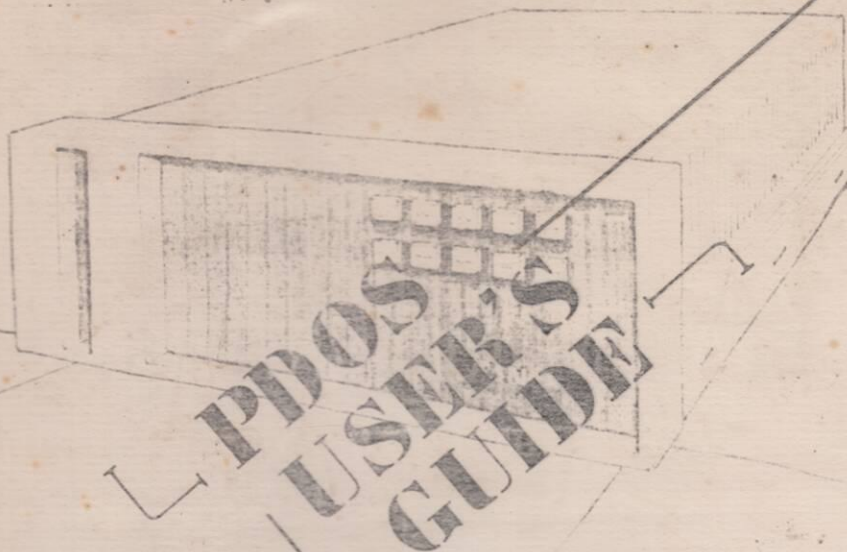


S-1000

**USER'S
GUIDE**

C.S.S.N



Tom Jennings

Fido's System Copy

Programmers Guide to the I/O devices

This is basically the same as above, but lists only the devices in the system, as a programmer (me) would like to see them.

Parallel A (04h)
 Parallel B (14h)
 Parallel C (24h)

A is the printer port. All three behave the same. Data in and out is non-inverted. No input data-ready strobe is available. An out to the port strobes the /STB line for one clock cycle.

The printer is wired as a Centronics, 8 bit data, 1 bit (LSB) in as BUSY status. The /STB line is the data strobe.

Front Panel (0bh)

This is yet another parallel port, exactly the same as above, except it drives the F.P. LED's and switches. Read 'em and write 'em.

Serial Ports:

Console (base 0h)
 Serial LST: (base 10h)
 Extra (base 0fh)

BASE+0 These are all UART like serial ports. Base+0 is status and baud registers:

7	6	5	4	3	2	1	0	
TBE	RDA	INT	SBD	FBD	SRV	OVE	FME	Status.
7	6	5	4	3	2	1	0	
1 stop	9600	4800	2400	1200	300	150	110	Baud.

BASE+1 Data in/out

BASE+2 Command register:

7	6	5	4	3	2	1	0	
x	x	TEST	BAUD	INTE	RST7	BRK	RESET	

TEST see manual. BAUD octuples the baud rates. INTE enables INTA interrupts. RST7 see manual. BRK sends a break. RESET clears the 5501.

BASE+3 IN: Interrupt address. See manual.
 OUT: Interrupt mask. See manual.

This is a basic ROM based debueer with commands as documented below. This is another in the endless series of little debuggers. This is the most complete and bus free one (so far...). The console I/O is inserted at assembly time, using a basic Debueer source module.

This is a relocating debugger; it contains a copy of RLOCATR2, the COM file relocater. This is also documented below.

REQUIREMENTS:

Z80 Processor,
40 bytes RAM,
a console.
a 2716 ROM.

OTHER ...

- Uses only 8080 registers internally; IX and IY nor any alternate registers are not modified by the debueer.
- Interrupts of any mode may be used.

ENTRY POINTS:

-On power up or whatever, jump to relative address zero. The console I/O will be initialized, and the default stack and return address will be initialized. The stack pointer and return address will be saved, and the Q command can be used to return to the caller.

Notes on use of the RELOCATE Command

The relocation feature can be used to move the entire debugger to any location in RAM memory. This provides for some "interesting" results if improperly used. First, there must be enough RAM at the desired address to accommodate the program, as this is not checked at relocation time. Note that memory wraps around the top to the bottom; i.e. FFFF and 0000 are sequential addresses. Try not to relocate over ROMs.

DEBUGGER COMMANDS

All debusser arguments may be separated by a space, comma, or carriage return (CR). When using args following commands, there may be no delimiting character between the command and the arg. A delimiter there implies the default value. (Listed for each command, below)

```
1/  G
2/  G<adr1>
3/  G<adr1>,<adr2>
4/  G.<adr1>
```

The Go command is used to run programs in the Z80 memory. A set of pseudo registers is kept by the debusser, and the Z80 registers are saved and restored here between G commands. These pseudo registers may be modified; see the X command.

The breakpoint facility uses location 38h, 39h, and 3ah, the RST 7 area. Do not modify this area. Note that the RST 7 area is not modified by the debusser until a breakpoint is set.

Form 2/ executes code from location <adr1>, with no breakpoint. 3/ executes from <adr1>, with a breakpoint at <adr2>. A register dump will be done when <adr2> is hit.

1/ executes from the PC as given in the register dump display, (see X command) with no breakpoint. 4/ executes as 1/, except a breakpoint is installed at <adr2>.

Note that if a breakpoint is to be used, the delimiter between the addresses MUST be a comma. Any other delimiter will assume you want form 1/ or 2/. In all cases if a comma immediately follows the G, the run address is the address displayed as PC in the X command.

When a breakpoint is hit, the default Dump address is set to the breakpoint address.

EXAMPLES:

```
G1000,1010      ;go to 1000, stop at 1010,
G,1020          ;go to address of last break
                ;point, stop at 1020.
G               ;continue from 1020, no
                ;breakpoint.
```

Note that the TRS-80 version uses RST 6's for break points. It uses the restart vector at 400F since 38h is in ROM.

```

1/   X.
2/   XA
3/   XF
4/   XB
5/   XD
6/   XH
7/   XX
8/   XY
9/   XS
10/  XP

```

Form 1/ displays the Z80 registers. The rest let you optionally modify the registers. A and F (PSW) are modified as single bytes, all others as pairs only.

Examples:

```

+x,
a  sz-a-p-c bc  de  hl  ix  iy  sp  pc  bKpt ^stk
01 01000001 0000 0000 0000 0000 0000 1000 2000 2000 0300
+xa0000=11                                ;after typing XA, 000 is
+                                           ;displayed. Change to 11.
+x,                                         ;examine the change.
a  sz-a-p-c bc  de  hl  ix  iy  sp  pc  bKpt ^stk
11 01000001 0000 0000 0000 0000 0000 1000 2000 2000 0300
+

```

BKPT and ^STK are used while tracing programs. These cannot be modified. BKPT is the address of the last set break point. ^STK is the value of the item on the top of the stack, the next element to be popped. (Usually a return address)

If a break point was set, and hit, (see G command) this address will be the same as the PC, which indicates the NEXT instruction to be executed. At this point the instruction has been restored. In addition, the default "dump address" is set to the PC address; S<cr> will list memory starting with the next instruction to be executed. If a RST 7 other than the break point was hit, this address will differ from the PC.

^STK can be used to put a break point just after the RET instruction for the subroutine. If you have traced down into a subroutine, and you want to execute the subroutine as is, but break on return, type:

```
G,<address>
```

where <address> is the displayed value of ^STK.

D<adr1>

D.

Dump 256 bytes of memory from <adr1>, as 16 lines of 16 bytes each, followed by it's ASCII equivalent. If <adr1> is omitted, the last used dump or breakpoint address will be used.

EXAMPLES:

```
D1000 ;dump memory from 1000 to
;10FF
D, ;following above example,
;dump from 1100 to 11FF.
```

S<adr1>

S.

Substitute memory, starting at <adr1>. If no address is supplied, the last used address, or the last break point address is used. The address and its contents are displayed, and new contents may be entered by typing valid hex digits. Only the last two entered are used, allowing retyping. Just a comma, space or <cr> leaves the contents unchanged. Following this, the next location is opened. Any other non-hex character terminates the command.

EXAMPLES:

```
S1000
1000 00 ca, ;displays current data (00),
1001 FF , ;enter new data (ca). Advance
1002 77 Q eh ? ;to next location. SKip 1001.
;the "Q" quits the "S" command,
;leaves 1002 intact.
```

F<adr1>,<adr2><byte>

Fill memory between <adr1> and <adr2> with constant <byte>. Does not verify memory after write.

EXAMPLE:

fill from 1000 to 3000 with 77's.
F1000,3000,77

M<from>,<end>,<to>

Move memory, between <from> and <end> to destination address <to>. Does not verify after copy, nor check for ROM.

EXAMPLE:

MF400,F7FF,100

I<port>

Read data from I/O port <port>. Display contents in hex.

EXAMPLE:

I97, ;input from port #97 hex,
D9 ;displayed data from port.

O<port>,<data>

Output data <data> to I/O port <port>.

EXAMPLE:

set the baud rate of a Cromemco TUART port A to 9600
+00,88
+

R<adr>

Relocate the debusser to location <adr>. The entire program is relocated to the indicated address. It is up to the user to avoid relocating over ROM, or over itself. All of the pseudo registers are moved and preserved. After relocation, the highest address used by the debusser is displayed, then executed. If the program is relocated over ROM, after the New End message is displayed, no prompt will appear.

EXAMPLE:

```
R5000 <cr>  
New End=5723
```

+

Q

This is used only if the debusser was called from another program. On entry to the call point, the caller's PC and stack were preserved. These are maintained during relocation as well. The stack pointer is restored, the return address is pushed on it, and control is returned there. All other 8080 registers are destroyed, plus the IX and IX if they were used during debussing. Make sure that during relocation, etc that there is still somewhere to return to!

If the debugger is the system monitor (i.e. no one called it) the Q command will jump to some unknown place in memory.