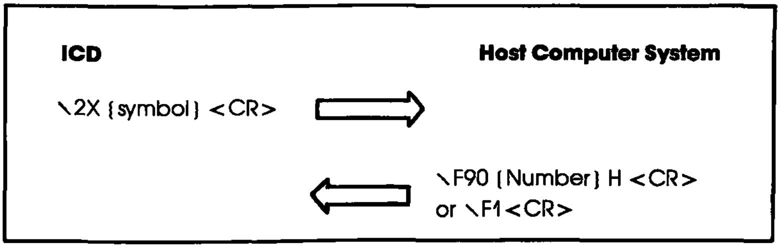
REMOTE MODE PROGRAM FLOW CHART



ORIGIN	**ACTION:**	The ICD uses this sequence to check the console input to the host computer. (The console input could be an inquiry about an interruption, restart or abort trace sequence, or dump output.)
ICD	\8A<CR>	CONSOLE KEY INPUT CHECK REQUEST RECORD. This record is sent to the host computer to request a console key input check.
HOST	\F8<CR>	NO CONSOLE INPUT RECORD. This record is sent to the ICD if there is no console input.
HOST	\F9{any ASCII code}<CR>	The host computer sends "<any ASCII code>" if there is any console key input.

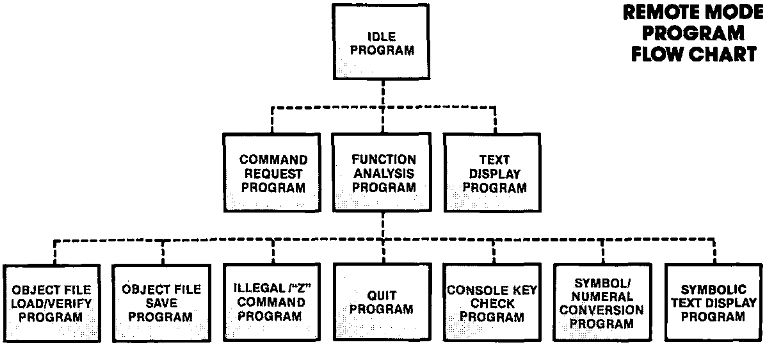
PROGRAM: SYMBOL/NUMERAL CONVERSION

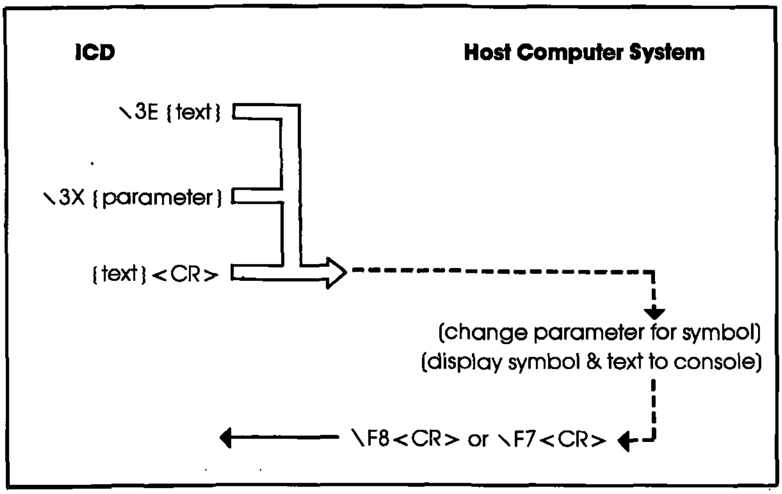




ORIGIN		ACTION:
		This sequence is used when the host computer requests a symbol or numeral conversion.
ICD	\2X{symbol} <CR>	SYMBOL/NUMERAL CONVERSION REQUEST RECORD. This is a record sent to the host computer requesting numeric conversion of a symbol. (The ICD sends {symbol}<CR> including "." which means a symbol.)
HOST	\F9 0 {number (hexadecimal ASCII)} H <CR>	NUMERIC RECORD. This record is sent to the ICD when the symbol received from the ICD has been converted to a numeral. (The host computer attaches 0 to the head of the converted value and "H"<CR> at the end.)
HOST	\F1<CR>	SYMBOL/NUMERAL CONVERSION ERROR CODE. This code is sent by the host computer when the symbol chosen cannot be converted to a numeral.

PROGRAM: SYMBOLIC TEXT DISPLAY





ORIGIN **ACTION:** The parameters sent from the ICD are displayed on the console screen after being converted to symbols.

ICD \ 3X{parameter} <CR> SYMBOL CONVERSION RECORD. This record tells the host computer to display the parameters after converting to symbols.

NOTE: The control codes <ACK>, <NAK>, and <ENQ> are not allowed in the symbolic text record.

NOTE: The header 3X before {parameter} *may contain values from 30 to 3F.*

HOST \F8<CR> DISPLAY COMPLETE CODE. This code is sent to the ICD when the symbol display and text in the symbolic text record have been completed.

HOST \F7<CR> DISPLAY INTERRUPT INDICATION CODE. The host computer sends this code to the ICD to interrupt the transmission of the symbolic text record.

PROGAM DESCRIPTION: **SYMBOLIC TEXT DISPLAY**

ACTION:

1. 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.

2. The host computer enters all data before <CR> into the input buffer and acts on the following:

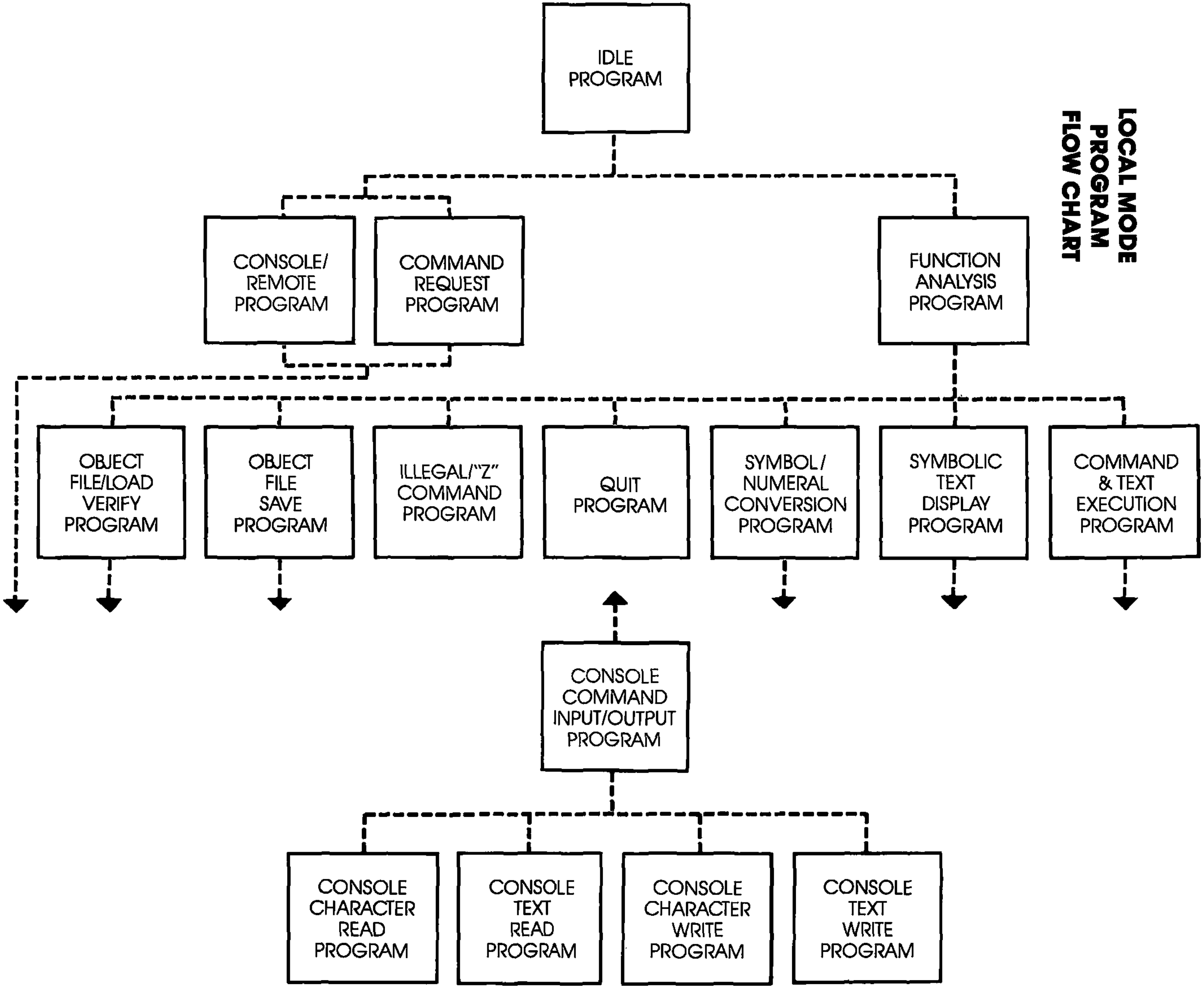
   a) If \3X{parameter} cannot be found in the input buffer, the host computer displays the contents of the input buffer already converted to symbols, sends \F8<CR> to the ICD, and then returns to the IDLE program.

   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 host computer returns to "a" (above) after converting \3X{parameter} to {parameter}.

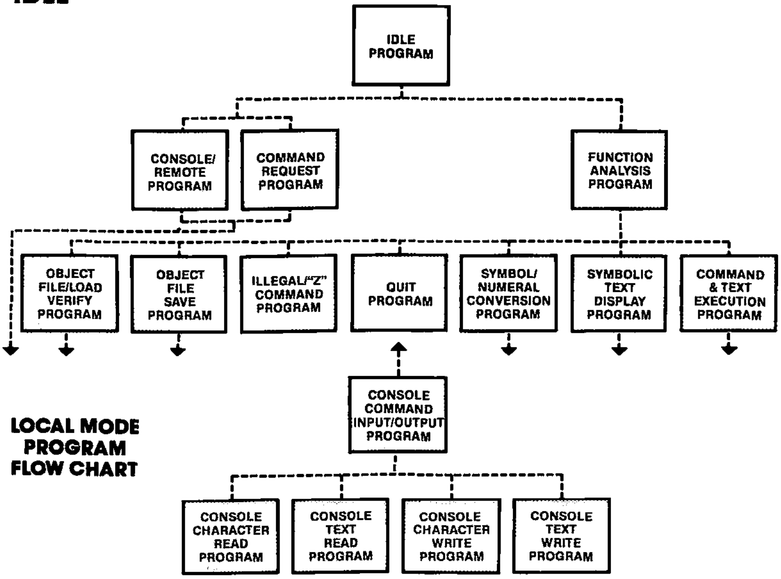
   If {parameter} is found in the symbol table, the host computer returns to "a" (above) after converting \3X {parameter} to a symbol.

LOCAL MODE PROGRAM FLOW CHART



LOCAL MODE: TERMINAL CONTROL OF THE ICD (WITH HOST DATA FILES)

PROGRAM: IDLE



LOCAL MODE PROGRAM FLOW CHART

DESCRIPTION: This program acts as the main intermediary program (transferring instructions and text only) between the ICD and the subprograms (Command Request and Function Analysis).

ACTION:

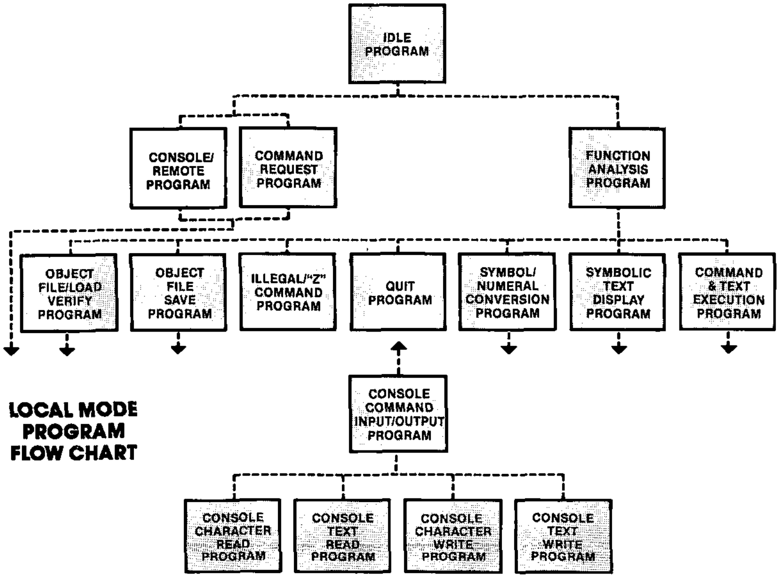
1. The host computer waits for an input from the ICD. (The host system must have an input buffer to hold the input code from the ICD.)

2. The host computer receives one line of data and places it in the input buffer.

3. The host computer then executes one of the following programs depending on the code it receives:

Code Received	Program Executed
\F0{text}<CR>	COMMAND REQUEST
any other	FUNCTION ANALYSIS

PROGRAM: COMMAND REQUEST—CONSOLE



LOCAL MODE PROGRAM FLOW CHART

ICD **Host Computer System**

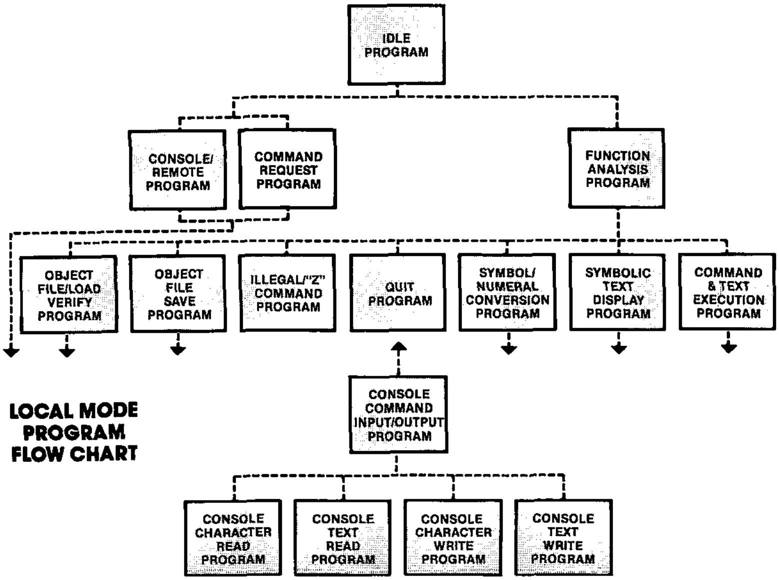
\F0{text}<CR> →

[
CONSOLE CHARACTER READ SEQUENCE
CONSOLE TEXT READ SEQUENCE
CONSOLE CHARACTER WRITE SEQUENCE
CONSOLE TEXT WRITE SEQUENCE
]

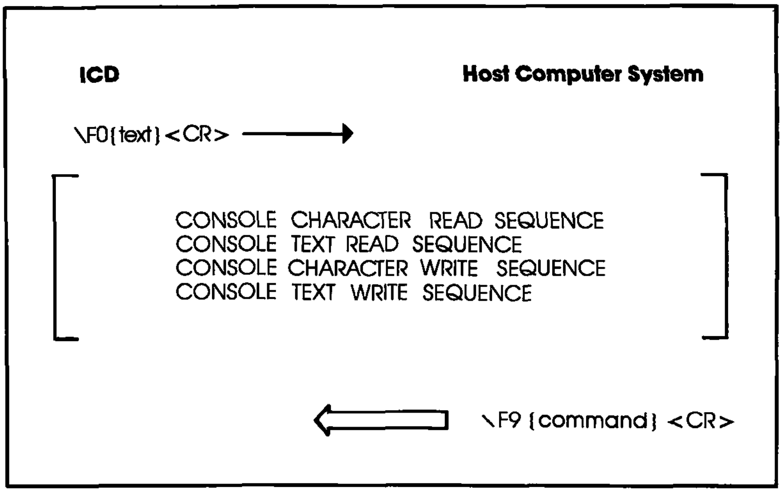
← \F1 <CR>

ORIGIN	**ACTION:**	These sequences allow commands to input to the ICD through the console terminal in the LOCAL mode.
ICD	\F0{text} <CR>	COMMAND INPUT STATUS WAIT CODE. This code is sent to the host computer before the ICD displays a prompt (>).
Optional sequences:		CONSOLE CHARACTER READ/WRITE SEQUENCE or CONSOLE TEXT READ/WRITE SEQUENCE
HOST	\F1<CR>	CONSOLE COMMAND INPUT REQUEST CODE. The ICD outputs a prompt to the console screen after receiving this code.

PROGRAM : COMMAND REQUEST — REMOTE



LOCAL MODE PROGRAM FLOW CHART



\F0{text}<CR>

CONSOLE CHARACTER READ SEQUENCE
CONSOLE TEXT READ SEQUENCE
CONSOLE CHARACTER WRITE SEQUENCE
CONSOLE TEXT WRITE SEQUENCE

\F9 {command} <CR>

ORIGIN **ACTION:** These sequences enable the ICD to directly execute commands in the LOCAL mode.

ICD \F0{text}<CR> COMMAND INPUT STATUS WAIT CODE. This code is sent to the host computer before the ICD displays a prompt (>).

Optional sequences: CONSOLE CHARACTER READ/WRITE SEQUENCE or CONSOLE TEXT READ/WRITE SEQUENCE

HOST \F9{ICD command}<CR> REMOTE COMMAND REQUEST RECORD. This record allows the ICD to execute commands directly. When the ICD receives this record from the host computer, it displays a prompt and the {command} on the console screen.

PROGRAM DESCRIPTION: **COMMAND REQUEST—CONSOLE/REMOTE**

ACTION:

1. The ICD requests a command by sending \F0{text}<CR> to the host computer from the ICD. Additionally, any of the following four console input/output sequences can be executed:

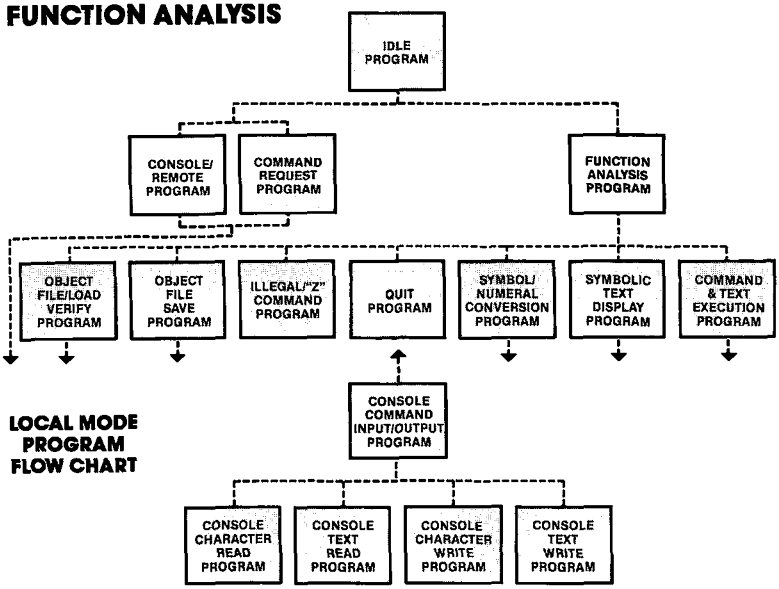
   a) CONSOLE CHARACTER READ PROGRAM
   b) CONSOLE TEXT READ PROGRAM
   c) CONSOLE CHARACTER WRITE PROGRAM
   d) CONSOLE TEXT WRITE PROGRAM

2. All of the console or remote commands can be executed when the host computer receives \F0{text}<CR>.

3. If console commands are used, the sequence ends with \F1<CR>, and the host computer returns to the IDLE program.

4. If remote commands are used, the sequence ends with \F9{ICD command}<CR>, and the host computer returns to the IDLE program.

PROGRAM: **FUNCTION ANALYSIS**



LOCAL MODE PROGRAM FLOW CHART

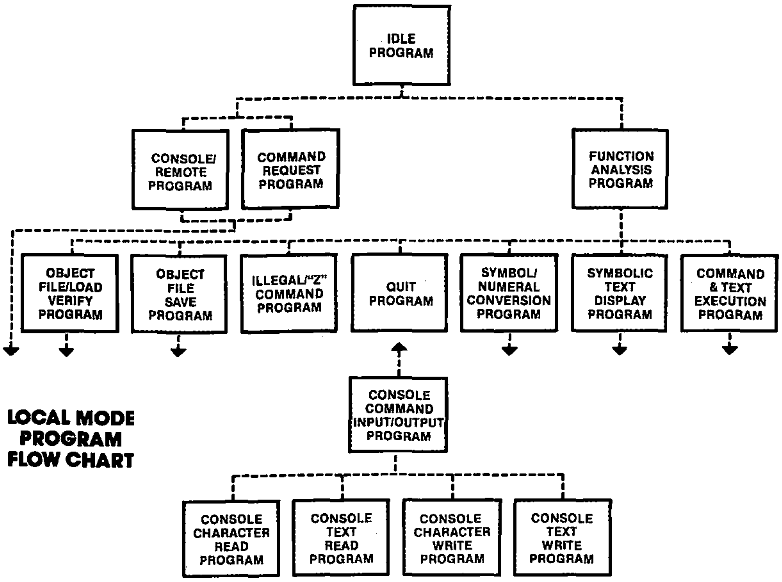
ACTION:

1. The host computer places one line of data (received from the ICD) into the input buffer and then analyzes the data.

2. The host computer then executes one of the following programs based on the contents of the input buffer:

Input Buffer Contents	Program Executed
\00{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

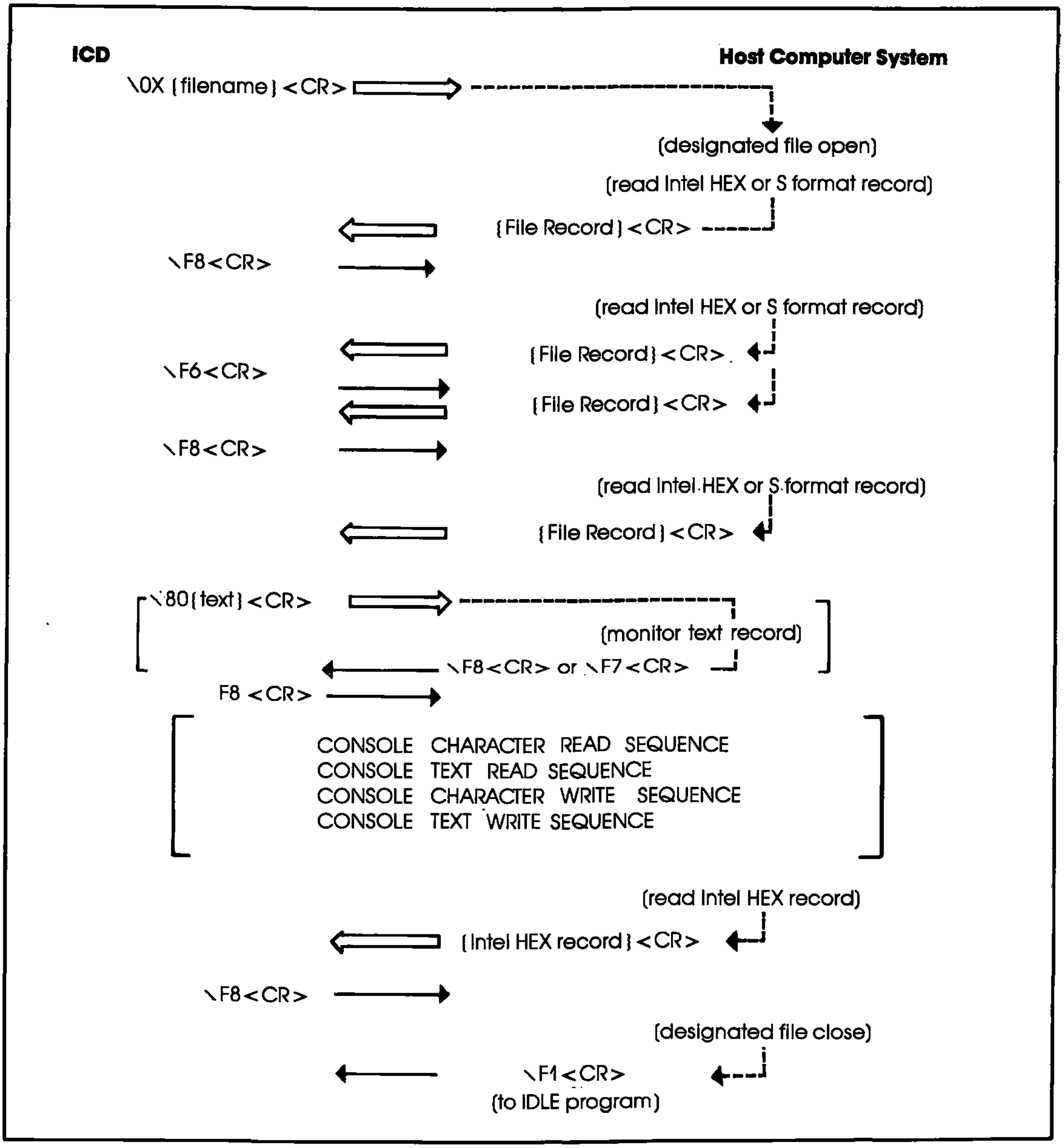
PROGRAM: OBJECT FILE LOAD/VERIFY



LOCAL MODE PROGRAM FLOW CHART

ORIGIN	**ACTION:**	
		An object file is sent from the host computer in response to a LOAD/VERIFY request from the ICD.
ICD	\00{filename} <CR> or \02{filename} <CR>	INTEL HEX LOAD REQUEST RECORD or S FORMAT LOAD REQUEST RECORD. This record is sent to the host computer when the ICD loads an object file.

ICD	\01{filename}<CR> or \03{filename}<CR>	INTEL HEX VERIFY REQUEST RECORD or S FORMAT VERIFY REQUEST RECORD. This record is sent to the host computer when the ICD verifies an object file with the memory.
HOST	{record}<CR>	OBJECT FILE RECORD. This Intel Hex or S format record is sent to the ICD from the host computer. This record may not contain any control code, and must end with <CR>.
ICD	\F8<CR>	OBJECT RECORD REQUEST CODE. This code is sent to the host computer to request an Intel Hex or S format record.
ICD	\F6<CR>	OBJECT RECORD RE-TRANSMISSION REQUEST CODE. This code is used when the ICD requests the host computer to re-transmit the object file. *NOTE: Most re-transmission requests are caused by an error occurring in the check sum of the Intel Hex or S format record.*
ICD	\F7<CR>	OBJECT FILE TRANSMISSION INTERRUPT CODE. When the host computer receives this request, it stops the LOAD/VERIFY sequence.
	Optional Sequences:	CONSOLE CHARACTER READ/WRITE SEQUENCE or CONSOLE TEXT READ/WRITE SEQUENCE
HOST	\F1<CR>	LOAD/VERIFY END CODE. The host computer sends this code to the ICD (after closing the file) if the file records are exhausted.
HOST	\F7<CR>	LOAD/VERIFY SEQUENCE ABORT INDICATION CODE. The host computer uses this code to inform the ICD that it is aborting the object LOAD/VERIFY sequence.


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)

PROGRAM DESCRIPTION: **OBJECT FILE LOAD/VERIFY**

ACTION:

1. The ICD sends \0X{filename}<CR> to the host computer to load or verify a user program. The host computer then opens the requested program file.

2. The host computer reads the Intel Hex or S format records from the file and acts on the following:

   a) If an error occurs when opening or reading the file, the host computer sends \F7<CR> to the ICD and returns to the IDLE program.

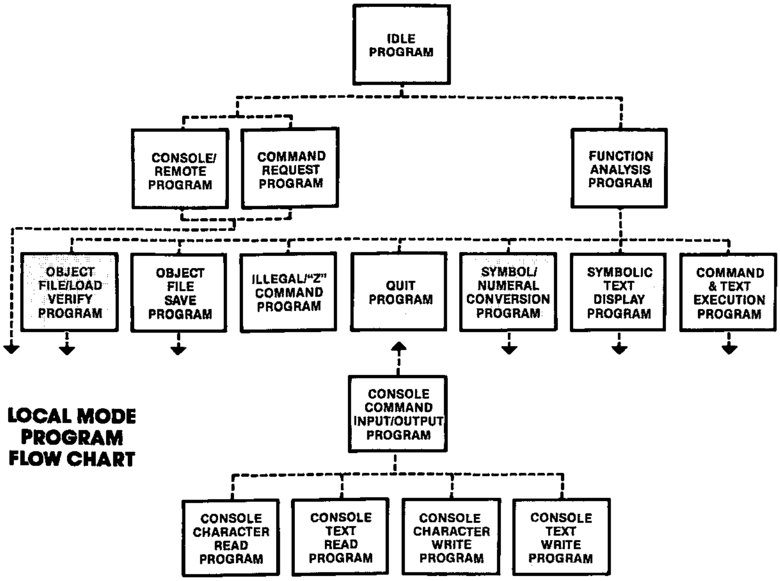
   b) If no error is detected, the host computer sends the Intel Hex or S format record to the ICD and then waits for \F8<CR> from the ICD.

      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.

   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.

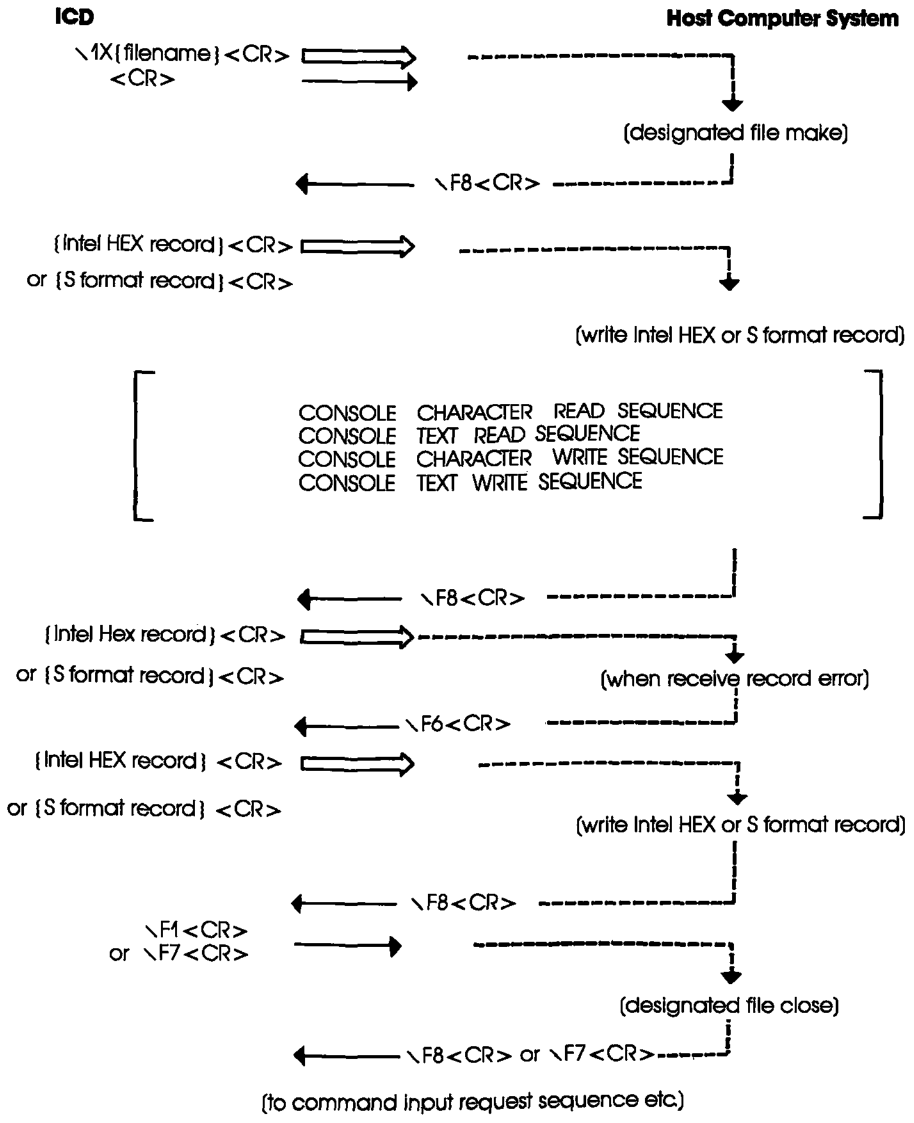
PROGRAM: OBJECT FILE SAVE



LOCAL MODE PROGRAM FLOW CHART

ORIGIN	**ACTION:**	When this command is issued, the host computer receives an object program and creates a file.
ICD	\10{filename} <CR> \12{filename} <CR>	INTEL HEX SAVE REQUEST RECORD or S FORMAT SAVE REQUEST RECORD. This record is sent by the ICD to request the host computer to save a file. The {filename} field may be used for a user-defined save message.

HOST	\F8<CR> (at file write)	OBJECT RECORD REQUEST INDICATION CODE. This code is sent to the ICD from the host computer to request an Intel Hex or S format record.
ICD	{record}<CR>	OBJECT FILE RECORD. This Intel Hex or S format record is sent to the host computer.
	Optional Sequences:	CONSOLE CHARACTER READ/WRITE SEQUENCE or CONSOLE TEXT READ/WRITE SEQUENCE
HOST	\F8<CR>	OBJECT RECORD REQUEST INDICATION CODE. The host computer sends this code to the ICD to request a record.
HOST	\F6<CR>	OBJECT RECORD RE-TRANSMISSION REQUEST CODE. This code requests the ICD to re-transmit an object code.
ICD	\F1<CR>	FILE END CODE. The ICD sends this code to the host computer when the transmission of the file records has been exhausted. The host computer ends the object save sequence by sending \F8<CR> after closing the file.
HOST	\F7<CR>	SAVE SEQUENCE ABORT INDICATION CODE. The host computer informs the ICD that it is aborting the object save sequence.
HOST	\F8<CR> (at file close)	FILE CLOSE END CODE. This code is sent to the ICD from the host computer when the file close is successful.
HOST	\F7<CR>	SAVE SEQUENCE ABORT INDICATION CODE. This code indicates that the host computer has stopped the object save sequence.


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.)

PROGRAM DESCRIPTION: **OBJECT FILE SAVE**

ACTION:

1. The ICD sends \1X{filename}<CR> to the host computer when saving a user program. The host computer then opens the selected user file.

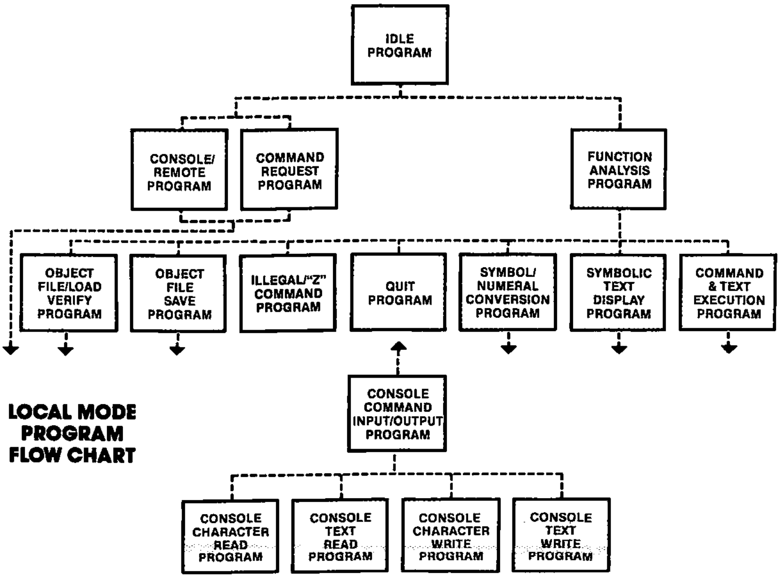
   If the file does not open, the host computer sends \F7<CR> to the ICD and returns to the IDLE program. If the file opens, the host computer sends \F8<CR> to the ICD.

2. The host computer 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.

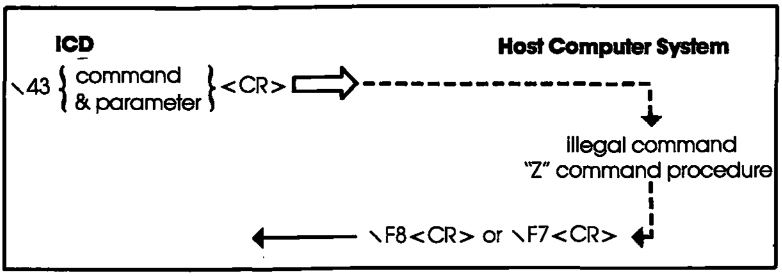
   After receiving an Intel Hex or S format record, the host computer then executes a file write of the record received from the ICD. If an error occurs during the file write operation, the host computer closes the user program file, sends \F7<CR> to the ICD, and returns to the IDLE program.

   If an error occurs in a sum check, the host computer sends \F6<CR> and waits for the Intel Hex or S format record to be retransmitted from the ICD. The host computer then waits for the next Intel Hex or S fsormat record (sending \F8<CR> to the ICD) if no error occurs during the file write.

PROGRAM: **ILLEGAL/"Z" COMMAND**



LOCAL MODE PROGRAM FLOW CHART



ORIGIN	**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 is interpreted and processed by the host computer.
ICD	\43{parameter} <CR>	ILLEGAL COMMAND/"Z" COMMAND RECORD. This record is sent to the host computer to process the ILLEGAL/"Z" command.
HOST	\F8<CR>	ILLEGAL COMMAND/"Z" COMMAND NORMAL END CODE. This code is sent to the ICD when the ILLEGAL/"Z" command has been processed successfully.

HOST \F7<CR> ILLEGAL COMMAND/"Z" COMMAND ABNORMAL END CODE. This code is sent to the ICD when the ILLEGAL/"Z" command has not been processed successfully.

PROGRAM DESCRIPTION: **"Z" COMMAND**

ACTION:

1. The ICD sends \43{parameter}<CR> (and the specified "Z" command) to the host computer.
2. The host computer then performs the specified "Z" command and acts on the following:

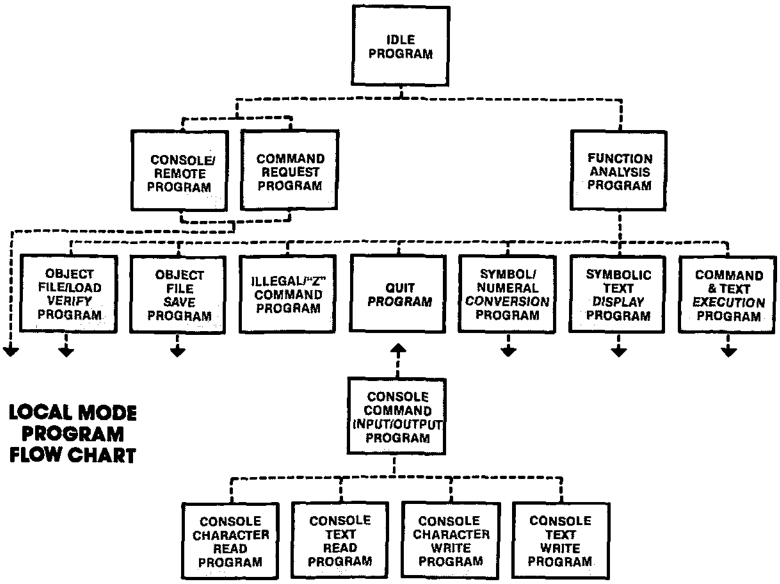
   If an error is contained in the "Z" command specification, the host computer sends \F7<CR> to the ICD and then returns to the IDLE program.

   If no error is detected, the host computer sends \F8<CR> to the ICD and then returns to the IDLE program.

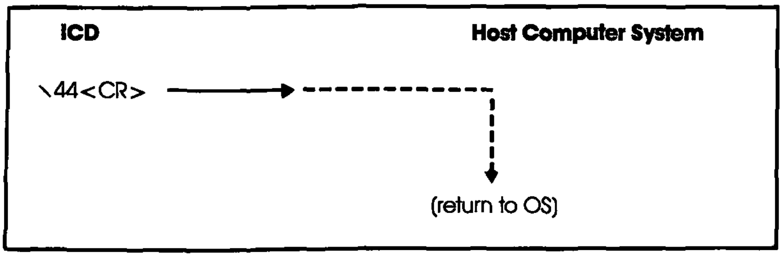
"Z" commands available with ZICE Z80 (V2.4) include:

ABBREVIATION	NAME	FUNCTION
HE	Help	Displays the command list.
DEF	Define	Adds a new symbol to the ZICE symbol table.
SL	SLoad	Reads a symbol file on diskette into the ZICE symbol table.
DEL	Delete	Deletes a symbol from the ZICE symbol table.
SS	SSave	Stores the ZICE symbol table on a diskette as the symbol file.
SH	Show	Displays the names and values of symbols in the ZICE symbol table and their qualities.
LOG	Log	Stores everything displayed into a specified file.
BA	Batch	Executes a file of ICD commands.

PROGRAM: **QUIT**



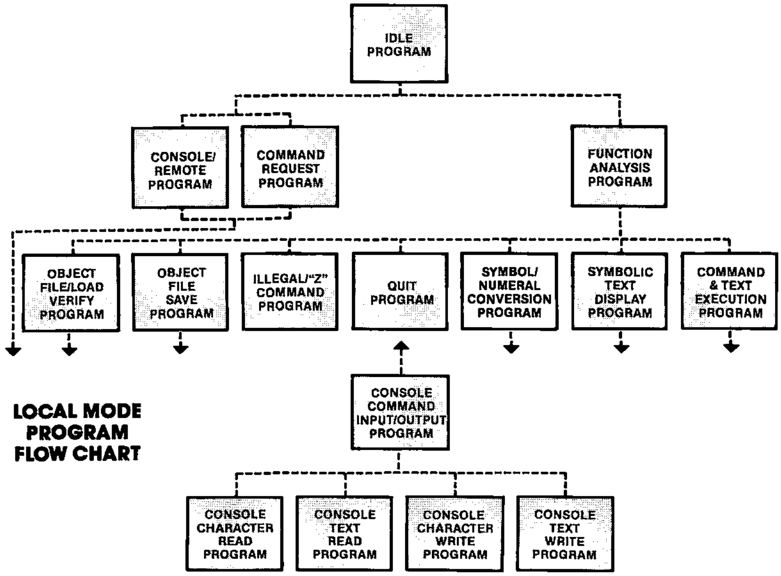
LOCAL MODE PROGRAM FLOW CHART



ORIGIN **ACTION:** The host computer returns to the operating system (OS) indicated by the ICD's code. (The ICD "HOST ON" mode is also cancelled.)

ICD \44<CR> QUIT RECORD.

PROGRAM: SYMBOL/NUMERAL CONVERSION



LOCAL MODE PROGRAM FLOW CHART

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>

ORIGIN ACTION: This sequence is used when the host computer requests a symbol/numeral conversion.

ICD \2X{symbol}<CR> SYMBOL/NUMERAL CONVERSION REQUEST RECORD. This record is sent to the host computer requesting the numeric conversion of a symbol. (The ICD sends {symbol}<CR> including ". " which means a symbol.)

Optional Sequences: CONSOLE CHARACTER READ/WRITE SEQUENCE or CONSOLE TEXT READ/WRITE SEQUENCE

HOST \F9 0 {number (hexadecimal ASCII)} H <CR> NUMERIC RECORD. This record is sent to the ICD when the symbol received has been converted to a numeral. (The host computer attaches 0 to the head of the converted value and "H"<CR> at the end.)

HOST \F1<CR> SYMBOL/NUMERAL CONVERSION ERROR CODE. This code is sent by the host computer when the symbol chosen cannot be converted to a numeral.

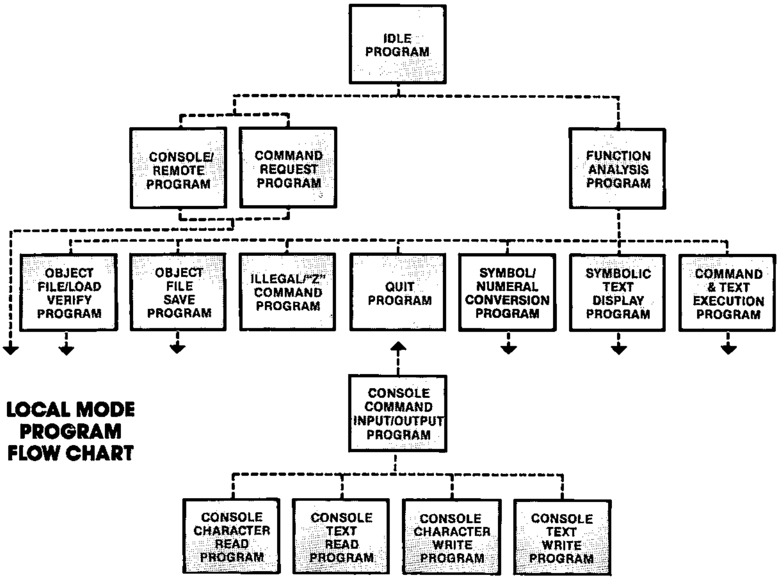
PROGRAM DESCRIPTION: **SYMBOL CONVERSION**

ACTION: 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:

1. If the conversion is successful, the host computer sends the numeral to the ICD with "0" attached at the head and "H" followed by <CR> at the end, and then returns to the IDLE program.

2. If the conversion is unsuccessful, the host computer sends \F1<CR> to the ICD and returns to the IDLE program.

PROGRAM: **NUMERAL CONVERSION**



ACTION:

1. The ICD sends \3E{text which includes \3X {parameter}} <CR> to the host computer when the numeral/symbol conversion program is executed.

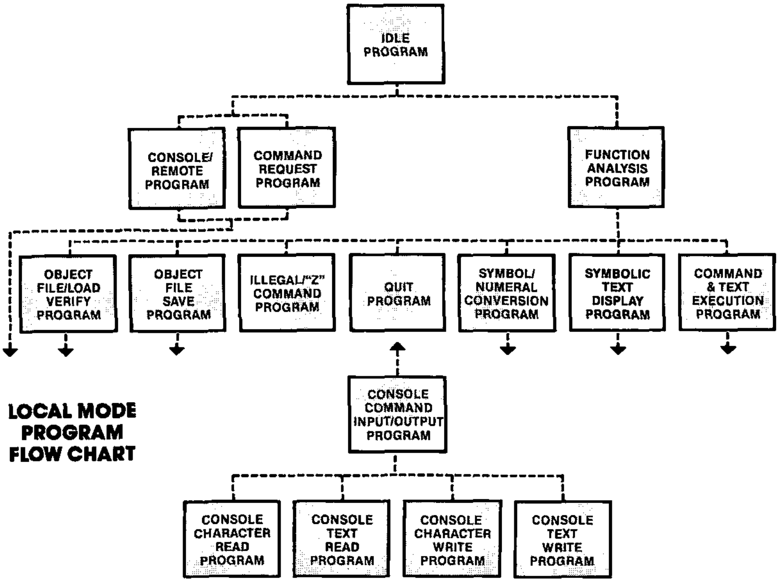
2. The host computer enters all data before <CR> into the ICD input buffer.

3. The host computer then searches the input buffer for \3X {parameter} and executes one of the following:

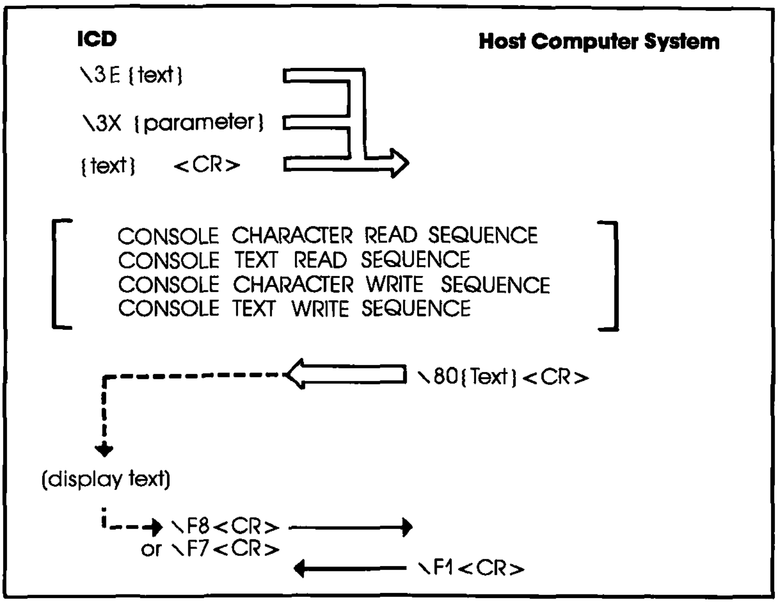
   a) If \3X{parameter} is not found, the host computer sends out the text—attaching \80 to the front and <CR> to the end of the text—and then waits for \F8<CR> from the ICD. When \F8<CR> is received from the ICD, the host computer sends \F1 to the ICD and returns to the IDLE program.

   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 returns to"3"(above). If {parameter} is found in the symbol table, the system converts \3X{parameter} to a symbol.

PROGRAM: SYMBOLIC TEXT DISPLAY

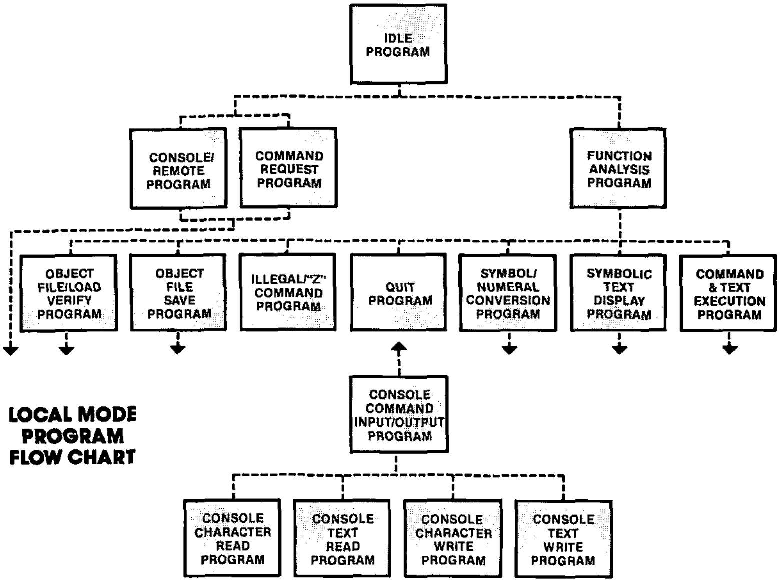


LOCAL MODE PROGRAM FLOW CHART

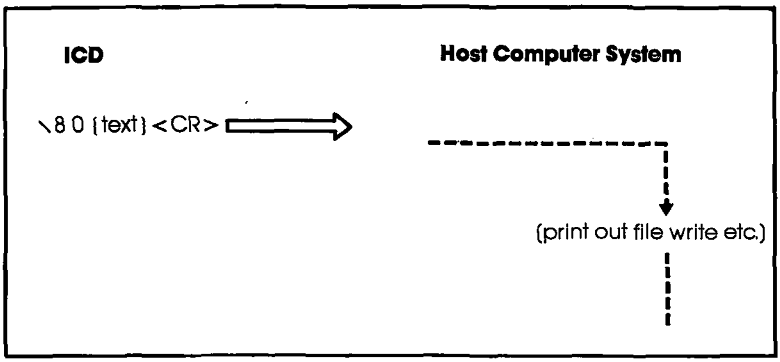


ORIGIN	**ACTION:**	The parameters sent from the ICD are displayed on the console screen after being converted to symbols.
ICD	\3E {text which includes \3X} <CR>	NUMERAL/SYMBOL CONVERSION RECORD. This record tells the host computer to convert a numeral to a symbol. (\3X is a header.)
Optional Sequences:		CONSOLE CHARACTER READ/WRITE SEQUENCE or CONSOLE TEXT READ/WRITE SEQUENCE
HOST	\80{text} {change to symbol} <CR>	NUMERAL/SYMBOL RECORD. The host computer sends this record if the numeral is successfully converted to a symbol.
ICD	\F8<CR>	DISPLAY END CODE. This code is sent to the ICD when the symbol display and text in the symbolic text record have been completed.
HOST	\F1<CR>	NUMERAL/SYMBOL CONVERSION END CODE. The host computer sends this code to the ICD to end the sequence.

PROGRAM: COMMAND AND TEXT EXECUTION

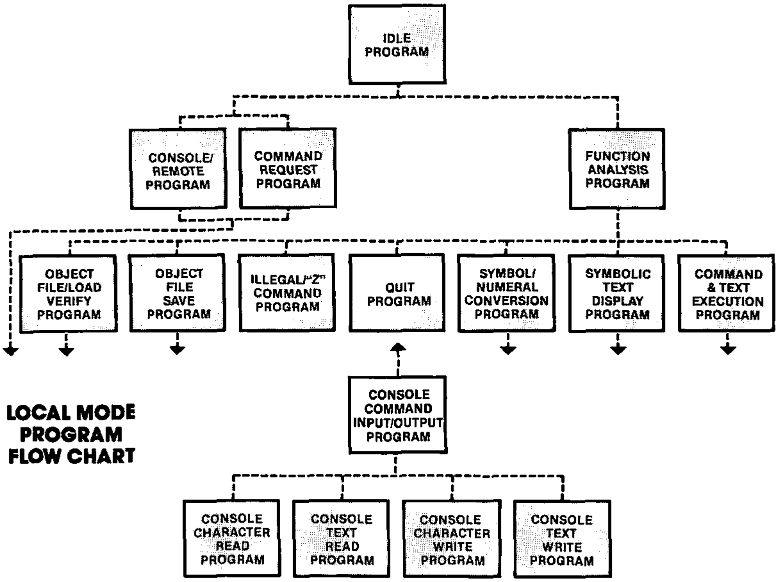


LOCAL MODE PROGRAM FLOW CHART



ORIGIN	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 treated as an Illegal/"Z" command after the HOST ON command is issued.*
ICD	\80{text} <CR>	COMMAND EXECUTION TEXT. Outputs one line of text after the execution of the command by the ICD.
HOST	\F8 <CR>	TEXT RECEPTION COMPLETE CODE. This code is transmitted to the ICD when the host computer has received the text and completed the output execution.

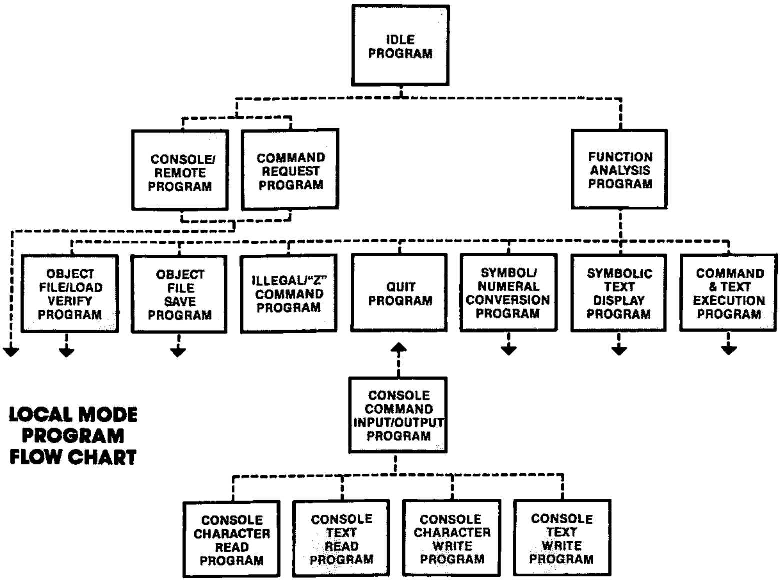
PROGRAM: **CONSOLE COMMAND INPUT/OUTPUT**



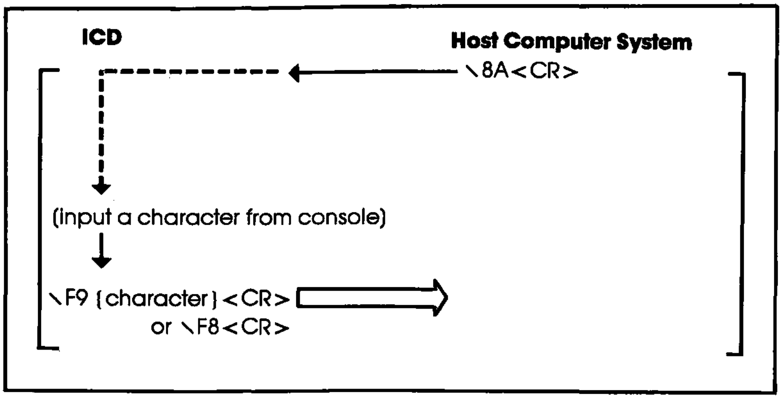
ACTION: There are four input/output sequences available when the ICD operates in the LOCAL mode:

1) CONSOLE CHARACTER READ
2) CONSOLE TEXT READ
3) CONSOLE CHARACTER WRITE
4) CONSOLE TEXT WRITE

PROGRAM: CONSOLE CHARACTER READ

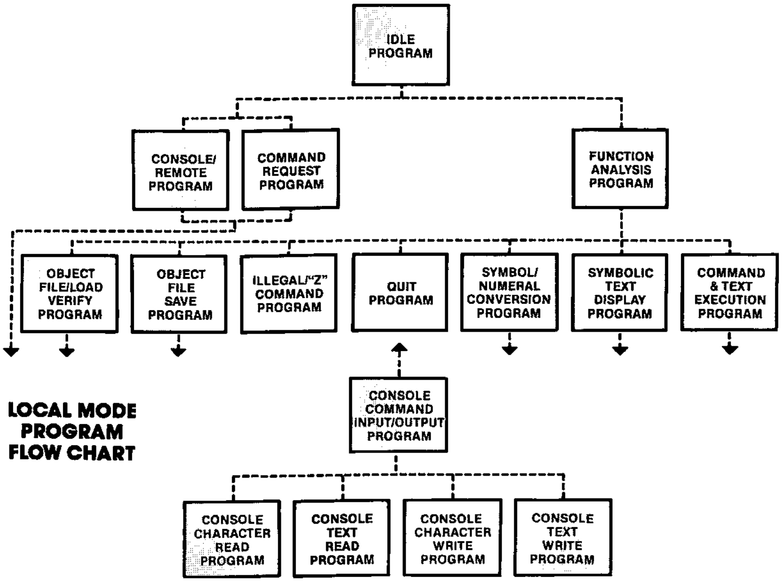


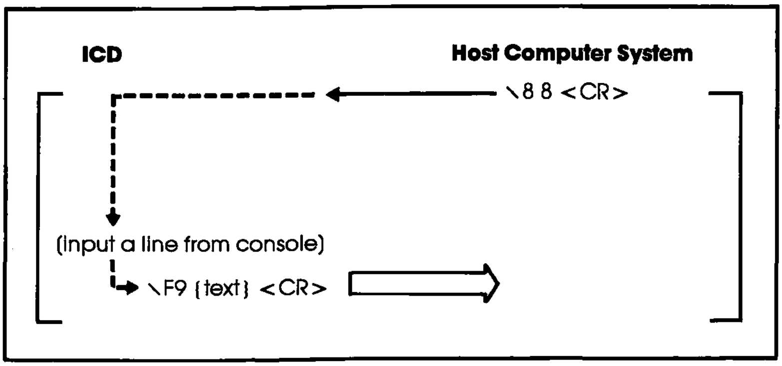
LOCAL MODE PROGRAM FLOW CHART



ORIGIN	**ACTION:**	The host computer uses this sequence to request a single character of data from the console through the ICD.
ICD	\8A	CONSOLE KEY INPUT REQUEST CODE.
HOST	\F9{input character} <CR>	CONSOLE INPUT CODE. This code is sent to the 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 back the console input.)
HOST	\F8<CR>	NO CONSOLE KEY INPUT CODE. The ICD sends this code to the host computer if there is no console input.

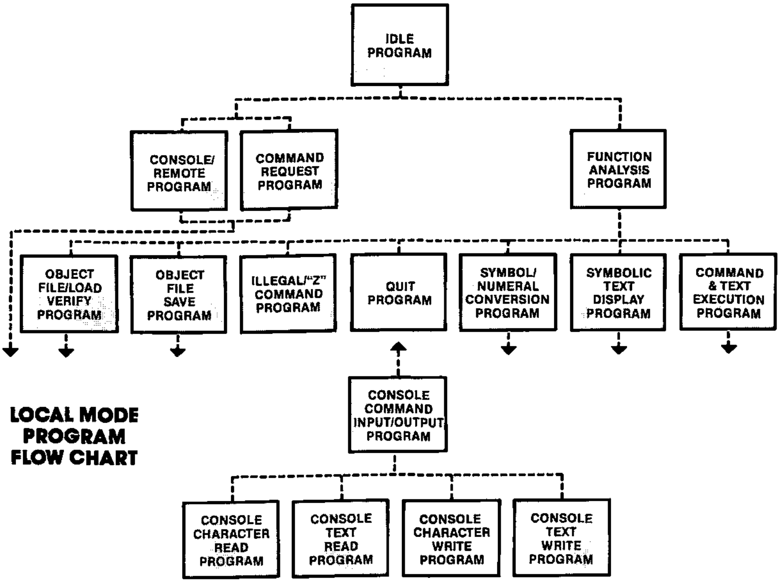
PROGRAM: CONSOLE TEXT READ



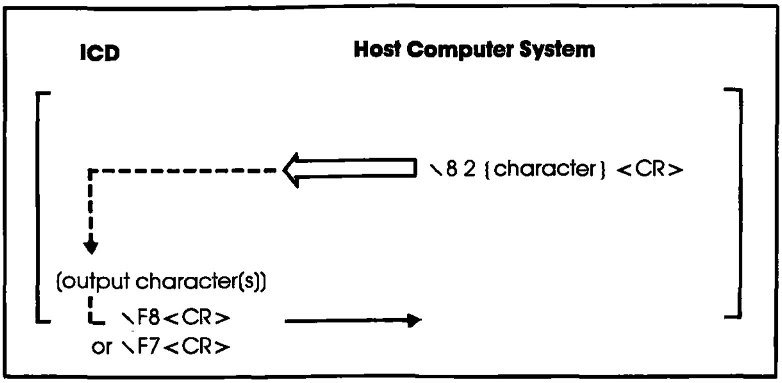


ORIGIN	**ACTION:**	This sequence is used when the host computer requests the ICD to input one line of data.
HOST	\88<CR>	DATA INPUT REQUEST CODE. The host computer sends this code to the ICD to request one line of data.
ICD	\F9{line of data} <CR>	DATA INPUT CODE. This code is sent to the host computer along with the line of data entered from the console terminal. The maximum number of input characters is limited to 255; subsequent characters are discarded.

PROGRAM: CONSOLE CHARACTER WRITE

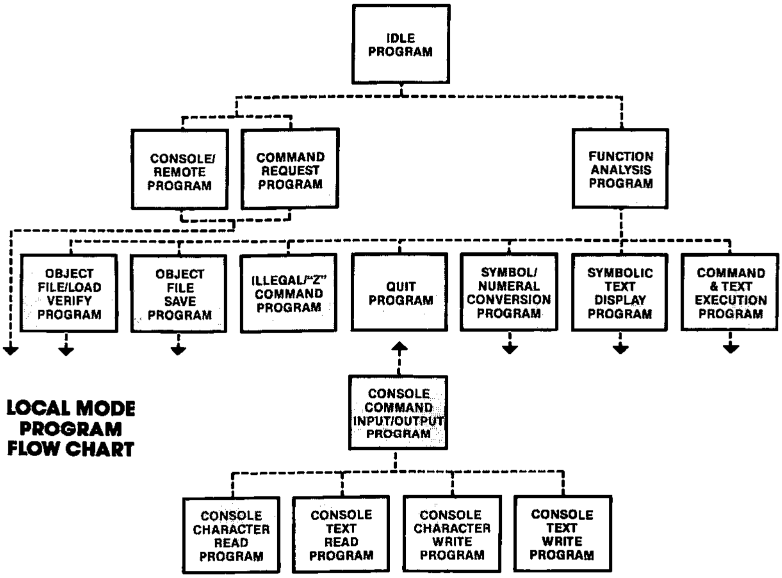


LOCAL MODE PROGRAM FLOW CHART

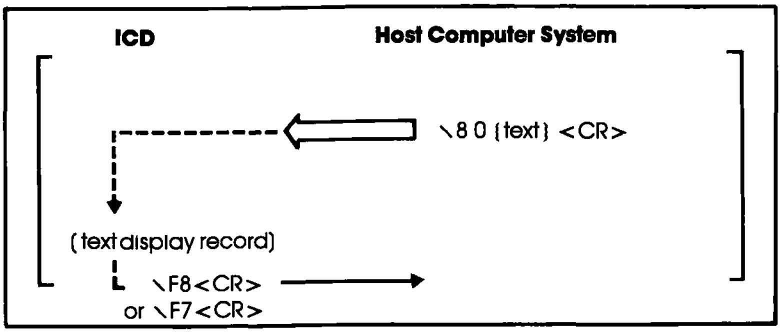


ORIGIN	**ACTION:**	This sequence is used when the host computer requests the ICD to output n characters to the console.
HOST	\82{characters} <CR>	N CHARACTERS OUTPUT REQUEST CODE. The host computer sends this code to the ICD, requesting the output of n characters to the console. The ICD then sends {characters} to the console, without being followed by a <CR><LF>.
ICD	\F8<CR>	N CHARACTERS OUTPUT END CODE. This code is sent to the host computer from the ICD when the n-character output to the console is completed.

PROGRAM: **CONSOLE TEXT WRITE**



LOCAL MODE PROGRAM FLOW CHART



ORIGIN	**ACTION:**	This sequence is used when the host computer requests the ICD to output one line of data to the console.
HOST	\80{text} <CR>	DATA OUTPUT REQUEST RECORD. This record requests the 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.
ICD	\F8<CR>	CONSOLE OUTPUT END CODE. This code is sent to the host computer when the sequence is completed.

NUMBER CONVERSION CODES
ICD278/Z80

number change code	format	description
\20	\20 { .symbol } <CR>	address symbol
\21—\2F	not used	

NUMBER CONVERSION CODES
ICD278/I8085

number change code	format	description
\20	\20 { .symbol } <CR>	address symbol
\21—\2F	not used	

NUMBER CONVERSION CODES
ICD178/I8086, I8088

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	

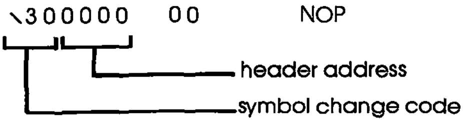
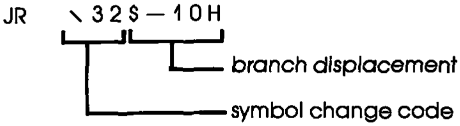
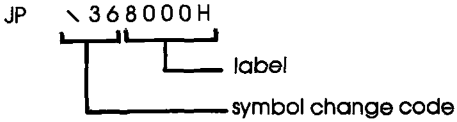
NUMBER CONVERSION CODES
ICD178/I8048

number change code	format	description
\20	\20 { .symbol } <CR>	address symbol
\21—\2F	not used	

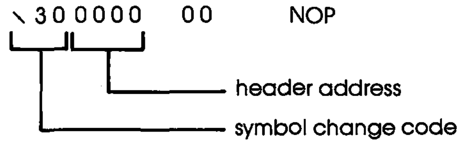
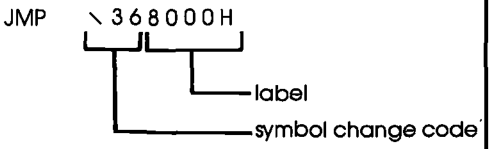
NUMBER CONVERSION CODES
ICD178/6800, 68010, 68008

number change code	format	description
\20	\20 {.symbol} <CR>	address symbol
\21—\2F	not used	

SYMBOL CONVERSION CODES
ICD278/Z80

symbol change code	description	example
\30	header	
\31	not used	
\32	branch displacement	
\33—\35	not used	
\36	label	
\37—\3F	not used	

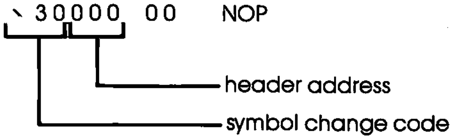
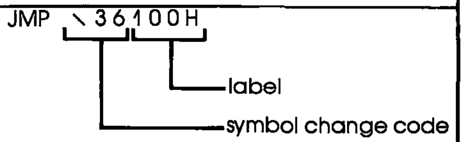
SYMBOL CONVERSION CODES
ICD278/I8085

symbol change code	description	example
\30	header	
\31—\35	not used	
\36	label	
\37—\3F	not used	

SYMBOL CONVERSION CODES
ICD178/I8086, I8088, I80186, I80188

symbol change code	description	example
\30	physical header	$\underbrace{\backslash 30}_{\text{symbol change code}}\underbrace{00000}_{\text{physical header address}}$ 90 NOP
\31	logical header	$\underbrace{\backslash 31}_{\text{symbol change code}}\underbrace{0000:0000}_{\text{logical header address}}$ 90 NOP
\32	branch displacement	JMP $\underbrace{\backslash 32}_{\text{symbol change code}}\underbrace{\$-10H}_{\text{branch displacement}}$
\33	not used	
\34	number	MOV AL, $\underbrace{\backslash 34}_{\text{symbol change code}}\underbrace{55H}_{\text{number}}$
\35 — \ 36	not used	
\37	label	JMP $\underbrace{\backslash 37}_{\text{symbol change code}}\underbrace{8000H}_{\text{label}}$ (or\370100H: 8000H)
\38	not used	
\39	variable	MOV AL, BYTE PTR $\underbrace{\backslash 39}_{\text{symbol change code}}\underbrace{0000H: 1000H}_{\text{variable logical address}}$
\3A—\3F	not used	

SYMBOL CONVERSION CODES
ICD178/I8048

symbol change code	description	example
\30	header	
\31—\35	not used	
\36	label	
\37—\3F	not used	

SYMBOL CONVERSION CODES
ICD178/68000, 68010, 68008

symbol change code	description	example
\30	header	$\underbrace{\texttt{\\30}}_{\text{symbol change code}}\underbrace{\texttt{000000}}_{\text{header address}}$
\31—\33	not used	
\34	number	$\texttt{MOVE. B }\underbrace{\texttt{\\34}}_{\text{symbol change code}}\underbrace{\texttt{\$00}}_{\text{number}}\texttt{,DO}$
\35	not used	
\36	label	$\texttt{JMP }\underbrace{\texttt{\\36}}_{\text{symbol change code}}\underbrace{\texttt{\$00001000}}_{\text{label}}\texttt{.W}$
\37	not used	
\38	variable	$\texttt{MOVE. B D0,}\underbrace{\texttt{\\38}}_{\text{symbol change code}}\underbrace{\texttt{\$00001000}}_{\text{variable}}\texttt{.W}$
\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:

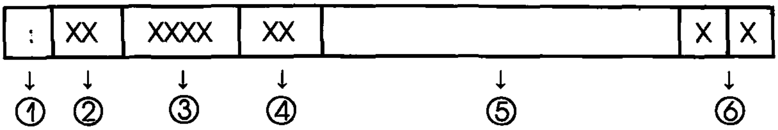
00_H	"00"(3030_H)
$9B_H$	"9B"(3942_H)

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
:2000000081000D00525A58608900040000CA00BAD95FFF4DF9AE52DA725FFD4FF2F808384
:0E0020000818001085A5A58B040000B0000490
:020000020200FA
:2000000008000000052EA1050A6401D0000B000FAF1DFFB0FFBBA50DAF35DFF5FF4FE008E1
:20002000081000D005A1A506000100D000000007AD05FFC0DFFAE125A585FFF2FF0F3009FD
:0400000301000000F8
: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
↓ ↓   ↓   ↓              ↓                          ↓
①②   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:

: 04 0000 03 51620005 41
↓ ↓ ↓ ↓ ↓ ↓
① ② ③ ④ ⑤ ⑥

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)

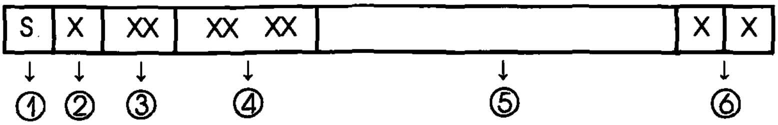
S FORMAT OBJECT FILE: All object files are described by ASCII codes. In the example, one byte of data is shown converted to a hexadecimal number ("0"—"9," and "A"—"F") of two digits:

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
```

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
↓ ↓ ↓   ↓     ↓     ↓
①② 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
↓ ↓ ↓   ↓                   ↓                   ↓
①②③    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:

```
S  8  04  010000  FA
↓  ↓  ↓     ↓     ↓
①  ②  ③     ④     ⑤
```

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.

Appendix A: Principles of Emulation, is being prepared now, and will be sent to you as soon as it becomes available.

ICD Product Demonstration: Features & Functions Of The ICD

Introduction

If this is the first time you are using a **ZAX** emulator, you've turned to the right place! In Appendix B, you'll be shown two exercises which you can use as a product training course. By following the exercises presented in this appendix, you'll not only demonstrate to yourself the powerful debugging capabilities of your ICD, but you'll learn more about emulation principles as well. Once you've familiarized yourself with some basic command functions and applications, you can then go back to the Master Command Guide in Section 2 and become an emulation expert!

Two Different Exercises!

You have two exercises to choose from in this appendix, and each exercise is designed to teach you something new about your ICD. The exercises are intended to work with whatever system configuration you are operating in (see "How To Connect Your ICD To Other Devices"). For example, if you're controlling the ICD with a terminal, and not using a target system, first construct the system configuration for that mode (Using The ICD Without A Target System: Terminal Controlled), and then find the exercise that is intended for that configuration (Exercise 1: Target System Not Used).

The system configurations and related exercises are shown below.

SYSTEM CONFIGURATION	EXERCISE
Using The ICD Without A Target System: Terminal Or Host Computer Control Of The ICD	See Exercise 1
Using The ICD With A Target System: Terminal Or Host Computer Control Of The ICD	See Exercise 2

Important! If this is the first time you are using a **ZAX** emulator, you should read through and then carry out Exercise 1: Using The ICD Without A Target System. This session reveals many of the ICD's capabilities, including performing actual emulation of a test program. (If you need a refresher course in emulation theory and practices, read through Appendix A before you try the exercises.)

Entering The Commands You don't need to know all about the command rules to use the ICD feature demonstration. Just carry out the instructions under ACTION and read the display on your terminal's screen. However, you must remember to enter the exact items as shown in the exercise—including feature characters (,/ =)—and provide spaces at the appropriate places as shown in the instructions.

If you make a mistake, the ICD will probably respond with an error message. It's usually not a big problem—just check to see that the proper characters, numbers, or spaces were used, and then re-enter the complete command statement.

Exercise 1: ICD Product Demonstration—Using The ICD Without A Target System.
System Configuration: Terminal Control of the ICD.
Operation Mode: LOCAL

ACTION	COMMAND DEMONSTRATED	COMMENT
PRESS: The yellow RESET button on the ICD and look at the screen.		You'll see the ICD's identification message followed by a prompt: ICD-278 for Z80 V2.0 > The prompt (>) indicates that the ICD is waiting for a command. After you've executed a command, or whenever an emulation process is completed, a new prompt will appear. Now return to your terminal's keyboard.
PRESS: G followed by a <cr>.	GO	The MONITOR lamp on the ICD goes off.
PRESS:The RESET switch and then look at the screen.		Nothing happened, right? That's because the ICD will not respond to a RESET input unless the MONITOR lamp is lit. This condition will occur whenever you're emulating a program as well.
PRESS: The MONITOR button on the ICD to exit.		A new prompt appears.
PRESS: The RESET switch.		
PRESS: R (followed by a RETURN; enter a RETURN after each action is executed).	REGISTER	Shows status of registers.
ENTER: F 0,2000,00	FILL	Fills 8K of internal memory with 00H (NOP instructions). It takes a few seconds for the ICD to do this—wait to see the prompt.

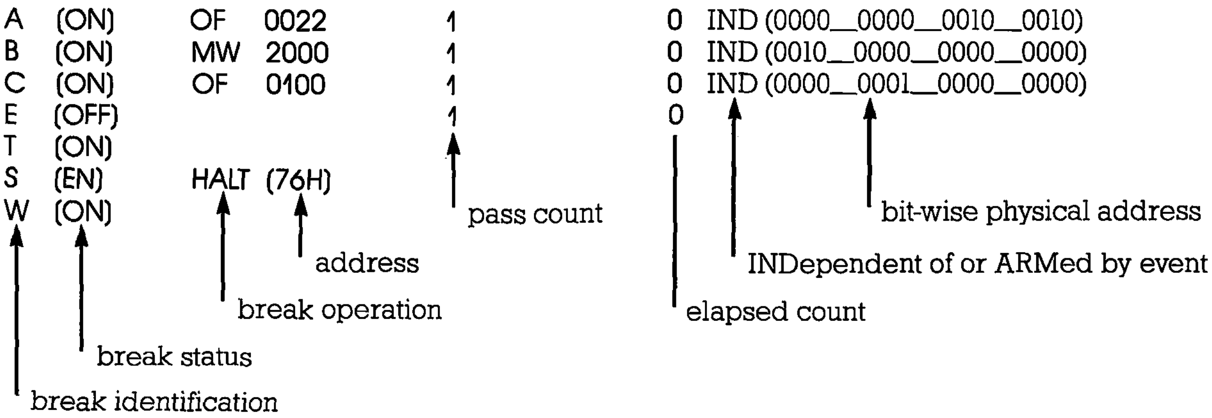
ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: D 0,FFF and, after a few seconds, PRESS: the space bar on your terminal.	DUMP	Displays contents of memory in HEX and ASCII. After the space bar is pressed, the scrolling will stop. Alternately press the space bar to start and stop the scrolling.
PRESS: The ESC key to exit.		
ENTER: DI 0,F	DISASSEMBLE	Disassembles contents of memory into assembly instructions. Now enter a program using the in-line assembler. This program will be used in the next examples as well.
ENTER: A 0		The ICD responds with 0000 and waits for your entry.

ICD displays: You enter:

```
0000    LD   SP,04000H
0003    LD   A,2
0005    LD   HL,1000H
0008    LD   BC,2000H
000B    CP   1
000D    JP   Z,15H
0010    LD   E,0AAH
0012    JP   17H
0015    LD   E,55H
0017    PUSH AF
0018    LD   (HL),E
0019    LD   A,(HL)
001A    CPI
001C    JP   NZ,0100H
001F    JP   PE,18H
0022    POP AF                     End of memory loop.
0023    SUB 1
0025    JP   NZ,5
0028    HALT                       End of test.
0029    NOP
002A
```

ACTION	COMMAND DEMONSTRATED	COMMENT
PRESS: The RETURN or the ESC key to exit the program.		
ENTER: DI 0,29	DISASSEMBLE	Displays the program just entered. The program just entered tests memory from 1000H to 3000H by writing alternate data patterns of 55s and AAs. After writing to a memory location, a verification is made by a read. In this first example, you will use this program to demonstrate how breakpoints are used, and emulation memory manipulated. You will also perform a trace of the program memory using the real-time trace buffer. In the second example (still using the same program) you will trace instructions and display the data in a single-step and jump-step manner. The third example demonstrates the remaining principal commands. THIS IS THE START OF EXAMPLE 1.
ENTER: B/C OF 100	BREAK	Sets the location of the ERROR message.
ENTER: B/A OF 22	BREAK	Sets a hardware (A) breakpoint.
ENTER: B/B MW,2000	BREAK	Sets a hardware (B) breakpoint.
ENTER: B S=HALT	BREAK	Uses the HALT code to implement the software breakpoint.

ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: B S=EN	BREAK	Enables (EN) software (S=) breakpoints to occur.
ENTER: B	BREAK	Displays status of the breakpoints. Compare with the display below:



```
A   (ON)    OF    0022          1          0  IND (0000_0000_0010_0010)
B   (ON)    MW    2000          1          0  IND (0010_0000_0000_0000)
C   (ON)    OF    0100          1          0  IND (0000_0001_0000_0000)
E   (OFF)                       1          0
T   (ON)
S   (EN)    HALT  (76H)
W   (ON)
```

NOTE: A,B,C = hardware break names, E = event break, T = ready timeout break, S = software break opcode, W = write-protect break.

ENTER: H CLR	HISTORY	Clears the real-time trace buffer.
ENTER: H BM	HISTORY	Sets the trigger mode of the real-time trace buffer (called up using the HISTORY command) to the Begin Monitor (BM) mode.

ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: H	HISTORY	Displays the trace status of the real-time trace buffer. Compare with the display below: Clock Counter = 00000000/0 Storage Mode = BM 2045 Storage Size = 0 "Clock Counter" shows the number of clocks (T-states) since the HISTORY command was cleared. "Storage Mode" shows the trace mode and trace range. "Storage Size" shows the number of cycles since the program began or since it was resumed.
ENTER: EV ST=MR,A=2000,D=55	EVENT	Sets an event (EV) for a memory read at address (A=) 2000H with data (D=) of 55H. NOW LET'S EMULATE!
ENTER: G 0	GO	Breakpoint B (display shows <Break Hardware B>) and displays the status of the registers.
ENTER: H D a RETURN, and then press the space bar several times.	HISTORY	Displays the contents of the real-time trace buffer. Remember this? This action causes the scrolling on the screen to stop and start.
PRESS: ESC key.		Exits the routine and brings the prompt back up on the screen.
ENTER: D 2000	DUMP	Displays memory location 2000H.

ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: B/B OFF	BREAK	Turns breakpoint (B) OFF.
ENTER: H EM	HISTORY	Changes to the End Monitor (EM) trigger mode.
ENTER: H	HISTORY	Displays the status of the real-time trace buffer. Compare with the display below: Clock Counter = 00032059/204889 Storage Mode = EM Storage Size = Full The word "Full" indicates a full buffer (2047 cycles).
		NOW LET'S CONTINUE EMULATION!
ENTER: G	GO	Starts the program again and stops when hardware breakpoint A (display shows <Break Hardware A>) occurs Now look at the address range 1000H-3000H—it should contain data of AA hex. Let's find out!
ENTER: D FF0,L30	DUMP	Dumps a total of 30 bytes in word units.
ENTER: D 2FF0,3030	DUMP	Notice the difference from the previous command.
ENTER: H D 100	HISTORY	Displays the last 100 locations in the real-time trace buffer. The space bar can be used to control the scrolling. Press the ESC key to exit.
ENTER: H EE	HISTORY	Changes to the End Event (EE) trigger mode.

ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: H	HISTORY	Displays the status of the real-time trace buffer. Compare with the display below: Clock Counter = 006405C/409692 Storage Mode = EE Storage Size = Full
ENTER: B/E ON	BREAK	Enables the event (E) breakpoint.
ENTER: G	GO	Starts emulation again and stops when an event break (display shows <Break Event>) occurs.
ENTER: H	HISTORY	Notice the status of the HISTORY command. Compare with the display below: Clock Counter = 000960C2/614594 Storage Mode = EE Storage Size = Full
ENTER: H D 60	HISTORY	Displays the last 60 pointers of the real-time trace buffer. Again, use the space bar to control the scrolling. Press the ESC key to exit.
ENTER: EV	EVENT	Displays the status of the event settings again.
ENTER: D 2000	DUMP	Memory location 2000 should contain 55 hex.
ENTER: G	GO	Starts emulation again and stops when hardware break A (display shows <Break Hardware A>) occurs. The address range 1000-7FFFH should now contain the data value 55 hex.

ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: D FF0,L30	DUMP	Dumps a total of 30 bytes in word units.
ENTER: D 2FF0,L30	DUMP	Notice the difference from the previous command.
ENTER: G	GO	Starts emulation again and stops when a user break occurs.
		THIS IS THE END OF EXAMPLE 1.
		THIS IS THE START OF EXAMPLE 2.
ENTER: R RESET	REGISTER	Resets the registers.
ENTER: DI 0,30	DISASSEMBLE	Checks to see that the program is still around.
ENTER: T A	TRACE	Traces all instructions to be displayed in a continuous manner.
ENTER: G	GO	Starts emulation.
PRESS: The space bar to start and stop the display of the instructions.		
PRESS: The ESC key to exit.		
ENTER: T J	TRACE	Traces all instructions but displays only Jump (JP) instructions.
ENTER: G	GO	Starts emulation.
PRESS: The ESC key to exit.		
ENTER: T/S A	TRACE	Traces instructions by a single-step method (one instruction at a time).

ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: T		ICD displays: <ON> /S All 0000 — FFFF This shows that the trace is active, that the single-step trace feature is active (/S), that all instructions are to be traced and displayed, and that the trace range is 0000 to FFFF (the default).
ENTER: G	GO	Starts emulation.
PRESS: The space bar to control steps.		
PRESS: The ESC key to exit.		THIS IS THE END OF EXAMPLE 2.
		THIS IS THE START OF EXAMPLE 3.
		This example will demonstrate other interesting features of the ICD. In this example, you will use different commands to MOVE, COMPARE, and SEARCH through memory, and also examine and change memory locations. Other commands allow reading and modification of I/O ports.
		NOTE: During this example, the space bar and ESC key may be used to control scrolling and to exit the display as shown in the previous examples.
ENTER: R RESET	REGISTER	Resets the registers again.

ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: F 2000,L10,00	FILL	Fills 16 bytes of memory with 00, starting at address 2000.
ENTER: D 2000	DUMP	Displays the 16 bytes of memory just filled.
ENTER: F 3000,L10,11	FILL	Fills 16 bytes of memory with 11, starting at address 3000.
ENTER: D 3000	DUMP	Displays the 16 bytes of memory just filled.
ENTER: F 4000,L10,22	FILL	Fills 16 bytes of memory with 22, starting at address 4000.
ENTER: D 4000	DUMP	Displays the 16 bytes of memory just filled.
ENTER: CO 2000,L10,3000	COMPARE	Compares memory locations starting at 2000 with those at 3000 and displays the locations which are different.
ENTER: S 4000,L10,22	SEARCH	Searches and displays memory locations which are equal to the searched data.
ENTER: S/D 4000,L10,22	SEARCH	Displays the locations which differ from the searched data.
ENTER: M 2000,L10,3000	MOVE	Moves a range of memory starting at location 2000 to address 3000.
ENTER: D 3000,L10	DUMP	Notice the difference from the previous command.
ENTER: E 4000	EXAMINE	The ICD responds with 4000 22 = .

ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: AA/		Examines and changes location 4000. *NOTE: The Slash (/) character terminates the response to the EXAMINE command.*
ENTER: P 0	PORT	Examines a port address and allows changes to be made. ICD responds with 0078=.
ENTER: A/		Changes the port address. *NOTE: The slash (/) character terminates the response to the PORT command.*
		THIS IS THE END OF EXAMPLE 3.
		This concludes the examples which feature the ICD controlled by a terminal (no target system used). If you have a host computer available, you can now use it (through the ZICE software) to control the ICD. To find out how to connect your ICD to the host computer, see "How To Connect Your ICD To Other Devices" in Section 1.
		NOTE: The following exercise was tested with an IBM PC as the host computer.
		You are now operating in the REMOTE mode, where the ICD is controlled by a host computer.
Execute the ZICE software program.		
PRESS: The RESET switch on the ICD.		The ICD will respond with an identification message and a prompt character (>).

ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: SA TEST.HEX,0,30,0	SAVE	Remember your test program? This just saved the program to your computer's disk. Let's prove it!
ENTER: ZD *.HEX or ZD,H		Displays all the files on the disk which end in .HEX. *NOTE: ZD is the ZICE command to display the directory of ZICE files. Different versions of ZICE may require a different command syntax. See your ZICE documentation for the proper command format used with your particular ZICE version.* Now reload the program (TEST.HEX) back to the ICD but at a different location.
ENTER: L TEST.HEX,1000	LOAD	Downloads TEST.HEX to the ICD starting at address 1000H. The offset is optional.
ENTER: DI 1000,L30	DISASSEMBLE	Displays the program after completing the download.
ENTER: Q	QUIT	Ends the ZICE program and returns to the system DOS. THIS IS THE END OF EXERCISE 1.

Exercise 2: ICD Product Demonstration—Using The ICD With A Target System
System Configuration: Host Computer Control Of The ICD
Operation Mode: REMOTE

ACTION	COMMAND DEMONSTRATED	COMMENT
Execute the ZICE software program.		THIS IS THE START OF EXERCISE 2.
ENTER: I 2	IN-CIRCUIT	Sets the mapping mode to target system memory only.
ENTER: DI	DISASSEMBLE	Disassembles user code.
ENTER: G	GO	Now you're emulating!
PRESS: The MONITOR break switch on top of the ICD to stop emulation.		
ENTER: H D 100	HISTORY	Displays your code that was executed right before the break.
ENTER: I	IN-CIRCUIT	This is the IN-CIRCUIT command. The following information describes the actions of this command. I2: Full target emulation mode (target performs as usual except microprocessor is replaced by ICD).
ENTER: I 0	IN-CIRCUIT	I0: No target emulation mode. (This mode does not require the use of any target resources. Used for software development when no hardware is available to execute code on. ICD depends on internal clock for operation.)

ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: I 1	IN-CIRCUIT	I1: Partial target emulation mode (median between I0 and I2 modes). Allows mapping of emulation memory to overlay portions of target memory. Also allows masking out of certain control pins on the microprocessor.
		THIS IS THE END OF EXAMPLE 1.
		THIS IS THE START OF EXAMPLE 2.
		This example will demonstrate the ability to execute part of memory out of the ICD and part out of the target system. In this example, you will move the target system's ROM into the ICD's memory space and then execute it out of the ICD.
STAY: In the In-circuit mode I1.		Assume the following memory map: 0 to 1FFF is ROM 2000 to EFFF is RAM F0000 to FFFFF is No Memory
ENTER: I 1	IN-CIRCUIT	Sets the ICD to partial emulation mode.
ENTER: MA	MAP	Displays the status of the MAP command. Notice that all the memory space of the Z80 defaults to being internal to the ICD. *NOTE: The resolution of the MAP command is in increments of 1K-byte blocks.*

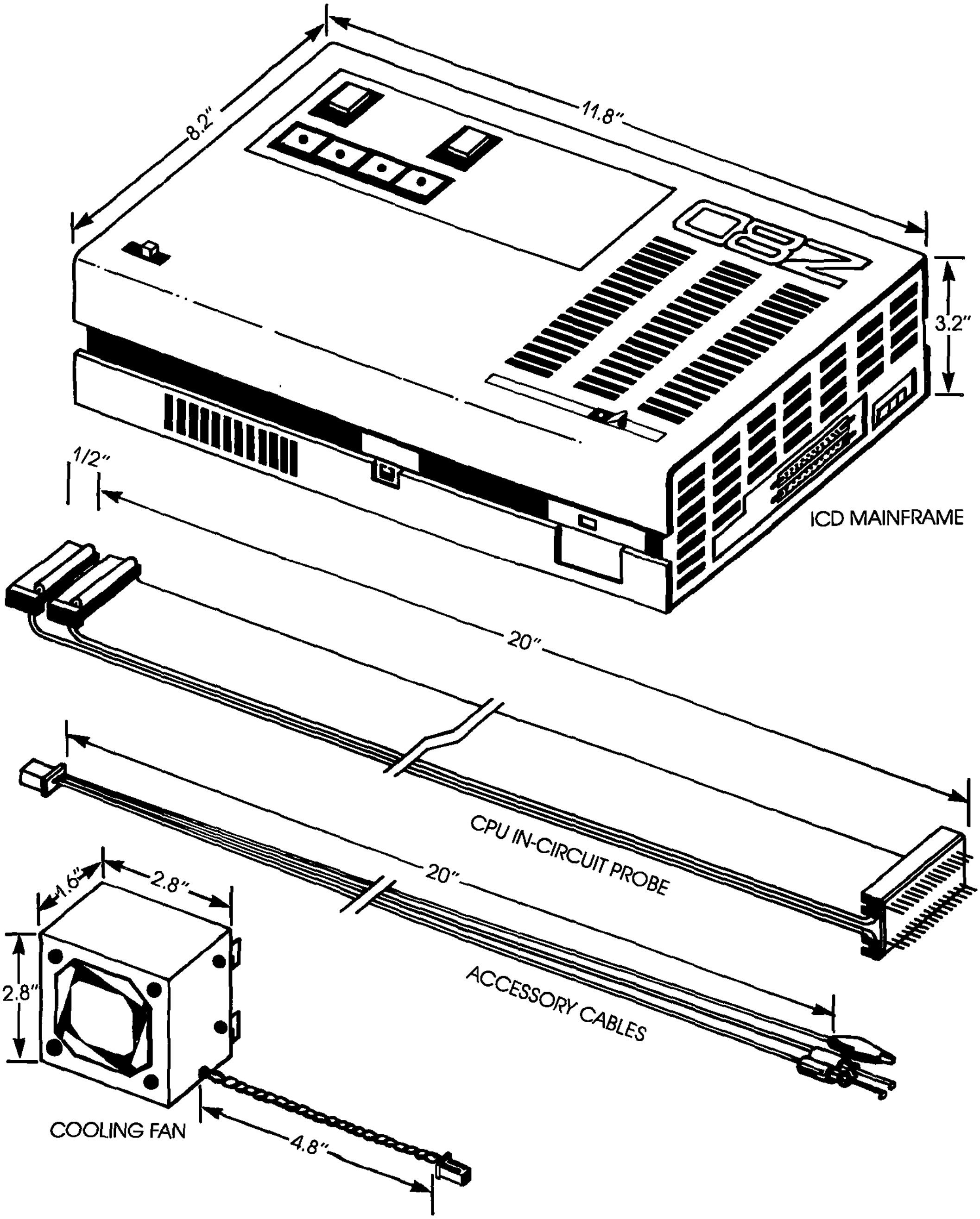
ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: MA 0,1FFF=RO	MAP	Maps to read-only instead of default read/write inside the ICD.
ENTER: MA 2000,EFFF=US	MAP	Maps to user memory.
ENTER: MA F000,FFFF=NO	MAP	Maps to non-existent memory.
ENTER: MA	MAP	Notice the status of the MAP command.
		Now you can move the contents of the target ROM into the ICD. If you are using a host computer, you could now download a program to the ICD which was meant for the same address space as the target ROM.
ENTER: M 0,1FFF,0,UP	MOVE	M means MOVE. 0,1FFF is the target address, the second 0 is the ICD start address, UP means move user (target) memory to program (ICD) memory.
ENTER: DI 0,FFF	DISASSEMBLE	Shows the disassembled code of the target ROM residing inside of emulation memory.
		Now that the target ROM contents are in ICD memory, you can begin emulation.
ENTER: R RESET	REGISTER	Resets the registers.
ENTER: DI	DISASSEMBLE	Shows the beginning of code.

ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: G	GO	Starts emulation.
		With the contents of the target ROM internal to the ICD, the code can now be modified using the in-line assembler, and then checked out for proper execution by setting breakpoints and using the real-time trace buffer, or simply tracing to the display. Once the new code has been checked, it can either be saved to a host computer or sent out the HOST/AUX port to a prom programmer for burning a new prom.
		Now let's examine the PIN command. This command allows you to manipulate certain control pins of the microprocessor in the I1 mode.
ENTER: I 1	IN-CIRCUIT	Sets mapping to I1 mode. *NOTE: The PIN command can only be used in the I1 mode.*
ENTER: PI	PIN	Displays the status of the PIN command. Notice the pins that can be controlled.
ENTER: PI INT=DI	PIN	Disables the interrupt pin to the emulation processor.
ENTER: PI	PIN	Notice the difference from the previous status request.
		THIS IS THE END OF EXERCISE 2.

Appendix C

Technical Specifications

Emulated Processors/ Clock Speed	Z80/2.5 MHz Z80A/4 MHz Z80B/6 MHz Z80H/8 MHz
Memory Size	64K bytes static RAM
Mapping Resolution	1K-byte blocks
Real-time Trace Buffer	2K deep x 32 bits wide
Debugger Commands—built into the ICD	24
Breakpoints	4 hardware, 8 software
Usable I/O Ports	256
Communication Ports	Two RS-232C/20mA current loop/TTL
Baud Rates	14 - 75 to 19,200bps (factory set at 9,600bps)
Physical Dimensions	300mm (11.8in) wide 210mm (8.2in) deep 80mm (3.2in) high
Probe Length	510mm (20in) long
Weight	3.3kg (7.3 lb)
Power Requirements	115VAC/230VAC; 50/60Hz
Power Consumption	40 watts
Operating Temperature	0 C to 45 C
Storage Temperature	– 10 C to 55 C
Humidity	30% to 85%; relative humidity (non-condensing)


11.8"
8.2"
3.2"
Z80
ICD MAINFRAME
1/2"
20"
CPU IN-CIRCUIT PROBE
20"
ACCESSORY CABLES
1.6"
2.8"
2.8"
COOLING FAN
4.8"

ICD Emulation Specifications

Memory Area

Program memory. The entire area of the program memory (64K bytes) is open. This memory is composed of high-speed static RAM.

User memory. The entire 64K-byte memory space is available to the target system.

Mapping. Both the program and user memory can be mapped in 1K-byte blocks. The mapping modes include: user memory, emulation read/write memory, emulation read-only memory, and non-memory.

I/O Port

All 256 ports are open.

Breakpoints

4 hardware and 8 software

Hardware Breakpoints. A,B,C, and Event trigger. All hardware breakpoints can be individually enabled and disabled.

A,B,C Breakpoints. Address 16 bits, BHE. Each bit may be specified 0,1, or "don't care." Status may be specified: OP-code fetch, memory access, memory read, memory write, I/O access, I/O read, I/O write, and instruction execution.

Event Trigger Breakpoint. Address 16 bits, BHE. Each bit may be specified 0,1, or "don't care." Status may be specified as: OP code fetch, memory access, memory read, memory write, I/O access/read/write, and instruction execution. Data: 8 bits. Each bit may be specified 0,1, or "don't care."

External Trigger Breakpoint. 1 channel—TTL level specified at high or low edge of signal.

Software Breakpoints. 8 points: 0 — 7. Any point may be specified as a software breakpoint by using the LDA, A or HALT instruction. All software breakpoints can be individually enabled and disabled. A break is caused in the target system when the CPU reads 7FH as an OP code (which represents an LDA, A instruction.) Execution of a software breakpoint does not effect the registers or flags.

Real-time Trace

Operation: The addresses, data, and status during emulation is stored in the real-time trace buffer.

Trace capacity: 2K deep x 32 bits wide.
Fixed trace data: A0-15, D0-7, MREQ•IORQ, RD•WR, M1.

Trigger functions include: End Monitor, Begin Monitor, End Event, Begin Event, Center Event, and Multiple Event.

Technical Bulletins & Application Notes

Introduction

Things are constantly changing in the microprocessor industry, and **ZAX** wants to help you stay on top of these changes. New products, emulation methods, and applications are always being devised and tested by us in an effort to provide you with the latest and most effective equipment possible. In the same manner, revising your existing equipment keeps it current with the latest ICD designs from **ZAX**.

One of the best ways we have of keeping you up-to-date is by issuing Technical Bulletins and Application Notes.

Technical Bulletins

Technical Bulletins inform you of major changes or revisions to the equipment's hardware or firmware. Usually they are the result of a problem that's recently been solved, or they could be a feature that's been revised to improve the performance of the emulator.

Application Notes

Application Notes are the result of new methods or procedures derived from emulation practices. They may also caution you against doing something a certain way, or they may show you a new way of accomplishing an old task.

Both Technical Bulletins and Application Notes are sent to you as soon as they become available—you should never need to request them. When you receive your documents, insert them into this appendix for easy reference. **That's all there is to it!**

address	A character or group of characters (such as a label, name, or number) that identifies a register, location, or unit where information is stored.
allocate	To assign blocks of data to specified blocks of storage.
argument	An independent variable upon whose value the value of a function depends. Usually the arguments of a function are listed in parentheses (sometimes within brackets) after the function name, if a function name is used.
ASCII	[ask-ee] American Standard Code for Information Interchange. A standard 8-bit information code used for information interchange between equipments of different manufacturers.
assemble	To prepare an object language program from a symbolic language program by substituting machine operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses. *[With **ZAX** ICD-series emulators, this operation is performed using the ASSEMBLE command.]*
assembler	A computer program which operates on symbolic input data to produce machine instructions. An assembler generally translates input symbolic codes into machine instructions—item for item—and produces the same number of instructions or constants which were defined in the input symbolic codes.
baud rate	The number of bits that are transmitted per unit of time (seconds). By definition, a baud is the reciprocal of time—in seconds—occupied by the shortest element of the code being transmitted, e.g., if the duration of the shortest signal element is 20 milliseconds, the modulation rate of the code is 50 bauds (per second). *[**ZAX** ICD-series emulators can transmit up to 19,200 bits per second.]*
bi-directional data bus	A data bus in which digital information is transferred in either direction, thus saving time and providing easy access to stored information.

binary	A numbering system based on 2's rather than 10's in which only the digits 0 and 1 are written.
bit	A binary digit.
branch	To depart from the normal sequence of executing instructions in the computer. (Synonymous with a jump.)
breakpoint	A point in a program as specified by an instruction where the program may be interrupted by some external intervention or by a monitor routine. This program break permits a visual check, print out, or other analysis of the program before resuming with the normal sequence. Used extensively in debugging operations. *[With **ZAX** ICD-series emulators, there are 4 hardware and 8 software breakpoints available using the BREAK command.]*
buffer	A storage device in which data is assembled temporarily during data transfers. It is used to compensate for the differences in the rate of flow of information when transferring information from one device to another.
byte	A sequence of adjacent binary digits operated upon as a unit and usually shorter than a computer word.
C	A high-level programming language designed to optimize run time, size, and efficiency. It was developed as the systems programming language of the UNIX operating system on the PDP 11/70 minicomputer from Digital Equipment Corporation.
CLK	clock
clock	Devices or units which control the timing of bits sent in a data stream, and controls the timing of the sampling of bits received in a data stream. One such clock device is a real-time clock, which measures the past or used time on the same scale as the external events it will be used to describe. Most microprocessor clocks operate in the range of 1 to 12 MHz.

code	A group of symbols that represent data or instructions in a computer. Digital codes may represent numbers, letters of the alphabet, control signals, etc. as a group of separate bits rather than continuous signals. (See microcode.)
compiler	A computer program, more powerful than an assembler, that will convert a higher level language into machine language.
computer control of the ICD	A remote code in which the ICD's input/output is controlled by a host computer using a utility software program designed for that computer. In this mode, the host computer sends commands and receives data via an RS-232C interface.
controller	A device which interfaces a peripheral to a computer in order to relieve the processor of device-controlled responsibilities.
CPU	Central Processing Unit. The module within a computer that is responsible for fetching, decoding, and executing instructions. It contains a main storage unit, arithmetic unit, and special register groups.
cross assembler or cross program	A program run on one computer for the purpose of translating instructions from a different computer.
cross compiling/assembling	A method in which an existing computer can be used to write and debug what will become a microcomputer program. The advantage to cross compiling/assembling is that designers can have access to all of the conventional peripherals so that the object code produced during development can then be loaded into the microcomputer system.
CRT	Cathode Ray Tube. A television tube used to display alphanumeric characters and graphics.
DCE	Data Communications Equipment. Refers to devices used for the transmission of information from one point to another.

debugging	A process of eliminating "bugs" in a system by isolating and correcting all malfunctions and/or mistakes in a piece of equipment or a program of operations.
decimal, binary-coded	Describing a decimal notation in which the individual decimal digits are represented by a pattern of ones and zeros, e.g., in the 8-4-2-1 coded decimal notation, the number 12 is represented as 0001 and 0010 for 1 and 2. This contrasts with a straight binary notation where 12 is represented as 1100.
default value	The choice among exclusive alternatives made by the system when no choice is made by the user.
development system	A system of devices, usually consisting of a diagnostic tool (such as an emulator), a computer, a printer, etc., that can be used together to develop and debug hardware (and software) for a given microprocessor.
development tools	Hardware and software devices that are used to develop and debug programs and/or microprocessor systems.
DIP	Dual In-Line Package. A standard IC package with two rows of pins at 0.1 ″ intervals.
DIP switches	A collection of small switches on a DIP that are used to select options on circuit boards without having to modify the hardware.
disassembly (disassembler)	Refers to a program that translates from machine language to assembly language. Usually used to decipher existing machine language programs by generating symbolic code listings of a program.
don't care	A term applied to an operation which can be changed or interrupted upon receipt of a control signal. The output of the operation is independent of the input.

downloading	A process whereby a file is loaded (using the LOAD command) "down" to the ICD from the host computer system.
DTE	Data Terminal Equipment. Equipment comprised of a data source (transmitter) or data sink (receiver) that provides for the communication control functions (protocol). *[**ZAX** ICD-series emulators use the DCE/DTE select switch to control the communications output for the HOST/AUX port.]*
dump	The process of transferring the contents of memory at a given instant of time onto a screen for viewing, or outputting the memory contents to a hard copy device (such as a printer). *[**ZAX** ICD-series emulators use the DUMP command to display the memory contents in either Hex or ASCII display.]*
duplex	A simultaneous two-way independent transmmission.
dynamic RAM	Memory that requires constant refreshing in order to store memory.
EAROM	Electronically Alterable Read Only Memory. A specialized random access read/write memory with a special slow write cycle and a much faster read cycle.
echo check	An accuracy check of a transmission in which the transmitted information received by an output device is returned to the information source and compared with the original information.
editor	A general-purpose text editing program that allows entry and maintenance of text in a computer system. The original text is entered and held in memory where it can then be changed and corrected by inserting, deleting, or changing lines of text or characters within a line.
EEPROM	Electronically Erasable Programmable Read Only Memory. An EEPROM is a device that can be erased electrically in one second and reprogrammed up to a million times.

EEPROM programmer	A unit that provides a means of programming a single EEPROM or an EEPROM module from a terminal.
EIA-RS-232C	A standard method adopted by the Electronic Industries Association to ensure uniform interface between data communications equipment and data processing terminal equipment. *[All **ZAX** ICD-series emulators use the EIA-RS-232C standard interface.]*
emulation	Techniques using software or microprogramming in which one system is made to behave exactly like another system, i.e., the emulating system executes programs in the native machine-language code of the emulated system.
emulation mode	The mode that the ICD assumes in order to execute instructions.
emulator	An instrument that imitates the control memory of future hardware. Also a device that causes a system (such as the target hardware) to accept certain software programs and routines and appear as if it were the other system.
emulator, stand-alone	An emulator whose execution is not controlled by a control program. It also does not share system resources with other programs, and excludes all other jobs from the computing system when it is being executed. *[All **ZAX** ICD-series emulators can operate as stand-alone or as program-controlled (by a host computer) devices.]*
EPROM	Erasable Programmable Read Only Memory. A specific type of ROM that can be programmed electrically. It can retain data even with the power disconnected but can be erased by exposure to short wavelength ultraviolet light, and may be reprogrammed many times thereafter. Other types of EPROMs may be electrically erased. (See EEPROM.)
field	A set of one or more characters which is treated as a whole.

firmware	Programs that are stored in a physical device (e.g. ROM) that can form part of a system or machine.
FORTRAN	FORmula TRANslator. A high-level language developed by the IBM Corporation, originally conceived for use on scientific problems but now used for many commercial applications as well. It requires the use of a compiler.
full duplex	A mode of communication in which data can be transmitted and received simultaneously.
gate	A device that has one output channel and one or more input channels such that the condition of the output state is determined by the state of the input channel. The NAND, NOR, AND, OR, XOR, and NOT functions are examples of gates.
GND	Ground
half-duplex	A mode of communication in which data may be transmitted in only one direction at a time.
halt	A condition which occurs when an operation in a program stops.
handshaking	A sequence of signals that are required for communication between different systems.
hardware	Physical (electrical, electronic, or mechanical) equipment—as opposed to a computer program—used for processing data. Contrast with software. In the development environment, hardware is the equivalent of your target system.
hex, hexadecimal	Pertaining to a number system with a base of 16. The digits 0 through 9 are used, then A through F, to represent decimal numbers 0 through 15, e.g., "FF" represents "11111111" binary, and "0A" is "00001010" binary.

PROM programmer	A module or external device used to program programmable read-only memories.
PROM programming	The process of altering PROMs (called "burning"), either by blowing (melting or vaporizing) fusible links in bipolar PROMs or by storing a charge on the floating gates of UVEPROMs.
RAM	Random Access Memory. This type of memory is random because it provides access to any storage location point in the memory by means of horizontal and vertical coordinates. Information can then be "written" in or "read" out very quickly.
read-only (RO)	Refers to a process where information can be read from memory only.
read-write (RW)	Refers to a process where information can be read from and written into memory.
real time	Pertains to the actual time during which a physical process transpires. In emulation, real-time operation is very important because of the necessity for the emulator to maintain a "transparent" condition with regard to the device being emulated. *[All **ZAX** emulators are capable of real-time emulation with no wait-states introduced for accessing memory.]*
register	A memory device capable of containing one or more computer bits or words to facilitate arithmetical, logical, or transferral operations.
ROM	Read Only Memory. A special memory that can be read into but not written into.
RS-232C	See EIA-RS-232C.
semantics	The relationship between symbols and their intended meanings independent of their interpretation.

journaling	Refers to a process where all information generated in an emulation session on a host computer is output to a storage file. The entire session can then be reviewed, line for line, just as it was initially entered.
kilobaud	Refers to the number of one thousand bits per second. *[ZAX ICD-series emulators are capable of transmitting at speeds to 19.2 kilobauds.]*
linking loader	A loader used to link compiled/assembled programs, routines, and subroutines, and turn the results into operations.
loader	A program required on practically all systems that load the user's program along with system routines into the central processor for execution. Loaders transfer the object code from some external medium (tape or disk) into RAM.
logic state analyzer (LSA)	A device that monitors a system or component board and displays the resulting information.
machine cycle	The time interval in which a computer (or similar device) can perform a given number of operations.
machine language	A set of symbols, characters, or signs, and the rules for combining them, that conveys instructions or information to a computer.
macro	Pertains to a specific type of instruction in assembly language that is implemented in machine language by more than one machine-language instruction, e.g., a group of instructions often designed to serve as an additive command or group of commands.
macro assembler	An assembler that is capable of assembling object programs from source programs written in symbolic language.
macro instruction	An instruction which stands for a predefined sequence of other instructions, called the "body" of the macro. Whenever a macro instruction is encountered in program text, it is "expanded," i.e., replaced by its body.

mainframe, main frame	Usually refers to large-scale computers (as opposed to microcomputers, microprocessors, and minicomputers). May also mean the fundamental portion of a computer, i.e., the portion that contains the CPU and controller units within the computer system.
microcode	A set of control functions performed by the instruction decoding and execution logic of a computer system. The microcode defines the instruction set of a specific computer.
mnemonic code	Refers to techniques used to assist human memory. A mnemonic code resembles the original word and is usually easy to remember, e.g., *mpy* for multiply, *acc* for accumulator.
monitor mode	Refers to a process where monitor commands from the ICD are executed. Dump, Fill, Disassemble, and Examine are all examples of commands used in the monitor code.
MOS	Metal-Oxide Semiconductor. A technology used for fabricating high-density ICs. The name refers to the three layers used in forming the gate structure of a field-effect transistor.
NOP, NOOP	No-OPeration. An instruction used to force a delay of one instruction cycle without changing the status flags or the contents of the registers.
object code	The code produced by a compiler or special assembler which can be executed by the processor when it is loaded, as with most microcodes, or it may require a linkage phase prior to loading and execution.
object program library	An organized set of computer programs, routines, or common or specifically designed software, containing various programs or routines, source or object programs, classified for intelligence or retrieval.

operating system	Software that is required to manage the hardware and logical resources of a computer system. Also a part of a software package (program or routine) dedicated to simplifying input/output procedures, sort-merge generators, data-conversion routines, or tests.
operation code	The symbols that designate a basic computer operation to be performed. This can be a combination of bits specifying an absolute machine-language operator, or the symbolic representation of the machine-language operator.
operator	A symbol in programming language that represents an operation to be performed on one or more commands (e.g., "add x").
parameter	A constant or variable in an equation or statement that may be assigned an arbitrary value.
parity bit	A redundant bit added to a group of bits so that an inaccurate retrieval of that group of bits is detected.
Pascal	A language designed to teach programming and the elements of computer science. Based on the language, ALGOL, it emphasizes aspects of structured programming.
peripheral devices	Various kinds of machines or devices that operate in combination with a computer but are not physically part of the computer. Peripheral devices typically display computer data, store data from the computer and return it to the computer on demand, prepare data for human use, or acquire data from a source and convert it to a form usable by a computer.
PIO interface	Abbreviation for Parallel Input-Output interface. PIO interfaces allow the computer to input and output parallel data to and from an external parallel device such as a keyboard or printer. Parallel means that all the data bits output at the same time.
PROM	Programmable Read Only Memory. A ROM that may be altered by the user. Some PROMs can be erased and reprogrammed through special physical processes.

high-level language	Any group of computer languages which use symbols and command statements an operator can read. High-level languages allow a user to write in a familiar notation rather than the machine-code language of a computer. BASIC, FORTRAN, FOCAL, and COBOL are all examples of high-level languages.
host computer or host computer system	The primary or controlling computer in a system operation. A host computer can also be reduced to a simple memory storage facility. *[You can use a host computer to control your ICD's operation, or use it to store data files only.* ***ZAX*** *emulators work with a wide variety of host computers, from PCs to powerful minicomputers.]*
ICE	See in-circuit emulation.
in-circuit emulation	Hardware/software facilities for real-time I/O debugging of chips. With in-circuit emulation, the actual microprocessor is replaced by a connector (usually 40-pin type) whose signals are generated by an emulation program. The emulated microprocessor can be stopped, its registers examined or modified, etc.
in-circuit probe	The connector (typically a cable with connector ends) that plugs into the target system's processor socket at one end, and into the emulator at the other end.
instruction	A coded program step that tells the computer what to do for a single operation in a program.
instruction set	The basic set of instructions that a particular computer can perform.
interface	The physical connection between two systems or two devices. *[****ZAX*** *ICD-series emulators interface to your target system for hardware development and debugging.]*
interrupt	A break in the normal flow of a system or program that occurs in such a way that the flow can be resumed from that point at a later time.

source program	A program coded in something other than machine language that must be translated into machine language before use.
stand-alone	Pertains to a device that requires no other piece of equipment to execute and complete its own operation.
stand-alone system	Usually, a microprocessor development system (MDS) that runs independent of the control of a computer. The MDS may contain a terminal and built-in display facility, which in effect makes it a full microcomputer with debugging capabilities.
statement	An instruction (macro) to the computer or other related device, to perform some sequence of operations.
static RAM	RAM that does not need to be refreshed or receive any further attention as long as power is applied.
step	One instruction in a computer routine.
stop bit	The last element of a character that defines the character space immediately to the left of the most significant character in accumulator storage.
symbolic debugging	Symbolic commands that are used to assist in the debugging procedure. Symbolic refers to codes which express programs in source language, i.e., by referring to storage locations and machine operations by symbolic names and addresses that are independent of their hardware-determined names and addresses.
symbolic trace	A process where addresses in a program trace are replaced with symbols. The symbol conversion process is performed in the host system using the appropriate software program.
syntax	Rules that govern sentence structure in a language, or statement structure in a language such as that of a compiler program.
target system	Refers to the processor under development.

trace	Refers to the operation of the real-time trace buffer (storage facility) and its ability to capture and store a portion of the program memory area.
transparency	The ideal emulation condition in which the operation of the target system is uneffected when the emulator is substituted for the microprocessor. Transparency can be broken down into two categories: functional and electrical. To be functionally transparent, the emulator should make no demands on any part of the target system's resources such as interrupts and memory allocation. To be electrically transparent, the emulator should duplicate as closely as possible the microprocessor's characteristics, such as timing and clock speed.
trigger	Refers to a user-specified reference point (external to the user program) which is used to determine where (and when) to begin and/or end a trace section.
TTL	Transistor Transistor Logic. A family of integrated circuit logic elements with a specific output structure, usually +5-volt "ones" and 0-volt "zeros."
universal emulator	A single emulator that is able to support several different processors.
uploading	A process whereby a file is transferred (using the "SAVE" command) from the ICD to the host computer system.
virtual memory	Refers to a technique that permits users to treat secondary memory (disk) storage as an extension of main memory and thus give the virtual appearance of a larger main memory.
XON/XOFF	Transmitter ON/OFF.
ZICE	Refers to the series of **ZAX**-developed support software programs used to interface different host computer systems to **ZAX** ICD-series emulators.

Acknowledgements

Written by Mark D. Johnson

Editing: Janet MacKenzie

Production: Mella Cathleen

Typesetting: A Typographic Service

Printing: California Lithograph Corporation

Binder Design: George Sakamoto

Special thanks to Jim Schmidt and the ZAX Engineering staff.

Let us know how we're doing! We invite your comments and suggestions by providing a Reader's Comments form at the end of this publication.