Z80 ICE User Manual   Code Version 0.63 or V0.64 5/23/09

Chapter 1
Introduction/Background

The Z80 microprocessor was very popular in the late 1970’s and early 80’s and was designed into a number of computer systems. (It is probably still being designed into new products, as a CPU core in highly integrated VLSI chips for consumer products. Other CPU’s from this era, such as the 6502 and 8051 are enjoying the same longevity.) Because there are a number of S-100 systems and other Z80-based CP/M systems being actively maintained and repaired by their owners, I decided to design a simple PIC-based circuit that could emulate a Z80 for displaying memory, writing to I/O ports, etc. as an aid to troubleshooting. Before I started the design, I became aware of a product developed in the early 80’s by Nicolet Paratronics called the Z80 NICE. It offered a lot more capabilities than I had planned to implement, with only a small increase in component cost. Using the capabilities of this device as a reference, I designed the Z80 ICE to which this manual applies. I have added some new capabilities as well, which I hope will prove useful. Many thanks to Jeff Jonas for the generous loan of his NICE unit so I could become familiar with it.

Chapter 2
Setup

The Z80 ICE was designed to be used with either a terminal, or a terminal emulation program such as TeraTerm or HyperTerm running on a PC or Mac.

Serial settings are 8 data bits, no parity, and 1 stop bit. No hardware handshaking is needed. Note: The R command can process data and store it in the target system’s memory at up to 19.2 K baud, if the target system is running at 1 MHz or higher. If a slower Z80 clock is used, higher baud rates can cause a data overrun condition, which will cause an error message to be displayed. (Using hardware handshake won’t fix this, as the PC takes too long to react to CTS going inactive.) In this case, simply switch to a slower baud rate.

The ICE automatically detects the baud rate. After powering up the unit, hit Enter on the terminal to send a carriage return. The ICE uses this to detect the baud rate and set its rate to match. Supported rates are 300, 600, 1200, 2400, 4800, 9600, and 19,200 baud. After the ICE has set its baud rate, it will send the sign-on message. If you don’t see this after pressing Enter, reset or power down the system and try again.

Note: The ICE’s PIC microcontroller is reset by the same active-low signal that resets the Z80. If there is a problem with the target system that causes this signal to stay low or float, the ICE’s microcontroller will be held in reset, and will never send anything to the terminal.

The ICE will initially come up in Quit mode. (Z80 paused) Press X to dump the registers and see if everything is OK. If the register display hangs, reset or power down the system and try again.

If you would prefer to have the ICE come up and start running automatically, as a standard Z80 chip would, use the EA command to enable “AutoRun” mode. This setting will be saved in EEPROM, and the next time you power up, the Z80 will start executing from address 0 after reset ends. The DA command can be used to disable AutoRun mode.

Chapter 3
Command Line Interpreter

Commands may be entered in either upper or lower case. All numeric values such as addresses or counts are entered and displayed in hex. When entering a 16-bit value (d16) 1-4 hex digits can be entered. 1 or 2 digits can be entered for an 8-bit value (d8). If a command accepts parameters it must be followed by a space. The parameters can be separated by either a comma or a space. Each command must be followed immediately by either a semicolon or a carriage return.

The command line can hold up to 31 characters. Once that limit is reached, you should hear a beep from the terminal. Any additional characters entered will replace the last character on the line.

The command(s) on the command line will be executed when a carriage return is typed.

There are two prompts:

OK====> indicates the last command was accepted and executed without error

ERR===> indicates that the last command was either not accepted, or caused an error when it was executed.

In either case, the ICE is ready for a new command to be entered.

Some commands may take a long time to execute, and you may decide to abort the command. The following commands may be either paused using Ctl-S, then resumed using Ctl-S or Ctl-Q, or stopped, by using Ctl-C:

D	Display memory
L	List memory
V	Verify (compare) memory
T	Trace
U	Untrace

It is possible to put more than one command on the same command line. In this case, use a semicolon instead of carriage return to signal the end of a command. Using the repeat line (RL) command, it is possible to repeat a sequence of commands in a loop. Delay can be inserted between commands with the Z command.

For example, the command

```
O 80 A5;Z 10;RL<CR>
```

will write A5 to I/O address 80, then delay for 16 (0x10) mSec, and repeat. To break out of the loop, press Ctl-C. The actual delay between iterations will be larger than 16 mSec if refresh is enabled.

The following commands always end command line processing, and will prevent looping if used in a command line repeat loop:

G, L, Q, R, S, T, U, W, X

Chapter 4
Overview of Operation

The ICE has two modes of operation – Go mode and Quit mode.

In Go mode, the ICE executes Z80 instructions at full speed, but not all of the commands are available. Basically, commands that don't require participation by the Z80 can be executed in this mode, such as enabling or disabling interrupts or bus requests to the Z80. Breakpoints can be changed in Go mode, since they are not used in this mode.

In Quit mode, the Z80 is paused, and can be used to implement the full set of commands, such as displaying or disassembling memory, computing checksums, writing to I/O ports, etc. The Trace and Untrace commands can be used in Quit mode to execute code, but not at full speed. Since the Trace mode displays all of the registers after each instruction, its speed is affected by the baud rate being used. This command is always slower than Untrace. The Untrace command does not display anything after each instruction, but it does grab the current PC value from the Z80 to check against breakpoints and printpoints. This takes some time. Also, refresh adds some delay if enabled. With refresh enabled, and the Z80 clocked at 4 MHz, Untrace executes roughly 1600 instructions/second. (This is not fast enough to keep up with data coming off of a floppy disk, so don't try to boot from a floppy in untrace mode.)

Interrupts:

The ICE can recognize interrupts in Go mode, but not in Quit mode. In Go mode, the user can enable or disable maskable or non-maskable interrupts reaching the CPU. Separate control of each is available. In Quit mode, no interrupts can reach the CPU.

Bus Requests:

The ICE can recognize bus requests in Go mode, but not in Quit mode. In Go mode, the user can enable or disable bus requests reaching the CPU. In Quit mode, no bus requests can reach the CPU.

Memory Refresh:

When the ICE is in Quit mode, the Z80 executes bursts of NOP instructions to provide for refresh of any DRAM present in the target system. This takes up about 25% of the processor's time, so if you know that your system doesn't use DRAM, you may want to disable refresh to speed up instruction tracing. In Go mode, the Z80 is executing at full speed, so no extra refresh provision is needed.

Since refreshing DRAM's requires a certain number of refresh cycles in a certain period of time, if the system clock is too slow, it will be impossible to meet this requirement. If the Z80's clock is below 200 KHz, refresh will be disabled at start-up. To meet a typical DRAM refresh requirement, the clock would need to be more like 2 MHz anyway.

Reset:

The PIC MCU uses the same reset signal as the Z80, so it comes up in a known state whenever the Z80 gets reset:

Maskable and non-maskable interrupts are enabled, or will be at least, when Go mode is entered. Bus requests are enabled as well.

The ICE comes up in Quit mode unless Autorun is enabled, then it comes up in Go mode, with the Z80 executing code.

All breakpoint and printpoint enables are cleared

The “saved memory address” is set to 0xFFFF. (More on this later)

The default PC starting value for the next G command is reset to 0000

Since the PIC gets reset whenever the Z80 is reset, you need to send it a carriage return to allow the auto-baud detection to complete. If characters are not echoed to the terminal after you see the version text displayed, reset or power down the target system and try again.

Chapter 5
Go Mode Commands

These commands can be entered in either mode (except Q):

BP	-	Set breakpoint (address)
BPC	-	Set breakpoint count
EBP	-	Enable breakpoint
DBP	-	Disable breakpoint
EPP	-	Enable printpoint
DPP	-	Disable printpoint
C	-	Clear all breakpoints and printpoints
EI	-	Enable maskable interrupts
DI	-	Disable maskable interrupts
EN	-	Enable non-maskable interrupts
DN	-	Disable non-maskable interrupts
EB	-	Enable bus requests (DMA)
DB	-	Disable bus requests
ER	-	Enable refresh
DR	-	Disable refresh
EA	-	Enable start-up Autorun (setting saved in EEPROM)
DA	-	Disable start-up Autorun (setting saved in EEPROM)
H	-	hexadecimal sum and difference
CF	-	Clock Frequency measurement (Z80 clock)
Q	-	Quit (pause Z80)
RL	-	Repeat (command) line
ST	-	Status display
Z	-	Sleep (insert delay)

BP: (Set breakpoint address)

Sets one of the three breakpoint addresses, which are used by the trace and untrace commands. Breakpoints end the current trace or untrace command when reached (and enabled) during either of these commands. They are compared to the current PC after each instruction is executed.

Breakpoints only work in Quit mode. Current breakpoint status can be displayed with the ST command.

Format:

BP n,d16 n is 1,2 or 3. d16 is a 16-bit address

Example:

```
OK ====> ST

---> 0ABC 00 D
---> 0246 00 D
---> 0369 00 D
INBR

OK ====> BP 1 123

OK ====> BP 2 654

OK ====> BP 3 982

OK ====> ST

---> 0123 00 D
---> 0654 00 D
---> 0982 00 D
INBR

OK ====>
```

BPC: (Set breakpoint pass count)

Sets the pass count for one of the three breakpoint addresses, which are used by the trace and untrace commands. (Count=0 and count=1 will produce the same result) Setting the count to 3 will cause a trace to stop on the third time that breakpoint matches the current PC. (If it is enabled)

If the pass count does not reach zero while a trace or untrace command is executing, its value is preserved and can be examined with the ST command to see how many times that breakpoint was hit. So if the pass count was set to 10 at the start of the command and is 7 after the trace or untrace stops, that address was hit 3 times.

Format:

BPC n,d8 n is 1,2 or 3. d8 is an 8-bit count

Example:

```
OK ====> ST

---> 0123 00 D
---> 0654 00 D
---> 0982 00 D
INBR

OK ====> BPC 1,33

OK ====> BPC 2,66

OK ====> BPC 3,99

OK ====> ST

---> 0123 33 D
---> 0654 66 D
---> 0982 99 D
INBR

OK ====>
```

EBP: (Enable breakpoint(s))

Enables one of the three breakpoints, which are used by the trace and untrace commands. If no parameter is entered, all three breakpoints will be enabled.

Format:

EBP [n]   n is 1,2 or 3

DBP: (Disable breakpoint(s))

Disables one of the three breakpoints, which are used by the trace and untrace commands. If no parameter is entered, all three breakpoints will be disabled.

Format:

DBP [n]   n is 1,2 or 3

Example:

```
OK ====> ST

---> 0123 33 D
---> 0654 66 D
---> 0982 99 D
INBR

OK ====> EBP

OK ====> ST

---> 0123 33 E
---> 0654 66 E
---> 0982 99 E
INBR

OK ====> DBP 3

OK ====> ST

---> 0123 33 E
---> 0654 66 E
---> 0982 99 D
INBR

OK ====>
```

EPP: (Enable printpoint(s))

Enables one of the three printpoints, which are used by the trace and untrace commands. If no parameter is entered, all three printpoints will be enabled.

Printpoints cause the current Z80 register values to be displayed when reached (and enabled) during an untrace command. They are compared to the current PC after each instruction is executed.

Note that printpoints and breakpoints share the same three address values, so a particular address can be enabled as a breakpoint, a printpoint, or both. (Breakpoint pass count values don't affect printpoint operation.)

Format:

```
EPP [n]         n is 1,2 or 3
```

DPP: (Disable printpoint(s))

Disables one of the three printpoints, which are used by the trace and untrace commands. If no parameter is entered, all three printpoints will be disabled.

Format:

```
DPP [n]         n is 1,2 or 3
```

Example:

```
OK ====> EPP

OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
INBR

OK ====> DPP 2

OK ====> DPP 3

OK ====> ST

---> 0123 33 E P
---> 0654 66 D
---> 0982 99 E
INBR

OK ====>
```

EI: (Enable maskable interrupts)

Enables maskable interrupts to reach the Z80 when in Go mode. If this command is entered while the CPU is in Go mode, interrupts will be enabled immediately. An “I’ appears on the last line of the status display if maskable interrupts are currently enabled. (All interrupts are always disabled in Quit mode, even if an I is displayed)

Format:

```
EI
```

DI: (Disable maskable interrupts)

Prevents maskable interrupts from reaching the Z80 when in Go mode. If this command is entered while the CPU is in Go mode, interrupts will be disabled immediately.

Format:

```
EI
```

Example:

```
OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
INBR

OK ====> DI

OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
 NBR

OK ====> EI

OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
INBR

OK ====>
```

EN: (Enable non-maskable interrupts)

Enables non-maskable interrupts to reach the Z80 when in Go mode. If this command is entered while the CPU is in Go mode, non-maskable interrupts will be enabled immediately. An “N’ appears on the last line of the status display if non-maskable interrupts are currently enabled. (All interrupts are always disabled in Quit mode, even if an N is displayed)

Format:

EN

DN: (Disable non-maskable interrupts)

Prevents non-maskable interrupts from reaching the Z80 when in Go mode. If this command is entered while the CPU is in Go mode, non-maskable interrupts will be disabled immediately.

Format:

DN

Example:

```
OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
INBR

OK ====> DN

OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
I BR

OK ====> EN

OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
INBR

OK ====>
```

EB: (Enable bus requests)

Enables bus requests to reach the Z80 when in Go mode. If this command is entered while the CPU is in Go mode, bus requests will be enabled immediately. A “B’ appears on the last line of the status display if bus requests are currently enabled. (Bus requests are always disabled in Quit mode, even if a B is displayed)

Format:

EB

DB: (Disable bus requests)

Disables bus requests from reaching Z80 when in Go mode. If this command is entered while the CPU is in Go mode, bus requests will be disabled immediately.

Format:

DB

Example:

```
OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
INBR

OK ====> DB

OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
IN R

OK ====> EB

OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
INBR

OK ====>
```

ER: (Enable refresh)

Enables refresh bursts to occur when the Z80 is in Quit mode. This setting has no effect on Go mode operation. An “R’ appears on the last line of the status display if refresh is currently enabled.

Format:

ER

DR: (Disable refresh)

Disables refresh bursts when the Z80 is in Quit mode.

Format:

DR

Example:

```
OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
INBR

OK ====> DR

OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
INB

OK ====> ER

OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
INBR

OK ====>
```

EA: (Enable Autorun)

Enables Autorun after power-up or reset. This setting is saved in EEPROM and will be maintained even when power is removed. Enabling Autorun causes the Z80 to immediately start executing code from address 0 at power-up. The ICE is in Go mode at this point. Maskable and non-maskable interrupts and bus requests are enabled. The current Autorun setting is part of the status display.

Format:

EA

DA: (Disable Autorun)

Disables Autorun after power-up or reset. This setting is saved in EEPROM and will be maintained even when power is removed. Disabling Autorun causes the ICE to enter Quit mode at power-up until a G command is executed.

Format:

DA

Example:

```
OK ====> EA

OK ====> ST

---> 0000 00 D
---> 0000 00 D
---> 0000 00 D
INBRA

OK ====> DA

OK ====> ST

---> 0000 00 D
---> 0000 00 D
---> 0000 00 D
INBR

OK ====>
```

H:	(Hex “calculator”)

Computes hexadecimal sum and difference. If values entered are A and B, in that order, command displays A+B, A-B. Results are limited to 16 bits and will wrap above 0xFFFF.

Format:

H d16,d16	Values entered as 1-4 hex digits.

Example:

```
OK ====> H 10A5,674C
77F1,A959
OK ====> H FFFE,0002
0000,FFFC
OK ====> H 2000,1000
3000,1000
OK ====>
```

CF: (Clock frequency)

Measure the frequency of the Z80 clock. Results are displayed with 1 KHz resolution. Minimum displayable clock rate is ~ 19 KHz.

Format:

CF

Example:

```
OK ====> CF

Z80 clock frequency is ~ 3.976 MHz
OK ====>
```

Q: (Quit)

Quit (stop Z80 and enter Quit mode)

Format:

Q

Example:

```
OK ====> Q

.Z.V..   A=00  BC=1234  DE=F74C  HL=5542  S=FFF0  P=824D  LD A,L
S....C  A’=4C  B’=1A64  D’=F29C  H’=0000  X=FFF0  Y=824D  I=00

OK ====>
```

RL (Repeat Line)

This command may be used to repeat a command or group of commands on a single command line. This could be used together with the sleep command to create a repetitive write to the same memory location, for instance, so that an oscilloscope can be used to observe timing and signal behavior. To break out of the loop, press Ctl-C.

Example:

```
OK ====>  P 8000 A5;Z 9;RL
```

Above command line will write 0xA5 to address 0x8000 then it will delay for 9 mSec, then repeat. (The actual delay between writes will be longer if refresh is enabled)

ST (Status)

Display breakpoint/printpoint and hardware status. The first three lines of the status display are devoted to breakpoint/printpoint information. The first value is the address which will be compared to the PC. The second value is the breakpoint pass count. The third value is either a D or E which indicates whether that address and pass count are currently enabled (E) or disabled (D) as a breakpoint. The fourth value, if displayed, is a P if the address is currently enabled as a printpoint.

The fourth line shows the current state of the three hardware enables. If maskable interrupts are enabled, an I will be displayed. If non-maskable interrupts are enabled, an N will be displayed. If bus requests are enabled, a B will be displayed. Note that all three of these are always disabled in Quit mode, regardless of what is displayed. If Quit mode refresh bursts are being generated, an R will also be displayed. If start-up Autorun is enabled, an A will also be displayed. Finally, if the Z80 is currently executing instructions at full speed (a Go command has been issued) the word RUNNING will be displayed.

Format:

ST

Example:

```
OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
INBR

OK ====> G

Execution begins at ====> 473F

OK ====> ST

---> 0123 33 E P
---> 0654 66 D P
---> 0982 99 E P
INBR RUNNING

OK ====>
```

Z (Sleep)

Causes the command processor to insert a variable delay before processing any other commands.

Format:

Z d8 The parameter is used to specify the desired delay in mSec. The actual delay will be somewhat longer if refresh is enabled.

Chapter 6
Quit Mode Commands

These commands can only be used in Quit mode:

CS	-	Generate and display 16-bit checksum of memory region
D	-	Display memory
E	-	Examine a single memory location (intended for ‘scope loops)
F	-	Fill memory
G	-	Go
I	-	Input from I/O port
L	-	List (disassemble) memory
M	-	Move memory block
MT	-	Memory test (destructive)
O	-	Output to I/O port
P	-	Put data into a single memory location (intended for ‘scope loops)
R	-	Read Intel Hex file into memory
S	-	Substitute into memory
SR	-	Soft reset
T	-	Trace
U	-	Untrace
V	-	Verify
W	-	Write Intel Hex file from memory
X	-	Examine/change registers

CS:	(Checksum memory region)

Adds memory contents from start address to end address (inclusive) and displays 16-bit sum. This is a simple 8-bit accumulation, with carry, so the result can be more than 8 bits. Only the low 16-bits of the result are displayed. This should match the checksum generated by many EPROM programmers.

Format:

CS start,end	Addresses entered as 1-4 hex digits.

Format:

CS

Example:

```
OK ====> CS 0000,03FF
Checksum: 0x73FD
OK ====>
```

D: (Display memory)

Displays memory contents from start address to end. Start address is truncated so that display always starts at nnn0. Multiples of 16 bytes are displayed, until end address is reached (or passed). ASCII equivalents for the data are also displayed. A dash is inserted after the first 8 bytes.

Format:

D [start],[end]	Addresses entered as 1-4 hex digits. Both parameters are optional. If no end address is entered, 0x80 bytes are displayed. If no start address is entered, display begins at the next address after the end of the last D or L command. (This is the “saved memory address”) So entering just D will start displaying from the end of the previous D command.

Example: (ASCII section is not correct)

```
OK ====> D 0002,0019

0000      C3 F1 A5 26 72 89 CE 27-F3 06 82 01 CD 00 77 BC  ....This text do
0010      3C 1F 5A 62 CE 98 CE 27-F3 06 82 01 CD 00 77 BC  es not go with t

OK ====> D

0020      C3 F1 A5 26 72 89 CE 27-F3 06 82 01 CD 00 77 BC  he data on the l
0030      3C 1F 5A 62 CE 98 CE 27-F3 06 82 01 CD 00 77 BC  eft.............
0040      C3 F1 A5 26 72 89 CE 27-F3 06 82 01 CD 00 77 BC  .I once shot an.
0050      3C 1F 5A 62 CE 98 CE 27-F3 06 82 01 CD 00 77 BC  elephant in my.p
0060      C3 F1 A5 26 72 89 CE 27-F3 06 82 01 CD 00 77 BC  ajamas. What he.
0070      3C 1F 5A 62 CE 98 CE 27-F3 06 82 01 CD 00 77 BC  was doing in my.
0080      C3 F1 A5 26 72 89 CE 27-F3 06 82 01 CD 00 77 BC  pajamas I’ll nev
0090      3C 1F 5A 62 CE 98 CE 27-F3 06 82 01 CD 00 77 BC  er know.........
```

E: (Examine a single memory location)

Input 8-bit data from selected Z80 memory address and display it. This command reads from a single Z80 memory location, unlike the D command, and is useful for creating a repeated memory read operation with the Z and RL commands, so that target system signals can be examined with a non-storage oscilloscope.

Format:

E address	16-bit memory address entered as 1-4 hex digits.

F: (Fill memory)

Fills memory from start address to end with the value given.

Format:

F start,end,value	Addresses entered as 1-4 hex digits. Value is 1-2 hex digits. Same value is written to every location. No attempt is made to verify that the write succeeded.

G: (Go)

Go (Start Z80 CPU executing at full speed)

Format:

G [starting address]	Starting address entered as 1-4 hex digits. If no starting address is entered, current PC value is used.

I: (Input from I/O port)

Input 8-bit data from selected Z80 I/O port and display it

Format:

I address	8-bit I/O address entered as 1-2 hex digits.

L:	(List memory)

Lists (disassembles) memory contents from start address to end as Z80 instructions.

Format:

L [start],[end]	Addresses entered as 1-4 hex digits. Both parameters are optional. If no end address is entered, 19 instructions will be displayed. If no start address is entered, display begins at the next address after the end of the last D or L command. (This is the "saved memory address") So entering just L will start displaying from the end of the previous L command.

Example:

```
OK ===> l 0
0000 F3             DI
0001 31 F0 80       LD SP,80F0
0004 3E A5          LD A,A5
0006 32 00 80       LD (8000),A
0009 DD 21 00 80    LD IX,8000
000D DD 34 0A       INC (IX+0A)
0010 FD 21 02 80    LD IY,8002
0014 FD CB 08 86    RES 0,(IY+08)
0018 0E 80          LD C,80
001A ED 78          IN A,(C)
001C DD 36 10 69    LD (IX+10),69
0020 FD 22 06 80    LD (8006),IY
0024 DD CB 06 C6    SET 0,(IX+06)
0028 3E 00          LD A,00
002A CB DF          SET 3,A
002C 11 34 12       LD DE,1234
002F ED 53 0C 80    LD (800C),DE
0033 21 00 00       LD HL,0000
0036 DD 21 00 00    LD IX,0000

OK ===> l
003A 7E             LD A,(HL)
003B 4F             LD C,A
003C 06 00          LD B,00
003E DD 09          ADD IX,BC
0040 2C             INC L
0041 7D             LD A,L
0042 A7             AND A
0043 30 F5          JR NC,003A
0045 DD E5          PUSH IX
0047 DB 20          IN A,(20)
0049 D3 40          OUT (40),A
004B 76             HALT
004C C3 00 00       JP 0000
004F FF             RST 38
0050 FF             RST 38
0051 FF             RST 38
0052 FF             RST 38
0053 FF             RST 38
0054 FF             RST 38

OK ===>
```

M: (Move memory block)

Moves a block of memory data from one area of memory to another. The destination area must be writeable for the command to work, but no attempt is made to confirm that the write actually succeeded.

Format:

M start,end,dest	The block of data between start and end (inclusive) is copied to a new area of memory, starting at dest.

MD: (Memory detect)

Reads and writes 16 bytes at start of all 64 1K pages of target memory and tries to detect read-only memory and RAM. If data always reads the same despite writing several patterns, we assume its read-only memory. If data always matches what we wrote, we assume it's RAM. Data in each 1K page is compared with data in lower 1K pages to detect stuck or unconnected upper address bits or the same memory appearing several times in the memory space. Page number displayed is lowest 1K page in target memory that contains the same data as this one. If no memory repeats, then each page will display its own page number. Note that if the bus always floats to the same value, or is terminated to a logic one level when floating, the data will always read the same, so it will be detected as read-only memory. Note: Executing this command will write to the first 16 bytes of every page, so memory contents will be changed.

Format:

MD

Example:

```
0000000000000000111111111111111122222222222222223333333333333333
0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
RRRRRRRR                        RRRRRRRR                RRRRRRRR
OOOOOOOO                        OOOOOOOO                WWWWWWWW
00000000                        00000000                33333333
01234567                        01234567                89ABCDEF
```

Above display shows 8K Read-only memory located in first eight 1K pages, and the same memory appears at 32K as well. An 8K RAM appears only in the top 8K pages.

```
0000000000000000111111111111111122222222222222223333333333333333
0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWOOOOOOOO
0000000000000000111111111111111122222222222222223333333333333333
0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
```

Above display shows 8K Read-only memory located in top eight 1K pages. The rest of the address space is filled with RAM that is fully decoded. (Does not repeat)

MT: (Memory test)

Performs a destructive test of memory by writing a rotating pattern of first a single 1 bit, then a single 0 bit and the rest ones to each memory location in the indicated range. After the pattern is written to the entire area, all locations are checked to see if the value read matches what was written. Any mismatches are displayed as address, value written, value read. The entire memory range is written and read 16 times, so any particular address may appear in the error list up to 16 times, if it fails for all data values.

Format:

```
MT start,end
```

O: (Output to I/O port)

Write 8-bit data entered to selected Z80 I/O port

Format:

O port #,d8	8-bit I/O port number/address entered as 1-2 hex digits.

P: (Put (write) data into a target system memory location)

Write 8-bit data entered to selected Z80 memory location. This command writes to a single memory location, and is useful for creating a repeated memory write operation with the Z and RL commands, so that target system signals can be examined with a non-storage oscilloscope.

Format:

P address,d8	16-bit memory address entered as 1-4 hex digits.

R: (Read Intel Hex file into memory)

Used to load target system memory with test programs, etc, saved in Intel Hex format. An optional 16-bit offset may be entered. This offset will be added to the addresses in the Hex file before storing the file data in memory. This is useful if the code being loaded is relocatable. If no offset is entered, an offset of 0 is used. (No offset)

The R command can process data and store it in the target system's memory at up to 19.2 K baud, if the target system is running at 1 MHz or higher. If a slower Z80 clock is used, higher baud rates can cause a data overrun condition, which will cause an error message to be displayed. (Using hardware handshake won't fix this, as the PC takes too long to react to CTS going inactive.) In this case, simply switch to a slower baud rate.

Format:

```
R [d16]
```

S: (Substitute into memory)

Used to load hex data values into system memory. No attempt is made to determine if write to memory was successful, so you can substitute into EPROM, but it won't accomplish much.

Format:

S d16	To advance to next address without changing contents, hit CR. To back up to previous address, hit a dash (-). To exit substitute memory mode, hit a period (.).

SR: (Soft reset)

Causes the ICE to reset all of the Z80’s registers to zeroes.

Format:

```
SR
```

T: (Trace)

This command may be used to step through target system code one instruction at a time. The Z80's internal registers are displayed after each instruction. An optional parameter may be entered to specify a number of instructions to be executed. Entering T 10 will execute 10 instructions and then stop. If an enabled breakpoint is reached before 10 instructions have executed, the trace will stop immediately. If the number of instructions to execute is entered as 0, the trace instruction will continue until a breakpoint is hit, and will not count instructions. If no parameter is entered, only one instruction will be executed.

If the Z80 encounters a Halt instruction when tracing, it will stop fetching opcodes and will no longer be controllable by the ICE. A message will be displayed in this case, and the system will need to be reset to restore normal operation.

Format:

```
T [d16]
```

U: (“Untrace”)

This command is similar to the trace command, but the registers are not displayed after every instruction, so execution is much faster. The registers may be displayed when specific PC values are hit using printpoints. Untracing will stop if the requested number of instructions has been executed, or if an enabled breakpoint has been reached. A parameter must be entered to specify a number of instructions to be executed. Entering U 10 will execute 10 instructions and then stop and display the registers. If an enabled breakpoint is reached before 10 instructions have executed, the untrace will stop immediately. If the number of instructions to execute is entered as 0, the untrace instruction will continue until a breakpoint is hit, and will not stop based on instruction count. When the untrace command stops, the total number of instructions executed during this one command is displayed. This could be used to count, for instance, how may Z80 instructions are executed between system reset and a prompt appearing. Ctl-C can also be used to stop the untrace command.

If the Z80 encounters a Halt instruction when untracing, it will stop fetching opcodes and will no longer be controllable by the ICE. A message will be displayed in this case, and the system will need to be reset to restore normal operation.

During an untrace command, the Z80’s PC is monitored and when instructions that modify it, such as branches, calls, and returns are executed, their address is saved in a 12-entry FIFO buffer. When the untrace command stops, these instructions are displayed above the CPU registers, and can be used to trace backwards from the current PC value to see what the recent sequence of calls or branches was before we stoppped. A maximum of 12 of these instructions will be displayed. If fewer than 12 were executed during the current untrace command, only that number will be displayed.

Format:

```
U d16
```

V (Verify/compare memory)

This command will verify/compare two regions of memory and display any differences found. Differences will be displayed as: address in first region, value in first region, value in second region.

Format:

V start,end,dest	Parameters are d16. All bytes from start to end (inclusive) will be compared.

W (Write memory out as Intel Hex file)

Used to save target system EPROM images, etc, in Intel Hex format. A 16-bit offset must be entered. This offset will be added to the addresses in the Hex file before storing the file data in memory. This may be useful if the code is located at an address other than 0, but you want the Intel Hex file addresses to start at 0 for loading into an EPROM programmer, for instance. Offset addition will wrap at 64K, so 0xF000 + 0x1000 = 0x0000. If you don't want any offset, just enter 0.

Format:

W start,end,offset	Parameters are d16. All bytes from start to end (inclusive) will be written. Start must be a lower address than end.

X (Examine and edit the contents of the Z80's registers)

Used to display or change Z80 register contents. Valid register selection arguments are: F, A, B, D, H, S, P, F', A', B', D', H', X, and Y. HL, BC, and DE must be edited as a 16-bit pair.

Format:

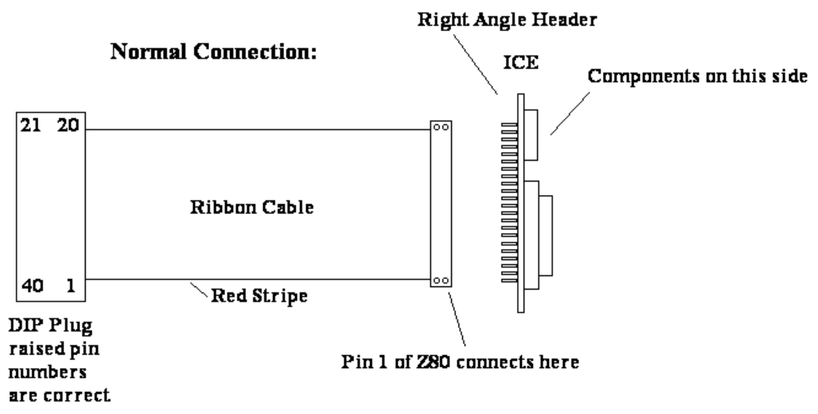
X [regsel] If no register selection argument is enterred, the complete register set will be displayed

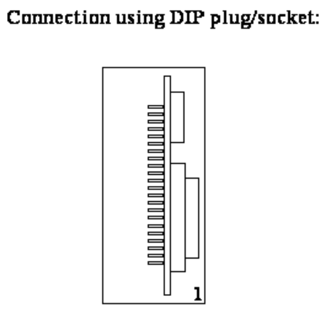
Example:

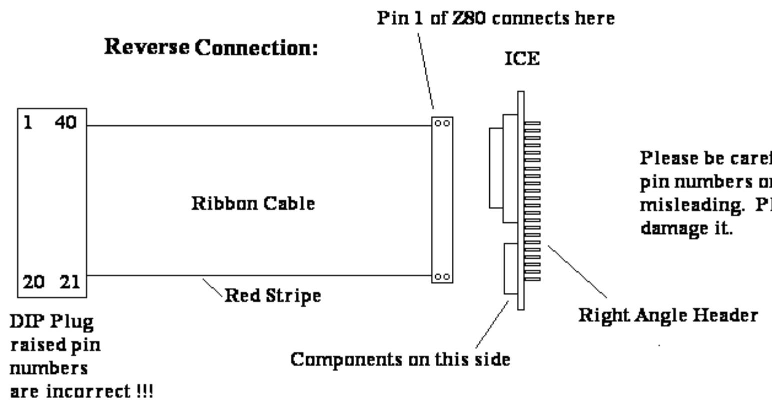
```
OK ====> x

.Z.V..   A=00  BC=1234  DE=F74C  HL=5542  S=FFF0  P=824D  LD A,L
S....C  A'=4C  B'=1A64  D'=F29C  H'=0000  X=FFF0  Y=824D  I=00

OK ====>
```

Chapter 7
Cabling Information:

Two ways to connect ribbon cable, depending on how you want ribbon cable to face on target system:







Please be careful when using this connection as the pin numbers on the DIP plug and the red stripe will be misleading. Plugging the ICE in incorrectly will probably damage it.

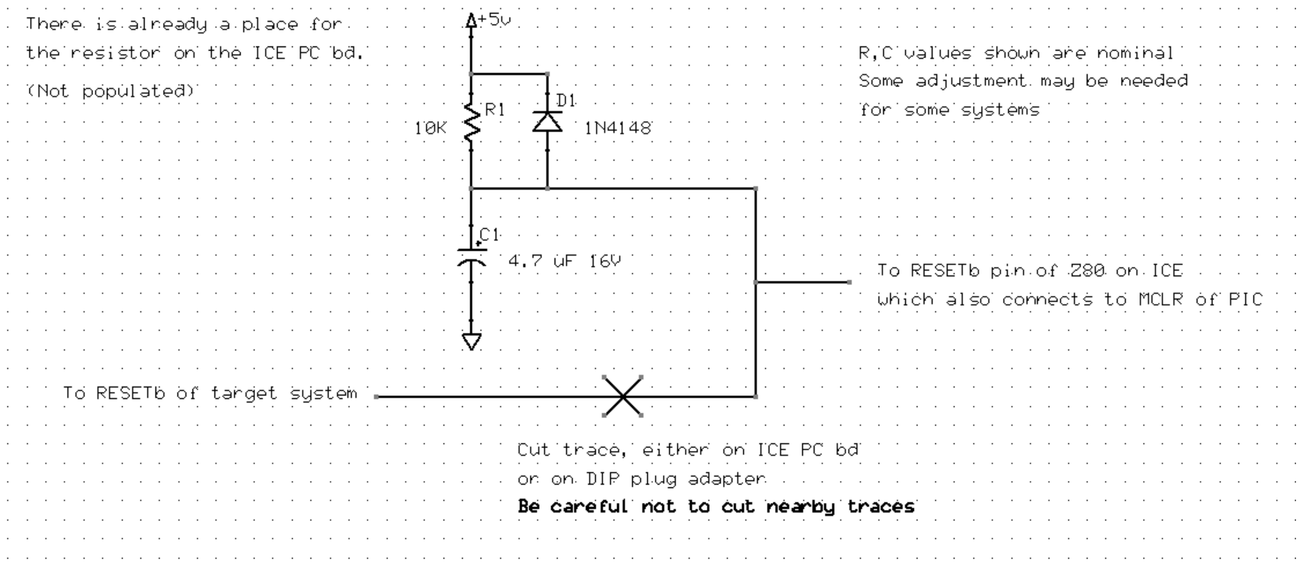
Chapter 8
How does it work?

The ICE consists of a 6 MHz Z80 CPU, a CPLD custom logic chip, an RS-232 level translator/charge pump chip, and a PIC microcontroller. The PIC chip communicates with the terminal through the RS-232 chip and controls the Z80. It is capable of reading Z80 register values, loading the Z80’s registers, single-stepping the Z80 through code, starting the Z80 running at full speed, and stopping the Z80. The CPLD chip contains logic to allow the PIC to control the Z80 and also to inhibit the bus request and interrupt signals from reaching the Z80. It can also prevent the Z80’s MREQ, IOREQ, RD, and WR signals from reaching the target system. In Quit mode, these four signals are forced to their inactive state in the target system, to insure that the target system is not using the data bus. This is necessary as the data bus is used for PIC-to-Z80 communication in this mode. In Go mode, the PIC chip starts up the Z80 and lets it run. In this mode, the ICE operates just like a normal Z80 chip. In Quit mode, the Z80 has been paused, and can be used to implement the commands that give the ICE its power. All communication with the target system’s memory takes place through the Z80. To read data from a memory location, the PIC arranges for the Z80 to read from that location, and then hand the data to the PIC. If the Z80 is halted, it is not fetching op codes, and cannot be controlled by the PIC chip. This condition is detected and a message is displayed. The system must be reset to restore normal operation.

Chapter 9
Watchdog Timer - Z80 Reset Issue

Some Z80-based arcade machines and pinball machines have a circuit called a "watchdog timer". When code is executing normally, the Z80 writes to or reads from a particular I/O or memory address maybe a few times each second. This action resets the timer and maintains normal operation. If the CPU crashes, it stops resetting the timer. As a result, the timer reaches a preset count, which causes the RESETb input to the Z80 to go low, resetting the CPU and restoring normal operation of the system. If an ICE is used, since the Z80 is paused a lot of the time, the watchdog timer will not get reset, and will keep resetting the Z80 (and the PIC chip), perhaps five times/second or faster. The ICE can't work if it keeps getting reset. There are two ways that the ICE can be used in systems with a watchdog timer:

1) Disable the watchdog timer. Some arcade games have a jumper on the main board that can be used for this purpose. You may need to solder across two pads temporarily to disable the timer. Other systems may have a different way to disable the timer (or no way at all).
2) Modify the ICE to supply its own RESETb signal to the Z80 and the PIC chip. If you do this, you'll need to cycle power to the ICE to reset it, since any target system reset switch will no longer function. You can add a few components to the ICE PC board to implement a RESETb circuit. Here is a schematic showing the modification needed:


There is already a place for
the resistor on the ICE PC bd.
(Not populated)
+5v
R1
10K
D1
1N4148
C1
4.7 uF 16V
R,C values shown are nominal
Some adjustment may be needed
for some systems
To RESETb pin of Z80 on ICE
which also connects to MCLR of PIC
To RESETb of target system
Cut trace, either on ICE PC bd
or on DIP plug adapter
Be careful not to cut nearby traces

Chapter 10
Code with a Halt Instruction

The PIC controls the Z80 by feeding it opcodes. When the Z80 executes a Halt instruction, it stops fetching opcodes and waits for an interrupt or reset. Some code may normally sit in a halted state waiting for an interrupt, then perform some action such as checking for a key press, serial character received, etc, then after any required tasks have been performed, execute a halt and wait for the next interrupt. The ICE will not work properly when executing this type of code, as the Z80 stops fetching instuctions when halted, and cannot be controlled by the PIC. The ICE can still be used in this type of system, but the code containing the Halt instruction cannot be executed. To avoid executing the Halt instruction as soon as the system is powered up, disable "autorun" mode, so that the Z80 will come up paused after reset. To do this, reset the system, hit <Enter> to get the sign-on message and prompt, then type

```
  DA <Enter>
```

This will disable autorun mode. (This setting is saved in EEPROM.) Now reset the system, and hit <Enter> to get the sign-on message and prompt. Now you should be able to use the ICE to test memory, access I/O ports, etc, but don't execute the code that includes the Halt instruction, as the ICE won't work after that.

Chapter 11
Differences between this unit and Nicolet's NICE

New commands not present in NICE:

CF - Measure clock frequency
CS - form 16-bit checksum of memory block
E - Examine and display a single memory location, for scope loops
MD - Try to detect RAM and EPROM in target memory space
P - Put data into a single memory location, for scope loops
W - Write out memory as Intel Hex
DN, EN - separate hardware control of NMI input
EA, DA - control start-up behavior (Go or Quit)
? - display help screen with command list

Other differences:

L command displays hex data bytes along with instructions

L command resolves branch target addresses

L command handles undocumented Z80 opcodes

M command checks for overlap between source and destination regions and changes move order if nec.

U command displays branch trace instead of the last few PC values as "backtrace"

U and T commands check for # of instructions = 0, which disables instruction count limit.

U command displays total instructions executed since start of command.

Z80 Halt output is monitored and a message is displayed if CPU is halted.

Z80 operation at much lower frequencies is supported because we measure the Z80 clock and scale our timings accordingly. The NICE does not operate properly at 500 KHz and below.

IC sockets enable using different Z80 chips, or replacing PIC or RS-232 chip if damaged.

RS-232 level translator chip is used to provide better voltage levels to PC or terminal. NICE uses 5V and Gnd.

R command (read Intel Hex) can operate at higher baud rates. The manual for the NICE unit implies that a maximum of 300 baud can be supported without hardware handshaking. I haven't seen any problems loading files at much higher baud rates, with no hardware handshake.

Operating current - NICE draws about 500 mA from the Z80 socket. Our unit draws about 70 mA with a CMOS Z80 installed, and about 130-200 mA with an NMOS Z80 installed. The PALs and MCU used in the NICE draw much more current than the CPLD and our PIC chip.

NICE supports assembly of Z80 instructions into RAM. (I don't) This is a nice feature, (no pun intended) but a lot of work, which I couldn't justify. It didn't seem useful enough to warrant the effort.