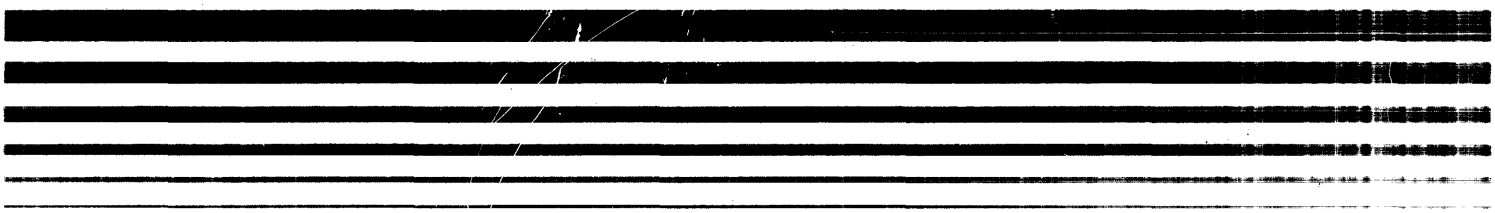
NICE® Z80+

In-Circuit Emulator for the Z80® Microprocessor Operation Manual



Nicolet

NICE Z80+

User's Guide and Reference Manual
for the Z80 Microprocessor

May 1986

Test Instruments Division
215 Fourier Avenue
Fremont, CA 94539

125-0061-0001 Rev. B

COPYRIGHT

LIMITED WARRANTY

Nicolet Test Instruments Division guarantees your emulator against all defects in materials and workmanship for a period of ninety (90) days from the date of delivery to the original user. All instruments returned to Nicolet Test Instruments Division FREIGHT PREPAID during the ninety day period will be replaced or restored to proper working condition and returned to the sender without charge. We neither assume nor authorize any representative or other person to assume for us any other liability in connection with the sale on any shipment of our products.

NOTE

> This warranty does not apply to damage caused by shipping, negligence, accident, unauthorized repair, abuse, misuse, or modification; or to inconveniences or consequential damages occasioned by the instrument, or by breach of any expressed or implied warranty with respect thereto. Further, no agreement extending or modifying this warranty in any way whatsoever will be binding upon unless executed by a duly authorized officer of the company.

This warranty gives you specific legal rights; you may also have other rights which depend on local jurisdiction.

We reserve the right to make changes and improvements in our products without incurring any obligation to similarly alter products previously purchased.

OUT OF WARRANTY REPAIRS

A flat repair cost will be charged for an out-of-warranty product that is returned to Nicolet Test Instruments Division. This price includes return surface freight charges and necessary component replacement.

Method of payment (included with emulator): check, credit card, COD or purchase order (approved by Nicolet Test Instruments Division).

If a bank card is used, the following information is required:

- Card type (ie. VISA or Mastercard)
- Name as shown on card
- Credit card number
- Expiration date

If requesting COD repair, please so state when returning the unit.

U.S. SERVICE CENTERS

Nicolet Test Instruments Division
5225-2 Verona Road
Madison, WI 53711-0288
Tel: 608/273-5008
Twx: 910/286-2737 (NICOLET MDS C)
Fax: 608/273-5061

Nicolet Test Instruments Division
215 Fourier Avenue
Fremont, CA 94539
Tel: 415/490-8870
Fax: 415/490-8063

Nicolet Test Instruments Division
12 New England Executive Park
Burlington, MA 01803
Tel: 617/273-4404

Nicolet Test Instruments Division
2181 Northlake Parkway, Suite 118
Tucker, GA 30084
Tel: 404/491-0127

OVERSEAS:

Contact your local distributor for the address of Nicolet Service Center nearest you.

NICE Z80+ MANUAL
TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION	1
About NICE, The Z80+ Emulator	1
Who Uses It	2
Basic Features	2
Functional Capabilities	3
Command Line Interpreter Characteristics	4
CHAPTER 2 SET-UP	5
Terminal	5
Communications Interface	6
Installation	7
CHAPTER 3 COMMAND LINE INTERPRETER	9
Command Format	9
Command Parameters	9
Command Line Prompts	9
Multiple Commands on One Line	10
Control Character Functions	10
Repeat Line Command	11
Computer Interface to the Command Line Interpreter	11
CHAPTER 4 OVERVIEW OF OPERATIONS	13
GO Mode vs QUIT Mode	13
Special Facts About NICE	13
CHAPTER 5 GO MODE COMMANDS	17
SBP - Breakpoint	18
SBPC - Breakpoint Count	19
SEBP - Enable Breakpoint	21
SDBP - Disable Breakpoint	22
SC - Clear ALL Software Breakpoints & Reset Pass Counters	23
SEPP - Enable Printpoint	24
SDPP - Disable Printpoint	25

NICE Z80+ OPERATOR'S MANUAL
TABLE OF CONTENTS
(continued)

CHAPTER 5 GO MODE COMMANDS (continued)

EI	-	Enable Interrupts	26
DI	-	Disable Interrupts	27
EB	-	Enable Bus	28
DB	-	Disable Bus	28
DR	-	Disable Refresh	29
ER	-	Enable Refresh	30
H	-	Hexadecimal Arithmetic	31
Q	-	Quit	32
RL	-	Repeat Line	33
STS	-	Status	34
Z	-	Sleep	36

CHAPTER 6 QUIT MODE COMMANDS 37

A	-	Assemble into RAM	38
D	-	Display Memory	40
E	-	Examine Input Port	42
F	-	Fill	43
G	-	Go	44
L	-	List in Assembler Format	45
M	-	Move	47
MT	-	Memory Test	48
O	-	Output	50
R	-	Read Intel Hex File	51
S	-	Substitute into Memory	56
SR	-	Soft Reset	56
T	-	Trace	57
U	-	Untrace	59
UP	-	Upload	61
V	-	Verify	62
X	-	Xamine	63
EOR	-	Enable Overlay RAM	65
DOR	-	Disable Overlay RAM	67

CHAPTER 6 QUIT MODE COMMANDS (continued)

PERFORMANCE MONITORING 68
INDIVIDUAL BUCKET ASSIGNMENTS 70
EBP - Enable Breakpoint 72
DBP - Disable Breakpoints 73
EPP - Enable Printpoints 74

APPENDIX A- QUICK REFERENCE COMMAND LIST A-1

APPENDIX B- INTEL HEX FORMAT B-1

APPENDIX C- TARGET SYSTEM MODIFICATIONS C-1

APPENDIX D- SELF TEST FUNCTIONS D-1

APPENDIX E- SAMPLE DOWNLOAD PROGRAM USING THE R COMMAND E-1

APPENDIX F- MECHANICAL SPECIFICATIONS F-1

CHAPTER 1
INTRODUCTION

ABOUT NICE, THE Z80+ EMULATOR

The NICE Z80+ begins a new generation of medium-to-high function emulators. Its revolutionary compact design provides the following benefits:

- **Reduction in Price.** NICE's low price finally brings In-circuit emulators out of the exclusive domain of large-scale operations and into the reach of computer repair shops and hobbyists.

- **Transportability.** NICE is only 3½" X 5½" and 1" thick. Not only can it be used in development labs, it can also be a part of the computer technician's portable repair kit. An RS232 compatible interface allows NICE to hook up to most terminals and modems for speedy diagnoses.

- **Full Speed Emulation with Minimal Target Disturbance.** NICE's compact design means the electronics are closer to the target system than previous generation emulators. The result is full speed emulation with minimal target disturbance. The NICE Z80+ includes a Molex connector to tap into an external power supply with a requirement of approximately 1A @ 5.2V.

- **Ease of Operation.** NICE's streamlined operation is consistent with its streamlined design. All it takes to get started is replacing the Z80 microprocessor with NICE (either directly or via the 40-pin cable assembly), connecting the terminal to NICE, resetting the target system, and hitting a carriage return.

WHO USES IT?

NICE is designed to meet the needs of four groups:

- Designers profit from NICE's ability to aid in debugging both hardware and software. Emulators provide a cost-effective means of integrating hardware and software.

- New Product Manufacturers can use NICE to pinpoint potential problems before beta testing. During the manufacturing, NICE can be used to bring up virgin CPU cards, then to download and run diagnostics. A more trouble-free product leads to greater customer satisfaction.

- Computer Repair Technicians can bring NICE along on on-site repair calls, possibly eliminating the need for in-shop repair. Smaller computer repair shops can finally afford an emulator because of NICE's low cost.

- Serious hobbyists can even use NICE. Its low cost and high versatility combine to make it a valuable tool for debugging home systems and designing custom hardware or software.

BASIC FEATURES

Despite its small size, NICE incorporates most of the features of the less portable and more expensive emulators.

- Full speed execution up to 8MHZ
- Refresh function maintained at all times
- All I/O ports available to user
- All memory addresses available to user
- Three software breakpoints with 8-bit loop counters
- Three software printpoints with 8-bit loop counters
- Sixteen real-time hardware breakpoints

- o Sixteen hardware printpoints
- o Eight Kbytes of overlay RAM hardware
- o Interface to user via standard 25-pin RS232 terminal connector
- o Local or remote operation
- o Automatic baud rate detection
- o Small size
- o Low cost
- o High reliability

FUNCTIONAL CAPABILITIES

NICE can perform the following:

- o Execute real time (hardware) breakpoints
- o Operate from overlay RAM
- o Perform user-definable histogramming
- o Display target system memory in hexadecimal and ASCII format
- o Display and modify any memory location in target system RAM
- o Display and/or modify any Z80 internal register
- o Examine any I/O port
- o Output single or multiple bytes to any I/O port
- o Perform hexidecimal arithmetic
- o Fill a block of RAM with a constant
- o Compare one block of memory to another
- o Test target system RAM
- o Move a block of memory from one location to another
- o Read and load an Intel Hex File into target system RAM
- o Trace and display all instructions
- o Trace and display only specified instructions
- o Upload Intel HEX File to Host
- o Disassemble memory into Z80 mnemonics
- o Assemble Z80 mnemonics into memory
- o Enable/Disable Z80 interrupts in hardware
- o Enable/Disable Z80 bus request in hardware
- o Enable/Disable Z80 refresh function

COMMAND LINE INTERPRETER CHARACTERISTICS

NICE's versatile command line interpreter allows you to perform the following:

- Enter one or more commands on the same line
- Enter a "Sleep" command to delay command execution
- Enter a "Repeat Line" command to repeat execution of the command line
- Erase the previous character on the command line
- Erase the entire command line

Additionally, a printout can be halted, restarted, or aborted.

CHAPTER 2
SET-UP

There are two aspects to setting up NICE for Z80 emulation:

1. Setting up the terminal,
2. Establishing the communications interface and installing NICE into the target system.

TERMINAL

Auto Baud Rate Detection

NICE is equipped with an automatic baud rate detection algorithm that is invoked whenever NICE is powered up. To determine the proper baud rate, NICE measures the length of the first start bit transmitted from the terminal. To start automatic baud rate detection, enter a carriage return following power up. After the baud rate has been determined, you will see the NICE prompt which, is a greater than sign (>). At this point, you may begin entering commands.

To work with NICE, the terminal must be set to one of the following baud rates:

- o 150
- o 300
- o 600
- o 1200
- o 2400
- o 4800
- o 9600
- o 19.2 k

Terminal Characteristics

To operate with NICE, the terminal must be set up with the following characteristics:

- o Full duplex operation
- o Auto line feed on; carriage return disabled
- o Line terminator set to either carriage return or line feed
- o Destructive space enabled

- o 8 bits of data
- o Parity disabled
- o 2 stop bits

COMMUNICATIONS INTERFACE

RS232 Considerations

NICE uses **RS232** for communications. RS232 is a standard serial asynchronous protocol. However, NICE uses only those signals defined in the RS232 specifications that are necessary for its operation. While the typical voltage levels for RS232 are +12 and -12 volts, NICE uses +3 and -3volts for the corresponding levels.

WARNING

The +12 and -12 volt signals from the terminal are clamped to +5 and 0 volts by diodes inside NICE. Since excessive current could damage the clamping diodes and/or custom circuits, be sure that you only use RS232 compatible devices with NICE. And **never** connect NICE to a device capable of supplying or sinking more than 15 mA of current.

Setting up the Communications Connector

Before you configure the pins on the communication cable, determine whether you will be connecting via the Data Terminal End (DTE) or the Data Computer End (DCE). The DTE connection is the one most commonly used with NICE. Therefore, the cable provided is wired for connection to a terminal.

The following list provides the pin numbers, signal names and functions for the communication cable connector provided with NICE, as well. To connect NICE directly to a computer, you must rewire the connector to conform to the pin numbers in parentheses.

PIN 3 (PIN 2) - **Received Data,** sent from NICE to the CRT terminal.

PIN 2 (PIN 3) - **Transmitted Data,** sent to NICE by the CRT terminal.

PIN 5 (PIN 4) - **Clear to Send.** This signal is sent by NICE to the terminal. A high signal (+5V) tells the terminal that NICE is ready to accept data. Use of PIN 5 ensures that the terminal will not transmit at a rate faster than NICE can accept.

PIN 20 (PIN 6) - **Data Terminal Ready.** This signal is sent to NICE by the terminal. A high signal tells NICE that the terminal is ready to accept data. Use of PIN 20 ensures that NICE will not transmit data faster than the terminal can accept it.

PIN 6 (PIN 20) - **Data Set Ready.** This signal is sent by NICE to the terminal. A high signal (+5V) tells the terminal that NICE has been installed and power has been applied to the target system.

PIN 7 (PIN 7) - **Ground,** is the return path for the previous signals.

INSTALLATION

To install NICE:

1. Remove the Z80 microprocessor from the target system.

2. Connect NICE, using either the pin plug provided or the short 40 pin cable connector.

    - Whenever possible, use the pin plug. The pin plug reduces signal noise. Use the 40 pin cable only if you cannot physically attach NICE to the target system.

    - Be sure that pin 1 of the target system is connected to pin 1 of NICE. Otherwise, you may damage NICE's custom circuits.

3. Attach the terminal to NICE using the communication cable provided.

4. Users may wish to use their own power supply; the pin-out from the emulator is as follows. Numbering from right to left as shown in Figure 1 below.

Pin 1	+5.2V
Pin 2	Ground
Pin 3	Sync Out (Positive True)
Pin 4	External Break In (Negative True)
Pin 5	Ground
Pin 6	+5.2V

NB: Pin 4 (Input) A negative true logic signal applied to this pin will cause a hardware (real time) breakpoint to be performed.

Pin 3 (output) This signal is active 95nS (worst case) following detection of the hardware breakpoint or printpoint address. May be used to initiate or trigger external hardware such as: oscilloscopes, logic analyzers, etc.

5. Ensure terminal characteristics are set correctly (see Page 5) and apply power to the system in the following order. First the TERMINAL next the EMULATOR finally the TARGET SYSTEM.

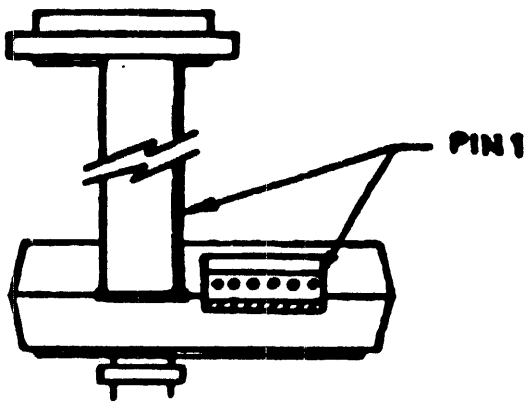


Figure 1 Molex Pin Orientation.

CHAPTER 3
COMMAND LINE INTERPRETER

COMMAND FORMAT

Spaces -- NICE is controlled by simple one- to three-character commands entered at the terminal. Spaces are ignored, with one exception: a space must follow a command when the command is used with a parameter. Character data may be entered either in upper or lower case.

Multiple Parameters -- When there are multiple parameters in a command, the parameters must be separated either by commas or by spaces.

Length -- The command line can contain a maximum of 31 characters. The terminal emits a beep once you have reached the 31-character limit. Any additional characters typed after the beep replace the last character on the line.

Execution -- NICE executes the commands on the command line when it receives a terminator character (Line Feed or Carriage Return) from the terminal.

COMMAND PARAMETERS

Certain commands require either alphabetic or numeric parameters. Numeric parameters must be entered in hexadecimal form, they can be either an 8 bit or 16 bit value, depending on the command.

COMMAND LINE PROMPTS

NICE displays two different prompts depending on the success of the previous command.

- The OK prompt (>) indicates the previous command was executed properly and that NICE is awaiting the next command.

- The ERR prompt (<) indicates the previous command was either entered incorrectly or its execution terminated abnormally. The ERR prompt is

displayed in conjunction with an audible warning. A new command may be entered following the arrow.

MULTIPLE COMMANDS ON ONE LINE

NICE will execute several commands entered on the same line. Each command must be separated by a semicolon.

For example, the following command

D E000; F 8000, 8800, AB

causes NICE to execute the Display memory command followed by the Fill memory command.

CONTROL CHARACTER FUNCTIONS

Four control codes are used to aid in entering commands and control printout. To enter a control character, hold down the CTRL key as you hit the indicated letter.

Table 3-1
Control Characters and Their Functions

Character	Function
Ctrl-H	Performs the same function as the backspace key: Backspaces and deletes the previous character.
Ctrl-U	Erases all the previous characters on the command line and positions the cursor at the beginning of the command line.
Ctrl-S	Starts or stops printout at the terminal. If printing, Ctrl-S stops the printing. If printing is already stopped, Ctrl-S or any other character causes printing to resume.
Ctrl-C	Aborts a printout or terminates a Repeat Line Command. For printouts, the abort takes place at the end of the next line of text.

REPEAT LINE COMMAND

The Repeat Line Command (RL) instructs NICE to repeat the command(s) on that line over and over again.

For example, the following command

D E000; F 8000, 8800, AB; RL

causes NICE to execute the Display memory command and Fill memory command over and over.

NICE only ceases executing the commands on the command line when it receives a Ctrl-C from the terminal or when the target system is reset.

COMPUTER INTERFACE TO THE COMMAND LINE INTERPRETER

When you communicate with NICE using a computer instead of a terminal, NICE's response to ASCII characters is not always obvious. The responses which cannot be easily determined by viewing a CRT are listed below:

- Non-control characters are echoed exactly as received.
- Control characters are not echoed.
- Lower-case characters are converted to upper-case.
- You can terminate a command line with either a Line Feed or a Carriage Return.
- A Line Feed followed by a Carriage Return signifies that NICE is sending a new line.
- The response to a backspace (Ctrl-H) is the sequence Ctrl-H, space , Ctrl-H. (On a terminal, this sequence deletes one character.)
- A Ctrl-U causes NICE to respond with the sequence Ctrl-H, space, Ctrl-H the number of times necessary to backspace and delete the entire line.

CHAPTER 4
OVERVIEW OF OPERATIONS

GO MODE VS QUIT MODE

NICE has two different states of operation.

In **GO mode,** NICE executes Z80 instructions at full speed, but only obeys a subset of its full repertoire of commands. NICE automatically enters the GO mode when you reset the target system. In **QUIT mode,** NICE obeys the full set of commands from the terminal, but it cannot execute instructions at full speed.

Chapters 5 and 6 contain valid GO mode and QUIT mode commands.

When you are not certain of NICE's current mode, you can issue the status command (STS M) from either mode. If the bottom line of the display includes the word "RUNNING," NICE is in the GO mode. Alternatively, depress the "return" key if a trancated version of the command set is displayed. The Z80 is then in the RUN mode.

SPECIAL FACTS ABOUT NICE

Because of NICE's compact new design, it behaves somewhat differently from larger emulators.

Interrupts

NICE has the capability of recognizing Z80 interrupts in the GO mode, but not in the QUIT mode. In GO mode, interrupts can be enabled or disabled from reaching the Z80 Microprocessor.

Bus Requests

NICE only recognizes Z80 Bus Requests when in the GO mode. In the GO mode, you can use one of the two GO mode commands to enable or disable Bus Requests to reach the Z80 Microprocessor. When NICE is in QUIT mode, Bus Requests are not allowed to reach the Z80 Microprocessor.

NOTE

If you use NICE in a system that requires continuous access to the memory bus (such as a CRT controller), you should expect an under run condition when NICE enters the QUIT mode.

Memory Refresh

When NICE is in the QUIT mode, the Z80 control lines are cycled so that systems containing dynamic RAM will not lose data. The Refresh function is always enabled following power up.

NOTE

If the target system does not have dynamic RAM, it is a good idea to disable the Refresh function. This allows NICE to operate 25 percent faster.

Memory Requests

When NICE is in the GO mode, memory requests are identical to Z80 memory requests. When NICE is in the QUIT mode, however, requests are extended.

NOTE

The lines which control memory access with NICE in the target system are active for a longer period of time than they would be ordinarily. Therefore, during a memory read, be sure that data from the target system RAM remains valid for the entire period as indicated by the Z80 control lines.

I/O Requests

When NICE is in the QUIT mode, I/O control signals are extended beyond ordinary duration. As with memory requests, then, be sure that I/O read data remains valid for the duration as indicated by the Z80 control lines.

Power Up Status

After power up, the following occur:

- NICE comes up in the GO mode, appearing to the target system as if it were a Z80 microprocessor.
- The interrupt request and bus request lines are enabled.
- All break point and print enable flags are cleared.
- Saved memory address is set to zero.
- Full speed execution starts at location zero.

Then, once NICE receives a carriage return from the terminal and sets its internal baud rate, NICE enters the command line interpreter.

CHAPTER 5
GO MODE COMMANDS

In the QUIT mode, NICE recognizes all commands. In the GO mode, however, NICE only recognizes a subset of these commands. The commands that NICE recognizes in the GO mode follow:

Page #			
18	SBP	-	Breakpoint
19	SBPC	-	Breakpoint Count
21	SEBP	-	Enable Breakpoint
22	SDBP	-	Disable Breakpoint
23	SC	-	Clear Software Breakpoints
24	SEPP	-	Enable Printpoint
25	SDPP	-	Disable Printpoint
26	EI	-	Enable Interrupts
27	DI	-	Disable Interrupts
28	EB	-	Enable Bus
28	DB	-	Disable Bus
29	DR	-	Disable Refresh
30	ER	-	Enable Refresh
31	H	-	Hexadecimal Arithmetic
32	Q	-	Quit
33	RL	-	Repeat Line
34	STS	-	Status
36	Z	-	Sleep
	?	-	Request Information on Defined Command

SBP - BREAKPOINT

Purpose

Sets any of the three breakpoint addresses that work with either the Trace or Untrace commands. **Software breakpoints only work in QUIT mode.**

Format

SBP Breakpoint-Number, Breakpoint Address

Breakpoint Number must be either 1, 2, or 3, corresponding to the desired break point. Breakpoint-Address is a 16-bit value which NICE saves for later comparison to the Z80 program counter during Trace and Untrace commands.

Examples

The following examples show how to set breakpoint addresses and use the Status (STS) command to display the results.

```
  ? SBP

SBP x,yyyy
   Set software breakpoint x (1-3) at address yyyy.


  > SBP 1,1800

  > SBP 2,1890

  > SBP 3,2000

  > STS

   ---> 1800 00 D
   ---> 1890 00 D
   ---> 2000 00 D
   IBR  RUNNING

  Overlay RAM is * DISABLED * at 0000-1FFF R/W
```

SBPC - BREAKPOINT COUNT

Purpose

Specifies the number of passes NICE makes before stopping at the selected break point.

NOTE

> Each of the three breakpoints has its own one-byte pass counter. Specifying a pass count allows NICE to trace a program past a particular address more than once before terminating with a breakpoint.

After NICE traces each instruction, it compares the enabled breakpoint addresses to the program counter. If a match is found, NICE determines whether the pass counter is zero.

- If the pass counter is zero, NICE stops tracing and sends the register display to the terminal.

- If the pass counter does not read zero, then it is decremented and NICE continues tracing.

Pass counters can also be used to determine the number of times a program passed a certain address. (NICE retains the decremented values even after it stops tracing.)

Format

SBPC Breakpoint-Number, Pass-Count

The Breakpoint Number must be either 1, 2, or 3. The Pass-Count is a one-byte value.

Examples

```
> ? SBPC

SBPC x,yy
   Change software breakpoint pass counter x (1-3) to yy.

> SBPC 1,9

> SBPC 2,20

> SBPC 3,7

> STS

 ---> 1800 09 D
 ---> 1890 20 D
 ---> 2000 07 D
 IBR  RUNNING

Overlay RAM is * DISABLED * at 0000-1FFF R/W
```

SEBP - ENABLE BREAKPOINT

Purpose

Enables one or all three breakpoints.

Format

This command has two forms:

1. **SEBP** - Enables all three breakpoints.

2. **SEBP Breakpoint-Number** - Enables breakpoint 1, 2, or 3, as specified.

Examples

```
> SEBP 3

> STS

 ---> 1800 09 D
 ---> 1890 20 D
 ---> 2000 07 E
 IBR  RUNNING

Overlay RAM is * DISABLED * at 0000-1FFF R/W

> SEBP

> STS

 ---> 1800 09 E
 ---> 1890 20 E
 ---> 2000 07 E
 IBR  RUNNING

Overlay RAM is * DISABLED * at 0000-1FFF R/W
```

SDBP - DISABLE BREAKpOINT

Purpose

Disables one or all three breakpoints.

Format

This command has two forms:

1. **SDBP** - Disables all three breakpoints.

2. **SDPB Breakpoint-Number** - Disables breakpoint 1, 2, or 3, as specified.

Examples

```
> ? SDBP

SDBP x
   Disable software breakpoint x (1-3).

> SDBP 1

> STS

 ---> 1000 09 D
 ---> 1090 20 E
 ---> 2000 07 E
 IBR  RUNNING

Overlay RAM is * DISABLED * at 0000-1FFF R/W

> SDBP

> STS

 ---> 1000 09 D
 ---> 1090 20 D
 ---> 2000 07 D
 IBR  RUNNING

Overlay RAM is * DISABLED * at 0000-1FFF R/W
```

SC - CLEAR ALL SOFTWARE BREAKPOINTS AND RESET PASS COUNTERS

FORMAT

SC

EXAMPLE

```
< ? SC

SC
   Clear all software breakpoints and reset pass counters.


< STS

 ---> 1800 09 E
 ---> 1890 20 E
 ---> 2000 07 E
 IBR  RUNNING

Overlay RAM is * DISABLED * at 0000-1FFF R/W

> SC

> STS

 ---> 0000 00 D
 ---> 0000 00 D
 ---> 0000 00 D
 IBR  RUNNING

Overlay RAM is * DISABLED * at 0000-1FFF R/W
```

SEPP - ENABLE PRINTPOINT

Purpose

Enables one or all of the three printpoints. Printpoints are used in combination with the Untrace Command to generate the register display. Enabling the print point function allows NICE to trace a large number of instructions while displaying the registers only at specific addresses. The registers are displayed when:

- o a printpoint is enabled, and
- o a match is made between program counter and associated breakpoint address.

NOTE

> The printpoint and breakpoint functions are independent. The only similarity is that they both use the same address for comparison.

Format

There are two forms of the SEPP command:

1. **SEPP** - Enables all three printpoints.
2. **SEPP Printpoint-Number** - Enables the specified printpoint.

```
> ? SEPP

SEPP x
   Enable software printpoint x (1-3).
> SEPP 2

> STS

 ---> 1800 09 D
 ---> 1890 20 D P
 ---> 2000 07 D
 IBR  RUNNING

Overlay RAM is * DISABLED * at 0000-1FFF R/W
> SEPP

> STS

 ---> 1800 09 D P
 ---> 1890 20 D P
 ---> 2000 07 D P
 IBR  RUNNING

Overlay RAM is * DISABLED * at 0000-1FFF R/W
```

SDPP - DISABLE PRINTPOINT

Purpose

Disables one or all of the three printpoints.

Format

There are two forms of the DPP command:

1. **SDPP** - Disables all three printpoints.

2. **SDPP Printpoint-Number** - Disables the specified printpoint.

Examples

```
> ? SDPP

SDPP x
   Disable software printpoint x (1-3).

> SDPP 1

> STS

 ---> 1800 09 D
 ---> 1890 20 D P
 ---> 2000 07 D P
 IBR  RUNNING

Overlay RAM is * DISABLED * at 0000-1FFF R/W

> SDPP

> STS

 ---> 1800 09 D
 ---> 1890 20 D
 ---> 2000 07 D
 IBR  RUNNING

Overlay RAM is * DISABLED * at 0000-1FFF R/W
```

EI - ENABLE INTERRUPTS

Purpose

Instructs NICE to allow hardware interrupts in GO mode if that capability has been disabled using the Disable Interrupts command. Interrupts are automatically enabled following power up of the NICE Z80+.

An "I" appears on the last line of the status display when interrupts are enabled.

Examples

```
> ? EI

EI
   Enable Z80 maskable and non-maskable interrupts (default).

> EI

> STS

 ---> 1800 09 D
 ---> 1898 20 D
 ---> 2000 07 D
 IBR  RUNNING
```

DI - DISABLE INTERRUPTS

Purpose

Instructs NICE to block interrupts so that they cannot reach the Z80 microprocessor. This function is useful for checking code since interrupts can be enabled or disabled during full-speed execution. If NICE did not have this capability, the alternative would be to disable interrupts in software. Not only does this method take time to compile the new object code, it corrupts the source. Further, it is not possible to disable the Z80 NMI interrupt in software.

Examples

```
Overlay RAM is * DISABLED * at 0000-1FFF R/W

> DI

> STS

 ---> 1000 09 D
 ---> 1090 20 D
 ---> 2000 07 D
 .BR  RUNNING

Overlay RAM is * DISABLED * at 0000-1FFF R/W
```

EB - ENABLE BUS

Purpose

Instructs NICE to allow bus requests in GO mode if that capability has been disabled using the Disable Bus command. Bus requests are automatically enabled following reset of the target system. A "B" appears on the last line of the status display when the bus is enabled.

DB - DISABLE BUS

Purpose

Instructs NICE to block bus requests so that they do not reach the Z80+ microprocessor. This function is useful for removing the asynchronous use of the address and data bus caused by DMA devices when you are checking a program.

WARNING

The DB command can stop a data transfer before it is completed. With devices such as disk drives, it is possible that data contamination could result.

Examples

```
> ? DB

DB
   Disable Z80 bus requests.

> DB

> STS

 ---> 1000 09 D
 ---> 1090 20 D
 ---> 2000 07 D
 ..R  RUNNING

Overlay RAM is enabled at E000-FFFF R/W

> EB

> STS

 ---> 1000 09 D
 ---> 1090 20 D
 ---> 2000 07 D
 .BR  RUNNING

Overlay RAM is enabled at E000-FFFF R/W
```

DR - DISABLE REFRESH

Purpose

When NICE is in the QUIT mode, data is read every 1.25 ms from a minimum of 128 consecutive memory addresses. This function maintains the dynamic memory refresh function. If your system does not contain dynamic RAM, you may disable the memory refresh function, allowing NICE commands to execute 25 percent faster.

```
  ? DR

DR
   Disable Z80 refresh during QUIT mode.

> DR

> STS

 ---> 1800 09 D
 ---> 1890 20 D
 ---> 2000 07 D
 IB.  RUNNING

Overlay RAM is enabled at E000-FFFF R/W
```

ER - ENABLE REFRESH

Purpose

Instructs NICE to allow memory refresh if it has been disabled. The refresh function is automatically enabled following system power up. An "R" appears on the last line of the status display when memory refresh is enabled.

Examples

```
> ER

> STS

 ---> 1800 09 D
 ---> 1890 20 D
 ---> 2000 07 D
 IBR  RUNNING

Overlay RAM is enabled at E000-FFFF R/W
```

H - HEXADECIMAL ARITHMETIC

Purpose

Enables NICE to add and subtract 16-bit hexadecimal numbers.

Format

H First-Number, Second-Number

After executing the command, NICE sends two values to the terminal: the sum of the two numbers appears first, followed by the difference of the two numbers in HEX, Decimal, and ASCII.

Examples

```
> ? H

H xxxx,yyyy
   Calculate sum and difference of two hex numbers, results are displayed
   as SUM,DIFF in hex, decimal and ASCII.

> H 6789,1234

 79BD,5555   31165,21845   y.,UU

> H 90,45

 00D5,004B   00213,00075   ..,.K
```

Q - QUIT

Purpose

Causes NICE to leave the GO mode and enter the QUIT mode. When NICE enters the QUIT mode, the following occur:

- o Full speed execution of Z80 microprocessor is terminated.
- o Interrupts and bus requests are inhibited.
- o A listing of current Z80 registers is sent to the CRT.

Format

Q

Examples

```
> ? Q

Q
   Transfers emulator from RUN mode to QUIT mode.

> Q

 F =....N.  A=AE BC =7A1E DE =F9E8 HL =01E6 SP=1FAB PC=0637
 F'=.Z.V.C A'=0F BC'=0018 DE'=E062 HL'=1FE6 IX=07B4 IY=E7EF  I=00
0637   10FE      DJNZ 0637
```

RL - REPEAT LINE

Purpose

Causes repeated execution of the command or commands currently on the command line. The RL command is especially useful when producing scope loops or looking for intermittent problems.

Format

RL must be the last command on the command line.

Examples

```
> ? RL

RL
   Repeat command line until ^C is pressed.

  D 1800,1820;RL

1800    3E 00 0F 0F F0 B0 0F 0F-F0 F0 0F 0F F0 F0 0F 0F    >...............
1810    0F 0F F0 F0 0F 0F F0 F0-0F 0F F0 F0 0F 0F F0 F0    ................
1820    F0 F0 0F 0F F0 F0 0F 0F-B0 F0 0F 0F F0 F0 0F 0F    ................

1800    3E 00 0F 0F F0 B0 0F 0F-F0 F0 0F 0F F0 F0 0F 0F    >...............
1810    0F 0F F0 F0 0F 0F F0 F0-0F 0F F0 F0 0F 0F F0 F0    ................
1820    F0 F0 0F 0F F0 F0 0F 0F-B0 F0 0F 0F F0 F0 0F 0F    ................

1800    3E 00 0F 0F F0 B0 0F 0F-F0 F0 0F 0F F0 F0 0F 0F    >...............
1810    0F 0F F0 F0 0F 0F F0 F0-0F 0F F0 F0 0F 0F F0 F0    ................
1820    F0 F0 0F 0F F0 F0 0F 0F-B0 F0 0F 0F F0 F0 0F 0F    ................

1800    3E 00 0F 0F F0 B0 0F 0F-F0 F0 0F 0F F0 F0 0F 0F    >...............
1810    0F 0F F0 F0 0F 0F F0 F0-0F 0F F0 F0 0F 0F F0 F0    ................
1820    F0 F0 0F 0F F0 F0 0F 0F-B0 F0 0F 0F F0 F0 0F 0F    ................

1800    3E 00 0F 0F F0 B0 0F 0F-F0 F0 0F 0F F0 F0 0F 0F    >...............
1810    0F 0F F0 F0 0F 0F F0 F0-0F 0F F0 F0 0F 0F F0 F0    ................
1820    F0 F0 0F 0F F0 F0 0F 0F-B0 F0 0F 0F F0 F0 0F 0F    ................

1800    3E 00 0F 0F F0 B0 0F 0F-F0 F0 0F 0F F0 F0 0F 0F    >...............
1810    0F 0F F0 F0 0F 0F F0 F0-0F 0F F0 F0 0F 0F F0 F0    ................
1820    F0 F0 0F 0F F0 F0 0F 0F-B0 F0 0F 0F F0 F0 0F 0F    ................

1800    3E 00 0F 0F F0 B0 0F 0F-F0 F0 0F 0F F0 F0 0F 0F    >...............
1810    0F 0F F0 F0 0F 0F F0 F0-0F 0F F0 F0 0F 0F F0 F0  * ^C abort
```

STS - STATUS

Purpose

Shows STS or STS M information about the following:

- o Breakpoints
- o Printpoints
- o Interrupts
- o Bus Requests
- o Refresh Function
- o GO or QUIT mode

The first three lines of the status display represent the first, second, and third software breakpoints respectively. Each line reads like this:

- o The first entry represents the breakpoint address.
- o The second entry shows the pass counter.
- o The "E" or "D" indicates whether the breakpoint is enabled or disabled.
- o A "P" indicates that the printpoint is enabled. A blank indicates that the printpoint is disabled.

The fourth line of the status display shows the state of the interrupt, bus request, refresh functions, and whether NICE is in GO mode or QUIT mode. An I, B, or R indicates the function is enabled, while a period indicates the function is disabled. The word "RUNNING" indicates that NICE is in GO mode.

STS B - displays breakpoint status
STS H - displays performance monitoring bucket status

(see examples)

Examples

```
< ? STS

STS mode
   Display emulator status.  Mode may be one of the following:
     M - software breakpoint and overlay ram status.
     B - hardware breakpoint status.
     H - histogram bucket status.

> STS M
 ---> 1000 09 D
 ---> 1090 20 D
 ---> 2000 07 D
 IBR

Overlay RAM is enabled at E000-FFFF R/W

> STS B
#  Addr Cntr Init PSE       Breakpoint Mode is * DISABLED *
-- ---- ---- ---- ---
00 0000 0000 0000 ...
01 0000 0000 0000 ...
02 0000 0000 0000 ...
03 0000 0000 0000 ...
04 0000 0000 0000 ...
05 0000 0000 0000 ...
06 0000 0000 0000 ...
07 0000 0000 0000 ...
08 0000 0000 0000 ...
09 0000 0000 0000 ...
0A 0000 0000 0000 ...
0B 0000 0000 0000 ...
0C 0000 0000 0000 ...
0D 0000 0000 0000 ...
0E 0000 0000 0000 ...
0F 0000 0000 0000 ...

> STS H

Sample count = 1000
#  Strt End  Count
00 0000 0FFE 0000
01 0FFF 1FFD 0000
02 1FFE 2FFC 0000
03 2FFD 3FFB 0000
04 3FFC 4FFA 0000
05 4FFB 5FF9 0000
06 5FFA 6FF8 0000
07 6FF9 7FF7 0000
08 7FF8 8FF6 0000
09 8FF7 9FF5 0000
0A 9FF6 AFF4 0000
0B AFF5 BFF3 0000
0C BFF4 CFF2 0000
0D CFF3 DFF1 0000
0E DFF2 EFF0 0000
0F EFF1 FFFF 0000
```

Z - SLEEP

Purpose

Introduces delay into the command line interpreter.

Format

Z Delay-Count

Each count represents 1 ms of delay. Since Delay-Count is a 16-bit value, you can generate delays from 1 ms to as long as 65 seconds.

The Sleep command is most useful when used in combination with the Repeat Line command (RL) to produce scope loops.

Example

Below, the Z command is used with the RL command to read data from the I/O port Hex '02'. The CRT update is slow enough to be viewed.

```
> ? Z

Z xxxx
   Command line time delay.  Delay is in milliseconds.

 E 02;Z 100;RL
---> E0
---> E0
---> E0
---> E0
---> E0
---> E0
---> E0
---> E0
---> E0
---> E0
---> E0
---> E0
---> E0
---> E0
--- * ^C abort
>
```

CHAPTER 6
QUIT MODE COMMANDS

In QUIT mode, NICE recognizes the following commands:

Page #		
38	A -	Assemble into RAM
40	D -	Display Memory
42	E -	Examine Input Port
43	F -	Fill
44	G -	Go
45	L -	List in Assembler Format
47	M -	Move
48	MT -	Memory Test
50	O -	Output
51	R -	Read Intel Hex File
54	S -	Substitute Into Memory
56	SR -	Soft Reset
57	T -	Trace
59	U -	Untrace
61	UP -	Upload
62	V -	Verify
63	X -	Xamine
65	EOR -	Enable Overlay RAM
67	DOR -	Disable Overlay RAM
65	RO -	Set Overlay to R/O Status
65	RW -	Set Overlay to R/W Status
70	PR -	Set Performance Monitor Bucket Range
68	PRE -	Set All Performance Monitor Ranges Evenly
69	PRH -	Set All Performance Monitor Ranges From Last Highest
68, 71	PRO -	Do Performance Monitoring
71	PS -	Change Performance Monitoring Sample Count
72	SBM -	Set Breakpoint Mode
72	EBP -	Enable Breakpoint
73	DBP -	Disable Breakpoint
73	BPC -	Set Breakpoint Counter
74	EPP -	Enable Printpoint

A - ASSEMBLE INTO RAM

Purpose

Allows you to enter Z80 assembly language at the terminal. Each line is assembled as it is entered; the result is placed in the target system RAM.

Format

Two forms of the command are allowed:

1. **A** - Begins placing the assembled code at the currently saved memory address. The new saved memory address becomes one more than the last byte used in the assembly process.

2. **A Start-Address** - Begins placing the assembled code at the specified memory address.

Once you enter the command, the terminal displays the memory address which will receive the first byte of the assembled code. Following the memory address, you enter the desired Z80 assembly language mnemonics and a carriage return. NICE assembles the instruction, places it in memory, and displays the address of the next instruction.

If an error in assembly is detected, the address is displayed again so that you can re-enter the instruction. This process repeats until you enter a blank. At that time, NICE returns to the OK prompt.

Examples

```
> A 1800

1800   LD A,98
1802   OUT 03,A
1804   LD  A,C0
1806   OUT 02,A
1808   LD SP,1FAF
180B   LD A,(1FE5)
180E   CP A5
1810   CALL NZ,03C1
1813   LD HL,1800
1816   CALL 05F6
1819   JR Z,0021
 * Parameter out of range?
1819   JRZ 0021
?
1819   JR Z,0021H
 * Parameter out of range?
1819   LD H,18
181B
```

D - DISPLAY MEMORY

Purpose

Displays the contents of the target system's memory space.

Format

Each line of the display begins with the address of the first byte, followed by sixteen bytes of data in hexadecimal form. The same sixteen bytes in ASCII form end the line. If no ASCII character exists for the corresponding byte, then a period is listed.

There are three formats for the Display Memory command:

1. **D** - Displays memory from the currently saved memory address through the next eight lines (128 bytes). Because the saved memory address is always one more than the last memory address, using a series of D commands displays successive blocks of eight lines of data.

   The saved address is set to zero following power up.

2. **D Start-Address** - Displays eight lines of data beginning with the specified Start-Address.

3. **D Start-Address, Last-Address** - Displays the block of data within the specified range.

Examples

```
  ? D

D (xxxx(,yyyy))
   Dump memory from address xxxx to yyyy.  If xxxx not specified, then
   dump from last memory location for 128 bytes.  If yyyy not specified
   then dump from xxxx for 128 bytes.

> D 0000,0020

0000    06 00 10 FE 3E 90 D3 03-3E C0 D3 02 31 AF 1F 3A    ....>...>...1..:
0010    E5 1F FE A5 C4 C1 03 21-00 10 CD F6 05 28 02 26    .......!.....(.&
0020    18 22 DC 1F 26 00 18 0A-E3 2B E3 22 E8 1F 18 0E    ."..&....+."....

> D 0020

0020    18 22 DC 1F 26 00 18 0A-E3 2B E3 22 E8 1F 18 0E    ."..&....+."....
0030    18 34 22 D2 1F 18 1D 71-E5 2A EE 1F E3 C9 32 E7    .4"....q.*....2.
0040    1F 2A E0 1F 3A E2 1F 77-3E 00 D3 02 3A E7 1F 2A    .*..:..w>...:..*
0050    E8 1F 00 C9 21 9F 1F 22-D0 1F AF 32 E6 1F DD 21    ....!..".../2...!
0060    9F 07 C3 D0 00 FF 32 E7-1F 3E 90 D3 03 3E C0 D3    ......2..>...>..
0070    02 3A E7 1F 22 E8 1F E1-22 DE 1F 22 DC 1F 2A E8    .:.."..."..".."*.
0080    1F ED 73 D0 1F 31 D0 1F-FD E5 DD E5 D9 E5 D5 C5    ..s..1..........
0090    D9 08 F5 08 E5 D5 C5 F5-ED 57 32 D3 1F 3E 00 E2    .........W2..>..

> D

00A0    A4 00 3E 01 32 D2 1F 31-AF 1F 2A D0 1F DD 21 B5    ..>.2..1..*...!.
00B0    07 2B CD F6 05 20 19 2B-CD F6 05 20 13 DD 21 AF    .+... .+... ..!.
00C0    07 00 00 11 62 E0 19 38-07 DD 21 B6 1F 37 18 04    ....b..8..!..7..
00D0    AF 32 E4 1F 3A E2 1F 2A-E0 1F 77 DC 0B 04 31 AF    .2..:..*..w...1.
00E0    1F CD FE 05 CD CB 06 18-F5 FE 10 38 24 21 E6 1F    ...........8$!..
00F0    CB C6 D6 10 FE 08 21 37-07 DA B0 03 DD 21 B6 1F    ......!7.....!..
0100    D6 08 21 E4 1F 77 21 E3-1F 36 00 21 41 07 C3 B0    ..!..w!..6.!A...
0110    03 4F 21 4B 07 3A E4 1F-C3 B0 03 21 57 07 18 F5    .O!K.:.....!W...
```

E - EXAMINE INPUT PORT

Purpose

Retrieves data from a given I/O port and displays the 8-bit hexadecimal result. The specified Port Address must be an 8-bit number.

Format

E Port-Address

Examples

```
> ? E

E port
   Examine I/O port.

> E 02
 ---> D0
```

F - FILL

Purpose

Loads a block of target system RAM with a given 8-bit constant. A good use of this command is to clear target system RAM prior to debugging code.

Format

F Start-Address, Last-Address, 8-Bit-Value

Examples

```
> ? F

F xxxx,yyyy,zz
   Fill memory from xxxx to yyyy with zz.

> F 2000,2010,00

> D 2000,2010

2000    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
2010    00 FF 00 FF 00 FF 00 FF-00 FF 00 FF 00 FF 00 7F    ................
```

G - GO

Purpose

Instructs NICE to enter the GO mode and start the target system running at full speed.

Format

There are two formats of the GO Command, as shown below:

1. **G** - Full speed execution begins at the address contained in the program counter.

2. **G Start-Address** - Full speed execution begins at the specified address.

Examples

```
) ? G

G (xxxx)
   Transfer from QUIT mode to RUN mode at address xxxx if specified
   else continue from last PC.

) G 009F

EXECUTION BEGINS AT ===) 009F
```

L - LIST IN ASSEMBLER FORMAT

Purpose

Disassembles instructions in the target system memory into assembly language format.

Format

Three forms of the command are illustrated:

1. **L** - Disassembles 19 consecutive Z80 instructions, beginning with the currently saved memory address. As each instruction is disassembled, the new saved memory address points to the start of the next instruction. Thus, repeated L commands in this form list sequential blocks of 19 instructions.

2. **L Start-Address** - Lists the block of 19 instructions beginning with the specified Start-Address.

3. **L Start-Address, Last-Address** - Lists the block of instructions beginning with the specified start address and ending with the specified last address.

Examples

```
> L

0039   AF        XOR A
003A   D301      OUT 01,A
003C   7B        LD A,E
003D   2F        CPL
003E   F6C0      OR C0
0040   D302      OUT 02,A
0042   0606      LD B,06
0044   DB00      IN A,00
0046   57        LD D,A
0047   CB1A      RR D
0049   3802      JR C,004D
004B   79        LD A,C
004C   08        EX AF,AF'
004D   0C        INC C
004E   10F7      DJNZ 0047
0050   DD23      INC IX
0052   7B        LD A,E
0053   E63F      AND 3F
0055   CB07      RLC A

> L 0900

0900   21E218    LD HL,18E2
0903   CBD6      SET 2,(HL)
0905   3EFF      LD A,FF
0907   32CD18    LD (18CD),A
090A   CDD20E    CALL 0ED2
090D   CD4A0C    CALL 0C4A
0910   E1        POP HL
0911   FDE1      POP IY
0913   18C1      JR 08D6
0915   CD160E    CALL 0E16
0918   CD9F0E    CALL 0E9F
091B   E1        POP HL
091C   C1        POP BC
091D   7A        LD A,D
091E   B8        CP B
091F   2804      JR Z,0925
0921   3E0C      LD A,0C
0923   1897      JR 08BC
0925   7B        LD A,E

> L 0999,099F

0999   70        LD (HL),B
099A   2B        DEC HL
099B   C1        POP BC
099C   FDE5      PUSH IY
099E   D5        PUSH DE
099F   E5        PUSH HL
```

M - MOVE

Purpose

Copies a block of target system memory from one location to another. The source data may be in ROM or RAM, but the destination must be in RAM.

Format

M Start-Address, Last-Address, Destination-Address

Examples

```
  ? M

M xxxx,yyyy,zzzz
   Move memory block xxxx-yyyy to address zzzz.

> D 0000

0000    06 00 10 FE 3E 90 D3 03-3E C0 D3 02 31 AF 1F 3A    ....>...>...1..:
0010    E5 1F FE A5 C4 C1 03 21-00 10 CD F6 05 28 02 26    .......!.....(.&
0020    18 22 DC 1F 26 00 18 0A-E3 2B E3 22 E8 1F 18 0E    ."..&....+."....
0030    18 34 22 D2 1F 18 1D 71-E5 2A EE 1F E3 C9 32 E7    .4"....q.*....2.
0040    1F 2A E0 1F 3A E2 1F 77-3E 00 D3 02 3A E7 1F 2A    .*..:..w>...:..*
0050    E8 1F 00 C9 21 9F 1F 22-D0 1F AF 32 E6 1F DD 21    ....!.."...2...!
0060    9F 07 C3 D0 00 FF 32 E7-1F 3E 90 D3 03 3E C0 D3    ......2..>...>..
0070    02 3A E7 1F 22 E8 1F E1-22 DE 1F 22 DC 1F 2A E8    .:.."..."..".."*.

> M 0000,0020,2000

> D 2000,2020

2000    06 00 10 FE 3E 90 D3 03-3E C0 D3 02 31 AF 1F 3A    ....>...>...1..:
2010    E5 1F FE A5 C4 C1 03 21-00 10 CD F6 05 28 02 26    .......!.....(.&
2020    18 FF 00 FF 00 FF 00 FF-00 FF 00 FF 00 FF 00 FF    ................
```

MT-MEMORY TEST

Purpose

Verifies that the target system RAM is functioning properly.

NOTE

The test used is sometimes called the "peaks and valleys test." First it tests the RAM by walking a one through each bit; then it walks a zero through each bit. The process is to write the particular pattern throughout the entire range and then go back and check each byte to see that it matches the pattern written. NICE repeats the process for each of the 16 patterns.

The MT Command takes approximately 15 seconds for each 1K of memory. Therefore, you may want to start by checking small blocks of memory.

Format

MT Start-Address, Last-Address

NB dots and explanation points are output to indicate activity.

Examples

```
> ? MT

MT xxxx,yyyy
   Perform peaks-and-valley memory test from xxxx to yyyy.

> MT 2200,2400
```

O - OUTPUT

Purpose

Sends one or more 8-bit data bytes to an I/O port.

Format

O Port-Address, I/O-Data, I/O-Data, . . .

The Port-Address and the I/O-Data are 8-bit values. You must supply the Port-Address and at least one I/O-Data byte. Any additional I/O-Data bytes, if provided, will also be sent to the I/O port. Sending multiple data bytes to an I/O port is especially useful with some of the newer LSI chips such as CRT, DMA and Floppy Disk controllers.

Examples

```
> ? O

O port,data[,data[...]]
   Output data to port.

> O 02,54,64,FE,;RL
 * ^C abort
> O 02,22;Z 50;RL
 * ^C abort
> O 02,FF
```

R - READ INTEL HEX FILE

Purpose

Loads an Intel Hex file into the target system RAM.

Format

A description of the format for an Intel Hex file is given in Appendix B. Two forms of the command are allowed:

1. **R** - Loads the file directly into RAM at the location indicated by the Intel Hex file.

2. **R Offset** - Adds the 16-bit offset value to the destination address prior to loading the data into RAM. The last record of an Intel Hex file specifies the program counter address and the offset is applied to this value as well. This form is useful for downloading relocatable code.

Once you enter the command, NICE only recognizes input which corresponds to an Intel Hex file. Each record begins with a colon (:), so NICE looks for a colon at the beginning of each record and ignores all other characters. This reduces NICE's susceptibility to noise and allows you to switch to a different download device following entry of the R command. As with the command line interpreter, all non-control characters are reflected as they are received. Thus, you can confirm that NICE received the correct sequence of characters.

NICE exits the command only at the end of the Hex file or when an error is detected.

NICE responds differently to good records and bad records:

- If the record is good, NICE sends this sequence:

  ACK, LF, CR

- If the record is bad, NICE sends this sequence:

  NAK, xx-ERR, LF, CR

  xx stands for RT (Record Type error), CS (Check Sum error), or HC (invalid Hex Code).

The Intel Hex file is composed of records. NICE processes each record as it is read. This means that NICE stores data in the record as it is received.

NOTE

When an error occurs, bad data may be stored in the target system RAM. To remove the bad data and continue downloading the file, issue another Read Intel Hex File Command. Begin downloading the file with the record that caused the error. Because each printable ASCII character is echoed as it is received, you may check that each character has been received properly by NICE.

Examples

```
OK ====> R
:0F8000003E00010000C04FE4020FA060018F686
:008000017F

OK ====> X

...... A=00  BC=0000  DE=0000  HL=0000  S=0000  P=8000  LD A,00
...... A'=00  B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

OK ====> D 8000

8000    3E 00 01 00 00 3C 04 FE-40 20 FA 06 00 18 F6 00    >....<..@ ......
8010    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8020    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8030    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8040    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8050    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8060    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8070    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................

OK ====> R 100
:0F8000003E00010000C04FE4020FA060018F686
:008000017F

OK ====> X

...... A=00  BC=0000  DE=0000  HL=0000  S=0000  P=8100  LD A,00
...... A'=00  B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

OK ====> D 8100

8100    3E 00 01 00 00 3C 04 FE-40 20 FA 06 00 18 F6 00    >....<..@ ......
8110    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8120    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8130    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8140    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8150    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8160    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8170    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................

OK ====> R
:0F800005 RT-ERR

ERR ===>
```

S - SUBSTITUTE INTO MEMORY

Purpose

Allows you to view and optionally enter data into the target system RAM.

Format

S Start-Address - Starts at the specified address.

Once the process is started, the terminal displays the data at the current address. At that point you may enter any of the following:

1. A single carrige return. In this case the data at the associated address remains unchanged, the current address is incremented, and the next data is byte displayed.

2. New data to replace the current data followed by a carriage return. The current address is then incremented and the next byte displayed.

$$\text{data} = \left[\frac{\text{hex \#}}{\text{string}}\right]\left\{,\left[\frac{\text{hex\#}}{\text{string}}\right]\left\{\ \ldots\ \right\}\right\}$$

$$\text{string} = \left[\begin{matrix}\cdot\\ \cdot\cdot\end{matrix}\right]\left\{\left\{\begin{matrix}\text{upper char}\\ \text{lower char}\end{matrix}\right\}\ldots\right\}\left[\begin{matrix}\cdot\\ \cdot\cdot\end{matrix}\right]$$

   Note: first and last quote must match.

3. A "-" followed by a carriage return. This form is used to back up to a previous location and correct an error. The data at the current address is left unchanged. The current address is decremented and the previous byte is displayed. At that point, you may type any of the options in this list.

4. A "." followed by a carriage return. This ends the substitute process and causes NICE to return to the command line interpreter.

Examples

```
> ? S

S xxxx
   Set memory at address xxxx.  Input line format:
        X{,X{...}}  Where: X = hex number, format: xx{H}
                           = string, format: ['/"]char['/"]
                           = - to move to previous address


  S 2000
2000 06 ---> FE
2001 00 ---> 45
2002 10 ---> 90
2003 FE ---> CE
2004 3E ---> "N"
2005 90 ---> "I"
2006 D3 ---> "C"
2007 03 ---> "E"
2008 3E ---> "+"
2009 C0 ---> .

> D 2000

2000    FE 45 90 CE 4E 49 43 45-2B C0 D3 02 31 AF 1F 3A    .E..NICE+...1..:
2010    E5 1F FE A5 C4 C1 03 21-00 10 CD F6 05 28 02 26    .......!.....(.&
2020    18 FF 00 FF 00 FF 00 FF-00 FF 00 FF 00 FF 00 FF    ................
2030    00 7F 00 FF 00 FF 00 DF-00 FF 00 FF 00 FF 00 FF    ................
2040    00 FF 00 FF 00 FF 00 FF-00 FF 00 FF 00 FF 00 FF    ................
2050    00 FF 00 FF 00 FF 00 FF-00 FF 00 FF 00 FF 00 FF    ................
2060    00 FF 00 FF 00 FF 00 FF-00 FF 00 FF 00 FF 00 FF    ................
2070    00 FF 00 FF 00 FF 00 FF-00 BF 00 FF 00 FF 00 FF    ................
```

SR - SOFT RESET

Purpose

Clears the Z80 PC and I registers and executes the Z80 DI instruction. This simulates a hardware reset to the Z80.

Examples

```
> ? SR

SR
   Soft reset Z80

> SR

> X

 F =....N.  A=70 BC =8506 DE =F9C2 HL =05CB SP=1FAB PC=0000
 F'=.Z.V.C A'=0F BC'=0010 DE'=E062 HL'=1FE6 IX=1FB7 IY=E7EF  I=00
0000   0600      LD B,00
```

T - TRACE

Purpose

Allows you to follow a program through one or more instruction steps. After each instruction is executed, NICE displays all internal Z80 registers and flags, as well as the mnemonic of the next instruction. The Trace Command is especially useful in debugging virgin code.

Format

The Trace Command has two formats:

1. **T** - Traces the program through one instruction.

2. **T Number-of-Instructions** - Traces the program for the specified number of instructions. (Number-of-Instructions is a 16-bit value.)

After NICE traces each instruction, all the Z80 registers and flags are printed at the terminal. If breakpoints are enabled, encountering a breakpoint will cause the trace sequence to terminate.

You can start or stop printout by entering CTRL-S, or terminate the command by entering CTRL-C.

Examples

```
< ? T

T (xxxx)
   Trace program with register display for xxxx steps.

> T

 F =....N.  A=78 BC =0006 DE =F9C2 HL =05C8 SP=1FAB PC=0002
 F'=.Z.V.C A'=0F BC'=0018 DE'=E062 HL'=1FE6 IX=1FB7 IY=E7EF  I=00
0002   10FE      DJNZ 0002

> T 4

 F =....N.  A=78 BC =FF06 DE =F9C2 HL =05C8 SP=1FAB PC=0002
 F'=.Z.V.C A'=0F BC'=0018 DE'=E062 HL'=1FE6 IX=1FB7 IY=E7EF  I=00
0002   10FE      DJNZ 0002

 F =....N.  A=78 BC =FE06 DE =F9C2 HL =05C8 SP=1FAB PC=0002
 F'=.Z.V.C A'=0F BC'=0018 DE'=E062 HL'=1FE6 IX=1FB7 IY=E7EF  I=00
0002   10FE      DJNZ 0002

 F =....N.  A=78 BC =FD06 DE =F9C2 HL =05C8 SP=1FAB PC=0002
 F'=.Z.V.C A'=0F BC'=0018 DE'=E062 HL'=1FE6 IX=1FB7 IY=E7EF  I=00
0002   10FE      DJNZ 0002

 F =....N.  A=78 BC =FC06 DE =F9C2 HL =05C8 SP=1FAB PC=0002
 F'=.Z.V.C A'=0F BC'=0018 DE'=E062 HL'=1FE6 IX=1FB7 IY=E7EF  I=00
0002   10FE      DJNZ 0002
```

U - UNTRACE

Purpose

The Untrace Command is identical to the Trace Command except that the Z80 registers and flags are printed only when the last instruction is traced. It allows you to trace a large number of instructions without having to wait for the printout at the terminal.

Format

The Untrace Command has two forms:

1. **U** - Traces the program through one instruction.

2. **U Number-of-Instructions** - Traces the program for the specified number of instructions. (Number-of-Instructions is a 16-bit value.)

Enabled breakpoints cause the Untrace Command to terminate and the display to be generated. Enabled printpoints are displayed but do not terminate the command. When the command terminates, the addresses of the previous four instructions are displayed. These are referred to as backtrace addresses -1, -2, -3, and -4.

Examples

```
> ? U

U (xxxx)
   Trace program without register display for xxxx steps.


> U

BACK TRACE    ------> -4=0000  -3=0000  -2=0000 -1=0050
 F =....N.  A=70 BC =EE06 DE =F9C2 HL =05C8 SP=1FAB PC=0051
 F'=.Z.V.C A'=0F BC'=0018 DE'=E062 HL'=1FE6 IX=1FB7 IY=E7EF  I=00
0051   1F        RRA

> U 9

BACK TRACE    ------> -4=0620  -3=0621  -2=0622 -1=0623
 F =S.HV..  A=B6 BC =EE06 DE =F9C2 HL =07B3 SP=1FAF PC=00E4
 F'=.Z.V.C A'=0F BC'=0018 DE'=E062 HL'=1FE6 IX=1FB7 IY=E7EF  I=00
00E4   CDCB06    CALL 06CB
```

UP - UPLOAD

Purpose

"Dumps target memory between specified addresses in Intel Hex format to the host terminal or computer.

Format

UP Start-Address, Last-Address

Example

```
>?UP
UP xxxx, yyyy
Upload to host . . . address range xxxx thru yyyy.

>
UP 8190,8200
:1081900001F0FD3602F0FD3603F0FD3604E0FD4649
:1081A00000CD7E80DCCB8010F8FD7E01FDAE02327A
:1081B00000F0252007FDCB010EFD66032D2007FDF5
:1081C000CB020EFD6E0418D6083EA0CDAD8008EDA2
:1081D0004D083EA1CDAD8008ED4D083E20CDAD80CF
:1081E00008FBED4D083E21CDAD8008FBED4D083E6E
:1081F00022CDAD8008ED4D083E55CDAD8008ED4552
:01820000CDB0
:0000000000
```

V - VERIFY

Purpose

Compares two blocks of memory within the target sytem and indicates any differences.

Format

V Start-Address, Last-Address, Compare-Address

Each of the parameters are 16-bit addresses. The Start-Address and Last-Address define one of the two blocks of memory as well as the length of both blocks. Compare-Address is the starting point of the second block of memory. If there is a discrepancy between the data at equivalent locations in the first and second blocks, NICE prints the address within the first block, the data at that address, and then the data within the second block.

Examples

```
) ? V

V xxxx,yyyy,zzzz
   Verify that memory block xxxx-yyyy matches block of same length
   at address zzzz.

) V 0000,0020,2000

0000 ---) 06 FE
0001 ---) 00 45
0002 ---) 10 90
0003 ---) FE CE
0004 ---) 3E 4E
0005 ---) 90 49
0006 ---) D3 43
0007 ---) 03 45
0008 ---) 3E 2B
```

X - XAMINE

The Xamine Command has two forms as indicated below:

1. **X** - Causes all the Z80 registers and flags to be displayed at the terminal.

   Both the primary and secondary sets of registers are displayed, the primary set on the top line and the secondary set on the bottom line.

   The two flag registers are displayed as a series of letters where each character represents one bit of the register. The letters represent the various flag bits as indicated below:

   S - Sign Flag
   Z - Zero Flag
   H - Half Carry Flag
   U - Parity or Overflow Flag
   N - Add/Subtract Flag
   C - Carry Flag

   The appropriate character is printed when the bit is set and a period is printed when the bit is reset. For example, if the Sign and Carry bits were the only ones set, the display would read:

   S....C

2. **X Reg-Id** - Displays and allows modification of the internal Z80 registers. Reg-Id indicates the desired register, and may be any of the following: F, F', A, A', B, B', D, D', S, P, X, Y, H, H' or I.

   After the data in the register is displayed, you may enter a carriage return to retain the old data, or you may enter new data to replace the old. If you enter new data, enter a one byte value for a one byte register and a two byte value for a two byte register. If you enter no data, the register remains unchanged.

Examples

```
> ? X

X (reg)
   Display/Modify Z80 registers.
        Registers are: F, F', A, A', B, B', D, D', S, P, X, Y, H, H', I
        Flags are: S, Z, H, U, N, C.

> X

 F =S.HV..  A=B6 BC =EE06 DE =F9C2 HL =07B3 SP=1FAF PC=00E4
 F'=.Z.V.C A'=0F BC'=0018 DE'=E062 HL'=1FE6 IX=1FB7 IY=E7EF  I=00
00E4   CDCB06    CALL 06CB

> X P

P=00E4 ---> 0624

> X S

S=1FAF ---> 1EFC

> X

 F =S.HV..  A=B6 BC =EE06 DE =F9C2 HL =07B3 SP=1EFC PC=0624
 F'=.Z.V.C A'=0F BC'=0018 DE'=E062 HL'=1FE6 IX=1FB7 IY=E7EF  I=00
0624   37        SCF

>
```

EOR- ENABLE OVERLAY RAM

EOR enables 8K overlay RAM from base address x000 where x is even.

```
> EOR 2000

> STS

 ---> 0000 00 D
 ---> 0000 00 D
 ---> 0000 00 D
 IBR

Overlay RAM is enabled at 2000-3FFF R/W
```

The RO and RW instructions define the overlay RAM as Read Only or Read/Write respectively.

```
>RO

> STS

 ---> 0000 00 D
 ---> 0000 00 D
 ---> 0000 00 D
 IBR

Overlay RAM is enabled at 2000-3FFF R/O
```

To overlay ROM area-operating from overlay RAM

```
> EOR 2000

> M 0000,0FFF,2000

> EOR 0000

> STS

 ---> 0000 00 D
 ---> 0000 00 D
 ---> 0000 00 D
 IBR

Overlay RAM is enabled at 0000-1FFF R/W

> G 0
```

Overlay RAM enabled, base address does not overlap area where ROM code resides.

Move contents of ROM to overlay RAM

Emulator RAM overlays system ROM.

The GO command is issued and operation commences from emulator overlay RAM.

In this example, overlay RAM is defined as R/W because the target system RAM starts at 1800 H and the overlay crosses the system RAM/ROM boundary.

DOR -DISABLE OVERLAY RAM

Disables entire 8K of overlay RAM. It does not change overlay RAM contents.

```
> ? DOR

DOR
   Disable 8k overlay ram.


< Q

 F =.ZH..C  A=30 BC =4800 DE =FFC1 HL =06E6 SP=1FAB PC=0637
 F'=.Z.V.C A'=0F BC'=0018 DE'=F9D0 HL'=1FE6 IX=079F IY=FFEF  I=00
0637   10FE      DJNZ 0637

> DOR

> STS

 ---> 0000 00 D
 ---> 0000 00 D
 ---> 0000 00 D
 IBR

Overlay RAM is * DISABLED * at 0000-1FFF R/W

>
```

PERFORMANCE MONITORING

Performance monitoring locates the area of program execution and reveals program bottlenecks.

The most general form of the command is: PRE XXXX, YYYY. Where XXXX thru YYYY is the address range for which the sixteen buckets are to be evenly distributed.

PRE evenly distributes the address buckets over the defined range. The monitoring is activated by the PRO command.

In the resulting histogram, 90 percent of program activity occurs in the address range of "bucket" number 00_H. This range or bucket may be further resolved by issuing the command PRH which "breaks down" the bucket; into an additional sixteen evenly distributed buckets.

```
> PRE 0000,FFFF

> PRO
................

 Sample count = 1000
 00 0000-0FFE:0F06 ¦ ##################################################
 01 0FFF-1FFD:0000 ¦
 02 1FFE-2FFC:0000 ¦
 03 2FFD-3FFB:0000 ¦
 04 3FFC-4FFA:0000 ¦
 05 4FFB-5FF9:0000 ¦
 06 5FFA-6FF8:0000 ¦
 07 6FF9-7FF7:0000 ¦
 08 7FF8-8FF6:0000 ¦
 09 8FF7-9FF5:0000 ¦
 0A 9FF6-AFF4:0000 ¦
 0B AFF5-BFF3:0000 ¦
 0C BFF4-CFF2:0000 ¦
 0D CFF3-DFF1:0000 ¦
 0E DFF2-EFF0:0000 ¦
 0F EFF1-FFFF:00FA ¦ ###
 ----------------¦-------------------------------------------------
                 ¦    ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
#  Strt-End :Count¦ 2%  10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
```

The PRH command can be used to perform several iterations until suitable resolution is obtained. The smallest bucket size will have two consecutive locations; to denote the upper and lower limits of the "bucket".

```
> PRH

> PRO
Sample count = 1000
00 0000-00FE:0EC9 ¦ ************************************************
01 00FF-01FD:0000 ¦
02 01FE-02FC:0000 ¦
03 02FD-03FB:0000 ¦
04 03FC-04FA:0000 ¦
05 04FB-05F9:0000 ¦
06 05FA-06F8:0000 ¦
07 06F9-07F7:0000 ¦
08 07F8-08F6:0000 ¦
09 08F7-09F5:0000 ¦
0A 09F6-0AF4:0000 ¦
0B 0AF5-0BF3:0000 ¦
0C 0BF4-0CF2:0000 ¦
0D 0CF3-0DF1:0000 ¦
0E 0DF2-0EF0:0000 ¦
0F 0EF1-0FFE:0000 ¦
-----------------¦------------------------------------------------
                 ¦    ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
#  Strt-End :Count¦ 2%  10%  20%  30%  40%  50%  60%  70%  80%  90%  100%

> PRH

> PRO
.................

Sample count = 1000
00 0000-000E:0000 ¦
01 000F-001D:0000 ¦
02 001E-002C:0000 ¦
03 002D-003B:00FF ¦ ***
04 003C-004A:00E0 ¦ **
05 004B-0059:0000 ¦
06 005A-0068:0000 ¦
07 0069-0077:0382 ¦ **********
08 0078-0086:02A0 ¦ ********
09 0087-0095:0620 ¦ ********************
0A 0096-00A4:00E0 ¦ **
0B 00A5-00B3:0000 ¦
0C 00B4-00C2:0000 ¦
0D 00C3-00D1:0000 ¦
0E 00D2-00E0:0000 ¦
0F 00E1-00FE:0000 ¦
-----------------¦------------------------------------------------
                 ¦    ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
#  Strt-End :Count¦ 2%  10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
```

INDIVIDUAL BUCKET ASSIGNMENT

Format

PR n, xxxx, yyyy

Where n is the bucket number and xxxx, yyyy is the bucket range. Useful for defining non-concurrent buckets eliminating areas of no-program activity or to include otherwise out of range buckets.

```
> PR 0,0033,0037
> PR 1,0038,003C
> PR 2,003D,0041
> PR 3,0064,0068
> PR 4,0069,006D
> PR 5,006E,0072
> PR 6,0073,0077
> PR 7,0078,007C
> PR 8,007D,0081
> PR 9,0082,0086
> PR A,0087,008B
> PR B,008C,0090
> PR C,0091,0095
> PR D,0096,009A
> PR E,009B,009F
> PR F,00A0,00A5
```

```
> PR0
.................


Sample count = 1000
00 0033-0037:0000 :
01 0038-003C:0189 : ****
02 003D-0041:00CF : **
03 0064-0068:0000 :
04 0069-006D:019E : *****
05 006E-0072:00CF : **
06 0073-0077:00CF : **
07 0078-007C:00CF : **
08 007D-0081:00CF : **
09 0082-0086:00CE : **
0A 0087-008B:019D : *****
0B 008C-0090:019E : *****
0C 0091-0095:026D : *******
0D 0096-009A:00CF : **
0E 009B-009F:0000 :
0F 00A0-00A5:0000 :
-----------------:-----------------------------------------------
                 :    :    :    :    :    :    :    :    :    :    :
#  Strt-End :Count: 2%  10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
```

PRO xxxx

Where xxxx is the address at which the monitoring is to commence

PRO

will commence sampling at the address following the last program counter value.

PS

sets the sample count up to 1000 $_H$. The sample count unless specified will default to 1000$_H$.

EBP - ENABLE BREAKPOINT

The breakpoints enabled in the Quit mode are effective in the GO mode (operate in real time). These breakpoints operate on several modes from which the following can be chosen: Memory Request, Memory Recall, Memory Write, on M1, I/O Request, I/O Write, and I/O Read.

Qualification is by address, so setting breakpoints on a memory read operation should include the address at which it is performed.

FORMAT

EBP x, yyyy, zzzz

x = breakpoint number 0 thru F

yyyy = address

zzzz = pass counter

Set Breakpoint Mode (SBM)

```
) ? SBM

SBM mode
   Set hardware breakpoint mode.  Modes are:
        M, MR, MW, M1, I, IR, IW, DB.
```

Where M is breaks on any memory operation

MR is break on any memory read

MW is break on any memory write

M1 is break on any instruction fetch

I is break on any I/O operation

IR is break on any I/O read

IW is break on any I/O write

DB is disable breakpoints on any operation

A single breakpoint is enabled at address 0004_H and coincides with a memory read operation.

BPC _ BREAKPOINT COUNT

When the breakpoint is encountered (after 9 passes through 0004_H) the instruction at this location is executed and the address of the next instruction is displayed with the register settings. The pass counter is initialized to its original value.

FORMAT

BPCx, yyyy

Set pass counter for breakpoint (O thru F) to a value (1 thru FFFF)

Example

```
> EBP 0,4
> SBM MR
> BPC 0,9
> STS B
#  Addr Cntr Init PSE       Breakpoint Mode is MEMRQ and RD
-- ---- ---- ---- ---
00 0004 0009 0009 .SE
01 0000 0000 0000 ...
02 0000 0000 0000 ...
03 0000 0000 0000 ...
04 0000 0000 0000 ...
05 0000 0000 0000 ...
06 0000 0000 0000 ...
07 0000 0000 0000 ...
08 0000 0000 0000 ...
09 0000 0000 0000 ...
0A 0000 0000 0000 ...
0B 0000 0000 0000 ...
0C 0000 0000 0000 ...
0D 0000 0000 0000 ...
0E 0000 0000 0000 ...
0F 0000 0000 0000 ...
> G 0
EXECUTION BEGINS AT ===> 0000
>  - Breakpoint at 0004
 F =....N.  A=90 BC =000C DE =F9C4 HL =04E6 SP=1FAB PC=0006
 F'=.Z.V.C A'=0F BC'=0018 DE'=F9D0 HL'=1FE6 IX=07A1 IY=FFFF  I=00
 0006   D303      OUT 03,A
```

DBP - Disable Breakpoints

The breakpoints are disabled; either one at a time or all at once.

```
> ? DBP

DBP x/ALL
   Disable hardware breakpoint x / ALL breakpoints.
```

EPP - ENABLE PRINTPOINTS

Printpoints are used with breakpoints to allow NICE Z80 to go through (in RUN mode) a large number of instructions while displaying the registers only at specific addresses. After the register display, printpoints allow program execution to continue (unlike breakpoints which cause the processor to stop at a specified location).

```
) EBP 0,4

) EPP 0

) SBM MR

) STS B
#  Addr Cntr Init PSE       Breakpoint Mode is MEMRQ and RD
-- ---- ---- ---- ---
00 0004 0000 0000 PSE
01 0000 0000 0000 ...
02 0000 0000 0000 ...
03 0000 0000 0000 ...
04 0000 0000 0000 ...
05 0000 0000 0000 ...
06 0000 0000 0000 ...
07 0000 0000 0000 ...
08 0000 0000 0000 ...
09 0000 0000 0000 ...
0A 0000 0000 0000 ...
0B 0000 0000 0000 ...
0C 0000 0000 0000 ...
0D 0000 0000 0000 ...
0E 0000 0000 0000 ...
0F 0000 0000 0000 ...

) G 0

EXECUTION BEGINS AT ===) 0000

)  - Printpoint at 0004
 F =....N.  A=90 BC =0018 DE =F9D0 HL =02E6 SP=1FAB PC=0006
 F'=.Z.V.C A'=0F BC'=001E DE'=F9E0 HL'=1FE6 IX=07A3 IY=E72F  I=00
 0006   D303      OUT 03,A
```

APPENDIX A
QUICK REFERENCE COMMAND LIST

Quit Mode Only

A	Assemble into memory beginning at the last referenced memory address.
A sa	Assemble into memory beginning at the specified start address (sa).
D	Display the block of memory beginning at the last referenced memory address.
D sa	Display the block of memory beginning at the specified start address (sa).
D sa, la	Display the block of memory beginning at the specified start address (sa) and stopping at the specified last address (la).
E pa	Examine the data at an I/O port address (pa).
F sa, la, d8	Fill memory from the specified starting address (sa) up to and including the specified last addess (la) with the specified 8 bit data byte (d8).
G	Go into full speed execution starting at the current program counter location.
G sa	Go into full speed execution starting at the specified start address (sa).
L	List in assembler mnemonics beginning at the last referenced memory address.

Quit Mode Only (continued)

L sa	List in assembler mnemonics beginning at the specified start address (sa).
L sa, la	List in assembler mnemonics beginning at the specified start address (sa) and stopping at the specified last address (la).
M sa, la, da	Move the block of memory between the specified start address and the last address to a destination address (da).
MT sa, la	Run a memory test on the target system RAM between the specified start address (sa) and a last address (la).
O pa, d8, d8, ...	Output to the specified port address (pa) one or more 8-bit data bytes.
R	Read an Intel hex file and load the data into the target system RAM.
R off	Read an Intel hex file, applying the specified 16-bit offset value prior to loading the target system RAM.
S sa	Substitute into memory beginning at the given start address (sa).
SR	Do a soft reset of the Z80 microprocessor.
T	Trace one instruction.
T d16	Trace one or more instructions.
U	Untrace one instruction.
UP sa, la	UPload to host in Intel Hex format between start address (sa) and last address (la)
U d16	Untrace one or more instructions.

Quit Mode Only (continued)

V sa, la, da	Verify that the target system memory is the same beginning at a given start address (sa) and ending at the last address (la) to the data block starting at the destination address (da).
X	Display all the Z80+ internal registers and flags.
X ?	Display and allow modification of the specified Z80+ internal registers. ? can be any one of the given registers: ? = F, F', A, A', B, B', D, D', H, H', S, P, X, Y, I
EOR x	Enable 8K overlay RAM (at base address x000).
DOR	Disable 8K overlay RAM.
RO	Set 8K overlay RAM to read-only status.
RW	Set 8K overlay to read-write status (default).
PR x,yyyy,zzzz	Set performance monitoring bucket x to address range yyyy thru zzzz.
PRE xxxx,yyyy	Set all performance monitoring buckets evenly over address range xxxx thru yyyy.
PRH	Set all performance monitoring buckets evenly over the address range defined by the current bucket with the highest activity.
PRO xxxx	Do performance monitoring starting at address xxxx.
PS xxxx	Set performance monitoring sample counter to xxxx.
DPR	Display last performance monitoring chart.
SBM mode	Set hardware breakpoint mode. Modes are: M, MR, MW, M1, I, IR, IW, DB.
EBP x(,yyyy(,zzzz))	Enable hardware breakpoint x (at address yyyy) and set pass counter to zzzz. Default pass counter to 0.
DBP x/ALL	Disable hardware breakpoint x / ALL breakpoints.
BPC x,yyyy	Set hardware breakpoint x pass to counter to yyyy.
EPP x	Enable hardware printpoint x.
DPP x/ALL	Disable hardware printpoint x / ALL printpoints.

Go or QUIT Mode

SBP 1, d16	Set breakpoint address 1 to the specified 16-bit value (d16).
SBP 2, d16	Set breakpoint address 2 to the specified 16-bit value (d16).
SBP 3, d16	Set breakpoint address 3 to the specified 16-bit value (d16).
SBPC 1, d8	Set breakpoint pass counter 1 to the specified 8-bit value (d8).
SBPC 2, d8	Set breakpoint pass counter 2 to the specified 8-bit value (d8).
SBPC 3, d8	Set breakpoint pass counter 3 to the specified 8-bit value (d8).
SEBP	Enable all three breakpoints.
SEBP n	Enable breakpoint 1, 2, or 3 where (n) is the breakpoint number.
SDBP	Disable all three breakpoints.
SDBP n	Disable breakpoint 1, 2, or 3 where (n) is the breakpoint number.
SEPP	Enable all printpoints.
SEPP n	Enable printpoint 1, 2, or 3 where (n) is the printpoint number.
SDPP	Disable all printpoints.
SDPP n	Disable printpoint 1, 2, or 3 where (n) is the printpoint number.

Go or QUIT Mode (continued)

EI	Enable interrupts to be received by the Z80 from the target system.
DI	Disable interrupts from the target system.
EB	Enable bus requests to be received from the target system.
DB	Disable bus requests from the target system.
ER	Enable NICE's automatic refresh function.
DR	Disable NICE's automatic refresh function.
H d16, d16	Using hex arithmetic, calculate the sum and difference of the two specified 16-bit data values.
Q	Quit full speed execution.
RL	Repeat the given command line.
STS mode	Display the emulator status, including breakpoint addresses pass counters breakpoint enable or disable printpoint enable or disable interrupts enable or disable bus requests enable or disable refresh enable or disable emulator mode
Z d16	Have the emulator wait a given amount of time before executing the next command.

APPENDIX B
INTEL HEX FORMAT

A short Intel Hex File is shown below:

```
:02319300923176
:00310001CE
```

Explanations of each field follow:

Record Mark Field: Frame 0 : 02319300923176

The ASCII code for Colon (:) is used to signal the start of a record.

Record Length Field: Frames 1 and 2 : 02319300923176

The number of data bytes in the record is represented by two ASCII hexadecimal digits in this field. The high-order digit is in frame 1. The maximum number of data bytes in a record is 255 (FF in dexadecimal). Two ASCII zeroes in this field indicate an end-of-file.

Regardless of the record type, a zero record length causes NICE to return to the command line interpreter following receipt of the checksum field.

Load Address Field: Frames 3-6 :02 3193 00923176

The four ASCII hexadecimal digits in frames 3-6 give the address at which the data is loaded. The high-order digit is in frame 3; the low-order digit is in frame 6. The first data byte is stored in the location indicated by the load address; successive bytes are stored in successive memory locations. This field in an end-of-file record contains zeros or the starting address of program.

Record Type Field: Frames 7 and 8 :023193 00 923176

The two ASCII hexadecimal digits in this field specify the record type. The high-order digit is in frame 7. All data records are type 0; end-of-file records are type 0 or 1. Other values for this field are not recognized and will cause an error in the downloading process.

Data Field: Frames 9 to 9 + 2* (record length - 1) :02319300 923176

A data byte is represented by two frames containing the ASCII characters 0-9 or A-F, which represent a hexadecimal value between 0 and FF (0 to 255). The high order digit is in the first frame of each pair. There are no data bytes in an end-of file record.

Checksum Field: Frames 9 + 2* (record length) to 9 + 2* (record length + 1)

:023193009231 76

APPENDIX C
TARGET SYSTEM MODIFICATIONS

CROMEMCO ZPU

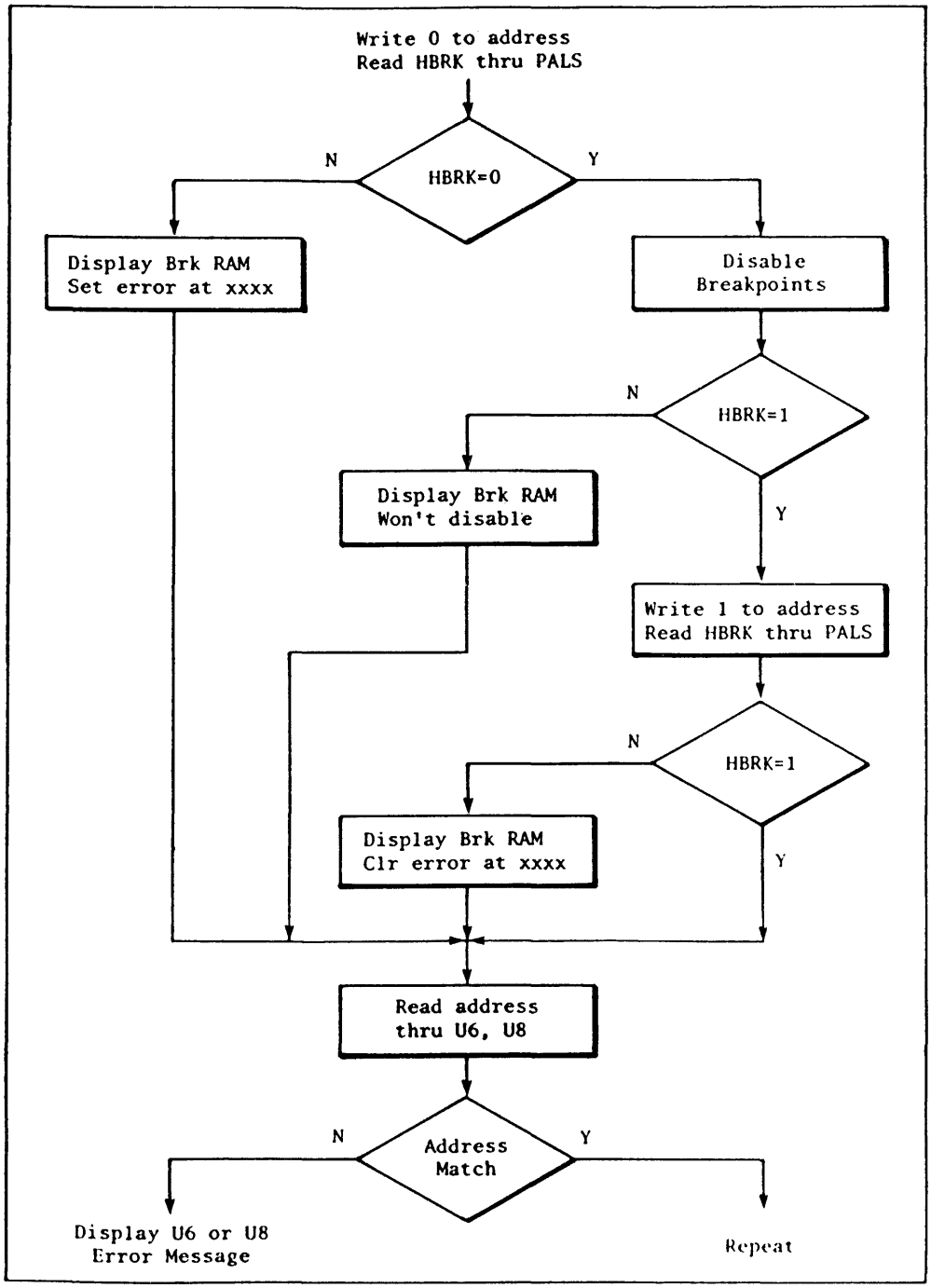
The ZPU card causes the data bus from the system RAM to be disabled for two or three cycles following the leading edge of RAM.

To modify the ZPU card so that it will work with NICE, tie up pin #1 of IC32 (floating is acceptable). This modifies the ZPU so that the Data Bus In is not disabled following the leading edge of MREQ.

APPENDIX D
SELF TEST FUNCTIONS

The NICE Z80+ has the following self-contained test routines:

Test B: Breakpoint Test performs the following to all 64k break RAM:



Test R:

Simple 00_H, FF_H test on 2K Z8 ROM — Tests write

Simple 00_H, FF_H test on 8K overlay RAM — Protect circuitry

RAM failed at xxxx

Write protect circuitry fail — Error messages

Test S: SCOPE TEST

Writes 00_H, FF_H through the 64K address range of the Z80.

Use to test chip select circuitry and output latches.

CAUTION

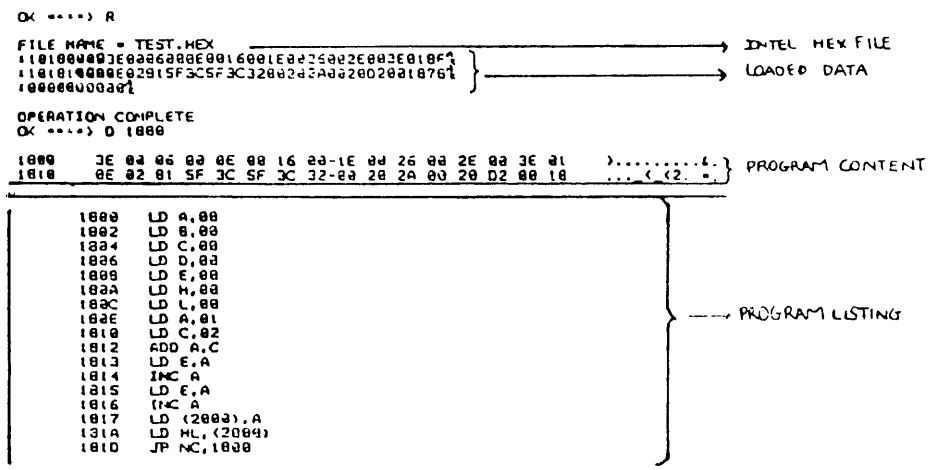
You must power down system to exit Test S.

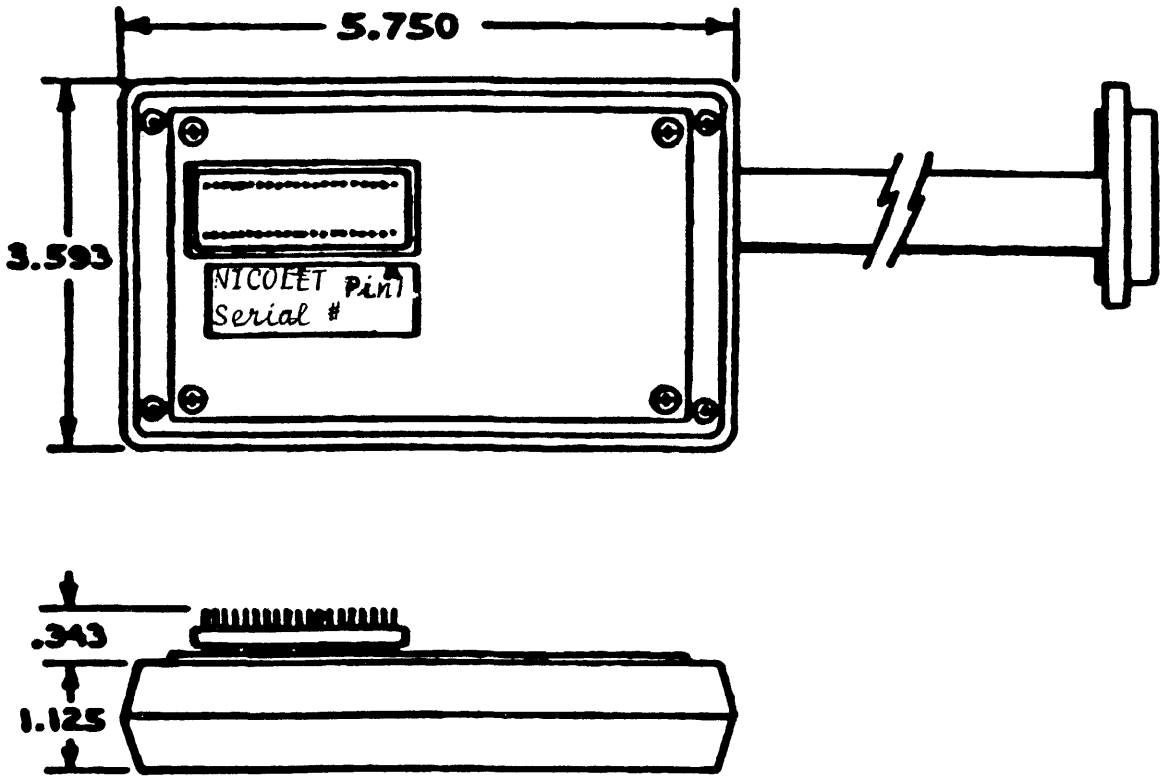
Test V:

Scans through all break RAM addresses, displaying all addresses that are set (=0).

APPENDIX E

SAMPLE DOWNLOAD PROGRAM USING THE R COMMAND




5.750
3.593
NICOLET Part
Serial #
.343
1.125

Nicolet