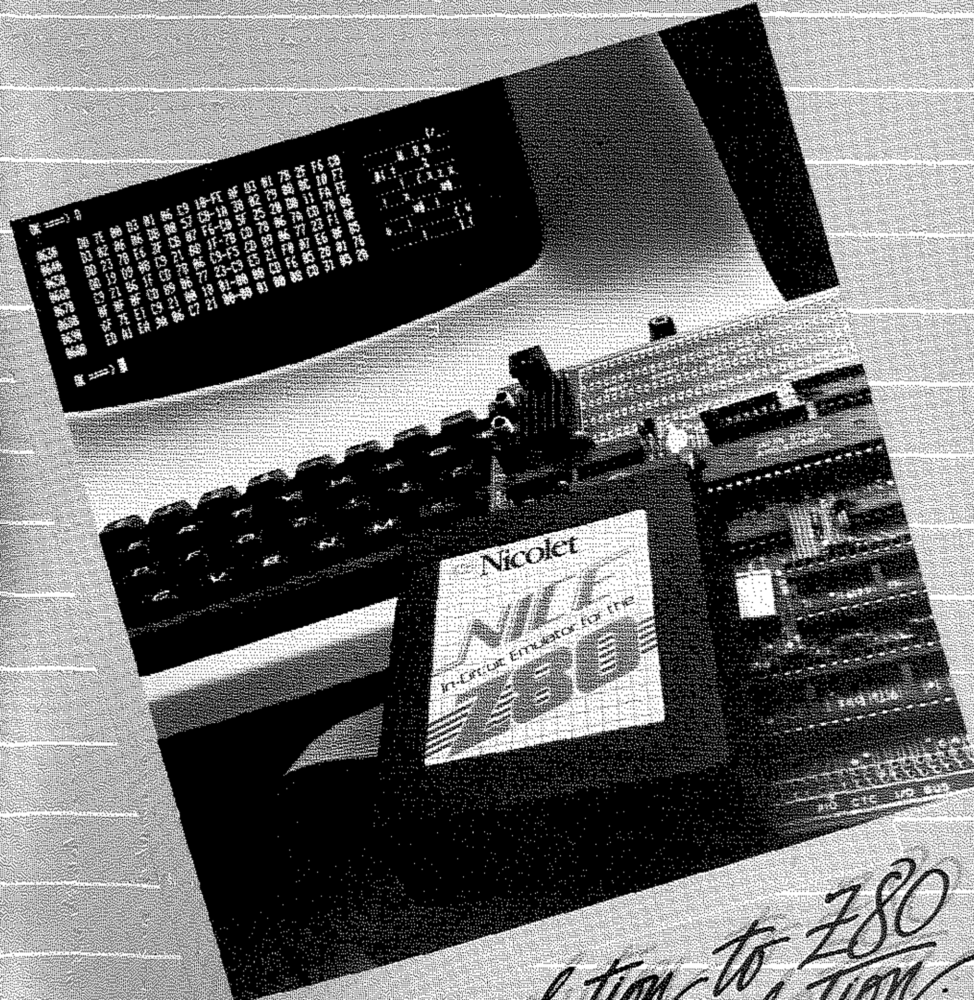
Nicolet

Nicolet Paratronics Corporation
201 Fourier Ave.
Fremont, California 94539
(415) 490-8300

Nicolet

NICE™
In-Circuit Emulator
for the Z80®

User's Guide



The low-cost solution to Z80 emulation.

COMM PORT ON 820

0 6 3
0 6 01
0 6 4
0 6 4C
0 6 5
0 6 EA

0 ϕ F

THEN TYPE H FOR HOST TERMINAL MODE

= OR =

B 07 A = 1200 BAUD
B 0E A = 9600 BAUD

NICE
User's Guide and Reference Manual

Nicolet

Nicolet Paratronics Division
201 Fourier Avenue
Fremont, CA 94539

125-0042-0001 Rev. A

LIMITED WARRANTY

Nicolet Paratronics Corp. ("NPC") 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 NPC 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 NPC unless executed by a duly authorized officer of the company.

This warranty gives you specific legal rights, and 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.

TRADEMARKS

NICE is a registered trademark of Nicolet Paratronics Corporation.
Z80 is a trademark of Zilog.

NICE USER'S GUIDE AND REFERENCE MANUAL
TABLE OF CONTENTS

NICE USER'S GUIDE AND REFERENCE MANUAL
TABLE OF CONTENTS
(continued)

NICE USER'S GUIDE AND REFERENCE MANUAL
TABLE OF CONTENTS
(continued)

CHAPTER I

INTRODUCTION

ABOUT NICE, THE Z80 EMULATOR

Using only six integrated circuits, NICE (for Z80 emulation) is the first of a new first 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 2 3/4" square and 1/2" 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. NICE does not even need a separate power supply and the associated wiring. It obtains its power of 500ma @ 5V directly from the target system.

- **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 revolutionary 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
Refresh function maintained at all times
All I/O ports available to user
All memory addresses available to user
Three break points with 8 bit loop counters
Three print points
Power derived from the target system

Interface to user via standard 25 pin RS232 terminal connector
Local or remote operation
Automatic baud rate detection
Small size
Low cost
High reliability

FUNCTIONAL CAPABILITIES

NICE can:

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

COMMAND LINE INTERPRETER CHARACTERISTICS

NICE's versatile command line interpreter allows you to:

- 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 reset. 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 reset. After the baud rate has been determined, you will see the NICE copyright notice followed by the OK prompt. At this point, you may begin entering commands.

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

- 150
- 300
- 600
- 1200
- 2400
- 4800
- 9600
- 19.2 K

Terminal Characteristics

In order to operate with NICE, the terminal must be set up with the following characteristics:

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

- 8 bits of data
- Parity disabled
- 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 +5 and 0 volts 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 15mA 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.

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.

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

o The ERR prompt (ERR =>) 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: Back-spaces 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 (ST) from either mode. If the bottom line of the display includes the word "RUNNING," NICE is in the GO 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 reset. However, you can use one of two commands to enable or disable this function. QUIT mode Refresh Timing Diagrams are provided in Appendix D.

> **NOTE**
>
> If the target system does <u>not</u> have dynamic RAM, it is a good idea to disable the Refresh function. This allows NICE to operate 25% 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.

Reset Status

After reset, 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:

BP -	Break Point
BPC -	Break Point Count
EBP -	Enable Break Point
DBP -	Disable Break Point
EPP -	Enable Print Point
DPP -	Disable Print Point
EI -	Enable Interrupts
DI -	Disable Interrupts
EB -	Enable Bus
DB -	Disable Bus
DR -	Disable Refresh
ER -	Enable Refresh
H -	Hexadecimal Arithmetic
Q -	Quit
RL -	Repeat Line
ST -	Status
Z -	Sleep

BP - BREAK POINT

Purpose

Sets any of the three break point addresses that work with either the Trace or Untrace commands. **Break points only work in QUIT mode.**

Format

BP Break-Point-Number, Break-Point Address

Break-Point Number must be either 1, 2, or 3, corresponding to the desired break point. Break-Point-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 break point addresses and use the Status (ST) command to display the results.

```
OK ====> ST

 ---> 0290 00 D
 ---> F098 00 D
 ---> E808 00 D
 IBR

OK ====> BP 1 123

OK ====> BP 2 456

OK ====> BP 3 789

OK ====> ST

 ---> 0123 00 D
 ---> 0456 00 D
 ---> 0789 00 D
 IBR

OK ====>
```

BPC - BREAK POINT COUNT

Purpose

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

> **NOTE**
>
> Each of the three break points 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 break point.

After NICE traces each instruction, it compares the enabled break point 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

BPC Break-Point-Number, Pass-Count

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

Examples

```
OK ====> ST

 ---> 0123 00 D
 ---> 0456 00 D
 ---> 0789 00 D
 IBR

OK ====> BPC 1,11

OK ====> BPC 2,22

OK ====> BPC 3,33

OK ====> ST

 ---> 0123 11 D
 ---> 0456 22 D
 ---> 0789 33 D
 IBR

OK ====>
```

EBP - ENABLE BREAK POINT

Purpose

Enables one or all three break points.

Format

This command has two forms:

1. **EBP** - Enables all three break points.

2. **EBP Break-Point-Number** - Enables break point 1, 2, or 3, as specified.

DBP - DISABLE BREAK POINT

Purpose

Disables one or all three break points.

Format

This command has two forms:

1. **DBP** - Disables all three break points.

2. **DPB Break-Point-Number** - Disables break point 1, 2, or 3, as specified.

Examples

```
OK ====> ST

 ---> 0123 11 D
 ---> 0456 22 D
 ---> 0789 33 D
 IBR

OK ====> EBP

OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 E
 ---> 0789 33 E
 IBR

OK ====> DBP 2

OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D
 ---> 0789 33 E
 IBR

OK ====>
```

EPP - ENABLE PRINT POINT

Purpose

Enables one or all of the three print points. Print points 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:

- a print point is enabled, and
- a match is made between the program counter and the associated break point address.

> **NOTE**
>
> The print point and break point functions are independent. The only similarity is that they both use the same address for comparison.

Format

There are two forms of the EPP command:

1. **EPP** - Enables all three print points.

2. **EPP Print-Point-Number** - Enables the specified print point.

DPP - DISABLE PRINT POINT

Purpose

Disables one or all of the three print points.

Format

There are two forms of the DPP command:

1. **DPP** - Disables all three print points.

2. **DPP Print-Point-Number** - Disables the specified print point.

Examples

```
OK ====> EPP

OK ====> ST

 ---> 0123 11 E P
 ---> 0456 22 D P
 ---> 0789 33 E P
 IBR

OK ====> DPP 1

OK ====> DPP 3

OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 IBR

OK ====>
```

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 reset of the target system.

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

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

```
OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 IBR

OK ====> DI

OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 .BR

OK ====> EI

OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 IBR

OK ====>
```

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

```
OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 IBR

OK ====> DB

OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 I.R

OK ====> EB

OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 IBR

OK ====>
```

DR - DISABLE REFRESH

Purpose

When NICE is in the QUIT mode, data is read every 1.25 msec 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% faster.

ER - ENABLE REFRESH

Purpose

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

Examples

```
OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 IBR

OK ====> DR

OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 IB.

OK ====> ER

OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 IBR

OK ====>
```

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.

Examples

```
OK ====> H 1000,45
1045,0FBB
OK ====> H 4000,1000
5000,3000
OK ====> H FFFF,1
0000,FFFE
OK ====>
```


Q - QUIT

Purpose

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

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

Format

Q

Examples

```
OK ====> Q
.Z.V..  A=00  BC=01C1  DE=E250  HL=E704  S=E8F4  P=E699  LD A,B
SZHVNC  A'=FF B'=FFFF  D'=FFFF  H'=FFFF  X=FFFF  Y=FFFF  I=00
OK ====>
```

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

```
OK ====> E FF
 ---> FE

OK ====> E FF;RL
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
```


ST - STATUS

Purpose

Shows information about the following:

- Break Points
- Print Points
- Interrupts
- Bus Requests
- Refresh Function
- GO or QUIT mode

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

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

Examples

```
OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 IBR

OK ====> G

EXECUTION BEGINS AT ===> E69F

OK ====> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 IBR  RUNNING

OK ====>
```


Z - SLEEP

Purpose

Introduces delay into the command line interpreter.

Format

Z Delay-Count

Each count represents 1.25 ms of delay. Since Delay-Count is a 16-bit value, you can generate delays from 1.25 ms to as long as 80 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 'FF.' The CRT update is slow enough to be viewed.

```
OK ====> E FF;Z 100;RL
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
 ---> FE
```

CHAPTER 6
QUIT MODE COMMANDS

In QUIT mode, NICE recognizes the following commands:

A - Assemble into RAM
D - Display Memory
E - Examine Input Port
F - Fill
G - Go
L - List in Assembler Format
M - Move
MT - Memory Test
O - Output
R - Read Intel Hex File
S - Substitute Into Memory
SR - Soft Reset
T - Trace
U - Untrace
V - Verify
X - Xamine

C - CLEAR BP & PP

A - ASSEMBLE INTO RAM

Purpose

Allows you to enter Z80 assembly language at the terminal. Each line is compiled 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

```
OK ===> A 8000

8000 LD A,00
8002 LD BC,0000
8005 INC A
8006 INC B
8007 CP 40
8009 JR NZ,8005
800B LD B,00
800D JR 8005
800F

OK ===>
```


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 reset.

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

```
OK ====> D E000

E000    C3 0F E0 C3 13 E0 C3 32-E0 C3 39 E0 C3 2D E0 AF    .......2..9..-..
E010    C3 15 E0 3E 80 32 49 E8-37 9F FA 1E E0 C7 31 00    ...>.2I.7.....1.
E020    E9 21 2B E0 E5 CD EF E6-C3 B4 E0 1A E0 3E FF C3    .!+..........>..
E030    41 E0 32 4E E8 AF C3 3E-E0 32 4E E8 3E 80 32 49    A.2N...>.2N.>.2I
E040    E8 22 4A E8 21 00 00 39-31 00 E9 E5 D5 C5 3C C2    ."J.!..91.....<.
E050    58 E0 CD EF E6 C3 76 E0-21 20 E8 35 F2 7B E0 34    X.....v.! .5.{.4
E060    CD B9 E6 CA B0 E0 3D CA-89 E0 3D CA A0 E0 3D CA    ......=...=...=.
E070    94 E0 3D CA 9A E0 3E 01-C3 29 E4 CD 50 E6 23 46    ..=...>..)..P.#F

OK ====> D

E080    3A 4E E8 77 2B 70 C3 76-E0 36 02 78 E6 0F 32 21    :N.w+p.v.6.x..2!
E090    E8 C3 76 E0 11 80 7F C3-A3 E0 11 00 7F C3 A3 E0    ..v.............
E0A0    11 7F 80 78 CD 8B E6 FA-76 E0 CD EC E5 C3 A3 E0    ...x....v.......
E0B0    78 32 21 E8 21 03 14 22-47 E8 31 FA E8 CD 50 E6    x2!.!.."G.1...P.
E0C0    5E 23 56 EB 23 22 4C E8-36 02 2B 11 4E E8 0E 07    ^#V.#"L.6.+.N...
E0D0    CD 82 E6 CD B9 E6 47 0E-C1 21 E2 E7 7E B7 CA 1E    ......G..!......
E0E0    E4 23 5E 23 56 23 05 F2-DC E0 07 DA 1E E4 D5 F5    .#^#V#..........
E0F0    3A 4E E8 CD 8B E6 FA 20-E4 C2 1F E4 3A 49 E8 07    :N...... ....:I..

OK ====> D E000,E01F

E000    C3 0F E0 C3 13 E0 C3 32-E0 C3 39 E0 C3 2D E0 AF    .......2..9..-..
E010    C3 15 E0 3E 80 32 49 E8-37 9F FA 1E E0 C7 31 00    ...>.2I.7.....1.

OK ====>
```


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

```
OK ====> E FF
 ---> FE

OK ====> E 05
 ---> FF

OK ====> E FE
 ---> FF

OK ====>
```

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

```
OK ====> F 8000,8800,AB

OK ====> D 8000

8000    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8010    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8020    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8030    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8040    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8050    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8060    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8070    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................

OK ====> F 8034,803C,00

OK ====> D 8000

8000    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8010    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8020    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8030    AB AB AB AB 00 00 00 00-00 00 00 00 AB AB AB     ................
8040    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8050    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8060    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................
8070    AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB     ................

OK ====>
```


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

```
OK ====> G

EXECUTION BEGINS AT ===> E687

OK ====> Q

..H...  A=00  BC=00A1  DE=100E  HL=EA00  S=E8F4  P=E6EB  M(P)=B
SZHVNC A'=7F  B'=7FFF  D'=FFFF  H'=FFFF  X=F9AB  Y=FBDF  I=00

OK ====> G E000

EXECUTION BEGINS AT ===> E000
```

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

```
OK ===> L 8000

8000 LD A,00
8002 LD BC,0000
8005 INC A
8006 INC B
8007 CP 40
8009 JR NZ,8005
800B LD B,00
800D JR 8005
800E NOP
800F NOP
8010 NOP
8011 NOP
8012 NOP
8013 NOP
8014 NOP
8015 NOP

OK ===>
```

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

```
OK ====> F 8000,8800,00

OK ====> M E000,E01F,8000

OK ====> D E000

E000    C3 0F E0 C3 13 E0 C3 32-E0 C3 39 E0 C3 2D E0 AF     .......2..9..-..
E010    C3 15 E0 3E 80 32 49 E8-37 9F FA 1E E0 C7 31 00     ...>.2I.7.....1.
E020    E9 21 2B E0 E5 CD EF E6-C3 B4 E0 1A E0 3E FF C3     .!+..........>..
E030    41 E0 32 4E E8 AF C3 3E-E0 32 4E E8 3E 80 32 49     A.2N...>.2N.>.2I
E040    E8 22 4A E8 21 00 00 39-31 00 E9 E5 D5 C5 3C C2     ."J.!..91.....<.
E050    58 E0 CD EF E6 C3 76 E0-21 20 E8 35 F2 7B E0 34     X.....v.! .5.{.4
E060    CD B9 E6 CA B0 E0 3D CA-89 E0 3D CA A0 E0 3D CA     ......=...=...=.
E070    94 E0 3D CA 9A E0 3E 01-C3 29 E4 CD 50 E6 23 46     ..=...>..)..P.#F

OK ====> D 8000

8000    C3 0F E0 C3 13 E0 C3 32-E0 C3 39 E0 C3 2D E0 AF     .......2..9..-..
8010    C3 15 E0 3E 80 32 49 E8-37 9F FA 1E E0 C7 31 00     ...>.2I.7.....1.
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 ====>
```


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

Examples

```
OK ====> MT 8000,8800

OK ====> MT 9800,A000

OK ====> MT B800,C000
C000 ---> 01/FF - W/R
C000 ---> 02/FF - W/R
C000 ---> 04/FF - W/R
C000 ---> 08/FF - W/R
C000 ---> 10/FF - W/R
C000 ---> 20/FF - W/R
C000 ---> 40/FF - W/R
C000 ---> 80/FF - W/R
C000 ---> FE/FF - W/R
C000 ---> FD/FF - W/R
C000 ---> FB/FF - W/R
C000 ---> F7/FF - W/R
C000 ---> EF/FF - W/R
C000 ---> DF/FF - W/R
C000 ---> BF/FF - W/R
C000 ---> 7F/FF - W/R

OK ====>
```


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

```
OK ====> O FF,1A

OK ====> O FF,1A,1B,1C,1D,1E

OK ====>
```

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
:0F8000003E000100003C04FE4020FA060018F686
: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
:0F8000003E000100003C04FE4020FA060018F686
: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

2. **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.

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

```
OK ====> S 80F8
80F8 7F --->
80F9 7F --->
80FA 7F --->
80FB 7F --->
80FC 7F --->
80FD 7F --->
80FE 7F --->
80FF 7F --->
8100 3E ---> 3E
8101 7F ---> 00
8102 7F ---> 01
8103 7F ---> 00
8104 7F ---> 00
8105 7F ---> 3C
8106 7F ---> 04
8107 7F ---> FE
8108 7F ---> 99
8109 7F ---> -
8108 99 ---> 40
8109 7F ---> 20
810A 7F ---> FA
810B 7F ---> 06
810C 7F --->
810D 7F ---> 18
810E 7F ---> F6
810F 7F ---> .

OK ====> D 80F0

80F0    7F 7F 7F 7F 7F 7F 7F 7F-7F 7F 7F 7F 7F 7F 7F    ................
8100    3E 00 01 00 00 3C 04 FE-40 20 FA 06 7F 18 F6 7F    >....<..@.......
8110    7F 7F 7F 7F 7F 7F 7F 7F-7F 7F 7F 7F 7F 7F 7F    ................
8120    7F 7F 7F 7F 7F 7F 7F 7F-7F 7F 7F 7F 7F 7F 7F    ................
8130    7F 7F 7F 7F 7F 7F 7F 7F-7F 7F 7F 7F 7F 7F 7F    ................
8140    7F 7F 7F 7F 7F 7F 7F 7F-7F 7F 7F 7F 7F 7F 7F    ................
8150    7F 7F 7F 7F 7F 7F 7F 7F-7F 7F 7F 7F 7F 7F 7F    ................
8160    7F 7F 7F 7F 7F 7F 7F 7F-7F 7F 7F 7F 7F 7F 7F    ................

OK ====>
```


SR - SOFT RESET

Purpose

Clears all the Z80 registers and flags so that they read zero. This command is especially helpful for tracing code; it is easier to see when registers are changed by the various instructions.

Examples

```
OK ====> X

S....C  A=FD  BC=0000  DE=1000  HL=0280  S=E8FC  P=E719  RRA
SZHVNC  A'=FF B'=FFFF  D'=FFFF  H'=FFFF  X=FFFF  Y=FFFF  I=00

OK ====> SR

OK ====> X

......  A=00  BC=0000  DE=0000  HL=0000  S=0000  P=0000  RST 38
......  A'=00 B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

OK ====>
```

.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 break points are enabled, encountering a break point 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.

CAUTION: USE OF THE T COMMAND WITHOUT THE . (DOT) MAY RESULT IN DESTRUCTION OF DATA IN A CONTROL REGISTER

Examples

```
OK ====> X P

P=0000 ---> E000

OK ====> X

......   A=00  BC=0000  DE=0000  HL=0000  S=0000  P=E000  JP E00F
......  A'=00  B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

OK ====> T

......   A=00  BC=0000  DE=0000  HL=0000  S=0000  P=E00F  XOR A
......  A'=00  B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

OK ====> T 5

.Z.V..   A=00  BC=0000  DE=0000  HL=0000  S=0000  P=E010  JP E015
......  A'=00  B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

.Z.V..   A=00  BC=0000  DE=0000  HL=0000  S=0000  P=E015  LD (E849),A
......  A'=00  B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

.Z.V..   A=00  BC=0000  DE=0000  HL=0000  S=0000  P=E018  SCF
......  A'=00  B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

.Z.V.C   A=00  BC=0000  DE=0000  HL=0000  S=0000  P=E019  SBC A,A
......  A'=00  B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

S.H.NC   A=FF  BC=0000  DE=0000  HL=0000  S=0000  P=E01A  JP M,E01E
......  A'=00  B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

OK ====>
```

• 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 break points cause the Untrace Command to terminate and the display to be generated. Enabled print points 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.

CAUTION: USE OF THE "U" COMMAND WITHOUT THE • (DOT) MAY RESULT IN DESTRUCTION OF DATA IN A CONTROL REGISTER

Examples

```
OK ====> STR

ERR ===> ST

 ---> 0123 11 E
 ---> 0456 22 D P
 ---> 0789 33 E
 IBR

OK ====> SR

OK ====> X P

P=0000 ---> E000

OK ====> U

BACK TRACE    ------> -4=0000  -3=0000  -2=0000 -1=E000
......   A=00  BC=0000  DE=0000  HL=0000  S=0000  P=E00F  XOR A
......  A'=00  B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

OK ====> U 100

BACK TRACE    ------> -4=E6DA  -3=E6DD  -2=E6DE -1=E6DF
S....C   A=00  BC=0000  DE=1000  HL=E902  S=E8FA  P=E6E0  LD (HL),D
......  A'=00  B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

OK ====> U 1001

BACK TRACE    ------> -4=E683  -3=E684  -2=E685 -1=E686
....N.   A=7F  BC=0008  DE=E81E  HL=E750  S=E8FA  P=E687  JP NZ,E682
......  A'=00  B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

OK ====>
```

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

```
OK ====> F 8000,9000,00

OK ====> M E000,E7FF,8000

OK ====> D E000

E000    C3 0F E0 C3 13 E0 C3 32-E0 C3 39 E0 C3 2D E0 AF     .......2..9..-..
E010    C3 15 E0 3E 80 32 49 E8-37 9F FA 1E E0 C7 31 00     ...>.2I.7.....1.
E020    E9 21 2B E0 E5 CD EF E6-C3 B4 E0 1A E0 3E FF C3     .!+.........>..
E030    41 E0 32 4E E8 AF C3 3E-E0 32 4E E8 3E 80 32 49     A.2N...>.2N.>.2I
E040    E8 22 4A E8 21 00 00 39-31 00 E9 E5 D5 C5 3C C2     ."J.!..91.....<.
E050    58 E0 CD EF E6 C3 76 E0-21 20 E8 35 F2 7B E0 34     X.....v.! .5.{.4
E060    CD B9 E6 CA B0 E0 3D CA-89 E0 3D CA A0 E0 3D CA     .......=...=...=.
E070    94 E0 3D CA 9A E0 3E 01-C3 29 E4 CD 50 E6 23 46     ..=...>..)..P.#F

OK ====> D 8000

8000    C3 0F E0 C3 13 E0 C3 32-E0 C3 39 E0 C3 2D E0 AF     .......2..9..-..
8010    C3 15 E0 3E 80 32 49 E8-37 9F FA 1E E0 C7 31 00     ...>.2I.7.....1.
8020    E9 21 2B E0 E5 CD EF E6-C3 B4 E0 1A E0 3E FF C3     .!+.........>..
8030    41 E0 32 4E E8 AF C3 3E-E0 32 4E E8 3E 80 32 49     A.2N...>.2N.>.2I
8040    E8 22 4A E8 21 00 00 39-31 00 E9 E5 D5 C5 3C C2     ."J.!..91.....<.
8050    58 E0 CD EF E6 C3 76 E0-21 20 E8 35 F2 7B E0 34     X.....v.! .5.{.4
8060    CD B9 E6 CA B0 E0 3D CA-89 E0 3D CA A0 E0 3D CA     .......=...=...=.
8070    94 E0 3D CA 9A E0 3E 01-C3 29 E4 CD 50 E6 23 46     ..=...>..)..P.#F

OK ====> S 8040
8040 E8 ---> 00
8041 22 ---> _

OK ====> V E000,E7FF,8000

E040 ---> E8 00
OK ====>
```

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, 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

```
OK ====> X

....N.  A=7F  BC=0008  DE=E81E  HL=E750  S=E8FA  P=E687  JP NZ,E682
......  A'=00 B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

OK ====> SR

OK ====> X

......  A=00  BC=0000  DE=0000  HL=0000  S=0000  P=0000  RST 38
......  A'=00 B'=0000  D'=0000  H'=0000  X=0000  Y=0000  I=00

OK ====> X P

P=0000 ---> E000

OK ====> X H'

H'=0000 ---> 1234

OK ====> X S

S=0000 --->

OK ====> X

......  A=00  BC=0000  DE=0000  HL=0000  S=0000  P=E000  JP E00F
......  A'=00 B'=0000  D'=0000  H'=1234  X=0000  Y=0000  I=00

OK ====> X F

F=......  ---> HCV

OK ====> X

..HV.C  A=00  BC=0000  DE=0000  HL=0000  S=0000  P=E000  JP E00F
......  A'=00 B'=0000  D'=0000  H'=1234  X=0000  Y=0000  I=00

OK ====>
```

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).
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 to clear all internal registers and flags.
T	Trace one instruction.
T d16	Trace one or more instructions.
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

Go or QUIT Mode

BP 1, d16	Set break point address 1 to the specified 16 bit value (d16).
BP 2, d16	Set break point address 2 to the specified 16 bit value (d16).
BP 3, d16	Set break point address 3 to the specified 16 bit value (d16).
BPC 1, d8	Set break point pass counter 1 to the specified 8 bit value (d8).
BPC 2, d8	Set break point pass counter 2 to the specified 8 bit value (d8).
BPC 3, d8	Set break point pass counter 3 to the specified 8 bit value (d8).
• C	CLEAR ALL BREAKPOINTS & PRINTPOINTS
EBP	Enable all three break points.
EBP n	Enable break point 1, 2, or 3 where (n) is the break point number.
DBP	Disable all three break points.
DBP n	Disable break point 1, 2, or 3 where (n) is the break point number.
EPP	Enable all print points.
EPP n	Enable print point 1, 2, or 3 where (n) is the print point number.
DPP	Disable all print points.
DPP n	Disable print point 1, 2, or 3 where (n) is the print point 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 in order to provide a scope loop.
ST	Display the emulator status, including break point addresses pass counters break point enable or disable print point 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. (Good for scope loops)

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

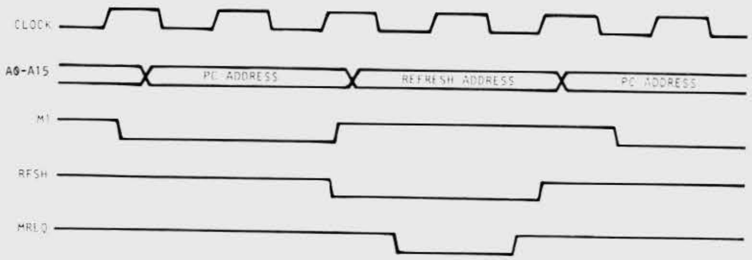
CROMEMCO ZPU

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

QUIT MODE REFRESH CYCLES FOR NICE

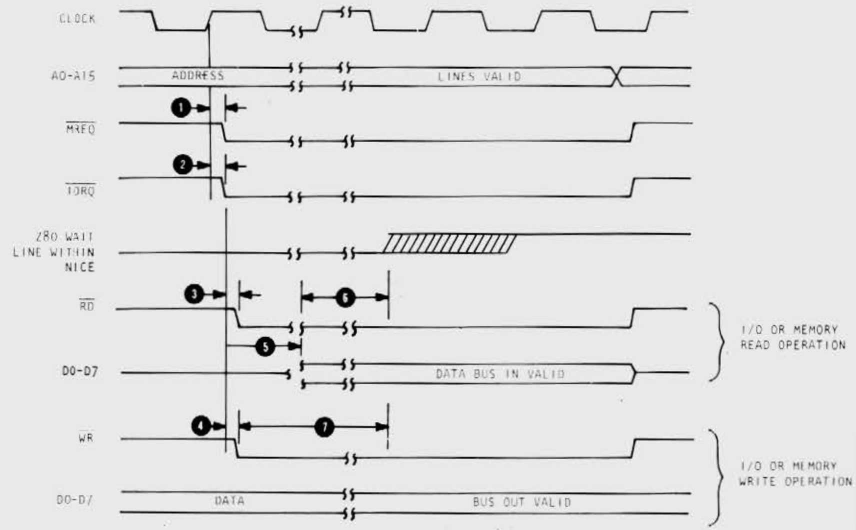


APPENDIX D

ADDITIONAL TIMING REQUIREMENTS FOR NICE MEMORY OR I/O CYCLES*

NOTE

These requirements do not apply when NICE is in full speed execution.



*		MIN	TYP	MAX
1	CLK↑ to MREQ↓	5ns	15ns	25ns
2	CLK↑ to IORQ↓	5ns	15ns	25ns
3	CLK↑ to RD↓	10ns	30ns	50ns
4	CLK↑ to WR↓	10ns	30ns	50ns
5	MREQ or IORQ to DATA IN VALID	2 s	3 s	4 s
6	DATA BUS IN VALID TO WAIT↑	2 s	3 s	4 s
7	WR to WAIT↑	2 s	3 s	4 s

APPENDIX E

SAMPLE DOWNLOAD PROGRAM USING THE R COMMAND

INDEX

(Commands appear in boldface)

For applications assistance please contact us on our Logic Analyzer Technical Hotline:

800/NICOLET (642-6538) - Toll Free Outside California

415/490-8300 - California

NICOLET PARATRONICS CORPORATION

201 Fourier Avenue

Fremont, California 94539

TWX: 910/381-7030

NICE USER'S GUIDE UPDATE

This update revises the emulator commands for the TRACE an
UNTRACE functions and adds a new command for a CLEAR functio

The TRACE command, T, pages 34, 54, and 63, of the Guide hav
been changed to .T and .T d16. The UNTRACE command, U, page
34, 56, and 63, of the Guide have been changed to .U and .U d16.

The new CLEAR command, .C, has been added to clear a
breakpoints and printpoints. Since the breakpoints and printpoin
are uncontrolled at powerup and upon reset, it is important t
perform the CLEAR function after power up and after each reset.

CAUTION: Use of the T only or U only commands without the . (dot) may result in destruction of data in a control register!

Nicolet

NICE NEWSLETTER

Nicolet is nearing the end of the second month of shipments of the NICE emulator. We have received a number of good comments, which we appreciate. We have also had a few problems which have inconvenienced some of you, and for this we apologize.

We would like to take this opportunity to summarize some of the common problems and the resolution of them. We will also mention a few of the unique problems with the request that if you have experienced a similar problem and resolved it, you will pass the information on to us.

I. COMMON PROBLEMS

1. Units through Serial #1070 experienced a malfunction when using TRACE (T) and UNTRACE (U) commands. This was remedied by adding the .C command and changing the TRACE command to .T and the UNTRACE to .U.

2. On all units, the SUBSTITUTE INTO MEMORY command S was inadvertantly left in the User's Guide. The only form of the substitute command is S START-ADDRESS.

3. The pin assignments for setting up the communications connector as described on page 6 of the User's Guide are incorrect. Received Data should read PIN 3 (PIN 2) and Transmitted Data should read PIN 2 (PIN 3).

   For those user's who may be establishing a communications interface other than with the RS232 cable supplied, please note that there may be a degradation in the performance of NICE as the cable length is extended.

4. Pages 36 and 43 of the manual show examples of assembly and disassemblys, commands "A" and "L". The mnemonics displayed at address 8009 should read JR NZ, $-4 and at address 800D, JR $-9.

Nicolet Paratronics Corporation
201 Fourier Avenue
Fremont, California 94539

II. UNIQUE PROBLEMS

1. NICE does not function properly with the Freedom 100 terminal. We have not had a chance to review specifications on this terminal and thus have not addressed this problem.

2. User's with Prog Log target systems have experienced problems in reading the Z80 registers correctly. Upon entering the QUIT mode, all Z80 registers are filled with F's. An explanation of this problem is as follows: The NICE unit uses the Z80 data bus of the target system. When NICE uses the bus, it drives $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$ high. At this point the target system should not have any devices driving the Z80 data bus. If the target system does drive the Z80 data bus while no memory or I/O requests are being processed, NICE will not function properly. On the Pro Log, this problem was fixed by cutting one trace on the target board. Careful examination of system timing is essential if you have a similar problem.

3. Downloading Intel Hex Files

   We have had one user indicate that he could download at 300 baud but not at 9600 baud. The reason for this is that NICE processes data at approximately 300 baud. NICE gives a Clear to Send signal for each byte of data. Thus the Clear to Send signal is present for one byte then is removed until that byte is processed. If the transmitting system is not monitoring or not observing the NICE Clear to Send signals, the NICE buffer will be overloaded causing NICE to fail.

   If you have a system that requires modification, such as the Cromenco ZPU described on page 69 of the User's Guide or the Pro Log previously described, we will be glad to supply you with such NICE specifications as are necessary to assist in identifying the modification.

   NOTE: If you intend to use your personal computer as the terminal, you will need to develop your own application program.