2

This is an informal report intended primarily for internal or limited external distribution. (The opinions and conclusions stated are those of the author and may or may not be those of the laboratory.) ~~This report is not to be given additional external distribution or cited in external documents without the consent of the author or LLL Technical Information Department.~~

UCID- 16251

LAWRENCE LIVERMORE LABORATORY

University of California/Livermore, California

SIMULATOR PROGRAM FOR THE INTEL MCS-8 800 CPU

Waldo Magnuson, Jr.

3/19/73

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

REFER QUESTIONS TO:		LAWRENCE RADIATION LABORATORY		ELECTRONICS ENGINEERING REPORT LIVERMORE, CALIFORNIA		UNIVERSITY OF CALIFORNIA		RECORDED		LER 72-103403	
Waldo Magnuson, Jr.		ORIG. W.C. Magnuson, Jr.		SIMULATOR PROGRAM FOR THE INTEL MCS-8 8008 CPU		DATE 3/19/73		REV.			
APPROVED						PAGE 1		OF 35			

Introduction

The INTEL 8008 CPU simulator program is now available on the LLL OCTOPUS CDC-7600 computer system. The program (MCS8SIM) provides a software simulation together with simulator execution commands to aid in program development for the MCS-8. Besides accepting 8008 machine code the simulator allows: Manipulation of the simulated memory and 8008 registers, breakpoint insertion, program tracing, symbolic and numeric dumps, single stepping, and other operations during the simulation. This report states how the simulator (MCS8SIM) is accessed and how it is run on the OCTOPUS network. The INTEL manual "8008 Simulator Software Package" appears as appendix A and describes the simulator, the simulator commands and error messages, and presents examples. A few differences in usage exist between the INTEL manual examples and the use at LLL. These differences will be pointed out. The program is called MCS8SIM at LLL rather than INTERP/8 as in the INTEL manual.

Reading the MCS8SIM Program from Photostore or EELIB

The MCS8SIM program is stored in the ELEPHANT photostore under the "take" directory: .558850:INTEL and can be read from the photostore as follows:

```
User:      ELF / .5 .3
           .RDS .558850:INTEL:MCS8SIM
           .END
Machine:    RDS
```

ALL DONE

The program is also stored in Public library file EELIB and can be read and started into execution by:

```
User:      X EELIB MCS8SIM / .5 .3
Machine:    TYPE INPUT FILE, OUTPUT FILE, CODE FILE, SYMBOL FILE
```

In the case where the program is read from the photostore, the program is transferred to disk. In the second case it is not only transferred but started in execution.

Input to MCS8SIM

When the simulator program is executed, the first line the program types is a request for information and is:

TYPE INPUT FILE, OUTPUT FILE, CODE FILE, SYMBOL FILE

the proper responses are:

```
INPUT FILE:      TTY or the disk file name where the code file in simulator
                  execution command format is stored. If TTY is typed the
                  program expects commands to be input from the teletype.
```

OUTPUT FILE: TTY or disk file name where all simulator output is to go. The disk file will be created by the program.

CODE FILE: Disk file name which contains 8008 machine code in "BPNF" format produced by the 8008 assembler (file MCSBIN).

SYMBOL FILE: Disk file name which contains the symbol table in the form produced by the 8008 assembler (file MCSOUT).

All disk file names must be seven or fewer characters in length. The code and symbol file entries are not required if all input is to be from the TTY or input file.

The LOAD command of the simulator causes the symbol tables and code to be loaded into the simulated MCS-8 memory. The proper unit numbers when running MCS8SIM are:

- LOAD 1. If input is from Teletype.
- LOAD 5. If running from disk file in simulator execution command format.
- LOAD 6. If running from disk file in code (BPNF) format.
- LOAD 7 6. If running from disk code file and disk symbol table file.

The IO formatting commands on pages 11, 14 and in the examples of the INTEL simulator write-up should be disregarded and the above LOAD assignments used instead.

Example Simulator Runs

To better show how the simulator is used at LLL two figures have been prepared. Figure 1 illustrates 1) the operations needed to transfer the MCS-8 assembler and simulator to disk file, 2) the creating of a sample assembly language 8008 program, (using TRIX), 3) the assembly of the program, and 4) adding to the code file a \$ (again using TRIX). The code file must end with a \$.

Figure 2 shows the use of the simulator with input and output on a Teletype. The example is a portion of the second example in the INTEL manual.

Acknowledgements

Thanks is given to W.J. Walsh, Lecturer, UC Berkeley for having the 9 track magnetic tape converted to punched cards. Terry Allison, Computations LLL, deserves the credit for getting the program to run on the OCTOPUS system.

Wald G. Magnuson, Jr.
W.G. Magnuson, Jr.
Electronics Engineering Department

WM:pv
Appendix A
Distribution:

W.G. Magnuson, Jr. (25 copies)
S.A. Nielsen (25 copies)
H.C. McDonald
EE Division Leaders
EE Group Leaders

T. G. Allison L-55
R. K. Cralle L-81
J. A. Johnson E-3, LASO

RECORDED

```

ELF / .5 .3
.RDS .558850:INTEL:MCS8
.RDS .558850:INTEL:MCS8SIM
.END
.RDS
RDS

```

ALL DONE

```

TRIX AC / .5 .3
.C(SIM)
.BLI
;* SAMPLE MCS-8 PROGRAM (PAGE 47 OF 8008 MANUAL)
*START LLI 200
&    LHI 0
&LOOP LAM
&    CPI 46
&    JNZ FOUND
&    CAL INCR
&    LAL
&    CPI 220
&    JFZ LOOP
&FOUND RET
&INCR INL
&    RFZ
&    INH
&    RET
&    END
&
.END

```

ALL DONE

```

MCS8 / .5 .3
PLEASE TYPE INPUT FILE NAME (A7)
SIM
TYPE LINE-FEED OR TAPE VAULT NO.

```

```

=====
8008 INTEL ASSEMBLER
=====

```

ALL DONE

```

TRIX AC / .5 .3
.O(MCSBIN)
66 LINES.
.AL66
&$
&
.END

```

ALL DONE

RECORDED

LER72-103403

Page 4

MCSKSIM / .5 .3

TYPE INPUT FILE, OUTPUT FILE, CODE FILE, SYMBOL FILE
TTY TTY MCSBIN MCSOUT
BEGIN

LOAD 7 6.

256 LOAD OK

DISPLAY SYMBOLS.

0000000 00000 0000H START
0000040 00004 0004H LOOP
0000230 00019 0013H FOUND
0000240 00020 0014H INCR

DI MEM 0 TO 100 HEX, MEM 0 TO 100 CODE.

00000 36H C8H 2EH 00H C7H 3CH 2EH 68H 13H 00H 46H 14H 00H C6H 3CH DCH
00016 48H 04H 00H 07H 30H 08H 28H 07H 00H 00H 00H 00H 00H 00H 00H
00032 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H

00096 00H 00H 00H 00H 00H

00000 LLI, C8H LHI, 00H LAM CPI, 2EH JTZ, 13H, 00H CAL, 14H, 00H LAL CPI, DCH
00016 JFZ, 04H, 00H RET INL REF INH RET HLT HLT HLT HLT HLT HLT HLT
00032 HLT HLT HLT HLT HLT, HLT HLT HLT HLT HLT HLT HLT HLT HLT HLT

00096 HLT HLT HLT HLT HLT

SET MEM 200 TO 210 = 43 46 48 20H 1111000B.

SET OK

GO.

EXECUTION ERROR 2 AT 22

DI MEM 19 COD.

00019 RET

SET M 19 = 0. DI MEM 19 CO.

SET OK

00019 HLT

SET CPU. GO.

SET OK

HLT CYCLE 117

DI CPU.

CYZSP A B C D E H L HL SP PS0
*0101*046*000*000*000*000*000*201*00201*000*00019

TIME.

TIME=117

END.

SEOF

ALL DONE

INTEL CORP. 3065 Bowers Avenue, Santa Clara, California 95051 • (408) 246-7501

MCS-8^{T.M.}

8008 Simulator Software Package

This manual describes the use of a FORTRAN IV program called INTERP/8. This program provides a software simulation of the INTEL 8008 CPU, along with execution monitoring commands to aid program development for the MCS-8.

Reproduced With The
Permission Of The
INTEL Corporation.
March 1973

NOVEMBER 1972

TABLE OF CONTENTS

I.	Introduction.....	1
II.	Basic Elements.....	1
	A. Numbers.....	1
	B. Symbolic Names.....	2
	C. Special Characters.....	2
III.	Interp/8 Commands.....	3
IV.	IO Formatting Commands.....	14
V.	Error Messages.....	16
VI.	Examples.....	17

I. INTRODUCTION.

This manual describes the use of a FORTRAN IV program called INTERP/8. This program provides a software simulation of the INTEL 8008 CPU, along with execution monitoring commands to aid program development for the MCS-8.

INTERP/8 accepts machine code produced by the INTEL 8008 Assembler, along with execution commands from a timesharing terminal, card reader, or disk file. The execution commands allow manipulation of the simulated MCS 8 memory and the 8008 CPU registers. In addition, operand and instruction breakpoints may be set to stop execution at crucial points in the program. Tracing features are also available which allow the CPU operation to be monitored. INTERP/8 provides symbolic reference to storage locations as well as numeric reference in various number bases. The command language is described in the paragraphs which follow.

II. BASIC ELEMENTS.

All input to INTERP/8 is "free form." Numbers, symbolic names, and special characters may be placed anywhere within the input line (see margin commands in Section IV). Comments may be interspersed in the input, but must be enclosed within the bracketing symbols /* and */.

1. Numbers. Numeric input to INTERP/8 can be expressed in binary, octal, decimal, or hexadecimal. The letters B, O, Q, D, and H following the integer number indicates the base, as shown below:

<u>number</u>	<u>value</u>
11011B	11011 ₂
28D	28 ₁₀
33O	33 ₈
33Q	33 ₈
1CH	1C ₁₆
28	28 ₁₀

A decimal number is assumed if the base is omitted. Note that although O is allowed to indicate octal integers, Q is also permitted to avoid confusion with the integer 0. Note that the leading digit of a hexadecimal number must be one of the digits 0,1,...,9. Thus, EF2₁₆ must be expressed as 0EF2H.

On output, INTERP/8 indicates octal integers with Q and omits the D on decimal values. The base used on output defaults to decimal, but may be changed by the user (see the BASE command in Section III).

2. Symbolic Names. Symbolic names are strings of contiguous alphabetic and numeric characters not exceeding 32 characters in length. The first character must be alphabetic. Valid symbolic names are:

SYMBOLICNAME
X3
G1G2G3
LONGSTRINGOFCHARACTERS

3. Special Characters. The special characters recognized by INTERP/8 are: \$ = . / () + - ' * , . All other special characters are replaced by a blank.

III. INTERP/8 COMMANDS.

The commands available in INTERP/8 are summarized briefly below. Full details of each command are given in following paragraphs.

<u>command</u>	<u>purpose</u>
LOAD	causes symbol tables and code to be loaded into the simulated MCS-8 memory
GO	starts execution of the loaded 8008 code
INTER	simulates an 8008 interrupt
TIME	displays time used in the 8008 simulation
CYCLE	allows the simulated CPU to be stopped after a given number of cycles
TRACE	enables tracing feature when particular portions of the program are executed
REFER	causes the CPU simulation to stop when a particular storage location is referenced
ALTER	causes the CPU simulation to stop when the contents of a particular memory location is altered
CONV	displays the values of numbers converted to the various number bases
DISPLAY	displays memory locations, CPU registers, symbolic locations, and IO ports
SET	allows the values of memory locations, CPU registers, and IO ports to be altered
BASE	allows the default number base used for output to be changed
PUNCH	causes output of machine code in BPNF format
END	terminates execution of an 8008 program.

The commands NOTRACE, NOREFER, and NOALTER are also defined. These commands negate the effects of TRACE, REFER, and ALTER, respectively. In all cases, the commands may be abbreviated (but not misspelled!). These abbreviations are indicated with the command description.

Commands are typed anywhere on the input line, with as many commands on a line as desired. The symbol "." must follow each command.

The end of data for the execution of INTERP/8 is indicated by a "\$EOF" starting in column 1 of the last card.

1. Range-lists. Many of the INTERP/8 commands accept a "range-list" as an operand. Tracing, for example, can be enabled for a specific range of addresses in the program. The range-list specifies a sequence of contiguous addresses in memory, or a range of numeric values to which the command is applied.

In its simplest form, a range-list is a number (binary, octal, decimal, or hexadecimal), or it may be a pair of numbers separated by the symbol "TO."

Thus, valid range-lists are:

10
63Q
50 TO 63Q
0FH TO 11001111B.

A range-list, however, can also reference a symbolic location, with or without a numeric displacement from the location. Suppose, for example, the

symbols START and INCR appear at locations 10 and 32 in the source program.

Valid range-lists involving these symbols are:

START	(same as 10)
START + 6	(same as 16)
START - 101B	(same as 5)
10 TO INCR	(same as 10 TO 32)
START+3 TO INCR-2	(same as 13 TO 30).

The range-list may also contain a reference to the current value of the program counter of the simulated 8008 CPU. The symbol "*" represents this value. If the value of the program counter is 16, for example, the following is a valid range-list:

START TO *	(same as 10 TO 16).
------------	---------------------

The exact use of the range-list is illustrated with the individual commands.

2. Notation. The following notation is used to describe the INTERP/8 command structure. Elements enclosed within braces { and } are optional, while elements enclosed within the brackets [and] are alternatives, where at least one alternative must be present.

A range-list, for example, can be specified as

range-element { TO range-element }

where a range-element is defined as

[number
symbolic-name * { [+ number] }]

As mentioned previously, command names can always be abbreviated.

The required portion of the command is underlined in the command description.

The symbol "TO" in the range list can be abbreviated as "T." Thus, the range list above can be redefined as:

range-element { TO range-element }.

Finally, the ellipses "..." indicate a list of indefinite length.

The commands are given alphabetically in the following paragraphs starting with a prototype statement using the above notation. A brief description is then given, followed by examples.

3. [ALTER
NOALTER] range-list { ,range-list,range-list,...,range-list }.

The ALTER command is an operand breakpoint command which causes the execution of the 8008 CPU to stop whenever an attempt is made by the CPU to store values into a memory location specified in the range-list. When the breakpoint is encountered, INTERP/8 prints ALTER x, where x is the value of the program counter. Execution can be started again with the GO, RUN, or INTER commands. Examples of the command are:

ALTER 0.

ALTER 0 TO 10.

ALTER 10 T INCR.

ALTER START+2 TO INCR - 0AH.

AL 5, START, X2, 7 T 10, INCR-3.

4. BASE $\left\{ \begin{array}{c} \underline{\text{BIN}} \\ \underline{\text{OCT}} \\ \underline{\text{DEC}} \\ \underline{\text{HEX}} \end{array} \right\}.$

This command causes the INTERP/8 system to use the number base specified by the second argument when printing results. This command has no effect on the number bases which are acceptable in the input.

5. CONV range-list {,range-list,range-list,...,range-list}.

The conversion command prints the values of the numbers specified in the range-list in binary, octal, decimal, and hexadecimal forms. Examples are:

CONV 23.

CONV *.

CON 10 TO START+3.

CO 10, 30, 28 Q, 1101B T 33H.

6. CYCLE number.

The cycle command causes a breakpoint to occur when the CPU cycle count reaches its current value plus the number specified in the cycle command (see the GO command, also).

7. DISPLAY display-element {,display-element, ..., display-element}.

The display command causes the values of memory locations, symbolic names, CPU registers, and IO ports to be printed. The output form of these values is determined by the current default base (see the BASE command). The width of the output line determines the output formatting (see the \$WIDTH command of Section IV).

In its simplest form, a display-element can be one of the 8008 CPU registers:

CY	(carry)	D	PS	(entire program stack)
Z	(zero)	E	PS	0
S	(sign)	H	PS	1 (program stack elements)
P	(parity)	L	...	
A		HL	(H & L)	PS 7
B		SP	(program stack pointer)	
C		PC	(program counter)	

In this case, valid DISPLAY commands are:

DISPLAY CY.
DISP CY, Z, H, HL.
D P, A, PS 0.

A display-element can also be the symbol CPU, in which case all registers are displayed.

The values latched into the IO ports can be displayed by using a display element of the form:

PORT range-list

The ports specified in the range-list (between 0 and 31) are printed. Examples are:

DISPLAY PORT 0.
DI PO 3, PO 5, PORT 5 TO 8, PO 1001B.

The contents of the symbol table can be examined by using a display-element of the form:

SYMBOLS { [*
symbolic-name
number] }

The form

DISPLAY SYMBOLS.

prints the entire symbol table, while the form

DISPLAY SYMBOLS number.

responds with the symbolic name (± a numeric displacement) which is closest to the address specified by the number. Examples are:

DISP SY.

DI SY 0FFH, SY 32.

If the symbol "*" is used in the command, the symbolic location closest to the current program counter is printed.

The values contained in memory locations can also be displayed. In this case, the display-element takes the form

MEMORY range-list { { CODE
 BIN
 OCT
 DEC
 HEX } }

The range of elements printed is specified in the range-list, while the form of the elements in the display is controlled by the command CODE (decoded instructions) or one of the number bases. If the form is omitted, the default number base is used in the display (see the BASE command). Valid DISPLAY commands are:

DISPLAY MEMORY 20.

DISP MEM 20 TO 30H.

DI M START T START+5.

DI MEM 0 TO 30 CODE.

D M 0 T 30 D, M 40 TO INCR+10 OCT.

The various display-elements may be mixed in a single DISPLAY command.

8. END.

The END command reinitializes the INTERP/8 system. If another program is subsequently loaded into memory, all break and trace points are reset. Otherwise, the currently loaded program may be rerun with all break and trace points remaining.

9. GO { { *
 { number } } }.

The GO command causes the execution of the loaded program to begin. In the case that a break point was previously encountered, the execution continues through the breakpoint. If the GO is followed by a *, the breakpoint addresses are printed as they are encountered, but the 8008 CPU does not halt until completion. If the GO is followed by a number, the effect is exactly the same as

CYCLE number. GO.

10. INTER { number { number { number } } }.

The INTER command simulates the 8008 interrupt system. The numbers which follow the INTER command correspond to an instruction and its operands which will be "jammed" into the instruction register. If no instructions follow the INTER command, the instructions from the last interrupt are used. If no previous command has been specified, a LAA (NOP) instruction is used. The INTER command causes the simulated execution to continue. Examples are:

INTER.

INT.

INTER 00010101B (this is an RST 20Q).

11. LOAD number { number }.

The LOAD command reads the symbol table and 8008 machine code into the simulated memory. The form

LOAD number.

reads only the machine code from the file specified by number (see file numbering in Section IV). The form

LOAD number number.

reads the symbol table from the file specified by the first number and the machine

code from the second file. The symbol table is in the form produced by the 8008 assembler (i.e., the first part of the listing file), and the machine code is in "BPNF" format (see the INTEL publication "Silicon Gate MOS LSI ROM"). This format is also produced by the INTEL 8008 assembler. The end of the code file is indicated by a "\$" appearing in the input. INTERP/8 responds to this command by printing the number of locations used by the program. Examples are:

LOAD 1.

LOAD 6 7.

12. $\left[\begin{array}{c} \text{REFER} \\ \text{NOREFER} \end{array} \right] \text{range-list } \{, \text{range-list}, \dots, \text{range-list} \}.$

This command is similar to the ALTER command except that a breakpoint occurs whenever any reference to the memory location takes place. Thus, an instruction fetch, an operand fetch, or an operand store all cause a breakpoint when this command is used. Examples are:

REFER 10.

RE 10 TO 30Q.

REF 5, 7, START TO START+5, 7IQ.

NOREF 0 TO 10.

13. RUN.

The RUN command has exactly the same effect as the command GO *.

14. SET set-element $\{, \text{set-element}, \dots, \text{set-element} \}.$

The SET command allows memory locations, CPU registers, and IO ports to be set to specific values. The register names described under the DISPLAY command can be used in the set-element:

register = $\left[\begin{array}{c} \text{number} \\ * \end{array} \right]$

The value of the specified register is set to the number following the "=" or to the value of the program counter if "*" is specified. Thus, valid commands are:

SET Z = 0.

SE A = 3, B = 77Q, PS 0 = 0EEH.

S HL = 28.

A set-element can also be the symbol "CPU" in which case all registers are set to zero, including the simulated 8008 timer. Examples are:

SET CPU.

S CP, PC = 25.

The values of IO ports can also be set by using a set-element of the form

PORT range-list = number {number number ... number}

In this case, the IO ports specified in the range-list are set to the list of numbers following the "=". If more ports are specified than there are numbers in the list, the number are reused starting at the beginning. Examples are:

SET PORT 5 = 10.

SET PO 6 TO 8 = 1 2 3.

S PO 10 TO 13 = 77Q 2.

S PO 8 = 10B, PO 12 = 13H, PO 30Q = 16.

The values contained in memory locations can be altered directly by using a set-element of the form

MEMORY range-list = number {number ... number}

As in the case of IO ports, the memory locations are filled from the list to the right of the equal sign, with numbers being reused if the list is exhausted.

Examples of this command are:

SET MEMORY 0 = 0.

S MEM 0 TO 50 = 0.

The SET command does not change break or trace points which are in effect.

S M START TO START+5 = 11111000B 22Q 33H.

As in the DISPLAY command, set-elements of each type may be intermixed:

SET CP, CY=0, M 5 = 10, PO 6 = 12, PC = 30.

15. TIME.

The TIME command causes INTERP/8 to print the number of states used by the simulated 8008 CPU since the last LOAD, END, or SET CPU command.

16. [TRACE
NOTRACE] range-list { ,range-list, ..., range-list }.

The TRACE command causes the INTERP/8 system to print the CPU register contents and the decoded instruction whenever an instruction is fetched from the memory region specified in the range-list. The form of the elements in the trace is defined by the current default base (see BASE command). The trace shows the register contents and operation code before the instruction is executed. The result of the operation is found in the next line of the trace, or through the DISPLAY CPU command.

A heading showing the various columns in the trace is printed after each tenth line of the trace. Examples of the TRACE command are:

TRACE 0 TO 100.

TR START TO START + 111B.

NOTRACE START, INCR, FOUND TO FOUND+3, 7Q.

17. PUNCH range-list { number }.

The PUNCH command causes the specified region of the simulated memory to be output in the BPNF format. If the number is present, the code is written into the corresponding INTERP/8 output file; otherwise the currently defined file is used. Examples are:

PUNCH 0 TO 0FFH.
PU START TO FINISH.

IV. IO FORMATTING COMMANDS.

INTERP/8 has a generalized IO formatting interface which is somewhat dependent upon the installation. In general, a number of files are defined by file numbers (not necessarily corresponding externally to FORTRAN unit numbers). These file numbers correspond to devices as follows:

INPUT

<u>INTERP/8 #</u>	<u>DEVICE</u>	<u>PDP-10 DEVICE</u>	<u>FILE NAME</u>
1	User's Console	TTY 5	
2	Card Reader	CDR 2	
3	Paper Tape	PAP 6	
4	Magnetic Tape	MAG 16	
5	Magnetic Tape	DEC 9	
6	Disk	DISK 20	FOR20.DAT
7	Disk	DISK 21	FOR21.DAT

OUTPUT

<u>INTERP/8 #</u>	<u>DEVICE</u>	<u>PDP-10 DEVICE</u>	<u>FILE NAME</u>
1	User's Console	TTY 5	
2	Printer	PTR 3	
3	Paper Tape	PAP 7	
4	Magnetic Tape	MAG 17	
5	Magnetic Tape	DEC 10	
6	Disk	DISK 22	FOR22.DAT
7	Disk	DISK 23	FOR23.DAT

IO functions are controlled through "\$" commands which may be interspersed throughout the input.

Any input line with a "\$" in column one, followed by a non-blank character is considered an IO command. The card is then scanned for an "=" followed by a decimal integer. The character following the "\$" and the integer value affect the IO formatting functions as follows:

<u>control</u>	<u>meaning</u>	<u>initial value</u>
\$COUNT = n	Start the output line count at the value n.	1
\$DELETE = n	Delete all characters after column n of the output.	120
\$EOF = 1	End-of-file on this device	0
\$INPUT = n	Read subsequent input from file number n.	1
\$LEFT = n	Ignore character positions 1 through n-1 of the input.	1
\$OUTPUT = n	Write subsequent output to file number n.	1
\$PRINT = n	Controls listing of the output. If n=0, input lines are not printed; otherwise input is echoed.	0
\$RIGHT = n	Ignore all character positions beyond column n of the input.	80
\$TERMINAL = n	INTERP/8 assumes conversational usage if n=1; otherwise batch processing is assumed.	1
\$WIDTH = n	This command sets the width of the output line. Note that this affects the format of the DISPLAY MEMORY command.	72

The default values shown above assume conversational use with a teletype or similar device. The defaults can easily be changed by recompiling the INTERP/8 program.

In the case of controls which take on only 0 or 1 values (e.g., \$PRINT, \$TERMINAL, and \$EOF), the equal sign and decimal number may be omitted. The value of the control is complemented in this case.

V. ERROR MESSAGES.

E R R O R M E S S A G E S

EXECUTION ERRORS

- 1 PROGRAM COUNTER STACK OVERFLOW
- 2 PROGRAM COUNTER STACK UNDERFLOW
- 3 PROGRAM COUNTER OUTSIDE SIMULATED MCS-8 MEMORY
- 4 MEMORY REFERENCE "

COMMAND MODE ERRORS

- 1 REFERENCE OUTSIDE SIMULATED MCS-8 MEMORY
- 2 INSUFFICIENT SPACE REMAINING IN SIMULATED MCS-8 MEMORY
- 3 END-OF-FILE ENCOUNTERED BEFORE EXPECTED
- 4 INPUT FILE NUMBER STACK OVERFLOW (MAX 7 INDIRECT REFERENCES)
- 5 UNUSED
- 10 IO FORMAT COMMAND ERROR (TOGGLE HAS VALUE OTHER THAN 0 OR 1)
- 11 UNUSED
- 13 INVALID SEARCH PARAMETER IN DISPLAY SYMBOL COMMAND (MUST BE SYMBOLIC NAME, ADDRESS, OR *)
- 14 DISPLAY SYMBOLS COMMAND INVALID SINCE NO SYMBOL TABLE EXISTS
- 15 UNUSED
- 16 UNRECOGNIZED COMMAND OR INVALID FORMAT IN COMMAND MODE
- 17 MISSING . OR EXTRA CHARACTERS FOLLOWING COMMAND
- 18 LOWER BOUND EXCEEDS UPPER BOUND OR IS LESS THAN ZERO IN RANGE LIST
- 19 THE FORMAT OF THE SYMBOL TABLE IS INVALID (MUST BE A SEQUENCE OF THE FORM N SY AD, WHERE N IS AN INTEGER, SY IS THE SYMBOLIC NAME, AND AD IS THE ADDRESS (IN OCTAL))
- 21 UNUSED
- 22 UNRECOGNIZED DISPLAY ELEMENT OR INVALID DISPLAY FORMAT
- 23 SYMBOLIC NAME NOT FOUND IN SYMBOL TABLE
- 24 INVALID ADDRESS OR NO SYMBOL TABLE PRESENT IN DISPLAY SYMBOL COMMAND
- 25 OUTPUT DEVICE WIDTH TOO NARROW FOR DISPLAY MEMORY COMMAND (USE \$WIDTH = N IO FORMAT COMMAND TO INCREASE WIDTH)
- 26 INVALID RADIX IN MEMORY DISPLAY COMMAND (MUST BE CODE, BIN, OCT, OR DEC)
- 27 UNRECOGNIZED SET ELEMENT IN SET COMMAND
- 28 MISSING SET LIST IN SET COMMAND
- 29 INVALID SET LIST OR SET VALUE IN SET COMMAND
- 30 MISSING OR MISPLACED * IN SET COMMAND
- 31 MISSING PROGRAM STACK ELEMENT NUMBER IN SET PS N COMMAND
- 32 INVALID INTERRUPT CODE SPECIFICATION (EITHER MORE THAN THREE BYTES, OR ELEMENT EXCEEDS 255)

VI. EXAMPLES.

Two sample INTERP/8 executions are given in this section which illustrate the commands available with the INTERP/8 system. The first example illustrates the basic commands. A simple program is constructed in the simulated MCS-8 memory. This program is then executed, showing the use of break and trace points. The second execution shows the use of symbol tables and 8008 code which is produced by the INTEL 8008 assembler. In each case, the actual commands which initiate the INTERP/8 system may vary from installation to installation.

.R INT8

BEGIN

/* THIS IS AN EXAMPLE OF THE USE OF THE INTERP/8 SYSTEM.

IN THIS EXAMPLE, THE BASIC COMMANDS WILL BE DEMONSTRATED

AND A SIMPLE PROGRAM WILL BE CONSTRUCTED AND EXECUTED */

/* THE NUMBER CONVERSION COMMAND IS USED FIRST */

CCNV 10.

1010B 120 10 AH
CON 100.

1000B 100 8 8H
CCN 3 TO 8.

11B 30 3 3H
100B 40 4 4H
101B 50 5 5H
110B 60 6 6H
111B 70 7 7H
1000B 100 8 8H

/* NEXT, THE VARIOUS DISPLAY AND SET COMMANDS ARE DEMONSTRATED */

DISPLAY CPU.

CYZSP A B C D E H L HL SP PS0
*0000*000*000*000*000*000*000*000*00000*000*00000
DISP A,D,HL.

A = 0
 L = 0
 HL = 0
 DIS PORT 4, PS 0, MEM 5.

P4=0
 PS0 = 0
 /* MEMORY LOCATION 5 WAS NOT DISPLAYED SINCE NO PROGRAM HAS BEEN
 LOADED */

SET H = 5, L=100. DISP CPU.

SET OK
 CYZSP A B C D E H L HL SP PS0
 0000 000 000 000 000 000*005*008*01288 000 00000
 /* NOTE THAT THE ELEMENTS WHICH HAVE CHANGED SINCE THE LAST DISPLAY
 ARE PRECEDED BY AN ASTERISK */

SET HL = 0EEFH. DIS CP.

SET OK
 CYZSP A B C D E H L HL SP PS0
 0000 000 000 000 000 000*014*239*03823 000 00000
 CONV 03823.

111011101111B 73570 3823 EEFH
 /* NOW CHANGE THE DEFAULT NUMBER BASE TO HEXADECIMAL */

BASE HEX. DISP CPU.

HEX BASE OK
 CYZSP A B C D E H L HL SP PS0
 0000 00H 00H 00H 00H 00H 0EH EFH 0EEFH 00H 0000H
 /* THEN CHANGE BASE TO OCTAL */

BASE OC. DI CP.

OCT BASE OK
 CYZSP A B C D E H L HL SP PS0
 0000 0000 0000 0000 0000 0000 0160 3570 073570 0000 000000

/* NOW PLACE A SIMPLE PROGRAM INTO MEMORY STARTING AT LOCATION 10.

THIS PROGRAM WILL ALTER THE VALUE OF MEMORY CELL 200 BY ADDING 1
 TO THE CURRENT VALUE OF THE CELL. IN SYMBOLIC FORM, THE PRO-
 GRAM IS AS FOLLOWS... LHI 0, LLI 200, LBM, INB, LNB, RLT.

THE LOAD OPERATION BELOW IS A 'DUMMY' OPERATION SO THAT MEMORY IS
 INITIALIZED PROPERLY. */

RECORDED

LOAD 1.

\$

00 LOAD OK
DISPLAY MEMORY 10 TO 20.

000120 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
BASE DEC.

DEC BASE OK
SET MEM 10 TO 20 = 00101110B 0 /* THIS IS LHI 0 */

00110110B 200 /* LLI 200 */

11001111B /* LBM */ 00001000B /* INB */

11111001B /* LMB */ 0 /* HLT */

SET CK
DI ME 10 TO 20.

00010 046 000 054 200 207 000 249 000 046 000 054
DI M 10 TO 20 CODE.

00010 LHI,00H LLI,C8H LBM INB LMB HLT LHI,00H LLI

/* NOTE THAT THE ',' SEPARATES ELEMENTS WHICH ARE PART OF THE
SAME INSTRUCTION (THE SECOND AND THIRD BYTES ARE IN HEX) */
CONV 0C8H.

11001000B 3100 200 C8H

/* WE CAN NOW EXECUTE THE PROGRAM BY SETTING THE PROGRAM COUNTER
TO LOCATION 10 */

SET PC=10. DI CP.

SET OK
CYZSP A B C D E H L HL SP PS0
0000 000 000 000 000 000 014 239 03823 000*00010
SE HL=0.

SET OK

CO.

HLT CYCLE 56

DI CPU.

CYZSP A B C D E H L HL SP PS0
0000 000*001 000 000 000*000*200*00200 000*00017
DI MEM 200.

00200 001

/* MEMORY LOCATION 200 HAS BEEN INCREMNTED -- NOW TURN ON THE

TRACE AND EXECUTE THE PROGRAM AGAIN */

TRACE 0 TO 100. GO.

TRACE OK

0000 000 001 000 000 000 000 000 200 00200 000 00017

HLT

HLT CYCLE 60

/* CPU MUST FIRST BE INITIALIZED TO ZERO */ SET CPU. GO.

SET OK

0000 000*000 000 000 000 000*000*00000 000*00000

HLT

HLT CYCLE 4

DI CPU.

CYZSP A B C D E H L HL SP PS0
0000 000 000 000 000 000 000 000 00000 000 00000
/* FORGOT TO SET PC = 10, TR\ AGAIN */ SET CPU, PC=10. GO.

SET OK

0000 000 000 000 000 000 000 000 00000 000*00010

LHI 0

0000 000 000 000 000 000 000 000 00000 000*00012

LLI 200

0000 000 000 000 000 000 000*200*00200 000*00014

LBM

0000 000*001 000 000 000 000 000 200 00200 000*00015

INB

0000 000*002 000 000 000 000 000 200 00200 000*00016

LMB

0000 000 002 000 000 000 000 200 00200 000*00017

HLT

HLT CYCLE 40

/* NOW TRY THE SAME EXECUTION WITH THE TRACE ENABLED OVER ONLY

PART OF THE PROGRAM */

NOTRACE 0 TO 100. TRACE 12 TO 14, 17.

TRACE OK

TRACE OK

SET CPU, PC=10. GO.

SET OK

0000 000*000 000 000 000 000*000*00000 000*00012
 LLI 200
 0000 000 000 000 000 000 000 000*200*00200 000*00014
 LBM
 *0001 000*003 000 000 000 000 200 00200 000*00017
 HLT
 HLT CYCLE 40
 /* SWITCH BACK TO FULL TRACE */ TR 0 TO 100.

TRACE OK
 DISP MEM 200.

00200 003
 /* NOW RUN THE CPU FOR ONLY A FEW INSTRUCTIONS AT A TIME. IN THIS
 WAY THE EXECUTION CAN BE MONITORED EASILY */
 GO 2.

GO OK

CYZSP	A	B	C	D	E	H	L	HL	SP	PS0
0001	000	003	000	000	000	000	200	00200	000	00017

 HLT
 HLT CYCLE 44
 SET CPU, PC=10. GO 2.

SET OK
 GO OK
 *0000 000*000 000 000 000 000*000*00000 000*00010
 LHI 0
 0000 000 000 000 000 000 000 000 00000 000*00012
 LLI 200
 CYCLE AT 14
 DI CPU.

CYZSP A B C D E H L HL SP PS0
 0000 000 000 000 000 000 000*200*00200 000*00014
 GO 1.

GO OK
 0000 000 000 000 000 000 000 200 00200 000 00014
 LBM
 CYCLE AT 15
 DI CPU.

CYZSP A B C D E H L HL SP PS0
 0000 000*003 000 000 000 000 200 00200 000*00015
 GO *.

0000 000 003 000 000 000 000 200 00200 000 00015
 INB
 0000 000*004 000 000 000 000 200 00200 000*00016
 LMB
 0000 000 004 000 000 000 000 200 00200 000*00017
 HLT
 HLT CYCLE 40

DI CPU.

```
CYZSP  A  B  C  D  E  H  L  HL  SP  PS0
0000 000 004 000 000 000 000 200 00200 000 00017
```

/* WE CAN SET BREAK POINTS IN THE CODE SO THAT AT EXECUTION STOPS
WHEN A PARTICULAR INSTRUCTION IS FETCHED. */

SET CPU,PC=10. TR 0 TO 100. REFER 12 TO 14.

SET OK
TRACE OK
REFER OK
GO.

```
*0000 000*000 000 000 000 000*000*00000 000*00010
LHI 0
0000 000 000 000 000 000 000 000 00000 000*00012
LLI 200
REFER AT 12
DI CPU.
```

```
CYZSP  A  B  C  D  E  H  L  HL  SP  PS0
0000 000 000 000 000 000 000 000 00000 000 00012
```

/* THE EXECUTION CAN ALSO BE STOPPED WHEN THE PROGRAM REFERS
TO MEMORY LOCATI ON 200 */

REFER 200. NOTRACE 0 TO 100. SET CPU,PC=10. GO.

REFER OK
TRACE OK
SET OK
REFER AT 14
DI CPU.

```
CYZSP  A  B  C  D  E  H  L  HL  SP  PS0
0000 000 000 000 000 000 000*200*00200 000*00014
DI MEM 14 CODE.
```

00014 LBM
GO 1. DI CP.

GO OK
CYCLE AT 15

```
CYZSP  A  B  C  D  E  H  L  HL  SP  PS0
0000 000*005 000 000 000 000 200 00200 000*00015
```

/* THIS SHOWS THE VALUE FETCHED FROM LOCATION 200. WE CAN STOP

RECORDED

LER72-103403

Page 29

THE PROGRAM ON A STORE INTO LOCATION 200 AS WELL */

NOREF 200. ALTER 200. SET CP, PC=10. GO.

REFER OK
ALTER OK
SET OK
ALTER AT 16
DI CPU.

CYZSP A B C D E H L HL SP PS0
*0001 000*006 000 000 000 000 200 00200 000*00016
D M 16 CO.

00016 LMB

/* THE REGISTER DUMP SHOWS THAT 6 WILL BE STORED AT LOCATION 200.

EXAMINE LOCATION 200, RUN THE MACHINE FOR ONE CYCLE, AND EXAMINE
THE CELL AGAIN */

DI MEM 200. GO 1. DI MEM 200.

00200 005
GO OK
CYCLE AT 17
00200 006

/* NOW GET A COMPLETE MEMORY DUMP IN BINARY */

DI MEM 0 TO 777Q BIN.

00000 00000000B 00000000B 00000000B 00000000B 00000000B 00000000B 00000000B
00006 00000000B 00000000B 00000000B 00000000B 00101112B 00000000B
00012 00110110B 11001000B 11001111B 00001000B 11111001B 00000000B
00018 00101110B 00000000B 00110110B 00000000B 00000000B 00000000B
00024 00000000B 00000000B 00000000B 00000000B 00000000B 00000000B

00198 00000000B 00000000B 00000110B 00000000B 00000000B 00000000B
00204 00000000B 00000000B 00000000B 00000000B 00000000B 00000000B
00510 00000000B 00000000B

/* AND THEN PUNCH THE CODE BETWEEN LOCATIONS 10 AND 20 (WE WILL USE
THE CONSOLE AS THE OUTPUT DEVICE) */

PUNCH 10 TO 20 1.

8 BNNNNNNNNF BNNNNNNNNF BNNPPPPPF BNNNNNNNNF
BNNPPPPNF BPPNNPPNNF BPPNNPPPPF BNNNNPPNNF
16 BPPPPPPNNF BNNNNNNNNF BNNPPPPNF BNNNNNNNNF
BNNPPPPNF BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF

END.

\$EOF

CPU TIME: 12.93 ELAPSED TIME: 46:12.73
NO EXECUTION ERRORS DETECTED

THIS EXAMPLE SHOWS A COMPLETE ASSEMBLY AND INTERP/8 EXECUTION

TYPE ASMI.DAT
* SAMPLE MCS-8 PROGRAM (PAGE 47 OF 8008 MANUAL)
START LLI 200
LHI 0
LOOP LAM
CPI 46
JNZ FOUND
CAL INCR
LAL
CPI 220
JFZ LOOP
FOUND RET
INCR INL
RFZ
INH
RET
END

.R ASM8

PLEASE TYPE INPUT FILE NAME
ASMI

8008 INTEL ASSEMBLER

CPU TIME: 3.72 ELAPSED TIME: 9.73
NO EXECUTION ERRORS DETECTED

EXIT
^C

.RENAME FOR20.DAT = LOGOU.DAT, FOR21.DAT = LOGBI.DAT
FILES RENAMED:
LOGOU.DAT
LOGBI.DAT

.TYPE FOR20.DAT

=====

SYMBOL VALUE

=====

```

1: START  00000
2: LOOP   00004
3: FOUND  00023
4: INCR   00024

```

```

=====IC
IC

```

.TYPE DORIU

TYPE FOR21.DAT

```

0 BNNPPNPPNF BPPNNPNNNF BNNPNPPPNF BNNNNNNNNF
  BPPNNNPPPF BNNPPPNPNF BNNPNPPPNF BNNPNPNPNF
8 BNNPNNPPF BNNNNNNNNF BNNPNNPPNF BNNPNPNPNF
  BNNNNNNNNF BPPNNNPPNF BNNPPPNNF BPPNPPPNF
16 BNNPNPNKNF BNNNNNPNNF BNNNNNNNNF BNNNNNPPPF
  BNNPPNNNNF BNNNNPNPPF BNNPNPNKNF BNNNNNPPPF
24 BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF
  BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF
32 BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF

```

THE CODE FILE MUST BE TERMINATED BY A \$ IN THE INPUT -- USE TECO
TECO FOR21.DAT

*N 32 \$\$

*OLST\$\$

32 BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF

*IS

\$\$

*EX\$\$

.R INT6

LOADING

LOADER 10K CORE
EXECUTION

BEGIN

/* THE SYMBOL TABLE AND CODE WILL NOW BE LOADED */

LOAD 6 7.

32 LOAD OK
DI SYMBOLS.

000000Q 00000 0000H START
000004Q 00004 0004H LOOP
000023Q 00019 0013H FOUND
000024Q 00020 0014H INCR
DI SYMBOL LOCP.

000004Q 00004 0004H LOOP
DI SYMBOL ZAP.

(00027) ERROR 23 NEAR ZAP
/* ERROR MESSAGE HAS LINE NUMBER ERROR NUMBER AND ITEM IN ERROR. IN
THIS CASE, THE SYMBOL COULD NOT BE FOUND IN THE TABLE */

DI SY 13H.

FOUND
DI SY 12H.

FOUND-1
DI SY 8.

LOOP+4
DI SY *.

START
/* NOW TAKE A LOOK AT MEMORY IN HEXADECIMAL AND IN CODE FORMAT */

DI MEM 0 TO 100 HEX, MEM 0 TO 100 CODE.

00000 36H C8H 2EH 00H C7H 3CH 2EH 68H 13H 00H 46H 14H 00H C6H 3CH DCH
00016 48H 04H 00H 07H 30H 0BH 28H 07H 00H 00H 00H 00H 00H 00H 20H
00032 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H

00096 00H 00H 00H 00H 00H
00000 LLI,C8H LHI,00H LAM CPI,2EH JTZ,13H,00H CAL,14H,00H LAL CPI,DCH
00016 JFZ,04H,00H RET INL RFZ INH RET HLT HLT HLT HLT HLT HLT HLT HLT
00032 HLT HLT HLT HLT HLT HLT HLT HLT HLT HLT HLT HLT HLT HLT HLT

00096 HLT HLT HLT HLT HLT

/* THIS PROGRAM SEARCHES FOR A 46 STARTING AT LOCATION 200 IN
MEMORY. WE WILL START BY PLACING A SEQUENCE OF NUMBERS IN THESE
LOCATIONS */

SET MEM 200 TO 210 = 43 46 48 20H 1111000B. DI MEM 200 TO 210.

SET OK

00200 043 046 048 032 120 043 046 048 032 120 043
 /* GET A COMPLETE TRACE OF THE PROGRAM */ TR 0 TO 0100.

TRACE OK
 GO.

CYZSP	A	B	C	D	E	H	L	HL	SP	PS0
*0000*000*000*000*000*000*000*000*00000*000*00000										
LLI 200										
0000 000 000 000 000 000 000 000 000*00200 000*00002										
LHI 0										
0000 000 000 000 000 000 000 000 200 00200 000*00004										
LAM										
0000*043 000 000 000 000 000 000 200 00200 000*00005										
CPI 46										
*1010 043 000 000 000 000 000 000 200 00200 000*00007										
JTZ 19										
1010 043 000 000 000 000 000 000 200 00200 000*00010										
CAL 20										
1010 043 000 000 000 000 000 000 200 00200*001*00013*00020										
INL										
*1011 043 000 000 000 000 000*001*00201 001 00013*00021										
RFZ										
1011 043 000 000 000 000 000 201 00201*000 00013										
LAL										
1011*001 000 000 000 000 000 201 00201 000*00014										
CPI 220										
CYZSP A B C D E H L HL SP PS0										
1011 201 000 000 000 000 000 201 00201 000*00016										
JFZ 4										
1011 201 000 000 000 000 000 201 00201 000*00004										
LAM										
1011*046 000 000 000 000 000 201 00201 000*00005										
CPI 46										
*0101 046 000 000 000 000 000 201 00201 000*00007										
JTZ 19										
0101 046 000 000 000 000 000 201 00201 000*00019										
RET										

EXECUTION ERROR 2 AT 22

/* THE ERROR OCCURS BECAUSE THE PROGRAM TERMINATES WITH A RET

RATHER THAN A HLT. FIX THE INSTRUCTION IN MEMORY */

DI MEM 19.

00019 007

DI MEM MEM 19 COD.

00019 RET

SET M 19 = 0. DI MEM 19 CO.

SET OK

00019 HLT
NOTR 0 TO 100. SET CPU. GO.

TRACE OK
SET OK
HLT CYCLE 117
DI CPU.

CYZSP A B C D E H L HL SP PS0
0101 046 000 000 000 000 000 201 00201 000 00019
/* THE PROGRAM TERMINATES CORRECTLY AFTER 117 MACHINE STATES */

TIME.

TIME=117
/* SET SELECTIVE BREAK POINTS */

REF START, INCR+1, LOOP. SET CPU. GO.

REFER OK
SET OK
REFER AT 0
DI SY *.

START
G.

REFER AT 4
DI SY *. GO.

LOOP
REFER AT 21
DI SY *. GO.

INCR+1
REFER AT 4
D SY.

0000000 00000 0000H START
0000040 00004 0004H LOOP
0000230 00019 0013H FOUND
0000240 00020 0014H INCR
NOREF START TO INCR+5.

REFER OK
/* SET SELECTIVE TRACE POINTS (TRACE AND REFER POINTS CAN BE IN EFFECT
AT THE SAME TIME, IF DESIRED) */

TR START, LOOP, FOUND. REFER FOUND. GO.

TRACE OK

REFER OK

*1011*201 000 000 000 000 000 201 00201 000*00004

LAM

*0101*046 000 000 000 000 000 201 00201 000*00019

HLT

REFER AT 19

DI CP.

CYZSP A E C D E H L HL SP PS0

0101 046 000 000 000 000 000 201 00201 000 00019

SET CP. GO.

SET OK

*0000*000 000 000 000 000 000*000*00000 000*00000

LLI 200

0000 000 000 000 000 000 000*200*00200 000*00004

LAM

*1011*201 000 000 000 000 000*201*00201 000 00004

LAM

*0101*046 000 000 000 000 000 201 00201 000*00019

HLT

REFER AT 19

GO.

0101 046 000 000 000 000 000 201 00201 000 00019

HLT

HLT CYCLE 117

/* THE ONLY REMAINING COMMANDS TO ILLUSTRATE ARE HTHE SET AND IDISPLAY

PORTS COMMANDS */

DI PORT 4.

P4=0

DI PORT 4, PO 3, PO 7 TO 100.

P4=0

P3=0

P7=0 P8=0

DI PO 20 TO 25.

P20=0 P21=0 P22=0 P23=0 P24=0 P25=0

SET PORT 5 = 11001100B, PO 10H = 550.

SET OK

DI POR 5 TO 17.

P5=204 P6=0 P7=0 P8=0 P9=0 P10=0 P11=0 P12=0 P13=0 P14=0 P15=0 P16=45 P

17=0

END.

SEOF

DISTRIBUTION

LLL Internal Distribution

TID File

3

External Distribution

Division of Technical Information, Ext.
Oak Ridge, Tennessee

2

Hank Smith
INTEL Corporation
3065 Bowers Avenue
Santa Clara, CA. 95051

1

NOTICE

"This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights."