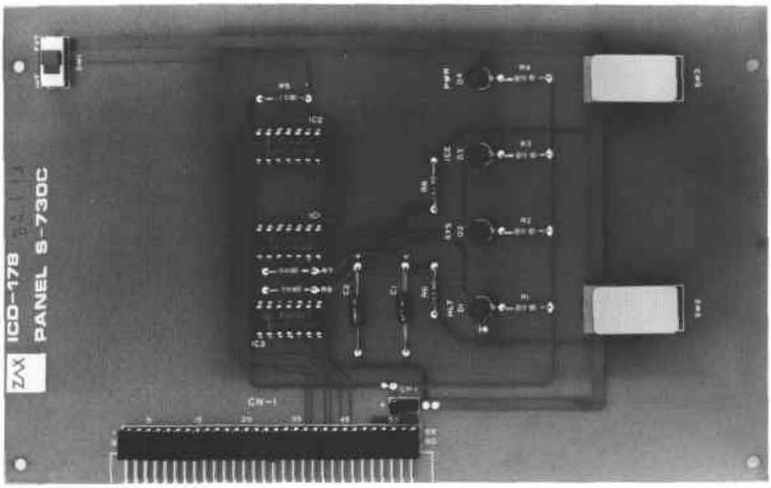
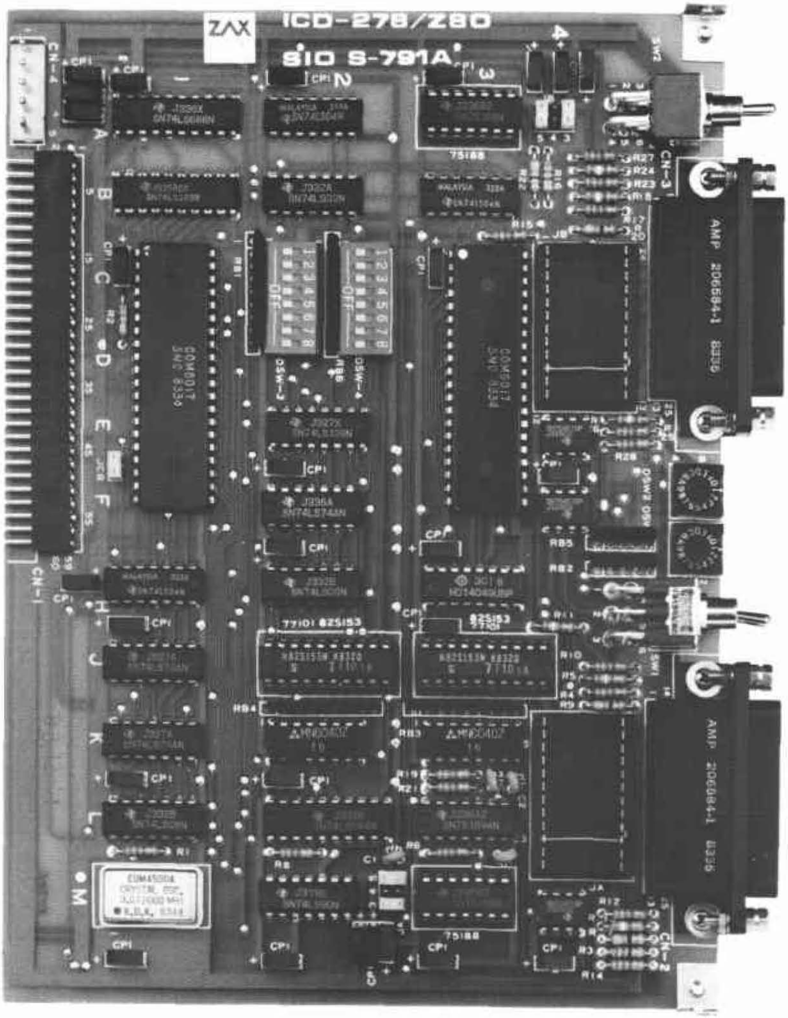
Indicator/Control Module

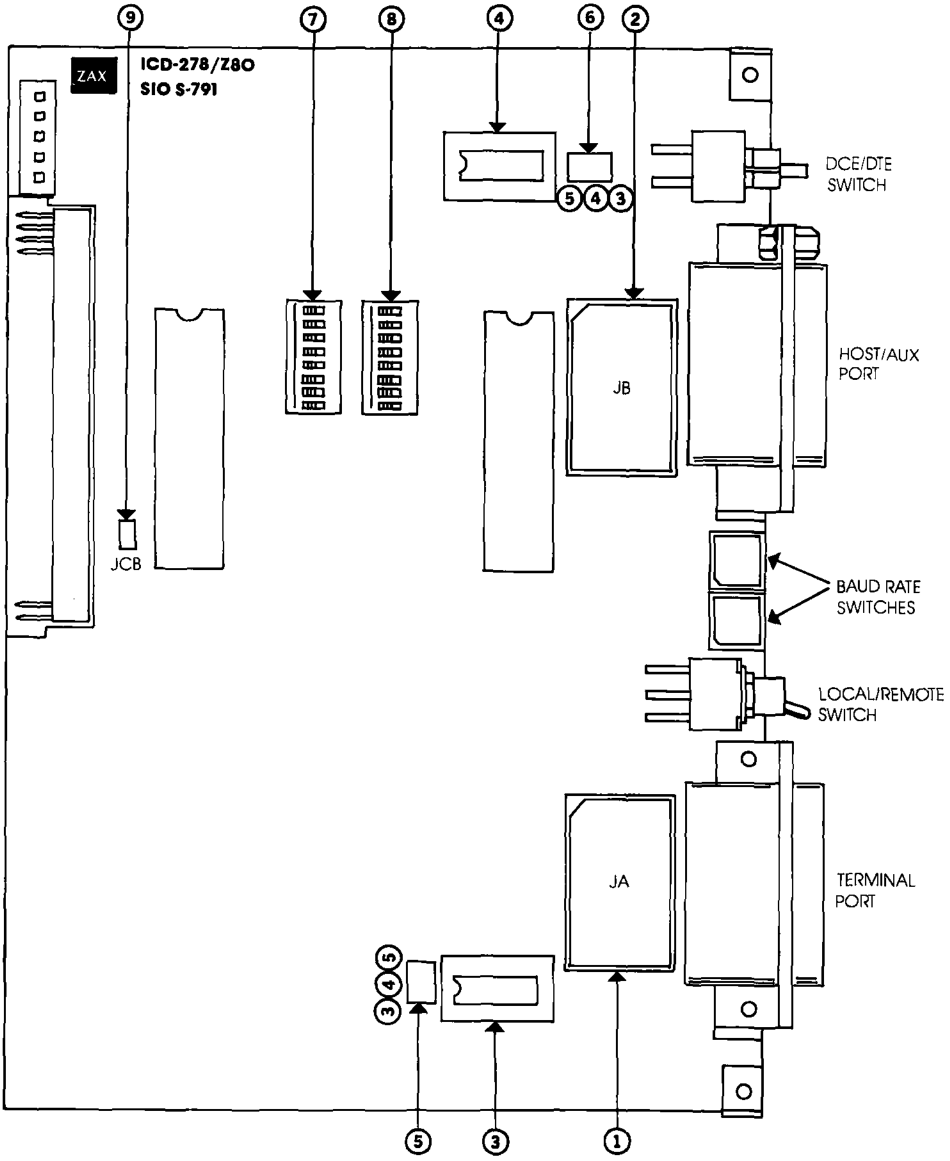
Description

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-4
CN-3
CN-1
CN-2
DSW-3
DSW-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.")

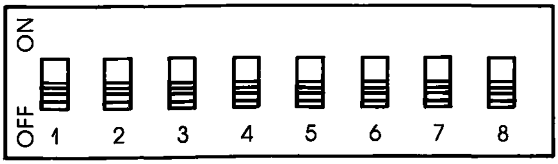
⑧ **DSW4 Transmission Format Switch.** Sets the data format and stop bits for the HOST/AUX port. (See "How To Set The Transmission Format Switches.")

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

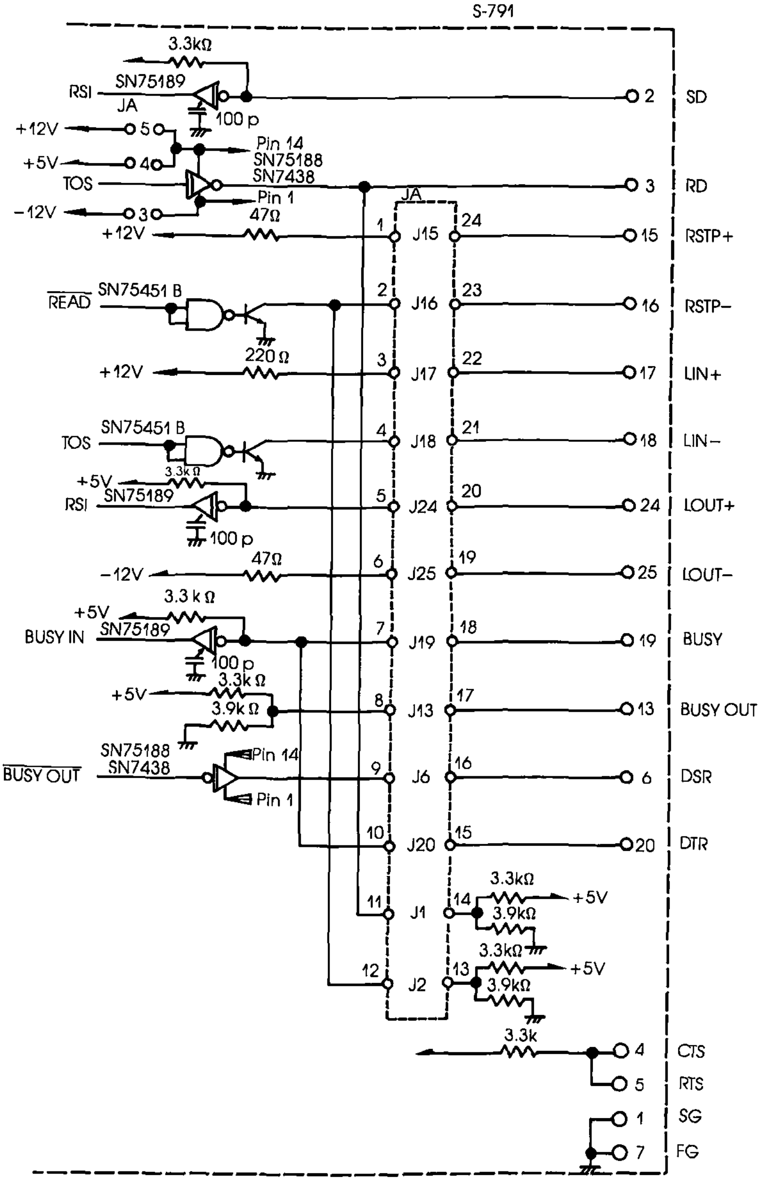
Signals for multiple ICDs can I/O through the HOST/AUX port by setting bits 6 and 7. When this feature is enabled, the External Break, Emulation Qualify, and Event Trigger signals can be monitored by more than one ICD. (I/O level is EIA.)

To activate this feature, set the following bits:

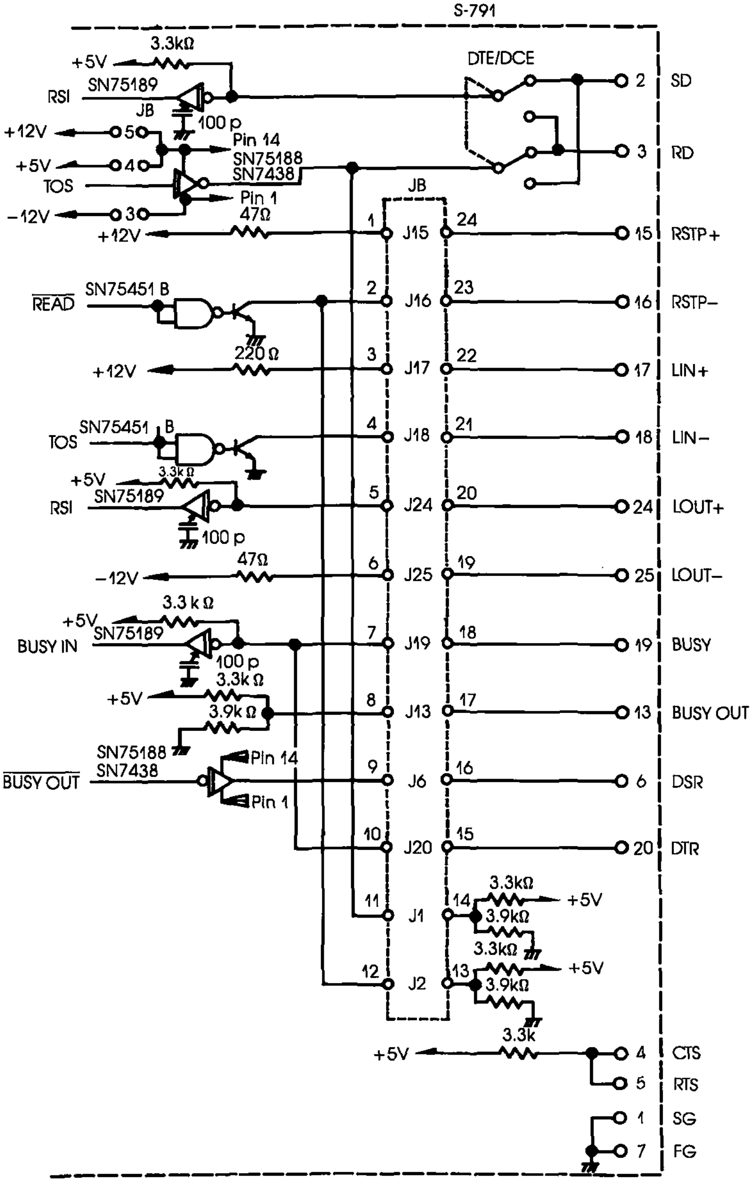
DSW3 bit 6 = ON   DSW4 bits 6 & 7 = ON

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



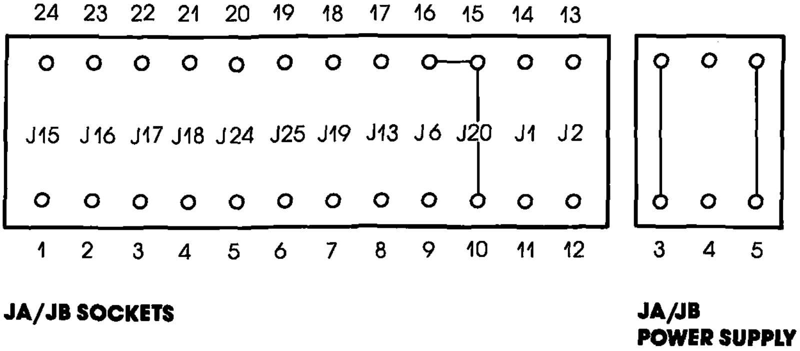
SIO S-791 DIAGRAM (TERMINAL PORT)



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.

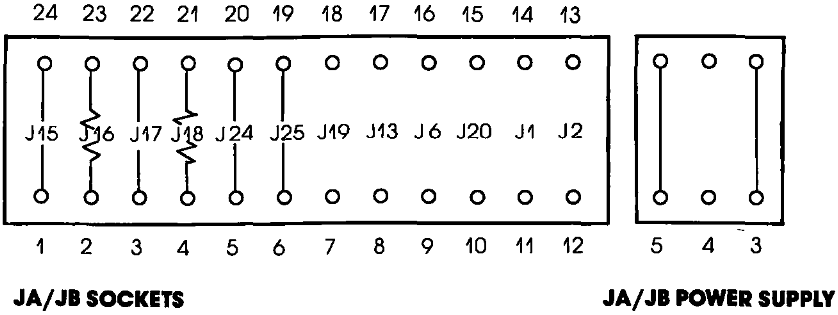
NOTE 2: CTS and RTS signals are looped back (null modem) within the ICD and pulled up to +5V.

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

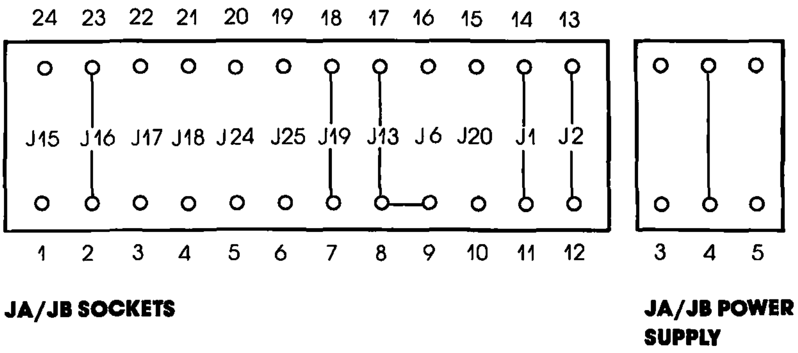
NOTE 1: Pin 25 is the current source pin for current loop input signals pulled down to −12V.

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:

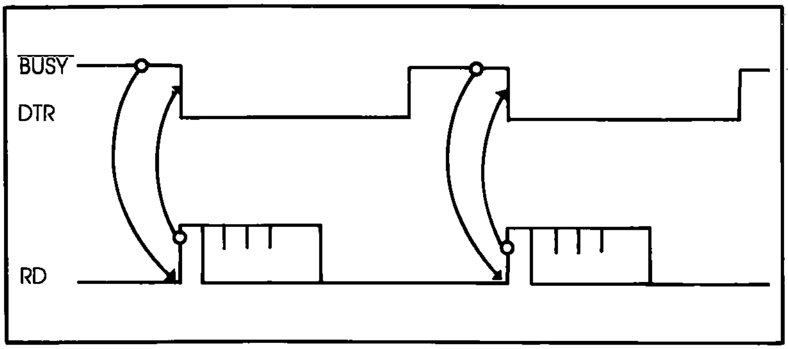
1. The host computer or terminal sends XOFF to the ICD before the reception buffer overruns.
2. When the reception buffer is ready, the host computer or terminal sends XON to the ICD and resumes reception.

The control codes for XON/OFF signals are:

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

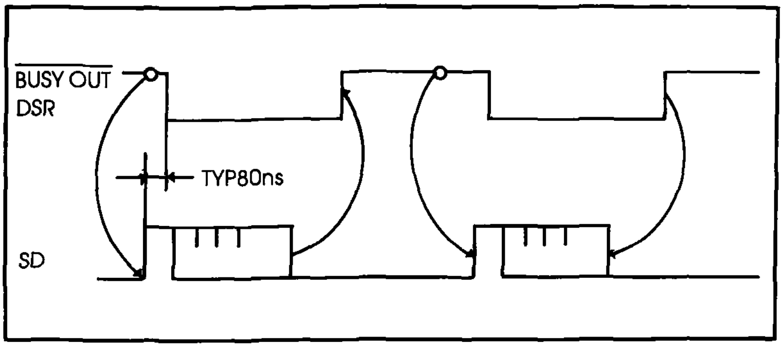
BUSY and DTR Input Signals

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.



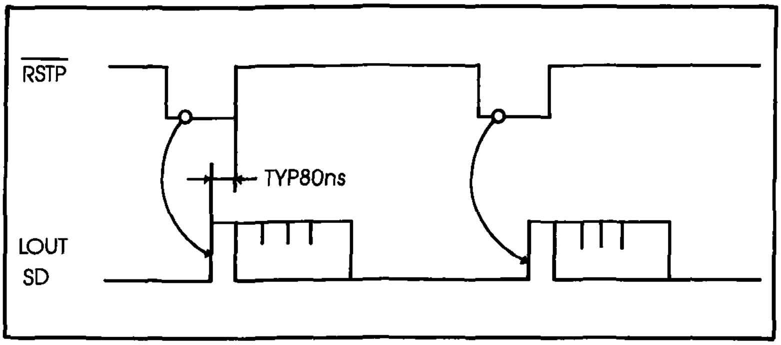
BUSYOUT and DSR Output Signals

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.

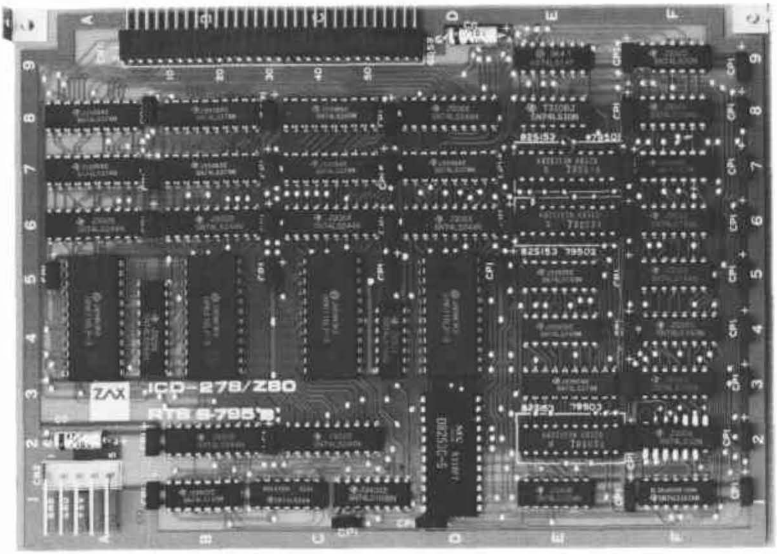


Real-time Trace Module

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.)

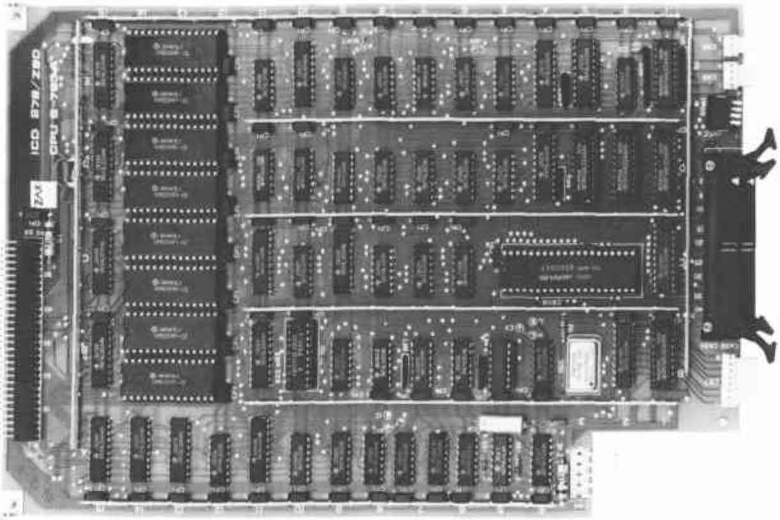


CPU Control Module

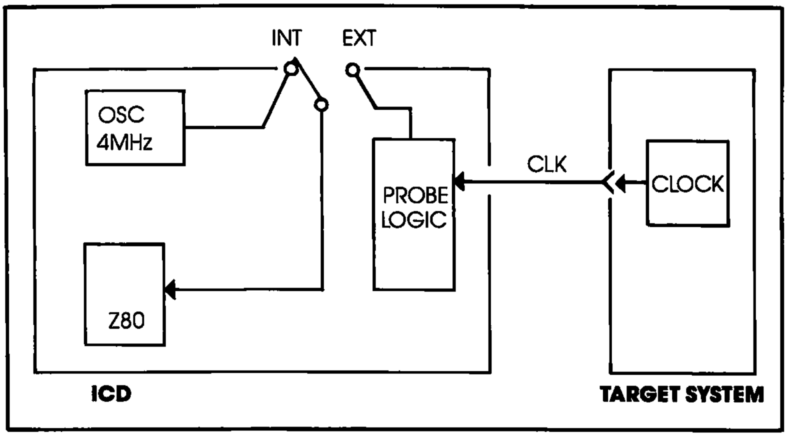
Description

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.



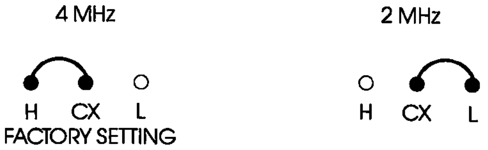
Internal and External Clock



▲ 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.

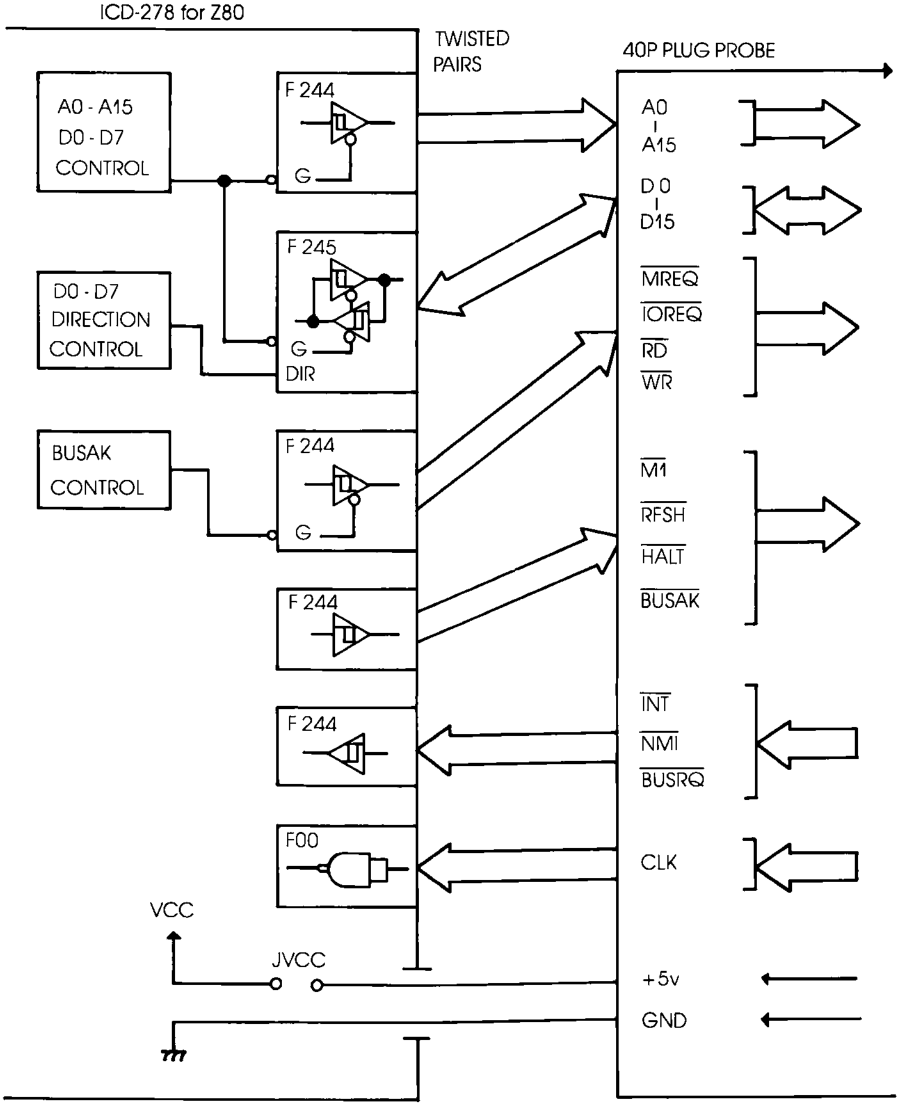


External Clock

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

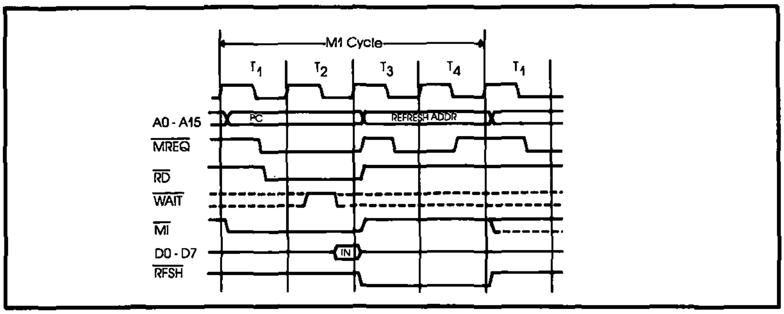
Z80 Pin Functions & Pin Assignments



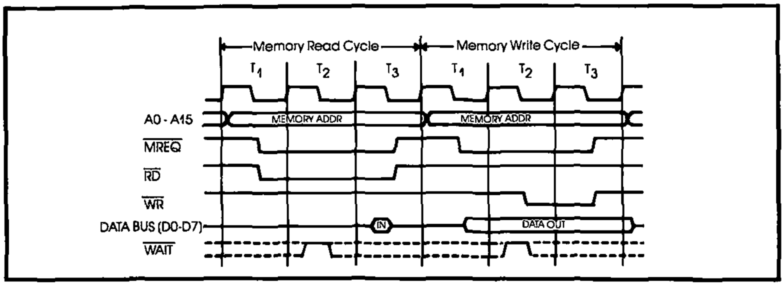
EMULATOR/TARGET SYSTEM INTERFACE

CPU Timing

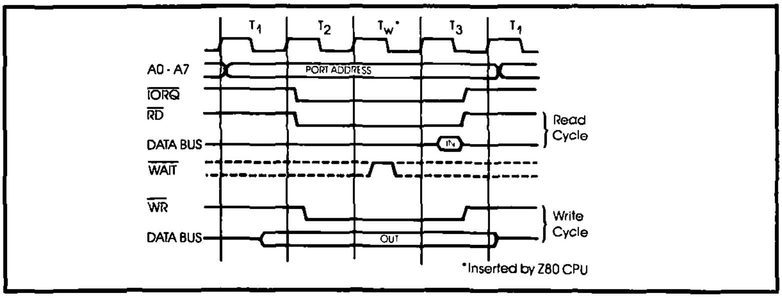
Instruction Opcode Fetch Cycle



Memory Read or Write Cycle

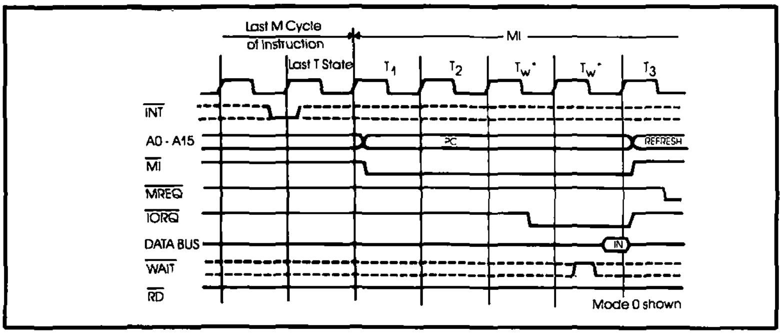


Input or Output Cycles



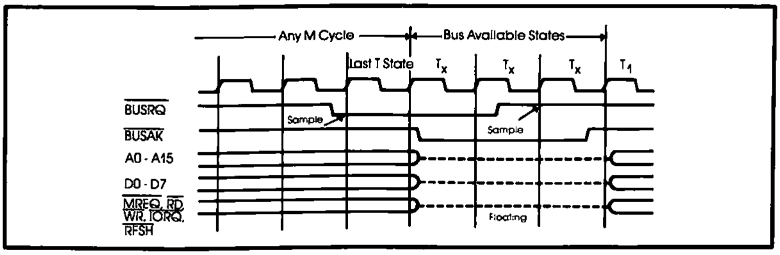
*Inserted by Z80 CPU

Interrupt Request/ Acknowledge Cycle



Mode 0 shown

Bus Request/ Acknowledge Cycle


Any M Cycle
Bus Available States
Last T State
T_x
T_x
T_x
T_1
$\overline{\text{BUSRQ}}$
$\overline{\text{BUSAK}}$
A0 - A15
D0 - D7
$\overline{\text{MREQ}}$, $\overline{\text{RD}}$
$\overline{\text{WR}}$, $\overline{\text{IORQ}}$,
$\overline{\text{RFSH}}$
Sample
Sample
Floating

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.

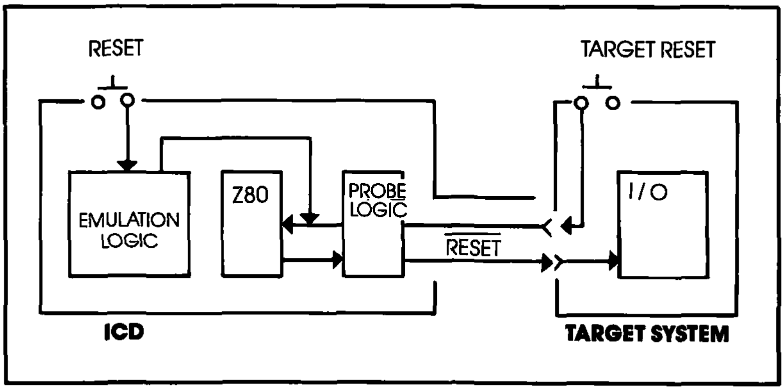
*3. RD signal suppressed when the Emulation Select switch's bit 2=OFF and bit 3=ON.

*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.



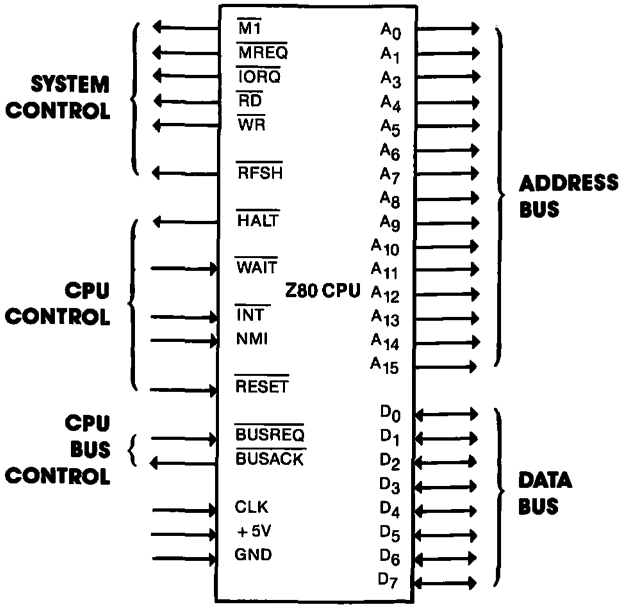
IN-CIRCUIT MODE 0,1,2

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

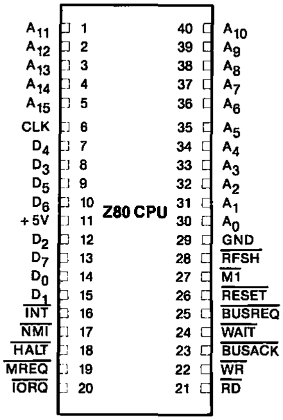
○: Valid △: Conditional valid x: Invalid

* Does not reset hardware in CPU.

Pin Functions



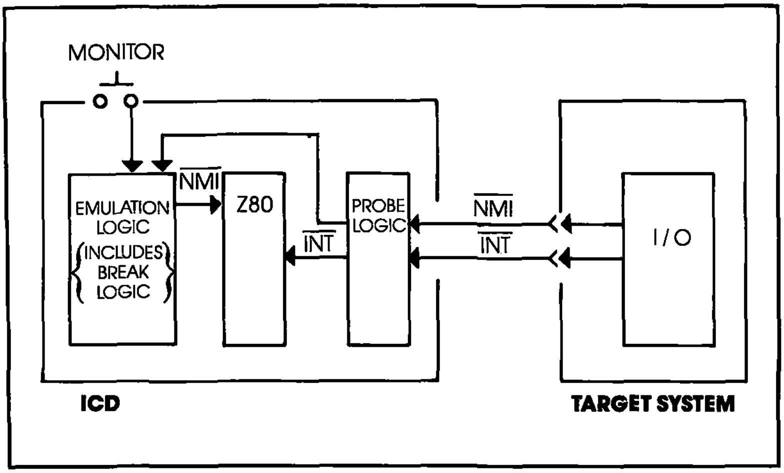
PIN ASSIGNMENTS



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	○
$\overline{\text{NMI}}$	x	x	△*2	○*1	△*2	○
$\overline{\text{INT}}$	x	x	x	○*1	x	○

○: Valid △: Conditional valid x: Invalid

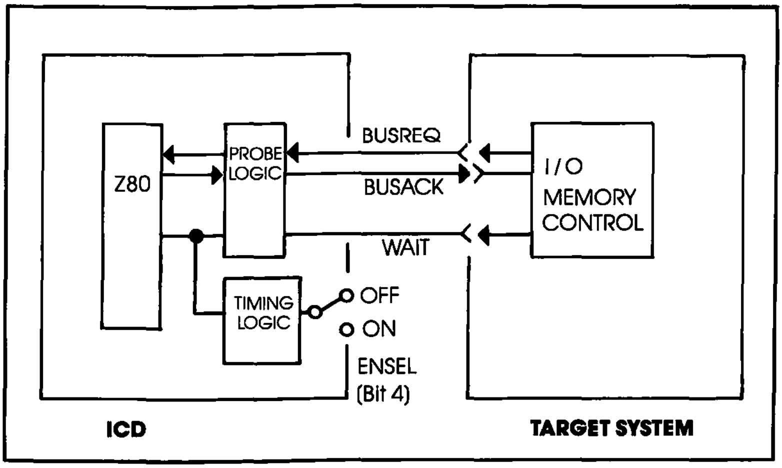
*1 Enable/disable can be set with a PIN command.

*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

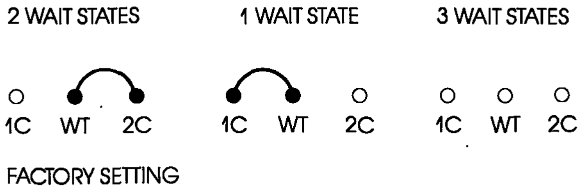
*$\overline{\text{WAIT}}$ signal is valid if the ICD monitor accesses the target system.

▲ 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.

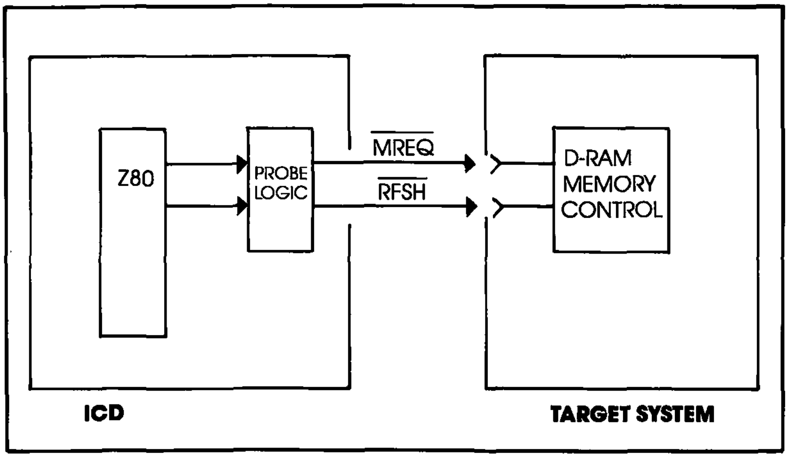
For 3 wait states—remove the jumper from WT and 2C pins.

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
	$t_{W(\phi H)}$	Check Pulse Width, Clock High	180		110		65		65		ns
	$t_{W(\phi L)}$	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
$A_{0\text{-}15}$	$t_{d(AD)}$	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
$D_{0\text{-}7}$	$t_{D(D)}$	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

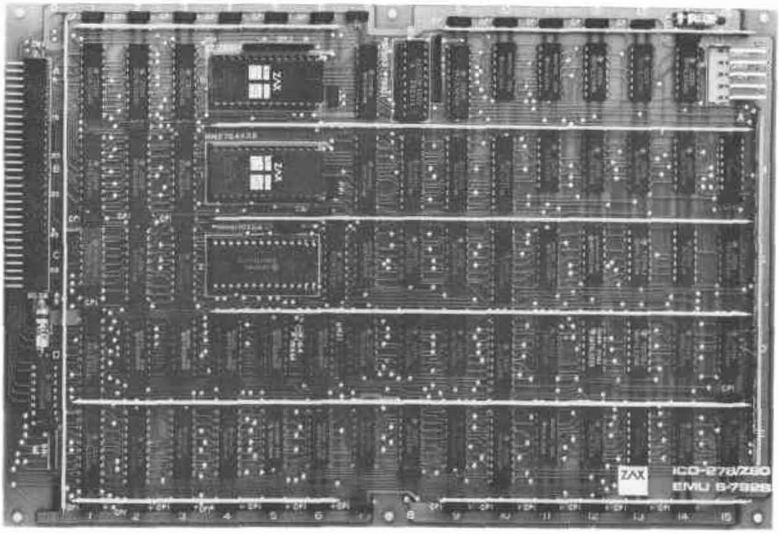
ICD-278 Signal Timing Diagram

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

Emulation Memory (Unit) Module

Description

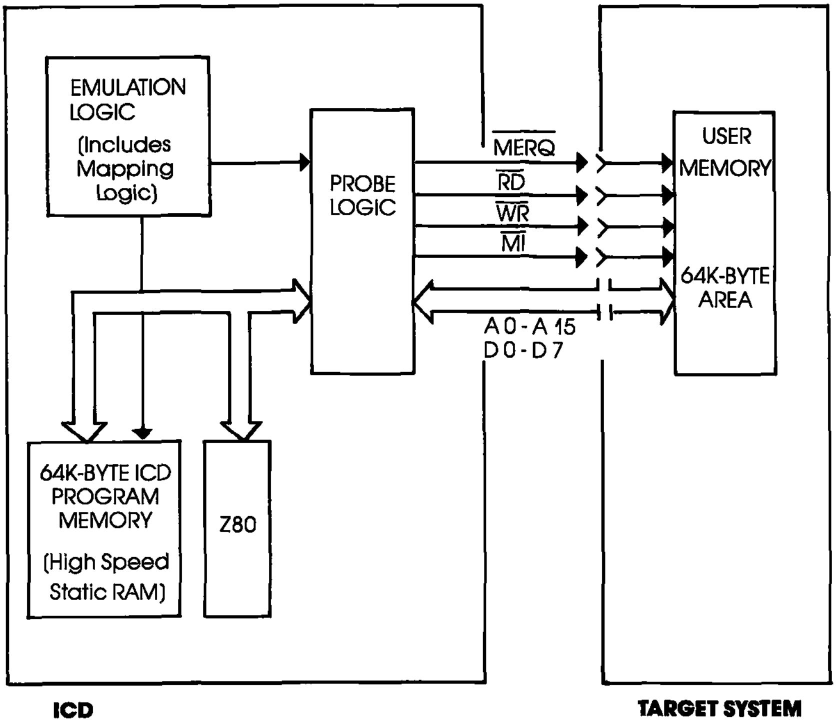
The Emulation Memory (EMU) module (S-792) manages the ICD emulation memory and the mapping of the target system's memory. The EMU module contains no user-serviceable controls or components; all functions are activated using the debugger commands.



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.

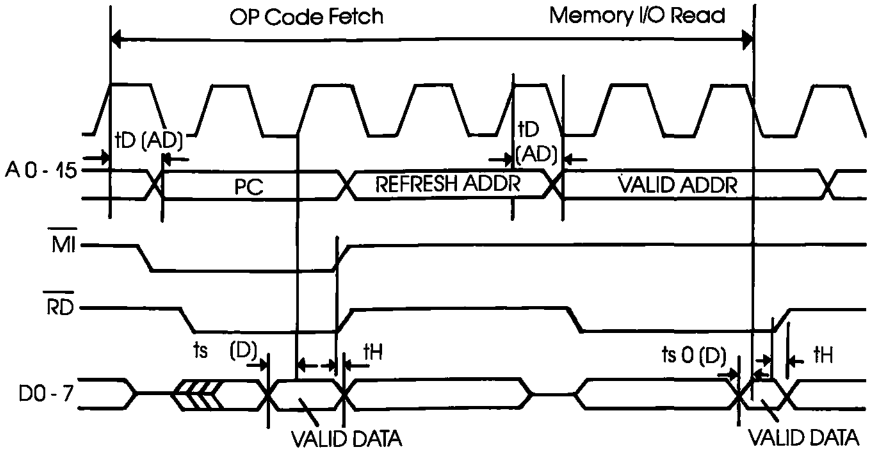


ICD Program Memory Access Time	min. 120 ns

Target System Memory

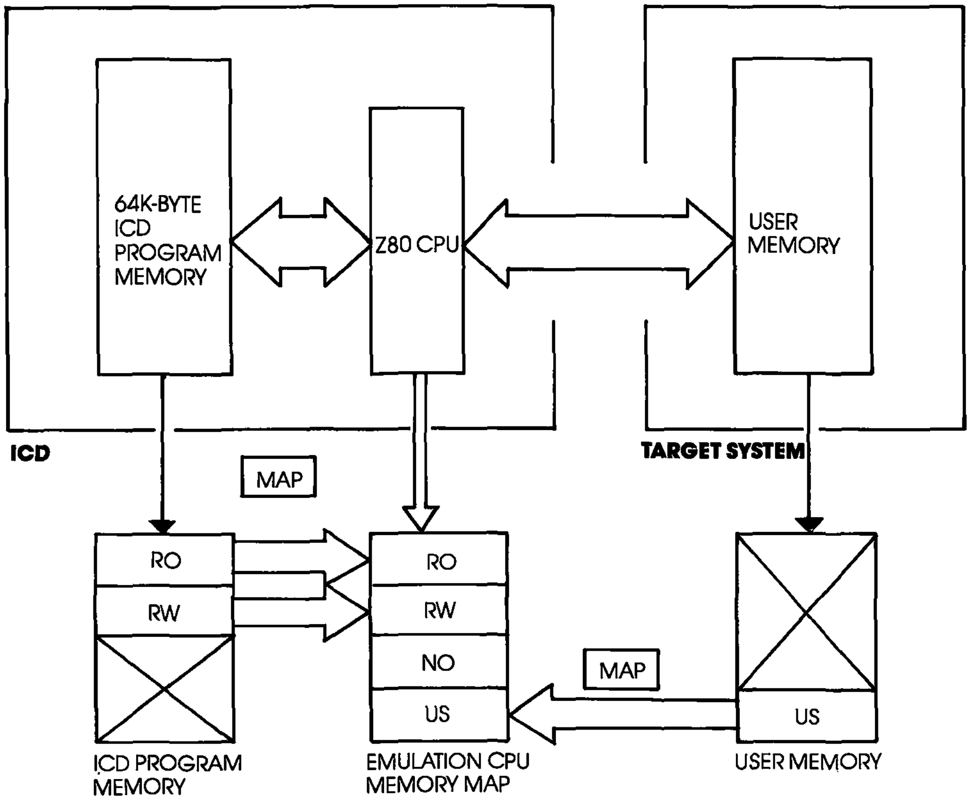
The memory contained in the target system is called target memory or user memory. The ICD can address up to 64K bytes of target memory.

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.)

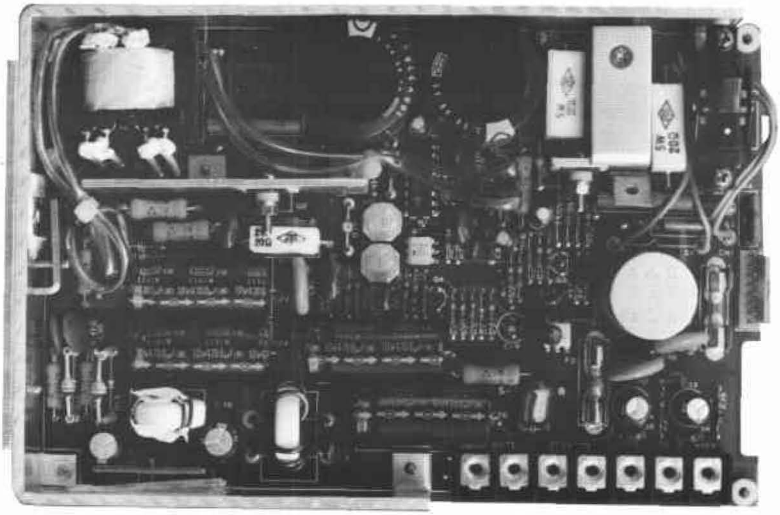


Power Supply Specifications

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.

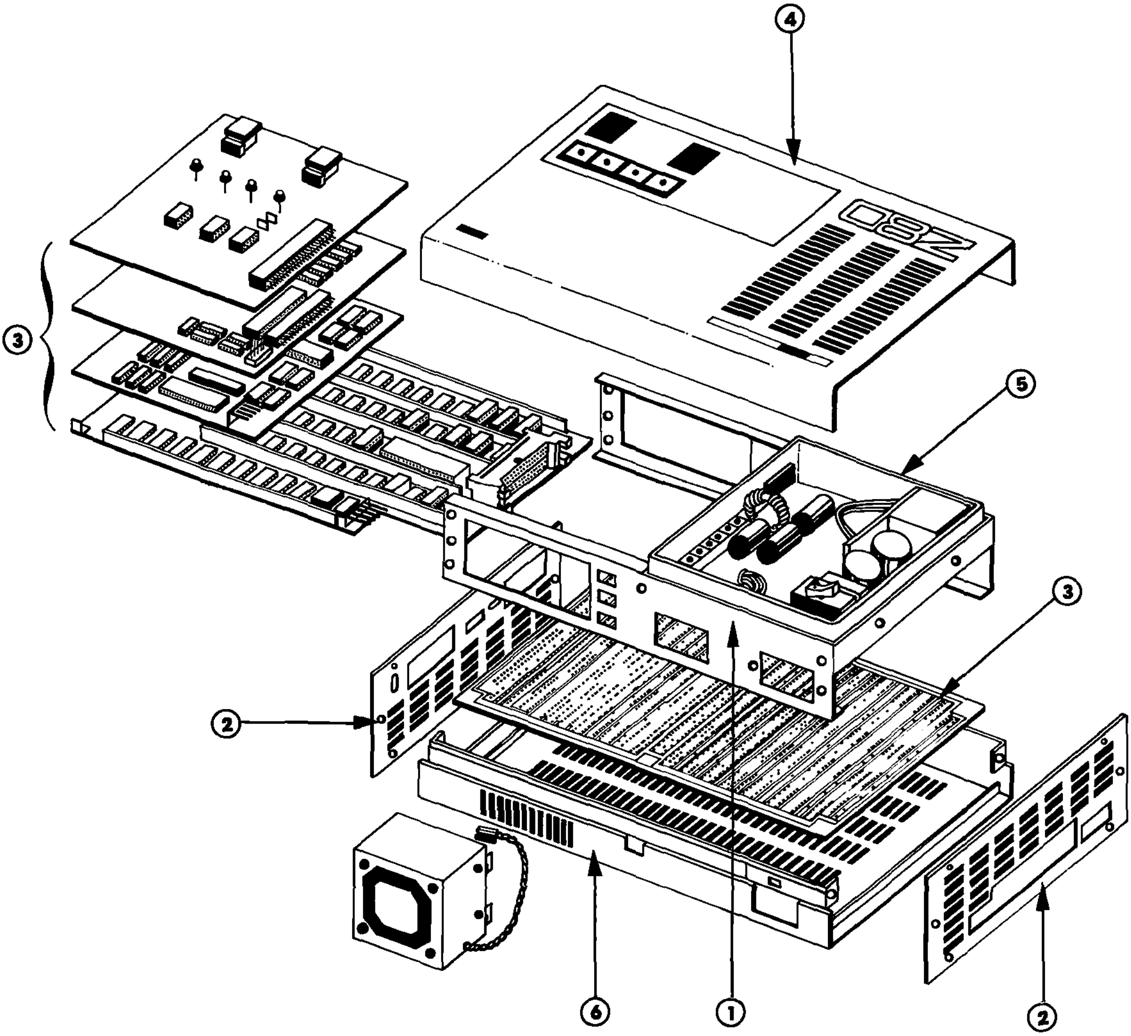
2. Do not adjust, modify, and/or in any way alter the controls or components on any of the three remaining modules (Indicator/Control, Real-time Trace, or Emulation Memory Unit) or the power supply.

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

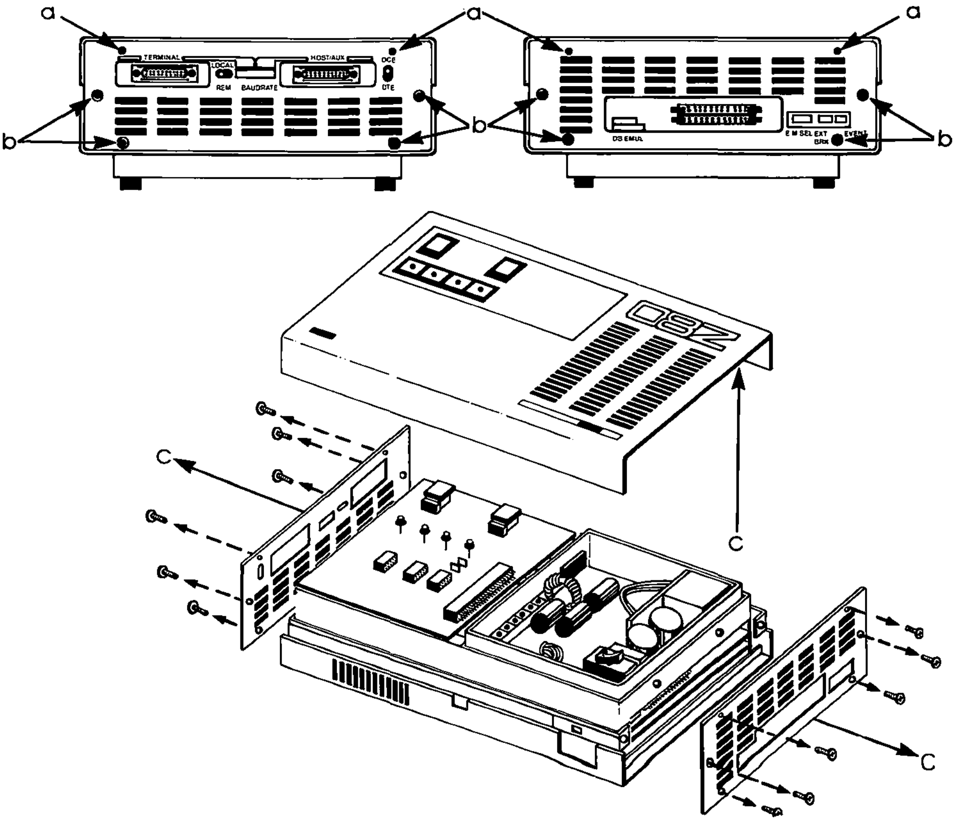
WARNING HAZARDOUS VOLTAGE IS PRESENT WITHIN THE ICD-278. DISCONNECT THE AC POWER PLUG BEFORE BEGINNING ANY INTERNAL WORK ON THE ICD-278.

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.



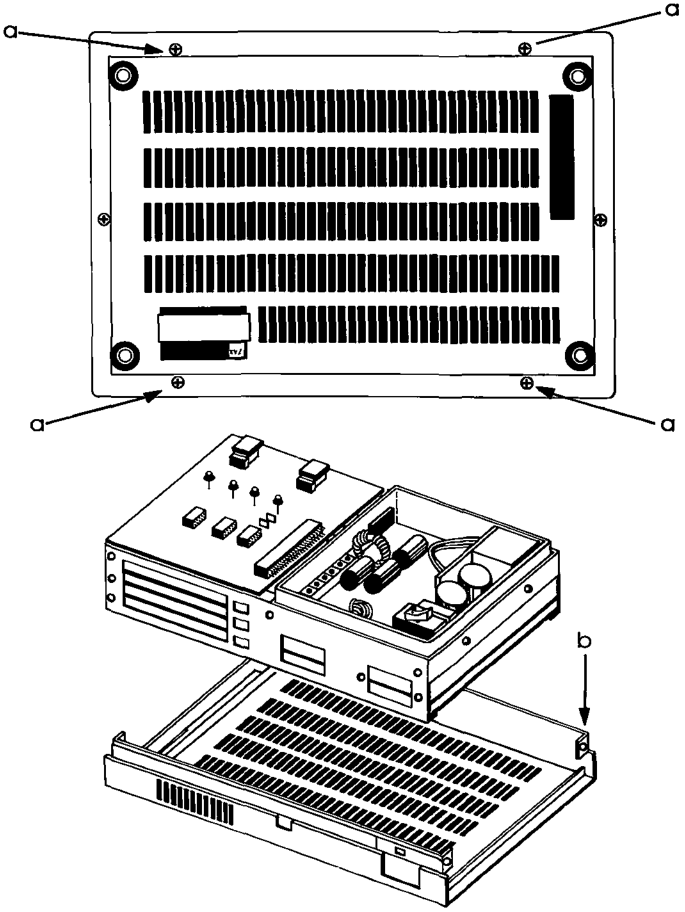
2. **Gently turn over the ICD and remove the bottom cover.**

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.

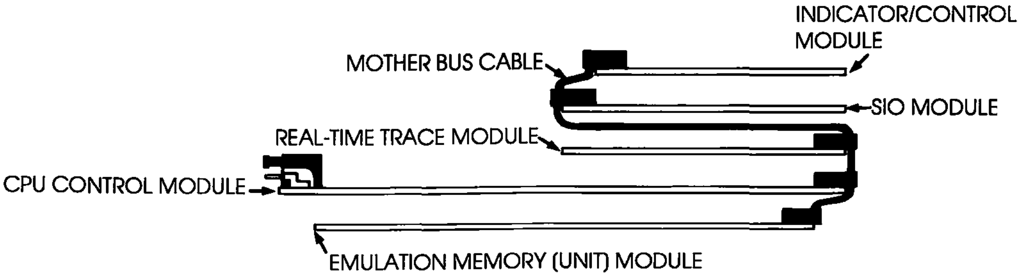


How The Modules Are Connected

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

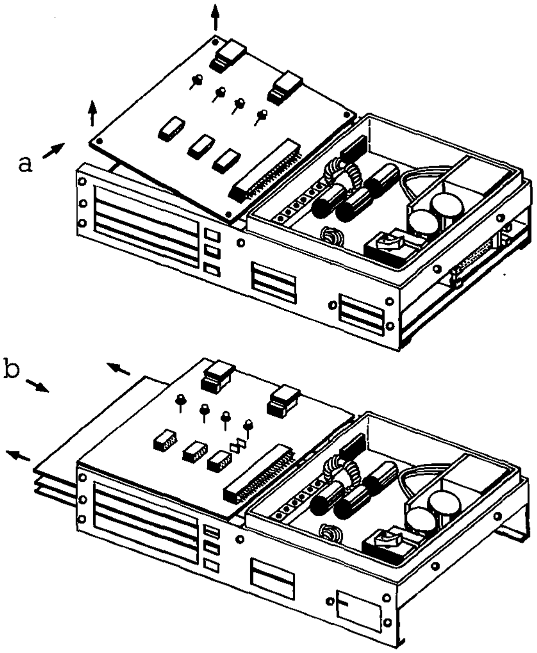
1. Free the power cable by disconnecting the five-pin socket (CN4) from the module.

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

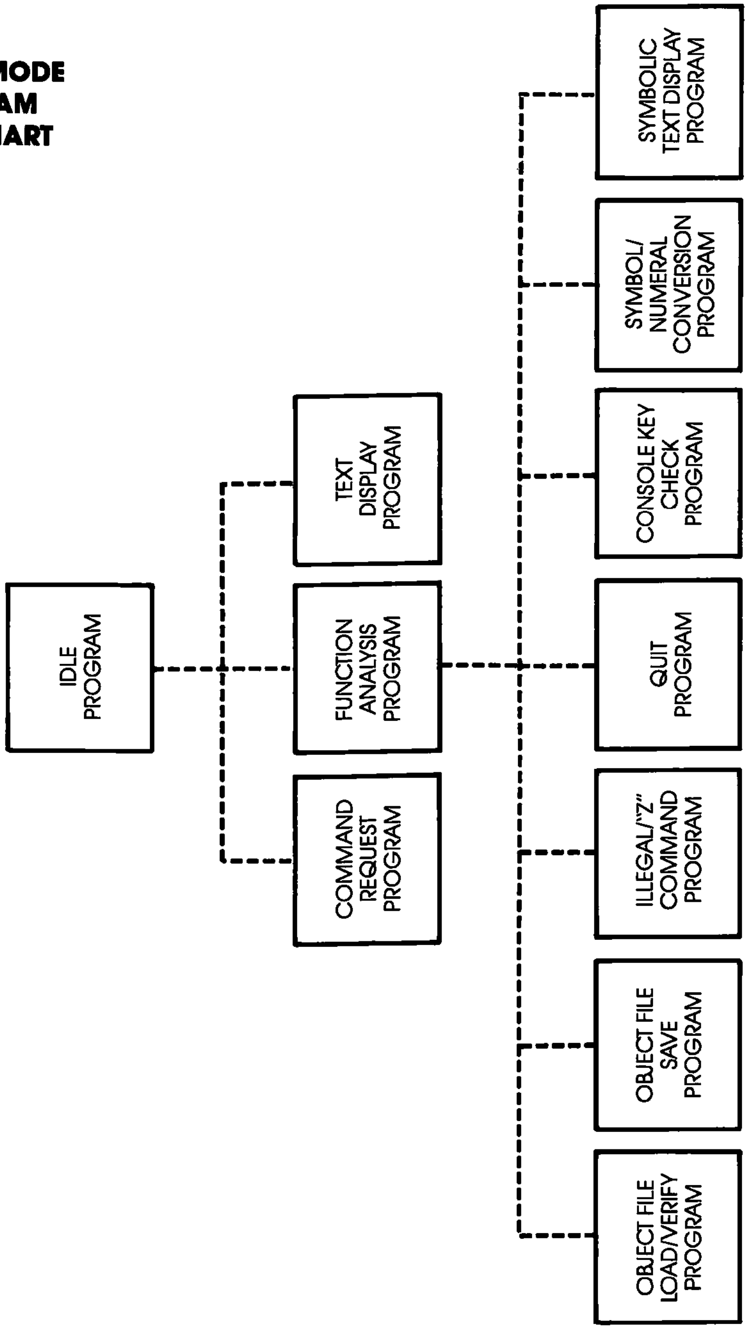
Your ICD can operate in one of two different system configurations with a host computer. In one configuration, a host computer is used to directly control the ICD via ZICE software; this is called the REMOTE mode. In the other configuration, the ICD is under the direct control of a console terminal and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

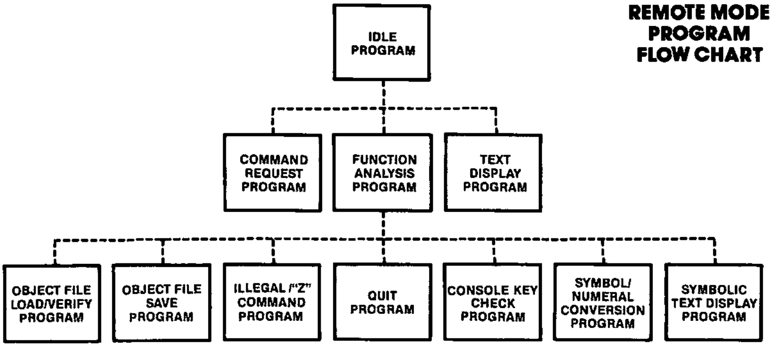
*NOTE: Although this manual is specifically designed for the ICD-278's Z80 CPU, this section can be used with ALL **ZAX** ICD-series emulators that feature "backslash" (\) protocol format. This format is structured as:* \code(text)<CR>. *A Number and Symbol Conversion Code chart which shows the correct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

REMOTE MODE
PROGRAM
FLOW CHART


IDLE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
TEXT DISPLAY PROGRAM
OBJECT FILE LOAD/VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
CONSOLE KEY CHECK PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM

REMOTE MODE: HOST COMPUTER CONTROL OF THE ICD

PROGRAM: IDLE


REMOTE MODE PROGRAM FLOW CHART
IDLE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
TEXT DISPLAY PROGRAM
OBJECT FILE LOAD/VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL /"Z" COMMAND PROGRAM
QUIT PROGRAM
CONSOLE KEY CHECK PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM

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:

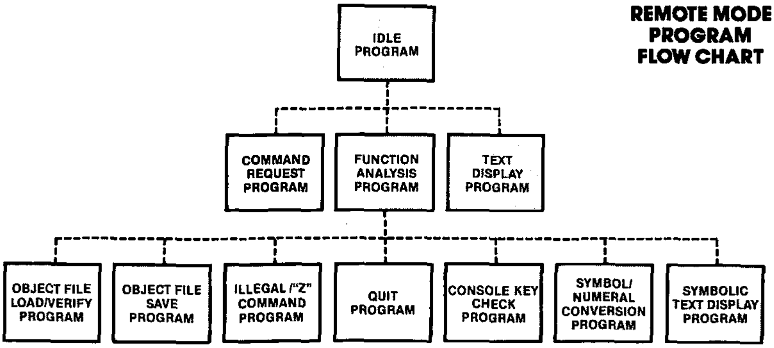
1. The host computer waits for an input from the ICD. (The host system must have an input buffer to hold the input code from the ICD.)

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:

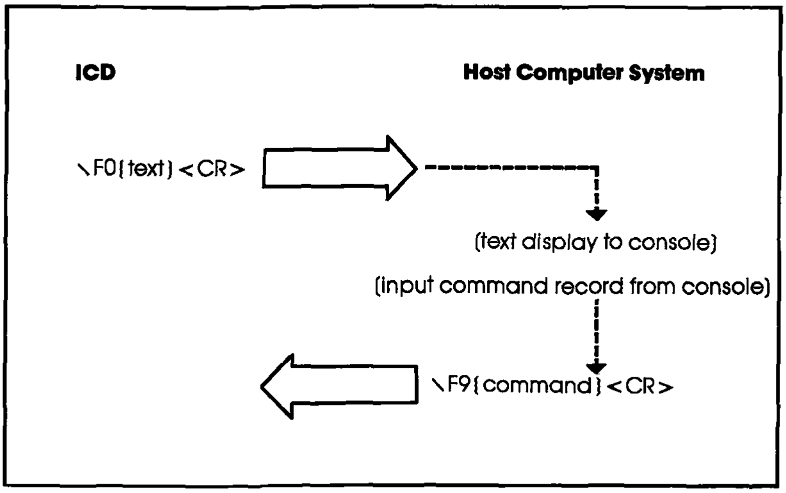
Code Received	Program Executed
\F0<CR>	COMMAND REQUEST
\80{text}<CR>	TEXT DISPLAY
any other	FUNCTION ANALYSIS

PROGRAM: COMMAND REQUEST



REMOTE MODE PROGRAM FLOW CHART

IDLE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
TEXT DISPLAY PROGRAM
OBJECT FILE LOAD/VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL /"Z" COMMAND PROGRAM
QUIT PROGRAM
CONSOLE KEY CHECK PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM



ICD
Host Computer System
\F0{text}<CR>
(text display to console)
(input command record from console)
\F9{command}<CR>

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**

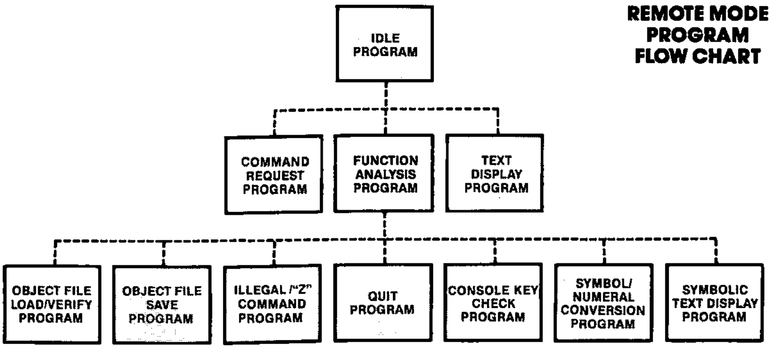
ACTION:

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

IDLE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
TEXT DISPLAY PROGRAM
OBJECT FILE LOAD/VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL /"Z" COMMAND PROGRAM
QUIT PROGRAM
CONSOLE KEY CHECK PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM

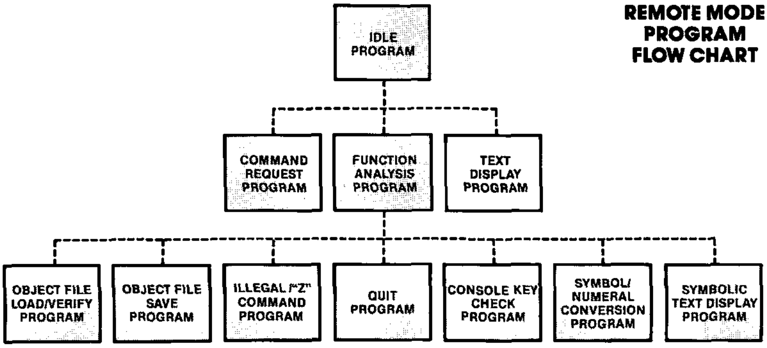
ACTION:

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

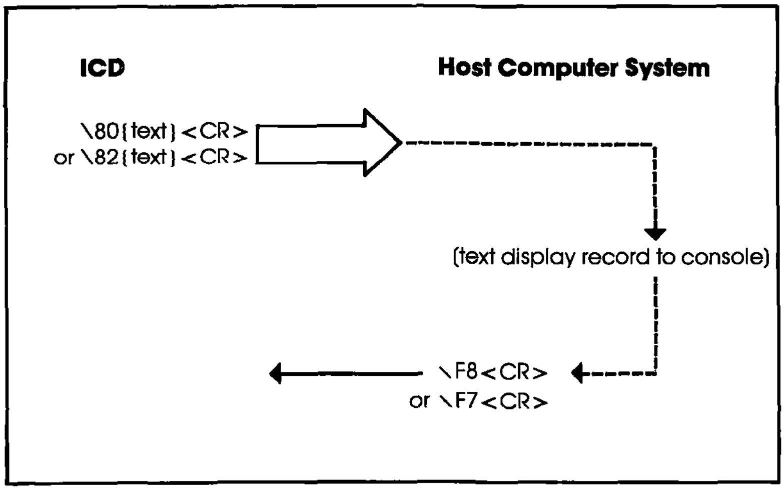
2. The host computer then executes one of the following programs based upon the contents of the input buffer:

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


IDLE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
TEXT DISPLAY PROGRAM
OBJECT FILE LOAD/VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL /"Z" COMMAND PROGRAM
QUIT PROGRAM
CONSOLE KEY CHECK PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM

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:

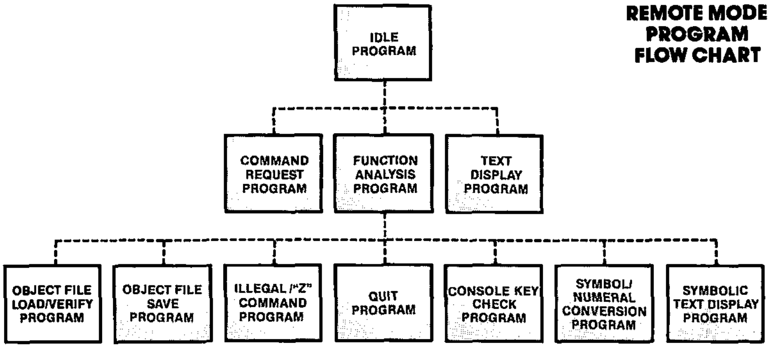
1. The TEXT DISPLAY program is requested when the ICD sends \80{text}<CR> to the host computer.
2. The host computer displays the text record on the console screen.
3. The host computer then checks the console input status and executes one of the following:

   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**


IDLE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
TEXT DISPLAY PROGRAM
OBJECT FILE LOAD/VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL /"Z" COMMAND PROGRAM
QUIT PROGRAM
CONSOLE KEY CHECK PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM

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.

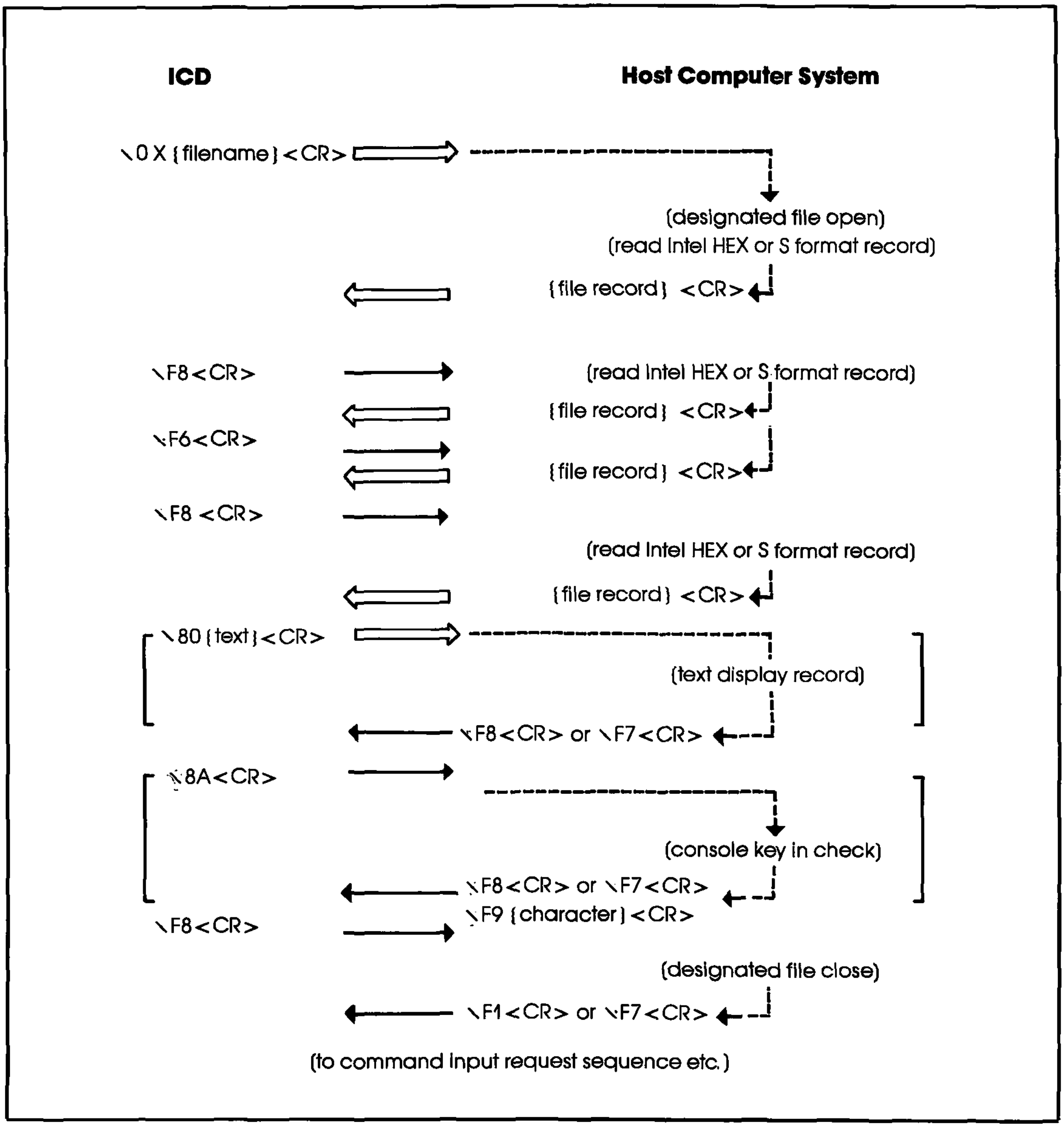
HOST {record}<CR>

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.

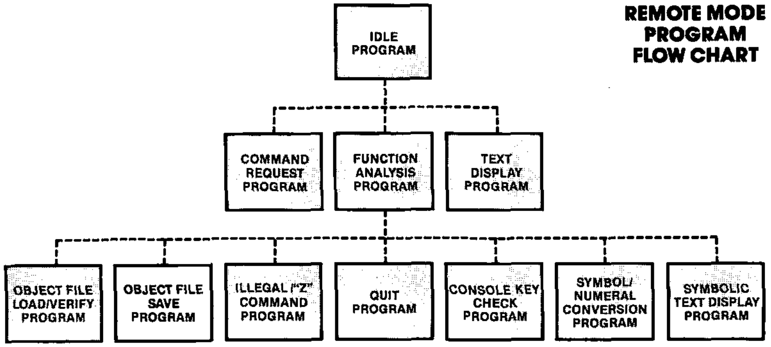
   If the code is \F6<CR>, the host computer waits for \F8<CR> after re-transmitting the Intel Hex or S format record to the ICD.

   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
IDLE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
TEXT DISPLAY PROGRAM
OBJECT FILE LOAD/VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL "Z" COMMAND PROGRAM
QUIT PROGRAM
CONSOLE KEY CHECK PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM

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**

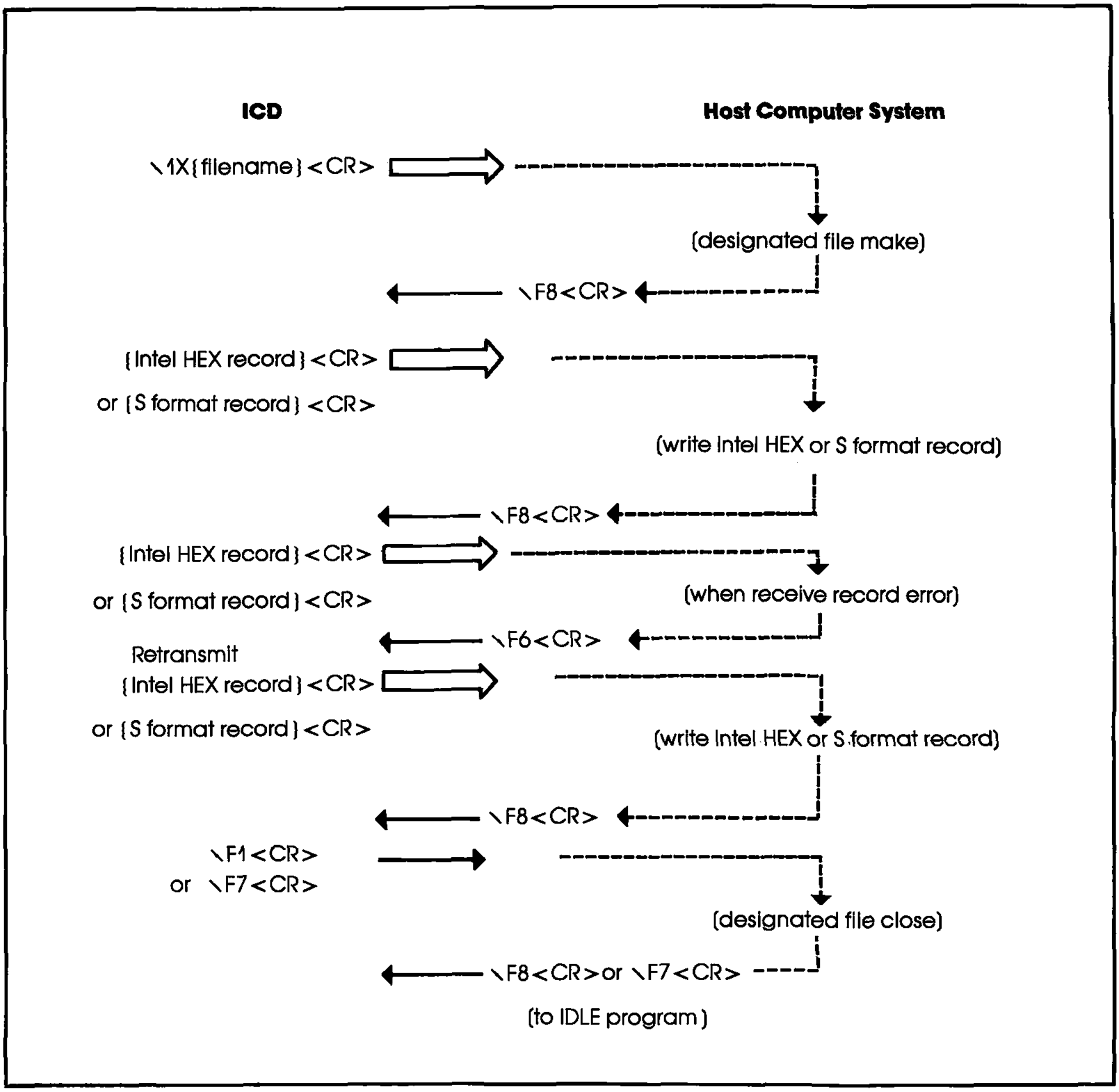
ACTION:

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.

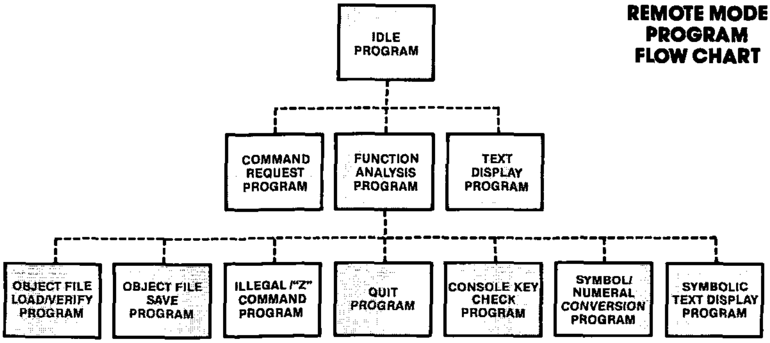
   If the file does not open, the host computer sends \F7<CR> to the ICD and returns to the IDLE program. If the file opens, the host computer sends \F8<CR> to the ICD.

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.

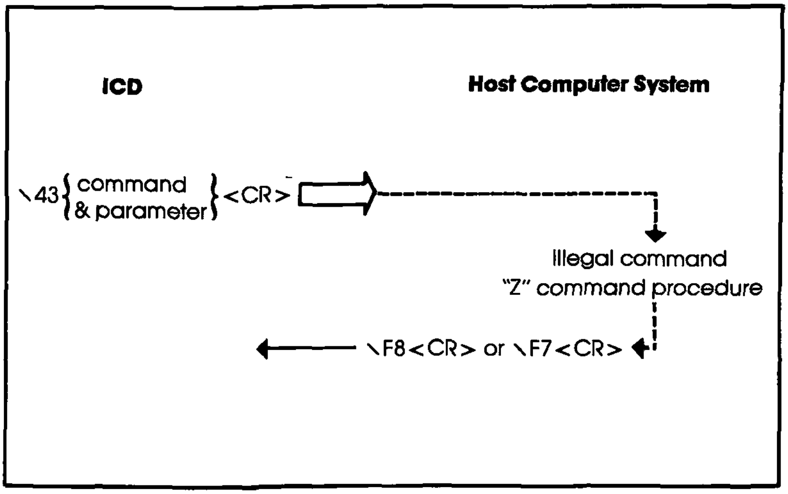
   If an error occurs in a check sum, the host computer waits for a retransmission of the Intel Hex or S format record from the ICD after sending \F6<CR>. The host computer then sends \F8<CR> to the ICD if no error occurs during the file write.


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


IDLE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
TEXT DISPLAY PROGRAM
OBJECT FILE LOAD/VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL /"Z" COMMAND PROGRAM
QUIT PROGRAM
CONSOLE KEY CHECK PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM

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.

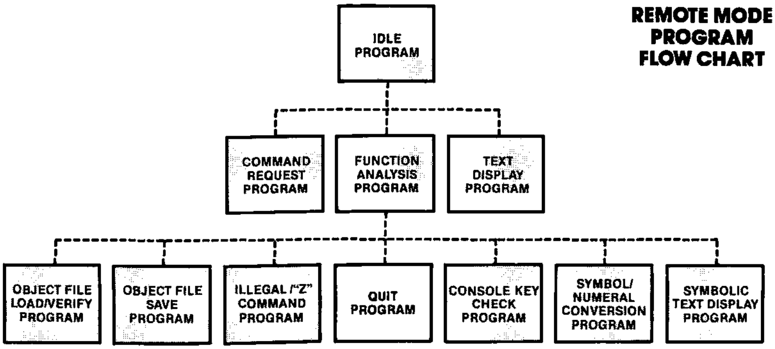
2. The host computer performs the specified "Z" command and then acts on the following:

   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.

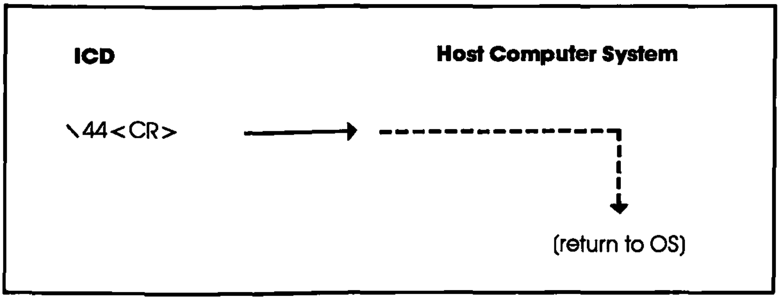
   If no error is detected, the host computer sends \F8<CR> to the ICD and then returns to the IDLE program.

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

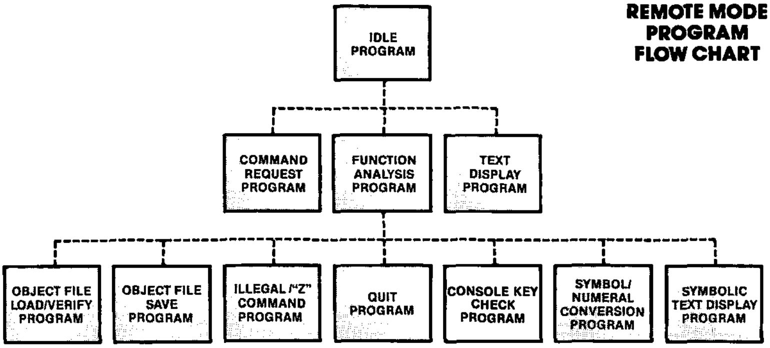

IDLE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
TEXT DISPLAY PROGRAM
OBJECT FILE LOAD/VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL "Z" COMMAND PROGRAM
QUIT PROGRAM
CONSOLE KEY CHECK PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM

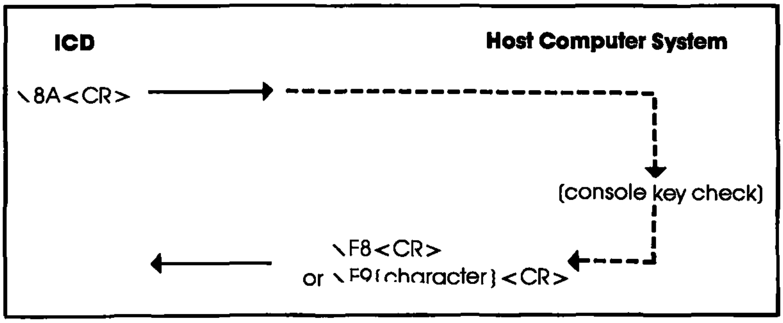
REMOTE MODE PROGRAM FLOW CHART



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

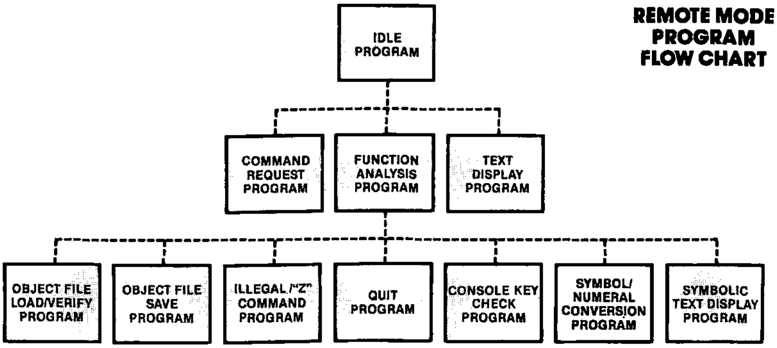
PROGRAM: CONSOLE KEY CHECK

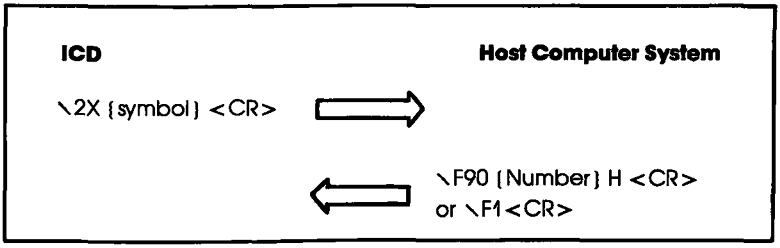

REMOTE MODE
PROGRAM
FLOW CHART
IDLE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
TEXT DISPLAY PROGRAM
OBJECT FILE LOAD/VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL /"Z" COMMAND PROGRAM
QUIT PROGRAM
CONSOLE KEY CHECK PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM



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


REMOTE MODE PROGRAM FLOW CHART
IDLE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
TEXT DISPLAY PROGRAM
OBJECT FILE LOAD/VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL "Z" COMMAND PROGRAM
QUIT PROGRAM
CONSOLE KEY CHECK PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM

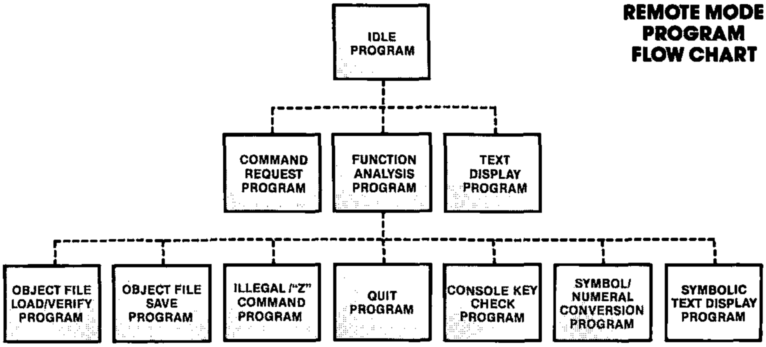

ICD **Host Computer System**

\2X {symbol} <CR> ⇨

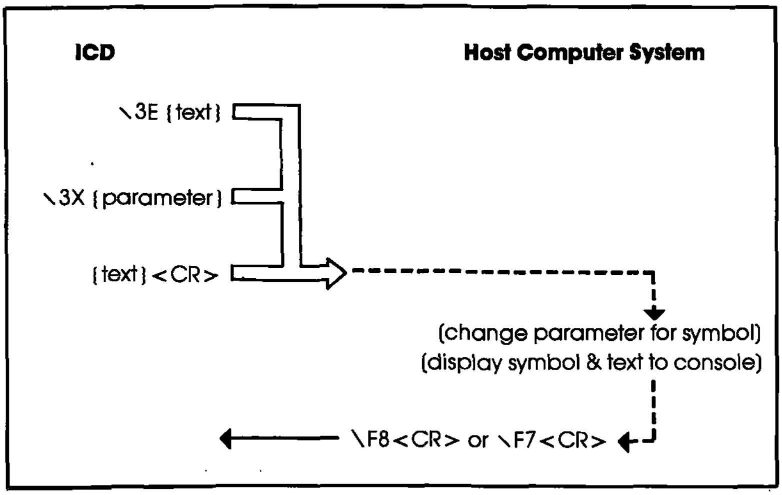
⇦ \F90 {Number} H <CR>
or \F1<CR>

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**


IDLE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
TEXT DISPLAY PROGRAM
OBJECT FILE LOAD/VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL /"Z" COMMAND PROGRAM
QUIT PROGRAM
CONSOLE KEY CHECK PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM

REMOTE MODE PROGRAM FLOW CHART



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:

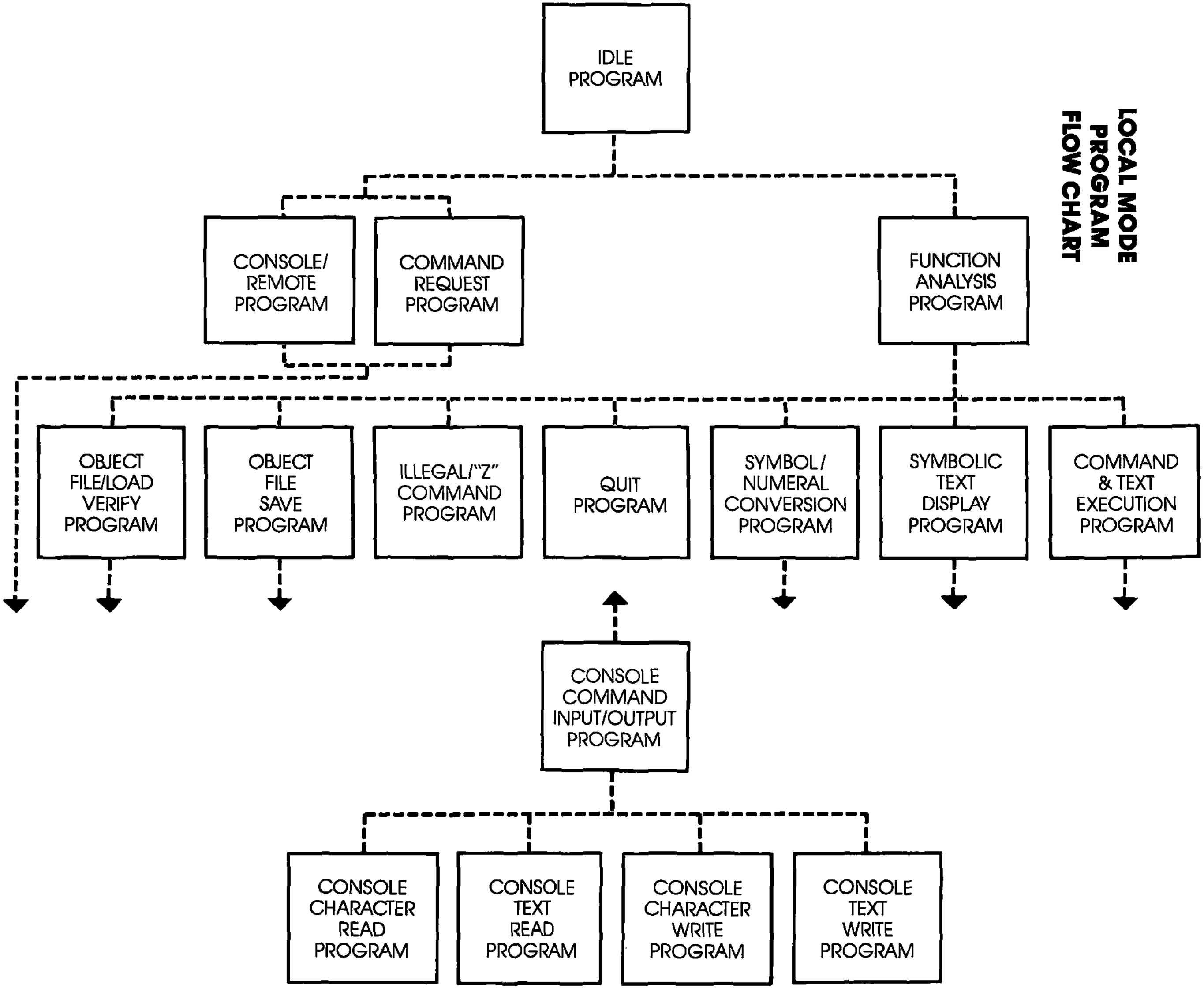
   a) If \3X{parameter} cannot be found in the input buffer, the host computer displays the contents of the input buffer already converted to symbols, sends \F8<CR> to the ICD, and then returns to the IDLE program.

   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}.

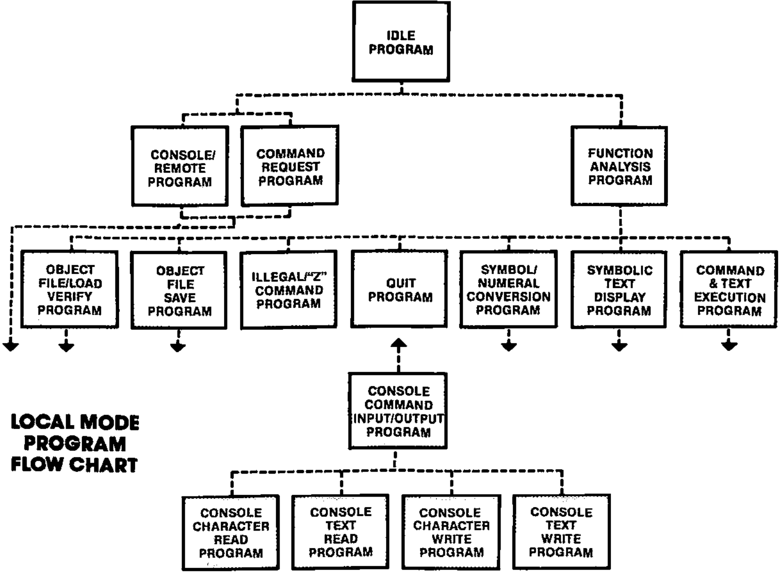
   If {parameter} is found in the symbol table, the host computer returns to "a" (above) after converting \3X {parameter} to a symbol.

LOCAL MODE PROGRAM FLOW CHART


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL / NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

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

PROGRAM: IDLE


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

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:

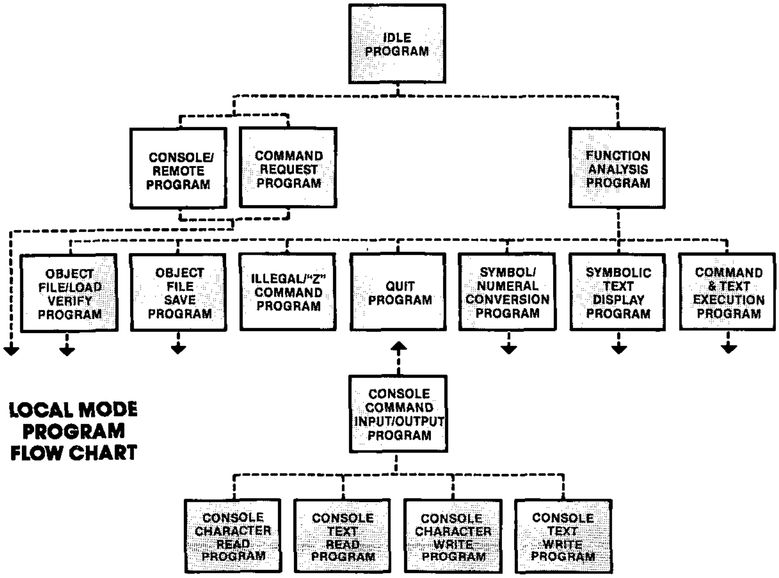
1. The host computer waits for an input from the ICD (The host system must have an input buffer to hold the input code from the ICD)

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

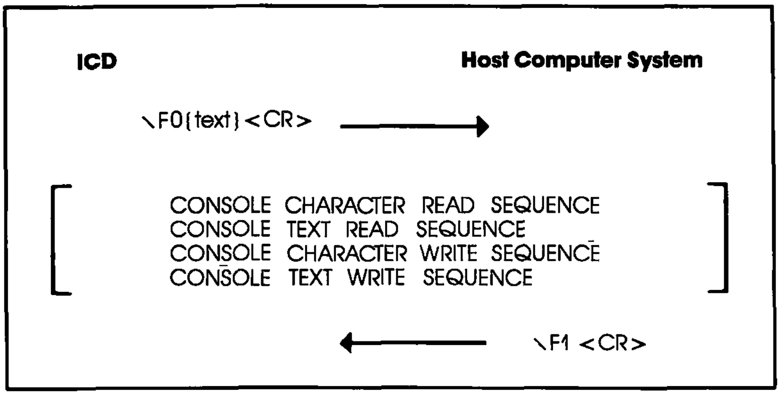
3. The host computer then executes one of the following programs depending on the code it receives:

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

PROGRAM: COMMAND REQUEST—CONSOLE


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

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>

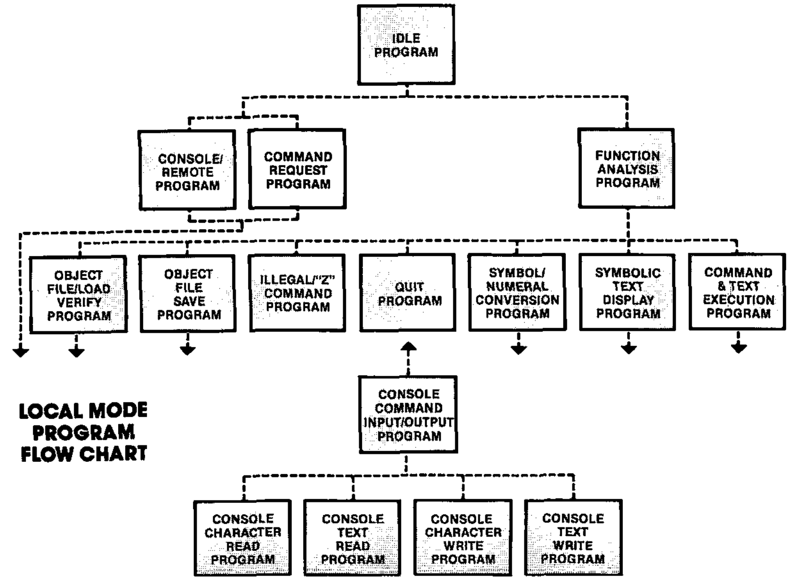
ORIGIN **ACTION:** These sequences allow commands to input to the ICD through the console terminal in the LOCAL mode.

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

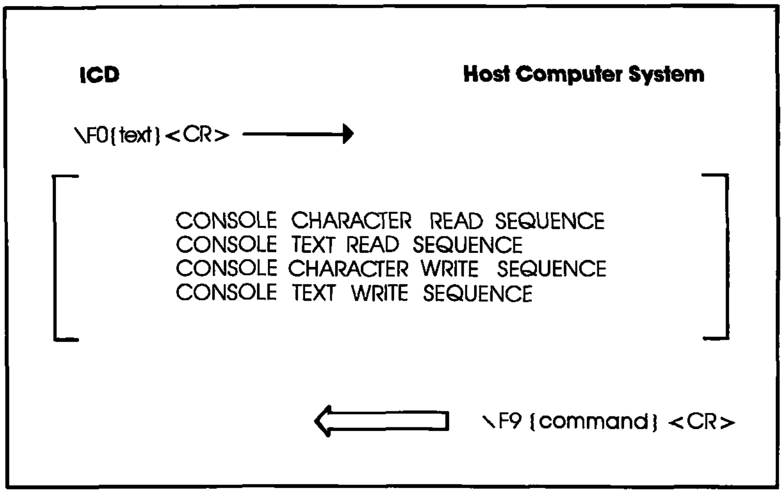
Optional sequences: CONSOLE CHARACTER READ/WRITE SEQUENCE or CONSOLE TEXT READ/WRITE SEQUENCE

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


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

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

← \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:

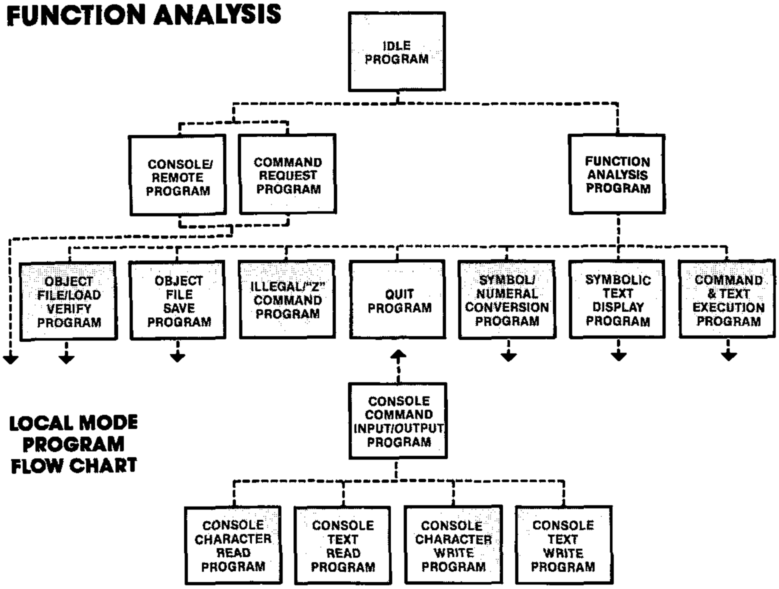
   a) CONSOLE CHARACTER READ PROGRAM
   b) CONSOLE TEXT READ PROGRAM
   c) CONSOLE CHARACTER WRITE PROGRAM
   d) CONSOLE TEXT WRITE PROGRAM

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


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

LOCAL MODE PROGRAM FLOW CHART

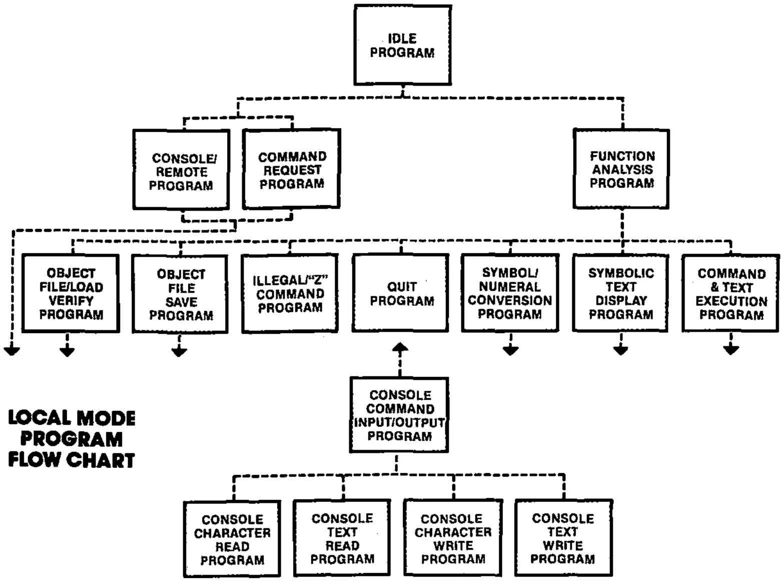
ACTION:

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


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

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:

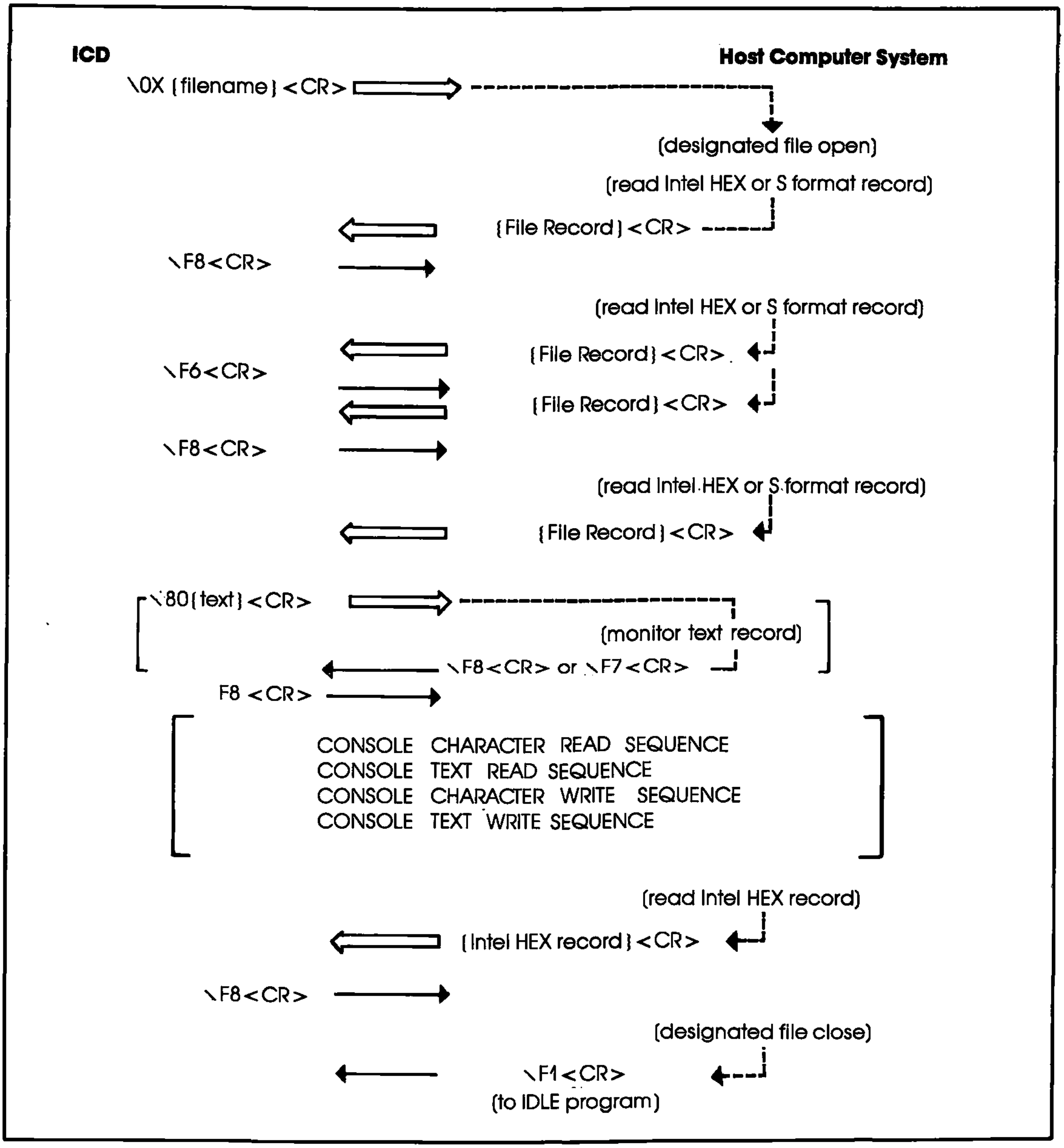
CONSOLE CHARACTER READ/WRITE SEQUENCE or
CONSOLE TEXT READ/WRITE SEQUENCE

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.

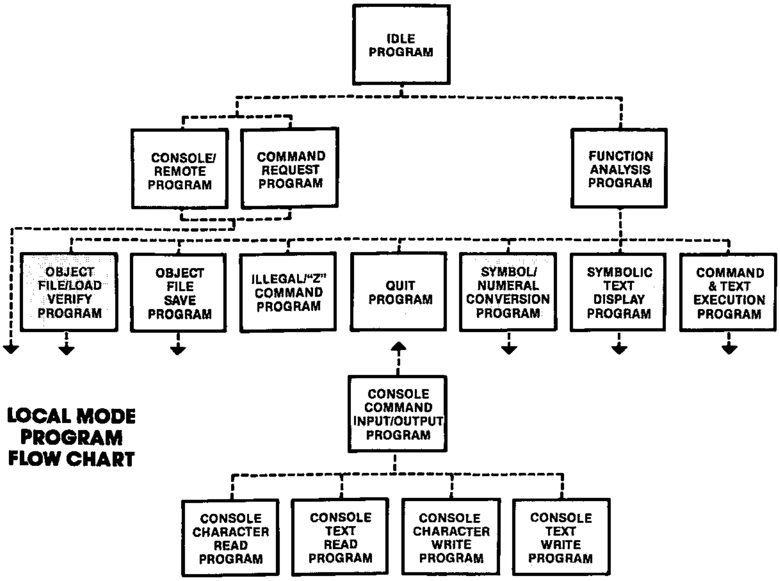
   b) If no error is detected, the host computer sends the Intel Hex or S format record to the ICD and then waits for \F8<CR> from the ICD.

   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


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
LOCAL MODE PROGRAM FLOW CHART
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

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.

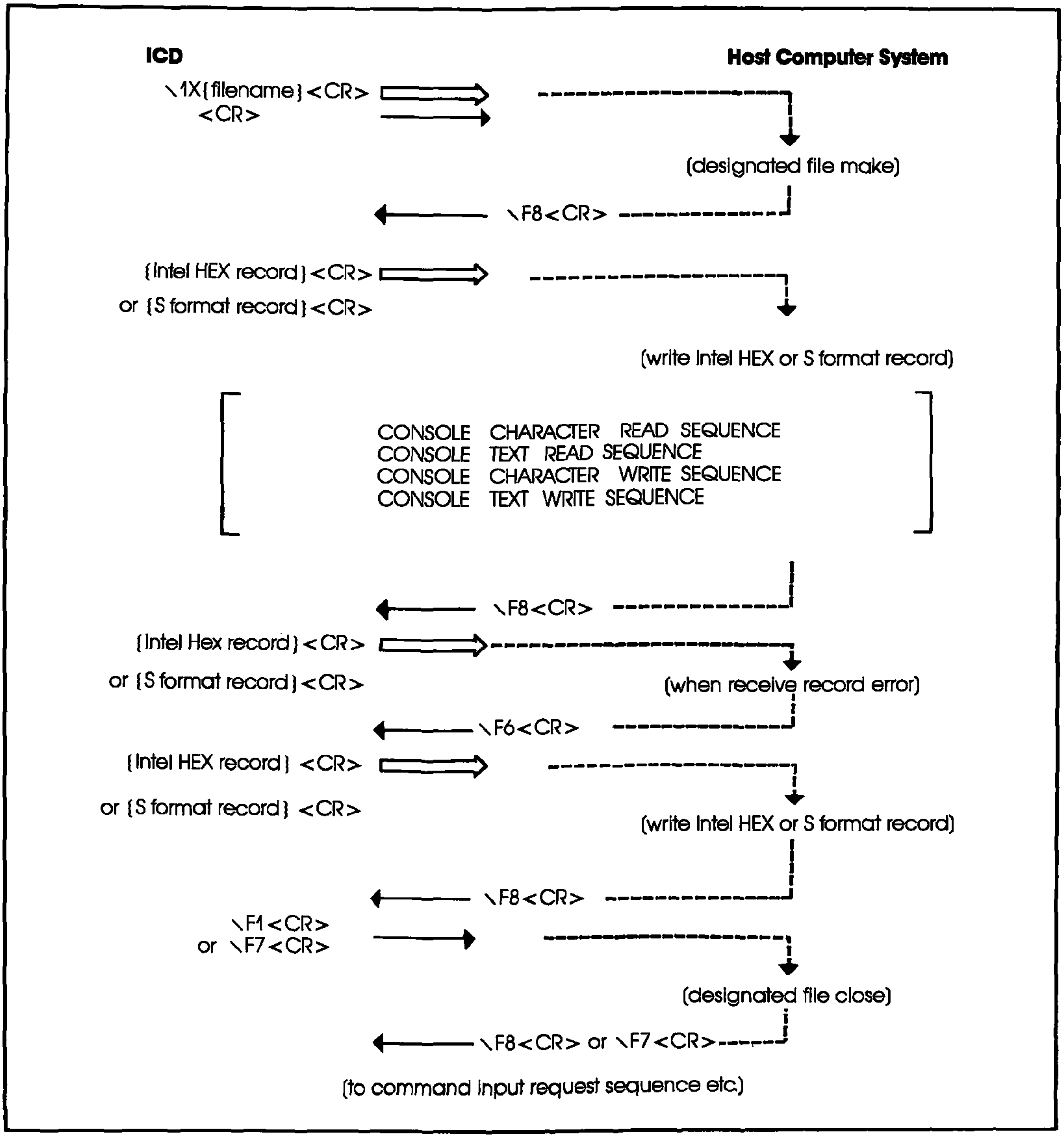
HOST \F6<CR>
OBJECT RECORD RE-TRANSMISSION REQUEST CODE. This code requests the ICD to re-transmit an object code.

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.

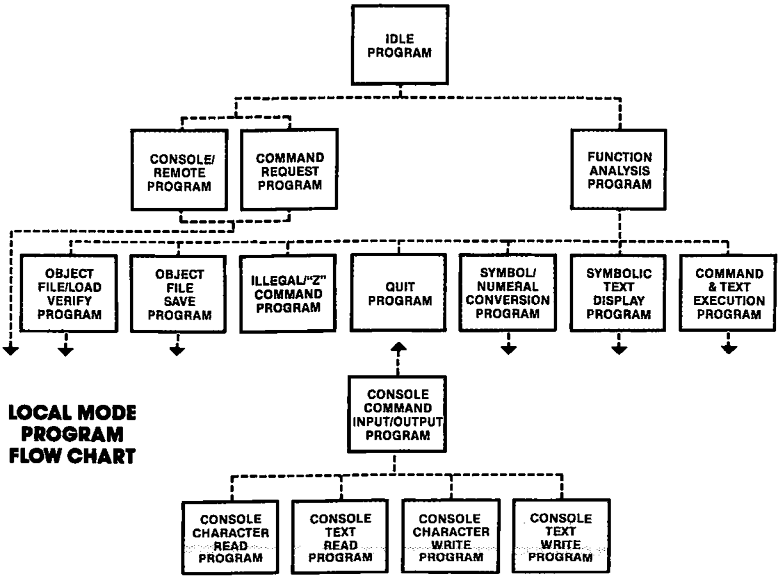
   If the file does not open, the host computer sends \F7<CR> to the ICD and returns to the IDLE program. If the file opens, the host computer sends \F8<CR> to the ICD.

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.

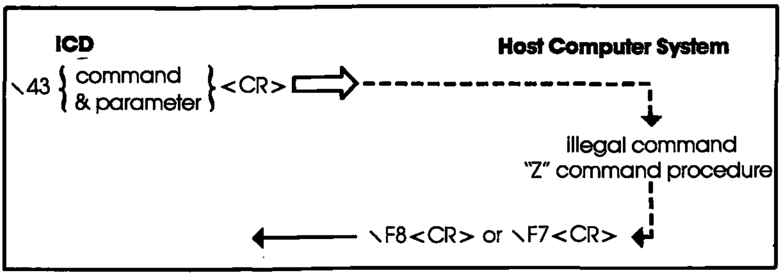
   After receiving an Intel Hex or S format record, the host computer then executes a file write of the record received from the ICD. If an error occurs during the file write operation, the host computer closes the user program file, sends \F7<CR> to the ICD, and returns to the IDLE program.

   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


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

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:

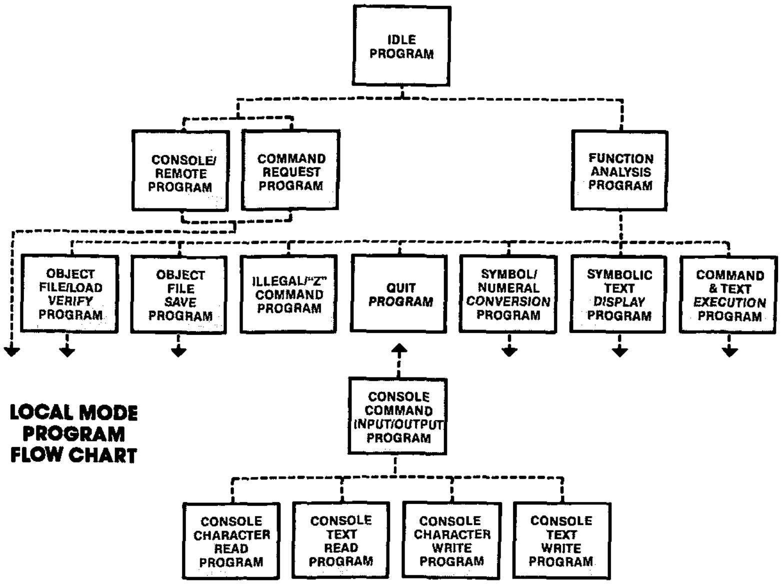
   If an error is contained in the "Z" command specification, the host computer sends \F7<CR> to the ICD and then returns to the IDLE program.

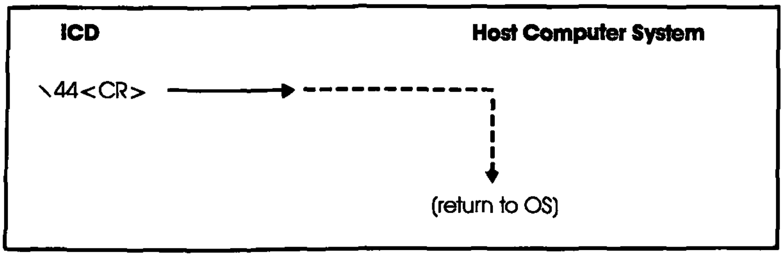
   If no error is detected, the host computer sends \F8<CR> to the ICD and then returns to the IDLE program.

"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

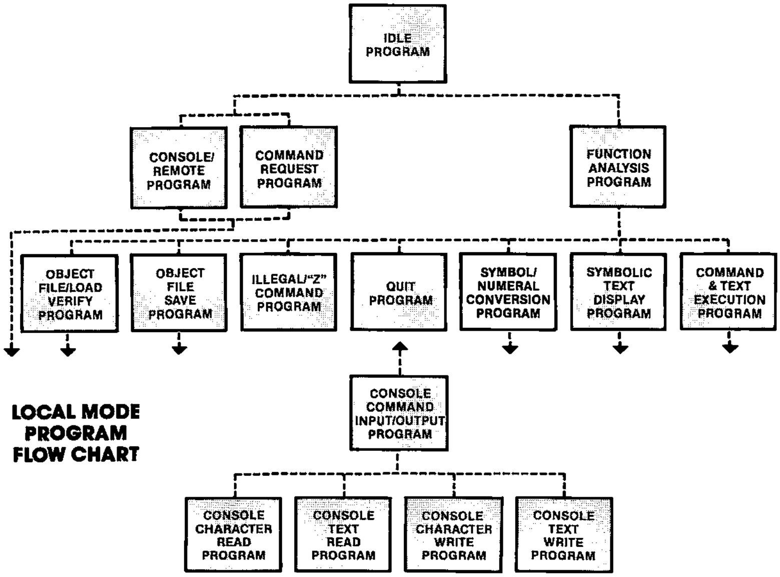

IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
LOCAL MODE PROGRAM FLOW CHART
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM



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


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

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.

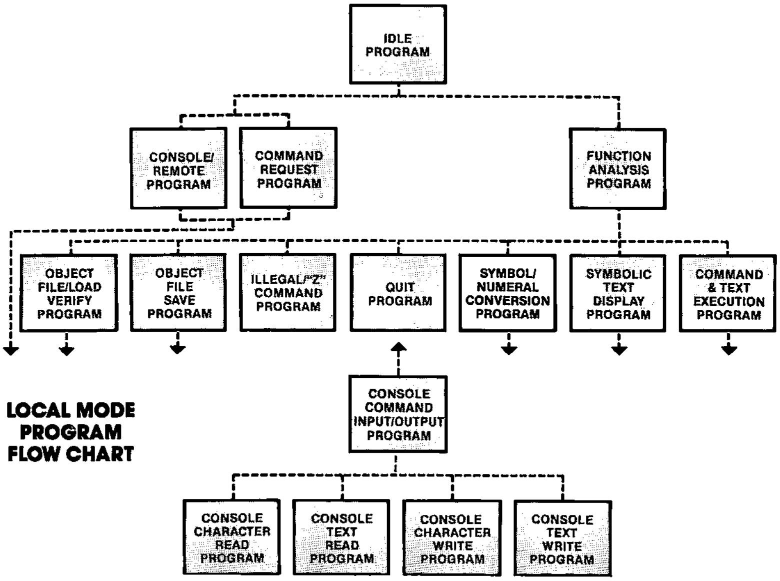
PROGRAM DESCRIPTION: **SYMBOL CONVERSION**

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


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
LOCAL MODE PROGRAM FLOW CHART
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

ACTION:

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

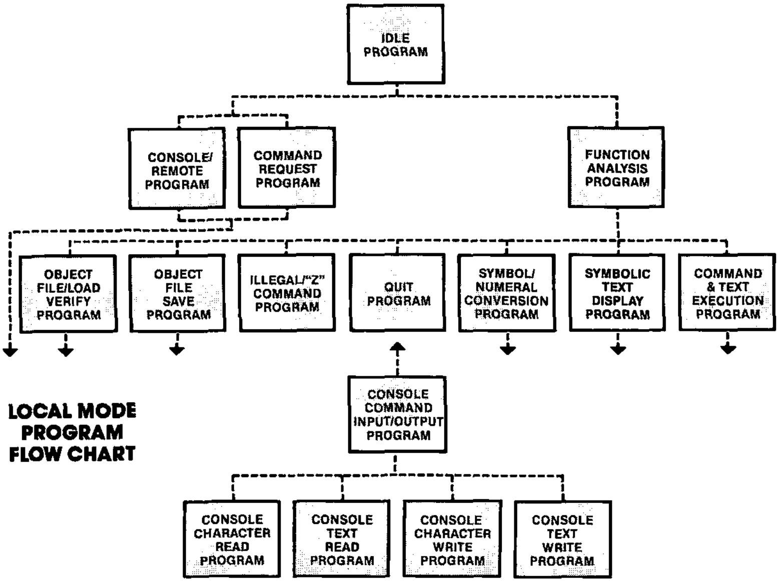
2. The host computer enters all data before <CR> into the ICD input buffer.

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

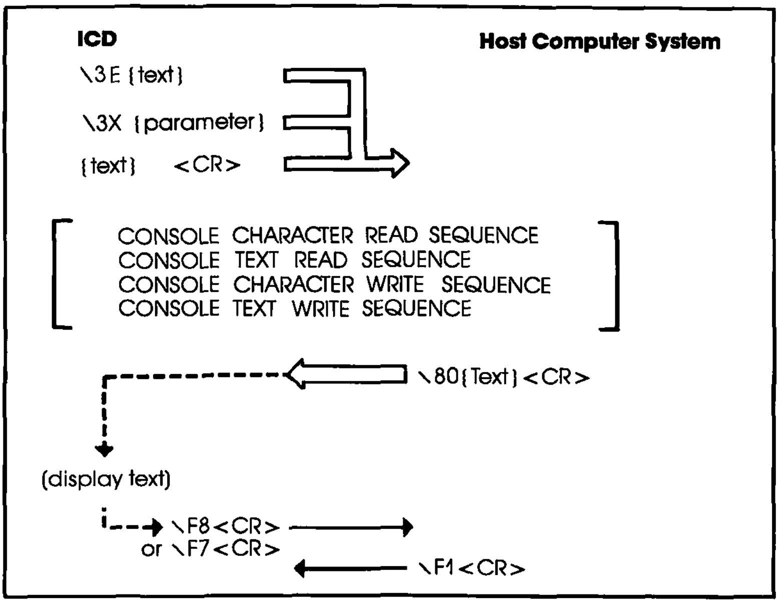
   a) If \3X{parameter} is not found, the host computer sends out the text—attaching \80 to the front and <CR> to the end of the text—and then waits for \F8<CR> from the ICD. When \F8<CR> is received from the ICD, the host computer sends \F1 to the ICD and returns to the IDLE program.

   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


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

LOCAL MODE PROGRAM FLOW CHART

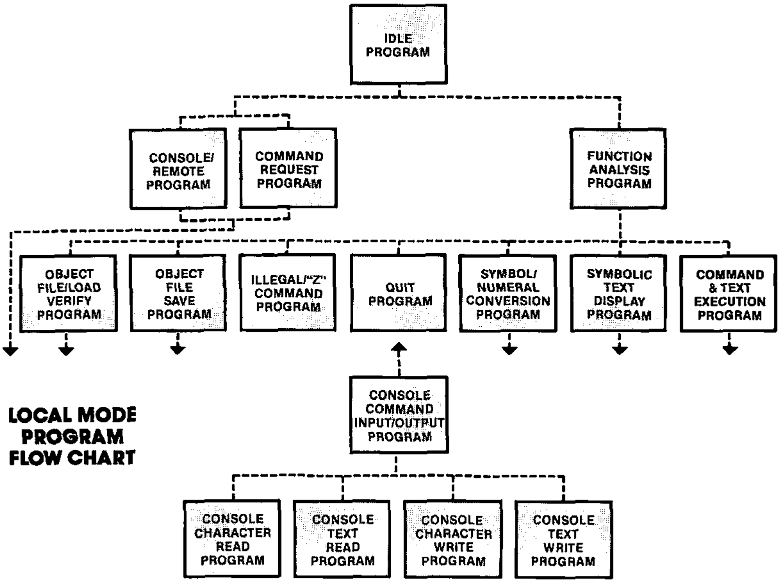

ICD
Host Computer System
\3E {text}
\3X {parameter}
{text} <CR>
CONSOLE CHARACTER READ SEQUENCE
CONSOLE TEXT READ SEQUENCE
CONSOLE CHARACTER WRITE SEQUENCE
CONSOLE TEXT WRITE SEQUENCE
\80{Text}<CR>
(display text)
\F8<CR>
or \F7<CR>
\F1<CR>

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

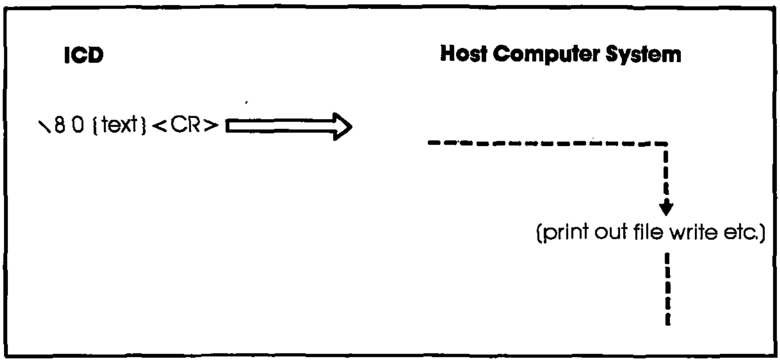
HOST	\80{text}
	{change to symbol}
	<CR>
		NUMERAL/SYMBOL RECORD. The host computer sends this record if the numeral is successfully converted to a symbol.

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


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

LOCAL MODE PROGRAM FLOW CHART

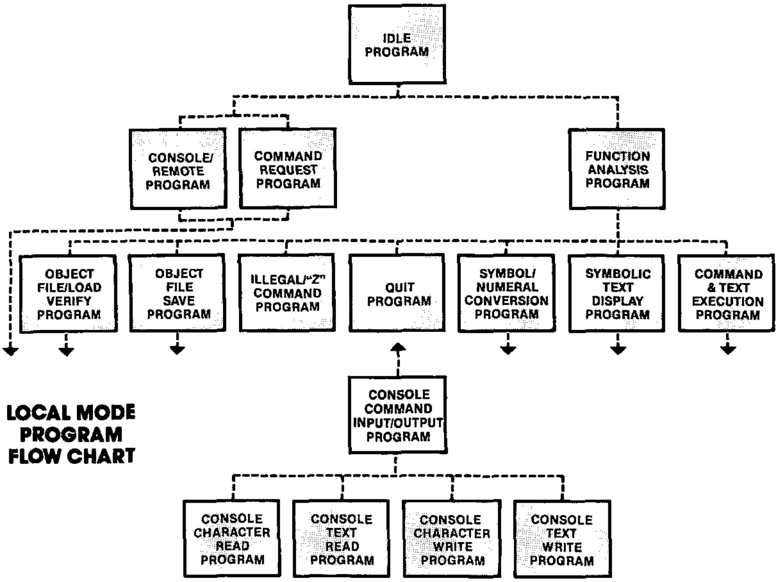


ORIGIN ACTION: The ICD outputs the command and the result of its execution to the host computer. The host computer can then output the text to a printer or onto a file. *NOTE: In the LOCAL mode, the PRINT ON command is treated as an Illegal/"Z" command after the HOST ON command is issued.*

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.

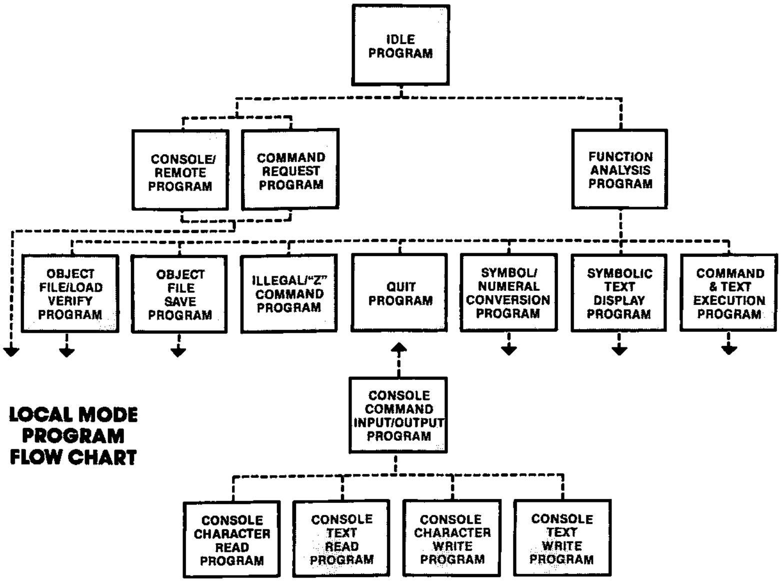
PROGRAM: **CONSOLE COMMAND INPUT/OUTPUT**


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
LOCAL MODE PROGRAM FLOW CHART
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

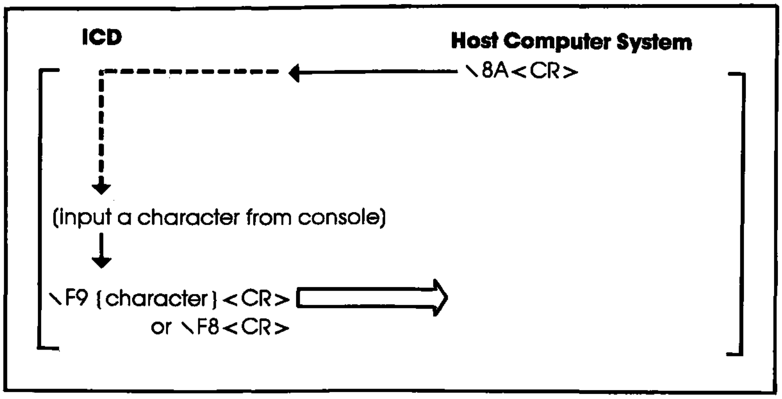
ACTION: There are four input/output sequences available when the ICD operates in the LOCAL mode:

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

PROGRAM: CONSOLE CHARACTER READ


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

LOCAL MODE PROGRAM FLOW CHART



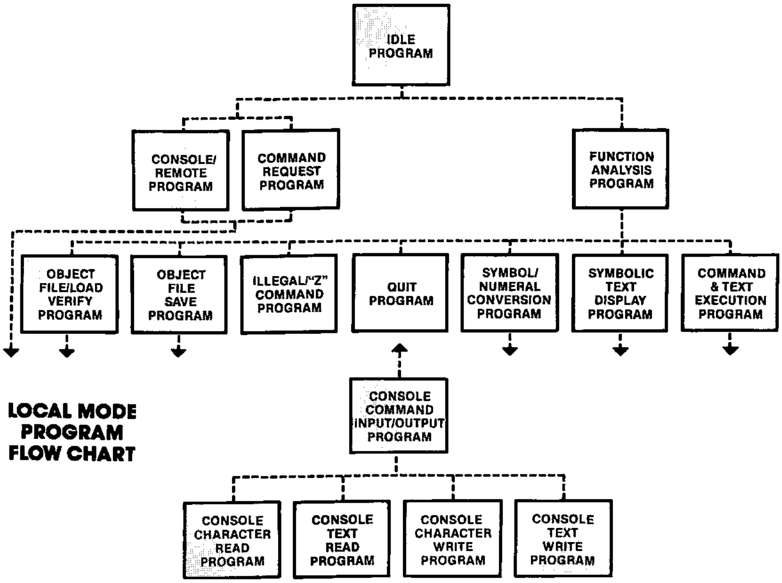
\8A<CR>

(Input a character from console)

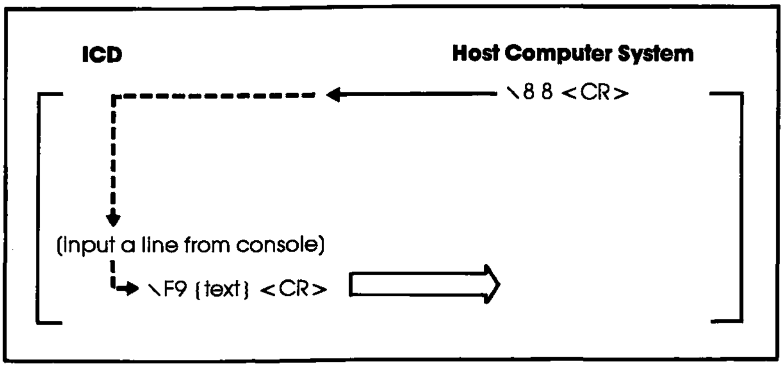
\F9 {character} <CR>
or \F8<CR>

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

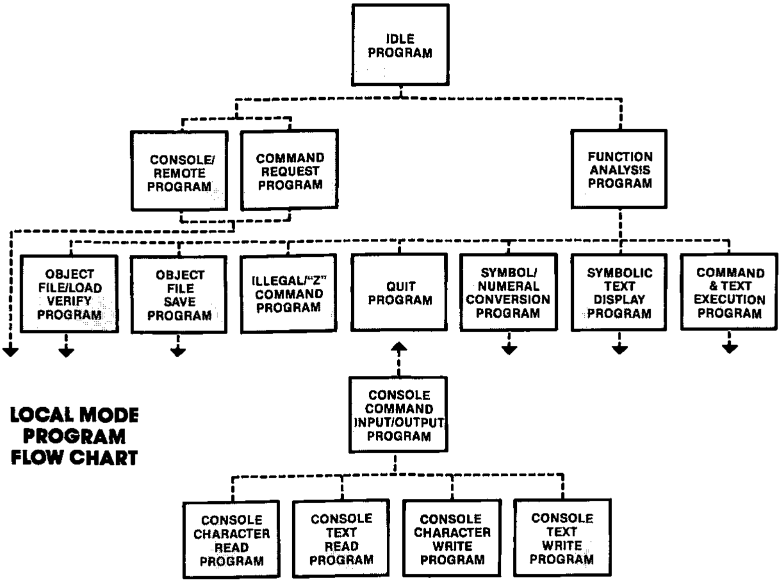

IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

LOCAL MODE PROGRAM FLOW CHART

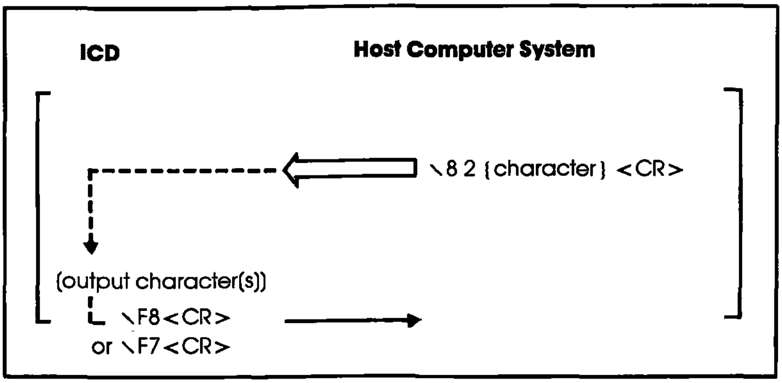


ORIGIN	ACTION:	This sequence is used when the host computer requests the ICD to input one line of data.
HOST	\88<CR>	DATA INPUT REQUEST CODE. The host computer sends this code to the ICD to request one line of data.
ICD	\F9{line of data} <CR>	DATA INPUT CODE. This code is sent to the host computer along 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

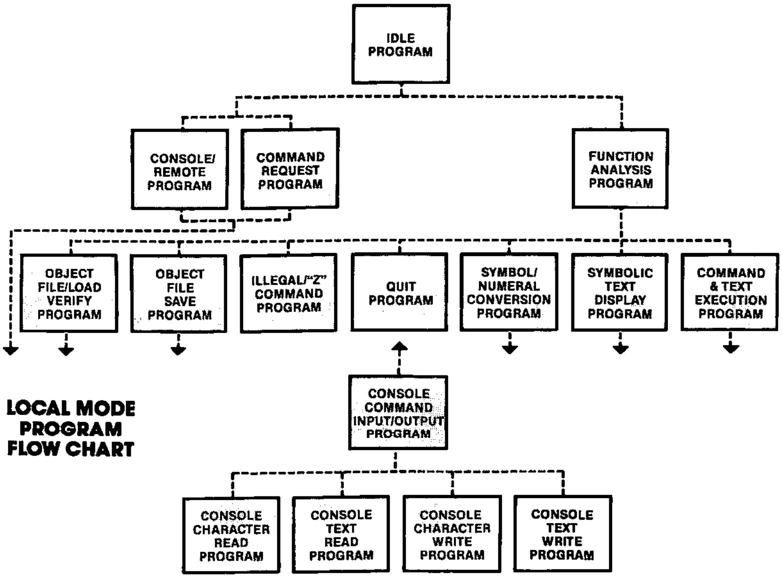

IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

LOCAL MODE PROGRAM FLOW CHART

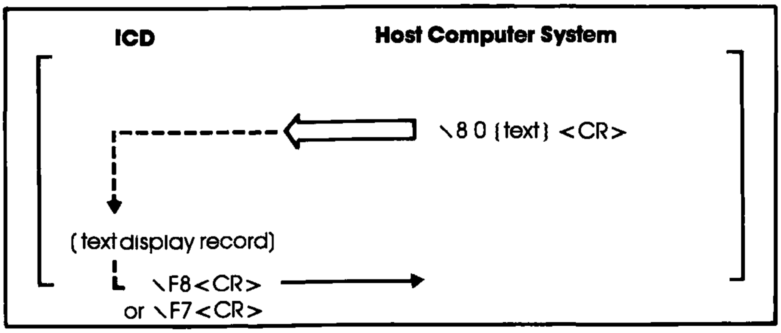


ORIGIN	ACTION:	This sequence is used when the host computer requests the ICD to output n characters to the console.
HOST	\82{characters}<CR>	N CHARACTERS OUTPUT REQUEST CODE. The host computer sends this code to the ICD, requesting the output of n characters to the console. The ICD then sends {characters} 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


IDLE PROGRAM
CONSOLE/ REMOTE PROGRAM
COMMAND REQUEST PROGRAM
FUNCTION ANALYSIS PROGRAM
OBJECT FILE/LOAD VERIFY PROGRAM
OBJECT FILE SAVE PROGRAM
ILLEGAL/"Z" COMMAND PROGRAM
QUIT PROGRAM
SYMBOL/ NUMERAL CONVERSION PROGRAM
SYMBOLIC TEXT DISPLAY PROGRAM
COMMAND & TEXT EXECUTION PROGRAM
CONSOLE COMMAND INPUT/OUTPUT PROGRAM
CONSOLE CHARACTER READ PROGRAM
CONSOLE TEXT READ PROGRAM
CONSOLE CHARACTER WRITE PROGRAM
CONSOLE TEXT WRITE PROGRAM

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	

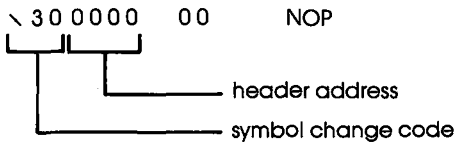
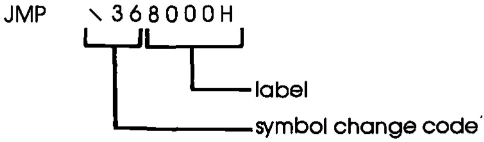
NUMBER CONVERSION CODES
ICD178/6800, 68010, 68008

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	$\underbrace{\backslash 30}_{\text{symbol change code}}\underbrace{0000}_{\text{header address}}$ 00 NOP
\31	not used	
\32	branch displacement	JR $\underbrace{\backslash 32}_{\text{symbol change code}}\underbrace{\$-10H}_{\textit{branch displacement}}$
\33—\35	not used	
\36	label	JP $\underbrace{\backslash 36}_{\text{symbol change code}}\underbrace{8000H}_{\text{label}}$
\37—\3F	not used	

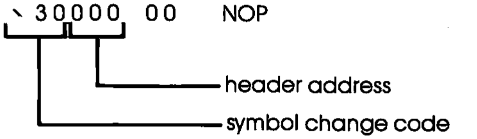
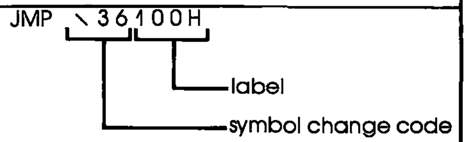
SYMBOL CONVERSION CODES
ICD278/I8085

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	\30000000 header address symbol change code
\31—\33	not used	
\34	number	MOVE. B \34$00,DO number symbol change code
\35	not used	
\36	label	JMP \36$00001000.W label symbol change code
\37	not used	
\38	variable	MOVE. B D0,\38$00001000.W variable symbol change code
\39—\3F	not used	

INTEL HEX OBJECT FORMAT:

All object files are represented by ASCII codes. This example shows one byte of data being converted to an ASCII hexadecimal number ("0"—"9" and "A"—"F") of two digits:

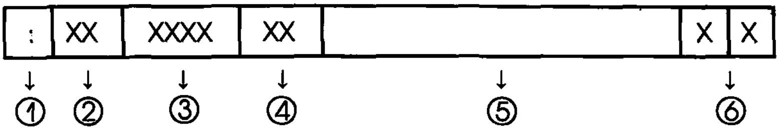
00_H	"00"(3030_H)
$9B_H$	"9B"(3942_H)

An object file is divided into units of records which include four types:

(1) Data Record
(2) End of File Record
(3) Extended Address Record
(4) Start Address Record

ICD/Z80, ICD/i8085, and ICD/8048 use Data and End Record only.

One record is formatted as shown below:



① Record mark
":" ($3A_H$)

Shows the beginning of an Intel Hex object record. The information preceding this mark is treated as a comment.

② Load address
"00"—"FF" (3030_H—4646_H)

Shows the number of data bytes contained in field ⑤.

③ Code address
"0000"—"FFFF" (30303030_H—46464646_H)

Shows the location address where a program or data is intended to be loaded. Normally contains "0000" as a dummy record.

④ Record type
Shows type of record:
"00" (3030_H) Data record
"01" (3031_H) End of File record

⑤ Data
Contains data bytes equal to the record length. (This field void if the record length is "00.")

⑥ Check Sum
2's complement of the value (one byte: carry ignored) of the total starting with the record length and the last data. *NOTE: Addition is made after the ASCII hexadecimal number of two digits has been converted to a 1-byte binary number.*

Example:

```
:020000020100FB
:2000000081000D00525A58608900040000CA00BAD95FFF4DF9AE52DA725FFD4FF2F808384
:0E0020000818001085A5A58B040000B0000490
:020000020200FA
:2000000008000000052EA1050A5401D0000B000FAF1DFFB0FFBBA50DAF35DFF5FF4FE008E1
:20002000081000D005A1A506000100D000000007AD05FFC0DFFAE125A585FFF2FF0F3009FD
:0400000301000000F8
:00000001FF
```

DATA RECORD: This record is used to show a program of 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
↓ ↓    ↓    ↓   ↓
①②   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
↓ ↓   ↓   ↓   ↓   ↓
①②   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
↓ ↓   ↓   ↓     ↓      ↓
①②   ③   ④     ⑤      ⑥
```

① Record mark
    ":" (3A$_H$)

② Record length
    "04" (3034$_H$)

Indicates that the data field in⑤contains 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$_H$)

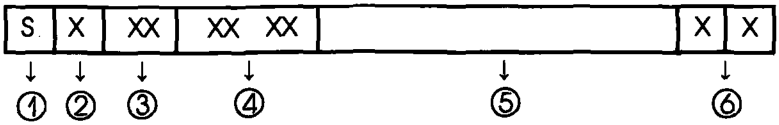
S FORMAT OBJECT FILE: All object files are described by ASCII codes. In the example, one byte of data is shown converted to a hexadecimal number ("0"—"9," and "A"—"F") of two digits:

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:

```
S00600004l424333
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

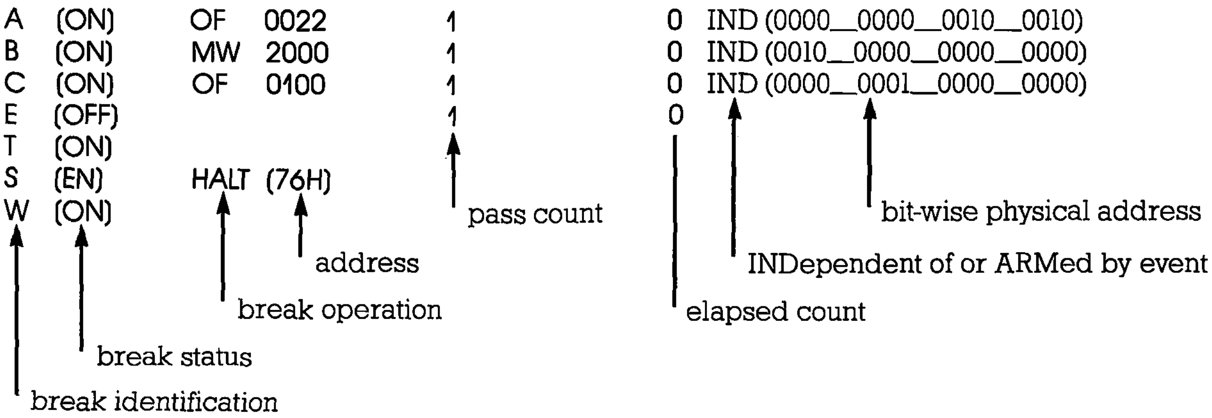
ACTION	COMMAND DEMONSTRATED	COMMENT
PRESS: The yellow RESET button on the ICD and look at the screen.		You'll see the ICD's identification message followed by a prompt: ICD-278 for Z80 V2.0 > The prompt (>) indicates that the ICD is waiting for a command. After you've executed a command, or whenever 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.

ACTION	COMMAND DEMONSTRATED	COMMENT
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.

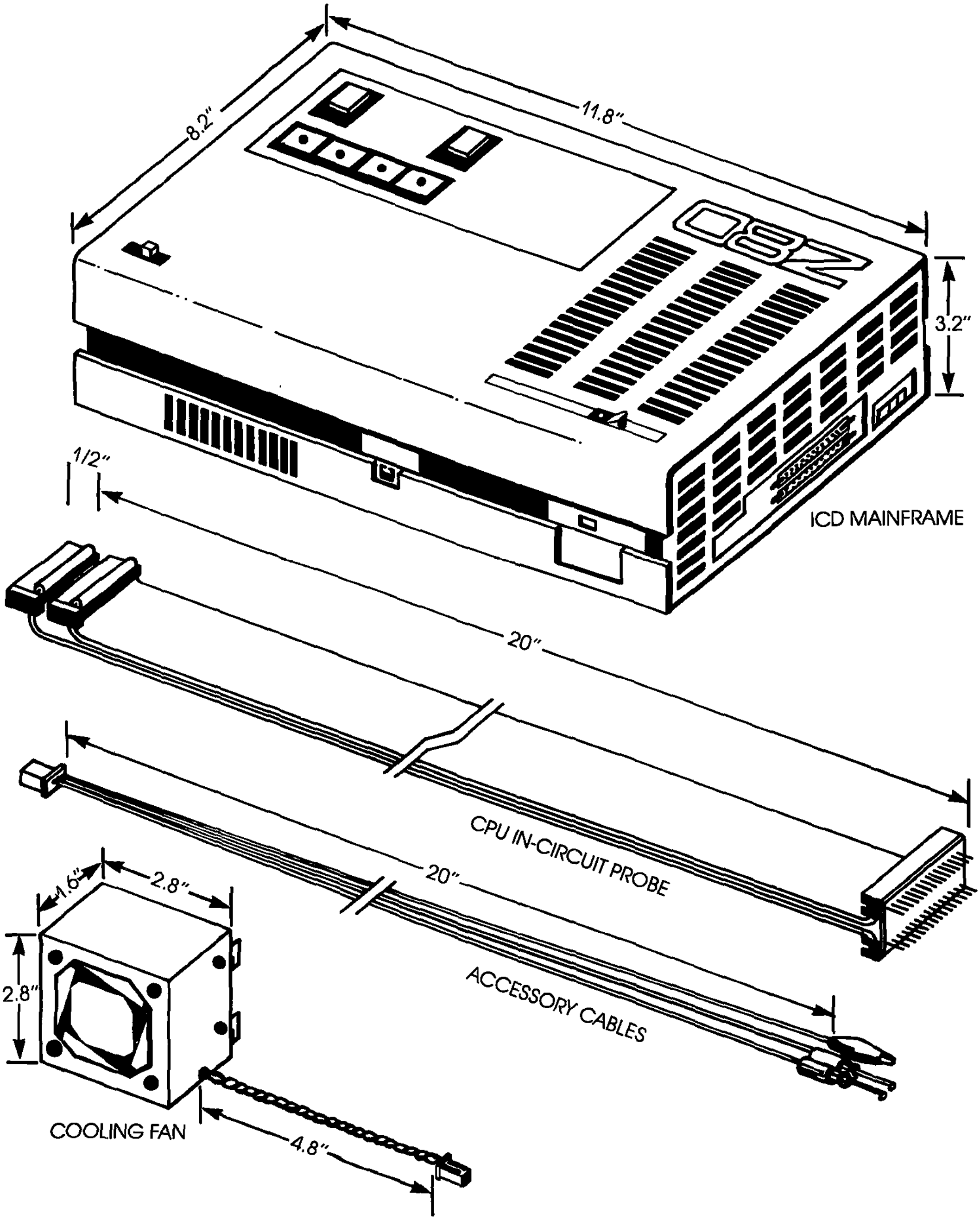
ACTION	COMMAND DEMONSTRATED	COMMENT
ENTER: MA 0,1FFF=RO	MAP	Maps to read-only instead of default read/write inside the ICD.
ENTER: MA 2000,EFFF=US	MAP	Maps to user memory.
ENTER: MA F000,FFFF=NO	MAP	Maps to non-existent memory.
ENTER: MA	MAP	Notice the status of the MAP command. Now you can move the 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)


8.2"
11.8"
3.2"
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.