ICE MONITOR COMMANDS

KEY TO THE CODE

All characters may be entered in upper or lower case.
Rubout and backarrow delete a character.
Control-X cancels the line.
# means a hex number.
(...) implies a choice.
[...] implies optional parameters.

MONITOR COMMANDS

The monitor prompt is a percent sign (%).
Active keys at the % are:

```
     Control-C Boots CP/M after reset.
     A         Sets 0-1FFFH and 4000H-4FFFH external, all
               internal memory is  write protected.
     C         Sets 4000H-7FFFH,C000-CFFFH,E000-EFFFH external,
               all internal memory is write protected.
     D         Enters debug mode.
```

DEBUG COMMANDS

The debug prompt is a minus sign (-).
Active keys at the - are:

```
     A         ASCII ON/OFF
               Switches ASCII character display in modify memory
               on/off.
     B[(M,P)(R,W)#]<CR> Breakpoint
               M = Memory
               P = Port
               R = Read
               W = Write
     Example: If you want to break on memory read at 38H, type
              at the prompt: BMR38<carriage return>.
              If you want to turn off the breakpoints type:
              B<carriage return>.
              Note: Step sets the breakpoint to the current
              instruction.
     D[#[#]]<CR>
               Dump from # to # memory locations.
     F#,#,#<CR>
               Fill memory from # to # with #.
     G[<SPACE>#]<CR>
               Go at PC or #
     I[(E,D,<CR>)]
               Set interupts enabled, disabled or display
               their status.
```

M#<CR>   Modify memory

Displays AAAA: DD

Where AAAA is the hex address and DD is the hex data.

Then it waits for input of form:

[#](<CR>,^,<LF>)

Entering a number modifies the location.

<CR> advances to the next location.

^ (up arrow) backs up one location,

<Line Feed> exits to the debug prompt.

P Modify Port

Functions exactly the same as modify memory.

Q Quit - Returns to the monitor

R(A,B,C,D,E,H,L,BC,HL,IX,IY,SP,PC)

Modify register

Displays VV or VVV the value of the register which can be modified by typing in a new value.

S   Step

Step sets the breakpoint to the current instruction then executes it, returns and displays the contents of the registers.

X   Examine registers

Displays current contents of the registers.

BNF grammer definition of TERSE

```
<numeric literal> ::= <ASCII digit> <numeric literal>¦<null>

<value> ::= <numeric literal>¦<value on stack>¦<constant name>

<name> ::= <ASCII character> <name>¦<null>

<verb cluster> ::= <verb>¦<if statement>¦<do statement>¦
                  <begin statement>¦<value> <verb cluster>¦<null>

<boolean value> ::= <zero value>¦<non-zero value>

<else statement> ::= ELSE <verb cluster> THEN

<if statement> ::= <boolean value> IF <verb cluster> THEN¦
                  <else statement>

<+loop statement> ::= <value> +LOOP

<do statement> ::= <limit value><start value> DO <verb cluster> LOOP¦
                  <+loop statement>

<while statement> ::= WHILE <verb cluster> REPEAT

<begin statement> ::= BEGIN <verb cluster> <boolean value>END¦
                     <while statement>

<: name> ::= <name> <code name> ::=<name>

<:definition> ::= : <:name> <verb cluster> ;

<code definition> ::= CODE <code name> <assembler op-codes> NEXT

<program> ::= <:name>

<variable name> ::= <name>

<variable definition> ::= <value> VARIABLE¦BVARIABLE <variable name>

<constant name> ::= <name>

<constant definition> ::= <value> CONSTANT <constant name>

<array name> ::= <name>

<array definition> ::= <value> ARRAY¦BARRY <array name>

<verb> ::= <:name>¦<code name>¦<variable name>¦<constant name>¦
          <array name>¦<table name>
```

<storage statement> ::= <value> B,¦,

<storage cluster> ::= <storage statement> <storage cluster>¦<null>

<table name> ::= <name>

<table definition> ::= TABLE¦BTABLE¦DATA <table name> <storage cluster>

<null> ::= the empty set

PATTERN BOARD

```
PORTS (in load order)
     7A - Status
     Bits: 0 - 0 = linear to area
               1 = area to linear

           1 - 0 = no expand
               1 = expand

           2 - 0 = use constant data
               1 = use pattern data

           3 - 0 = no flush
               1 = flush (do not use with expand)

           4 - 0 = no flip
               1 = flip

           5 - 0 = flop
               1 = no flop

           6 - unused

           7 - unused

    *78 - linear address low

    *79 - linear address high

     7B - area address low

     7C - area address high

     7B - Xmod (port is used twice)
          for plop = (80 - width) see next port for definition
                      of width.
          for flip = (-80 - width)
          for flop = (80 + width)
          for flop + flip = (-80 + width)

     7D - width = (Xsize - 1) or
             if expand then (Xsize * 2) - 1, if flush add 1

    *7E - height = (Ysize - 1)
         this port fires the operation
```

* Only those ports need to be repeated if another pattern of the same width is to be started where the previous pattern left off.

DEBUG 81 Glossary
09/21/81

The interactive debugger provides a means for examining the processing of TERSE compiled programs in a detailed fashion, either by stepping through a program or setting breakpoints in a program stream.

Certain functions have made use of the particular implementation of TERSE on the Z-80. Perhaps the most important of these is that NEXT is implemented as a PCIY (jump to adr in IY register) which is the address of the inner interpreter. When in debug step mode, this address is replaced by code in the debugger. In other implementations, this could be gotten around by temporarilly overwriting the inner interpreter with a jump.

The other useful function of DEBUGging package is the UNCOMpiler, which allows verbs to be listed out with the adress of their component verbs, a function which is quite easily implemented by virtue of the simple structure of compiled TERSE code.

Note also that when executing TERSE code in breakpoint mode, the debuggers inner interpreter is the one being used, which has a modest overhead involved in doing the address comparisons, so code will run quite a bit (about 1/4 speed) slower.

As a reminder, the following registers are utilized in the Z-80 TERSE implementation:

BC	Inner interpreter pointer.
IY	Address of NEXT, the inner interpreter.
IX	Return stack pointer.
SP	Parameter stack pointer.

To begin DEBUGging a TERSE verb, see STEP.

$BC --- p

Return the address of a variable containing the current interpreter pointer used by the debugger.

$BRK --- p

Return the address of a variable which when set to 1, means a breakpoint is set; thus a STEP or an S will proceed until the address contained in BPNT is encountered.

' ' nnnn --- n

Return the begin of code adr of verb NNNN. (Part of system verbs)

BM --- p

Return the addres of the Break Mode Variable. It will contain either:

0 - Stop execution when a breakpoint is encountered.
1 - Print status information on encountering a breakpoint and continue execution.

BPNT --- p

Return the address of a variable containing the address where the breakpoint is set.

BRK n ---

Set a breakpoint at the specified address.

CLRBRK

Clear a previously set breakpoint.

DFG --- p

Return the address of the Display Mode Variable. It will contain either:

0 - No display after the verb is executed.
1 - Display only the verb being executed.
2 - Display the parmater stack, the verb about to be executed, and the top of the return stack (I). This is the default value.

DISPLAY FORMAT: [Parameter Stack] VERB= nnnn [I= n]

PS --- n

Returns the value of the parameter stack pointer (same as SP@).

PSD

List the contents of the ENTIRE parameter stack.

Q

Execute the entirety of the verb about to be executed.
Note: Q places the breakpoint pointer at the adr 2 + the interpreter pointer. Verbs that use the 2nd word as data or a jump address (IF ELSE CASE LIT) will not work. No actual change in the memory location is made (PROM programs can be debugged this way), but note that the verb is actually being stepped one instruction at a time when in Q or breakpoint mode so that it will be slower.

RS --- n

Returns the value of the return stack pointer.

RSD

List the contents of the ENTIRE return stack.

S

Execute one instruction (one pass throught the inner interpreter). the program is actually being stepped an instruction at a time when in Q or breakpoint mode.

SCT --- p

Return the address of the Step Count Variable which is set to the number of verbs to be executed before control returns to the user.

STEP STEP nnnn

Prepare to debug verb NNNN. The verbs about to be executed will be printed.
Example: ' TESTPROG 1+ BRK
Set a breakpoint at the first instruction of TESTPROG (note skipping the header byte).

UNCOM n ---

Uncompile (list) the verb compiled at adr n.

VERB n ---

Display the name of the verb whose code start adr is n. Very handy !

------ END OF DEBUG 81 Glossary ------

GAS Port Assignments
9/21/81

This is a description of the various ports used by the GAS system. Each port number is a hex value and for each bit that is not specified, then that bit is unused.

GAS Input Port Assignments

10		Gun Handle #1 Switches
	Bit 0	Up
	Bit 1	Down
	Bit 2	Left
	Bit 3	Right
	Bit 4	Trigger
11		Gun Handle #2 Switches
	Bit 0	Up
	Bit 1	Down
	Bit 2	Left
	Bit 3	Right
	Bit 4	Trigger
12		Gun Handle #3 Switches
	Bit 0	Up
	Bit 1	Down
	Bit 2	Left
	Bit 3	Right
	Bit 4	Trigger
13		Gun Handle #4 Swithces
	Bit 0	Up
	Bit 1	Down
	Bit 2	Left
	Bit 3	Right
	Bit 4	Trigger
14		Front Panel Switches
	Bit 0	Switches 0
	Bit 1	Switches 1
	Bit 2	Switches 2
	Bit 3	Switches 3
	Bit 4	Switches 4
	Bit 5	Switches 5
	Bit 6	Switches 6
	Bit 7	Switches 7
15		Front Panel Switches
	Bit 0	Switches 8
	Bit 1	Switches 9
	Bit 2	Switches 10
	Bit 3	Switches 11
	Bit 4	Switches 12
	Bit 5	Switches 13
	Bit 6	Switches 14
	Bit 7	Switches 15
16->1B		Not Used
1C		Gun Handle #1 Knob
1D		Gun Handle #2 Knob

1E		Gun Handle #3 Knob	
1F		Gun Handle #4 Knob	
20		Dart Channel A Data	
21	CRT	Dart Channel B Data	
22		Dart Channel A Status	printer
23		Dart Channel B Status	
24		9511 APU Data	
25		9511 APU Status	
26->28		Not Used -- NOTE: No input should be done!	
29		Miscellaneous	
	Bit 0	Cassette Tape Data In	
	Bit 1	Light Pen Switches	
	Bit 2	Bit Pad Strobe Status	
2A		Bit Pad Strobe Status Reset	
2B		Bit Pad Strobe Status Reset	
2C		PERSCI Floppy Disk Data	
2D		PERSCI Floppy Disk Status	
2E		Bit Pad Data	
2F		Bit Pad Status	

GAS Output Port Assignment

00	Color Register 0
01	Color Register 1
02	Color Register 2
03	Color Register 3
04	Color Register 4
05	Color Register 5
06	Color Register 6
07	Color Register 7
08	Low/High Resolution
09	Horizontal Color Boundary, Background Color
0A	Vertical Blank Register
0B	Color Block Transfer
0C	Magic Register
0D	Interrupt Feedback Register
0E	Interrupt Enable and Mode
0F	Interrupt Line
10	Master Oscillator
11	Tone A Frequency
12	Tone B Frequency
13	Tone C Frequency
14	Vibrato Register
15	Tone C Volume, Noise Modulation Control
16	Tone A Volume, Tone B Volume
17	Noise Volume Register
18	Sound Block Transfer
19	Expand Register
20	Dart Channel A Data

21		Dart Channel B Data
22		Dart Channel A Command
23		Dart Channel B Command
24		9511 APU Data
25		9511 APU Command
26		Front Panel LEDs
	Bit 0	LED 0
	Bit 1	LED 1
	Bit 2	LED 2
	Bit 3	LED 3
	Bit 4	LED 4
	Bit 5	LED 5
	Bit 6	LED 6
	Bit 7	LED 7
27		Front Panel LEDs
	Bit 0	LED 8
	Bit 1	LED 9
	Bit 2	LED 10
	Bit 3	LED 11
	Bit 4	LED 12
	Bit 5	LED 13
	Bit 6	LED 14
	Bit 7	LED 15
28		Cassette Tape Motor Controls
	Bit 0	Input Cassette Motor
	Bit 1	Output Cassette Motor
29		Cassette Tape Data Out Individual Bits are Not Used; Each output Instruction causes the logic state of the flip-flop to change.
2A		Reset Bit Pad Strobe Status
2B		reset Bit Pad Strobe Status
2C		PERSCI Floppy Disk Data
2D		PERSCI Flopppy Disk Command
2E->CB		Not Used

```
CC                Memory Mapping
    Page 1 - 4000-7FFF
  Bit 0-3         0 -- Screen RAM
                  1 -- RAM first Board
                  2-15 RAM one of the other boards
    Page 0 -- 0-3FFFH
  Bit 4           0 -- EPROM
                  1 -- RAM
  Bit 5           0 -- Read/Write
                  1 -- Write Protected
    Page 2 -- 8000-BFFF
  Bit 6           0 -- EPROM
                  1 -- RAM
  Bit 7           0 -- Read/Write
                  1 -- Write Protected
```

5 1
0001
0101 0
1110 0001
E 1

1111 0000

------ End Of GAS Port Assignments ------