ParaSol Debugger Kit

(Part No. 900038)

USER'S MANUAL

Processor Technology Corporation

7100 Johnson Industrial Drive
Pleasanton, CA 94566
Telephone (415) 829-2600

First Printing, April, 1978
Manual No. 730018

CONTENTS

FIGURES

TABLES

SECTION 1

INTRODUCTION

1.Ø SCOPE OF THIS MANUAL

The ParaSol Debugger Kit User's Manual is divided into 6 sections. The first, which you are now reading, introduces the manual itself and then the ParaSol Debugger hardware/software system.

Section 2, Assembly and Hardware Checkout, provides the directions for building the Debugger and Cable Adapter boards. After the ParaSol is assembled, a test procedure is provided to assure it is ready for use. At the end of Section 2 is a complete parts list for the ParaSol.

Section 3, Hardware Theory of Operation, describes the interfaces between the Debugger Board and the known-good Sol (Master Sol) and the Sol being debugged (Slave Sol). It describes the signal flow between these three components, the logic operation of the debugger board, and the sequence of operations which control the Debugger.

Section 4, Software Theory of Operation, describes the procedure used to transfer the source files for the Debugger Program from cassette tape to disk. The basic subroutines used to control the Debugger are described in addition to the input/output routines of the Debugger Program. A set of flow charts for the test routines and a complete listing of the Debugger Program is also included in this section.

Section 5, Operating Instructions, contains instructions for connecting the Debugger to the Master and Slave Sols, and the procedure for loading the Debugger Program. Each test performed by the Debugger program is described, and the commands available during the trace mode are listed and described.

Section 6, Drawings, provides the assembly drawings and schematic which are used to build, understand and maintain the ParaSol Debugger.

1.1 INTRODUCTION TO THE ParaSol

1.1.1 PHYSICAL CONFIGURATION

The ParaSol consists of a main logic board called the Debugger Board and an adapter board which interfaces the Debugger to the Master Sol (known-good Sol) through its parallel interface connector. The Debugger board plugs into the S-1ØØ bus connector of the Slave Sol (unit to be debugged). The kit includes necessary cables and a cassette containing the Debugger Program.

1.1.2 OPERATION

The ParaSol Debugger allows the known-good Sol to automatically test the slave Sol. If the tests do not uncover a problem in the buses or memory of the slave Sol, the ParaSol then allows you to trace through the personality module program of the slave Sol, single stepping one cycle at a time, or running at high speed until a breakpoint is reached. You can even examine the contents of the CPU registers of the slave Sol while the program is running.

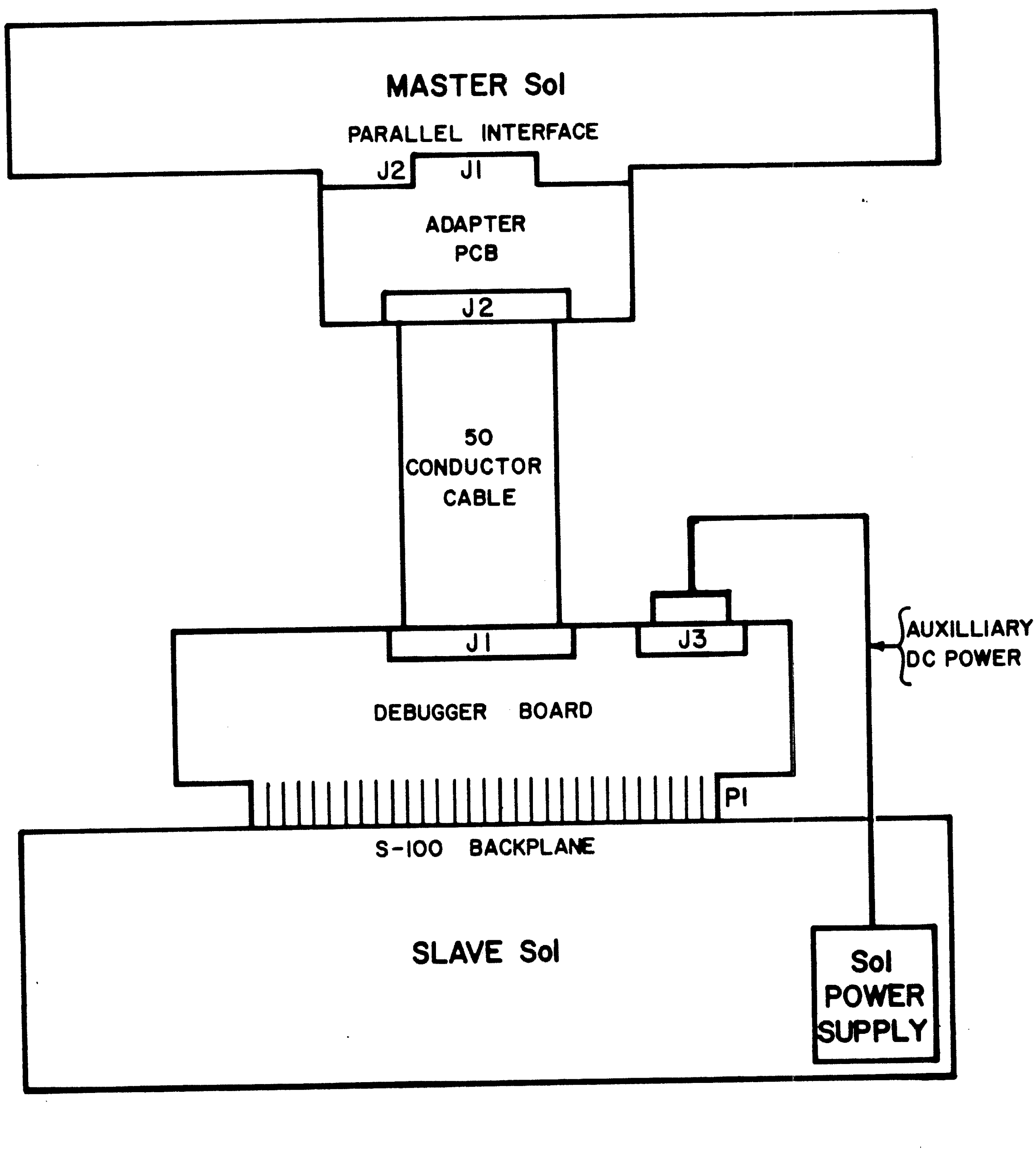


Fig. 2-1. ParaSol Debugger System Interconnect Diagram

SECTION 2

ASSEMBLY AND HARDWARE CHECKOUT

2.Ø INTRODUCTION

1. Unpack the unit and check contents against parts lists in 2.4 of this section.

2. Visually check the PCBs for defects.

3. Using an ohmeter, check to see there are no shorts between the DC power buses and GND.

2.1 ASSEMBLY OF ParaSol DEBUGGER BOARD

(Refer to Fig. 6-1, ParaSol Debugger PCB Assembly and 2.4, Parts List.)

1. Install R1.

2. Install 14-pin DIP socket at U1.

3. Install 16-pin DIP sockets at U2 through U12.

4. (Optional:) Install test point loop at P1, pin 5Ø, GND.

5. Install capacitors C1 through C8. Observe polarity of C7 and C8.

6. Install header J1. Pin 1 is indicated by molded triangle on header. Orient with triangle as shown on assembly drawing.

7. Install 2Ø-pin header J2. Observe same polarity as J1.

8. Install regulator IC U13. Apply heatsink compound between PCB surface and bottom of the regulator. (Refer to Detail A-A on Fig. 6-1.

9. Install J3, Sol backplane DC power cable assembly. Proper orientation is critical! The two white wires should be toward the regulator. The white/yellow wire should be toward J2. Fasten the cable assembly with plastic cable tie for strain relief.

10. Using a VOM, check for shorts between +8 VDC and GND (across C7). Check between +5 VDC and GND (across C8).

11. Plug the PCB into the known-good Sol backplane and check +8 VDC across C7. Check for +5 VDC across C8.

12. If OK, unplug PCB from backplane and install ICs U1 through U12.

13. Sight under ICs for bent out or bent under pins.

14. Mark Rev. level on component side of PCB. (Refer to Fig. 6-1, note 1.)

2.2 ASSEMBLY OF CABLE ADAPTER BOARD

(Refer to Fig. 6-2, ParaSol adapter PCB and corresponding parts list in Sec. 7.)

2.2.1 INSTALLATION OF CONNECTORS

1. On the component side of the PCB, install the 25-pin PC-mount socket connector at J1. Solder all pins and check for shorts and bad solder joints.

2. Mount the header end of the 50-conductor ribbon cable assembly on the component side of the adapter board. Solder and check for shorts and bad connections.

2.2.2 INSTALLATION OF Sol REV LEVEL CONFIGURATION SOCKET

The adapter board is wired to match the parallel I/O connector signals of Sols of rev E and above. Below rev E, the signals PID0 through PID7 are in reverse order. The following procedure allows the adapter board to be configured for any revision of the Sol. (Parts for this modification are not provided in the kit.

1. On the trace side of the cable adapter PCB, cut each of the 8 traces between the two vertical rows of socket pads provided for a 16-pin socket at J3. Cut them down the middle.

2. Install a 16-pin socket on the component side at J3.

3. Make a Rev. E configuration plug from a 16-pin component carrier by reconstructing with jumper wire the pattern of the traces which you cut in step 1. Check for shorts and bad solder joints.

4. Wire a second configuration plug for rev D and lower from another 16-pin component carrier by reversing the connections to pins 9 through 16 as shown below in Table 2-1, Pin Connections for Configuration Plugs. Check for shorts and bad solder joints.

Table 2-1. Pin Connections for Configuration Plugs

REV E PIN CONNECTIONS FROM	TO	REV D PIN CONNECTIONS FROM	TO
1 --	16	1 --	9
2 --	15	2 --	10
3 --	14	3 --	11
4 --	13	4 --	12
5 --	12	5 --	13
6 --	11	6 --	14
7 --	10	7 --	15
8 --	9	8 --	16

2.3 TEST PROCEDURE

(Refer to Fig 2-1, System Interconnect Diagram.)

1. Connect the female end of the 50-conductor flat cable to the header J1 of the Debugger PCB. Observe the orientation of pin 1. A colored stripe on one side of the cable and an arrow on the cable connector indicates pin 1. Proper orientation will result in the cable leading off from J1 toward J2. (Refer to Fig. 2-2, Orientation of Debugger Cable Connections.)

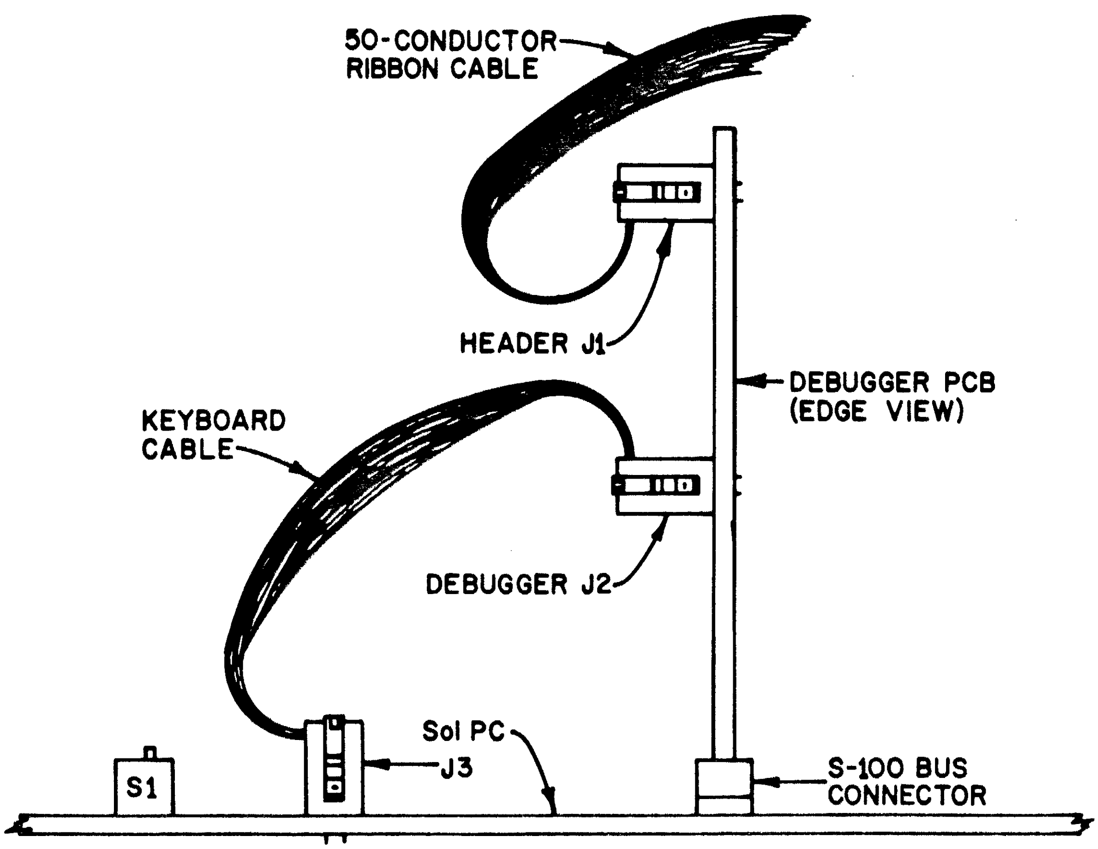


Fig. 2-2. Orientation of Debugger Cable Connections.

CAUTION

> Once the flat cable is installed, do not remove unless necessary. The cable and header are not built for frequent removal and re-installation.

2. Disconnect the backplane power cable from the backplane of a Sol, and mate it with the J3 power connector on the debugger board. Do not plug the debugger Board into the S-100 Bus of the Sol. Place the debugger Board on a piece of cardboard or other non-conductive material so there is no possibility of shorts.

3. Next plug the 25-pin connector on the cable adapter board into the parallel I/O connector (J2) of another Sol.

4. Apply power to both Sols.

5. Use the ENTR command to load the following program into the Sol which has the adapter board installed (referred to as the master Sol).

```
C900  AF          XRA A
C901  D3 FA       OUT 0FAH     PIE HIGH
C903  3E 08       MVI A,8
C905  D3 FA       OUT 0FAH     PIE LOW
C907  C3 00 C9    JMP 0C900H   OVER AND OVER
```

6. Now EXEC C900. The PIE signal on pin 5 of U1 on the debugger board should be a square wave with a normal TTL swing.

7. Change the byte at C904 to 10 hex; then EXEC C900 again. The PUS signal on pin 1 of U1 on the debugger board should be a square wave with a normal TTL swing.

8. Enter the following program to check the keyboard register:

```
C900  3E 00       MVI A,0
C902  D3 FA       OUT 0FAH     SELECT KEYBOARD REG.
C904  3C          INR A
C905  D3 FD       OUT 0FDH     LOAD KEYBOARD REG.
C907  C3 04 C9    JMP 0C904H   LOOP
```

9. EXEC C900. PIE should be high and PUS should be low. All 8 outputs of the keyboard register should be square waves. KBD1 should be oscillating at half the rate of KBD0, KBD2 should be half the rate of KBD1, etc. The outputs of the keyboard register are pins 11, 12, 13, and 14 of U4 and U5.

10. Enter the following program to check the DIO register, and the bit of the control register that enables the DIO register outputs:

```
C900  3E 10      MVI A,10H
C902  D3 FA      OUT 0FAH    SELECT CONTROL REG.
C904  AF         XRA A
C905  D3 FD      OUT 0FDH    ENABLE DIO REGISTER
C907  3E 18      MVI A,18H
C909  D3 FA      OUT 0FAH    SELECT DIO REG.
C90B  3C         INR A
C90C  D3 FD      OUT 0FDH    LOAD DIO REG.
C90E  C3 0B C9   JMP 0C90BH  LOOP
```

11. EXEC C900. PIE should be low, PUS should be high. Pins 1 and 2 of the DIO register ICs, U3 and U7 should be low. All 8 outputs of the DIO register should be square waves with the same rate relationship as the keyboard register outputs during the previous test. The DIO register outputs are pins 3, 4, 5, and 6 of U3 and U7.

12. Enter this program to test the control register:

```
C900  3E 10      MVI A,10H
C902  D3 FA      OUT 0FAH     SELECT CONTROL REG.
C904  3C         INR A
C905  D3 FD      OUT 0FDH     LOAD CONTROL REG.
C907  C3 04 C9   JMP 0C904H   LOOP
```

13. EXEC C900. The outputs of the control register should all be square waves as in the previous two tests. These outputs are pins 3, 4, 5, and 6 of U6 and U8. The signal at pin 8 of U1 should be a narrow pulse which goes low for a period of 1/2 to 1 microsecond.

All debugger board outputs are now tested. Inputs are tested at the beginning of the debugger program. If a bad input is detected the system stops with all signals in a static state so the error can be easily traced.

2.4 PARTS LIST - PARASOL DEBUGGER, TOP ASSEMBLY (900038A)

ITEM #	PART #	QTY	REFERENCE CODE	STANDARD PART#/DESCRIPTION
3	727051	1		Cassette, Software, Parasol Debugger

PARTS LIST - PARASOL DEBUGGER, PCB ASSEMBLY (900043A)

ITEM #	PART #	QTY	REFERENCE CODE	STANDARD PART #/DESCRIPTION
1	900040	1		Fab, PCB, Parasol Debugger
4	717002	1	J2	Header, Male, PC, 20 Pin
5	717003	1	J1	Header, Male, PC, 50 Pin
6	103003	1	J3	Assy, Power Cable, Sol Backplane Board
8	720024	1		Pan Head Machine Screw, 6-32 x 3/8
9	720011	1		Hex Nut, 6-32
10	720041	1		Internal Tooth Lock Washer, #6
12	707034	1	R1	Resistor, 560 ohms, 1/4 W, 5%
13	707036	1	C7	Cap, 15uf, Tant, 20 V, 10%
14	707032	1	C8	Cap, 1.0 uf, Tant, 35 V, 10%
15	707023	5	C2,3,4,5,6	Cap, .047 uf, Disc Cer, +80-20%
16	707011	1	C1	Cap, 680 pf, Disc Cer, 10%
18	701162	1	U13	<u>7805</u>,* LM340T-5, Volt Reg., +5
19	701124	1	U1	74LS132, Quad 2-Input Nand
20	701188	1	U2	8T98, Hex Buf/Inv
21	701134	4	U9,10,11,12	74LS153, Dual 4-to-1 Line MPX
22	701077	4	U3,6,7,8	<u>74173</u>, 8T10, TRI-State 4 Bit Latch
23	701142	2	U4,5	74LS163, Synch 4-Bit Bin Cntr
25	713004	1	U1	Socket, DIP, 14-Pin
26	713006	11	U2-U12	Socket, DIP, 16-Pin
27	722011	1		Tie, Cable, Plastic
28	721000	A/R	U13	Heatsink Compound

PARTS LIST - PARASOL ADAPTER, PCB ASSEMBLY, (900042A)

ITEM #	PART #	QTY	REFERENCE CODE	STANDARD PART #/DESCRIPTION
1	900041	1		Fab, PCB, Parasol Adapter
4	900032	1	J2	Assy, Cable, 50 Cond, Gen. Purpose
5	717011	1	J1	Socket, 25-Pin, PC
6	720013	2		Pan Head Machine Screw, 4-40 x 7/16"
7	720038	2		Internal Tooth Lock Washer, #4
8	720020	2		Hex Nut, 4-40

*The underline number is the standard vendor part number, others are possible equivalents.

SECTION 3

HARDWARE THEORY OF OPERATION

3.1 OVERVIEW

The ParaSol Debugger contains the logic necessary to connect the parallel interface of a master Sol to S-100 bus of a slave Sol. This is a master slave relationship since the Sol whose parallel interface is connected to the ParaSol is in control of the slave Sol. The master Sol can act as a front panel for the slave Sol, displaying the address bus, data bus, and status signals of the slave Sol. The master Sol can start, stop, or single step the Slave. The master Sol can automatically test the buses and memory of the slave Sol as well as trace through a program sequence.

A program in the master Sol controls all the operations of the ParaSol. The logic on the Debugger board allows input/output instructions which reference the parallel interface of the master Sol to access the address, data, status, and control signals of the S-100 bus of the slave. These control signals are XRDY, $\overline{\text{DIGI}}$, PRESET, MWRITE, FRDY, $\overline{\text{DO DSBL}}$, and $\overline{\text{PHANTOM}}$. XRDY is used to stop the CPU of the slave Sol mid-way through a machine cycle. While stopped, the master Sol can then input the data on the buses of the slave, or use $\overline{\text{DIGI}}$ to substitute data from a register on the Debugger board for the data from the slave's memory.

An 8080 instruction cycle is defined as the time required to fetch and execute an instruction. Every instruction cycle consists of one to five machine cycles. The ParaSol debugger single steps one machine cycle at a time. A machine cycle is the sequence of events that occur when the 8080 accesses memory or an I/O port. Each machine cycle consists of three to five T-states of 500 nanoseconds each. The timing of signals during the T-states of a machine cycle cannot be monitored or controlled by the ParaSol Debugger. Refer to the 8080 Microcomputer Systems Users Manual from Intel for detailed information on the operation of the 8080.

3.2 THE Sol PARALLEL INTERFACE

The ParaSol Debugger interfaces to the master Sol through the parallel interface connector at the rear of the unit. Data is sent to the Debugger on the eight lines named POD0 to POD7. The signal POL (low active) is the output strobe. Data is received from the Debugger on the eight lines named PID0 to PID7. The 8080 accesses these data lines through port FD (hex). The signals PIE and PUS are Sol output lines used to select the source or destination of data within the debugger. The 8080 controls PIE through bit 3 of port FA (hex); PUS is controlled by bit 4 of the same port.

3.3 THE PARASol PARALLEL INTERFACE

A 50-conductor flat cable forms a path for the parallel interface signals between the master Sol and the debugger board. An adapter board on the Sol end of the flat cable connects the signals to the even-numbered conductors, and ground to all odd-numbered conductors. The interface signals enter the debugger board through a 50-conductor header designated J1.

The eight PID lines connect to the outputs of 74LS153 four-to-one multiplexers, U9,U10,U11, and U12. The POD lines connect to the data inputs of three 8-bit registers. The keyboard data register is made up of two 74LS163 ICs, U4 and U5. The DIO register, U3 and U7, consists of two 74173 ICs. The control register is made up of two more 74173s, U6 and U8. The signals PIE and PUS connect to the select inputs of the four-to-one multiplexer to control which group of S-100 bus signals will be placed on the PID lines. PIE and PUS also connect to the enable inputs of the ICs that make up the three 8-bit registers. The clock inputs of the three registers are all connected to the signal $\overline{\text{POL}}$. When $\overline{\text{POL}}$ makes a transition from low to high the data on the POD lines is loaded into the register enabled by PIE and PUS.

3.4 THE PARASol S-100 INTERFACE

Inputs to the ParaSol Debugger from the S-100 bus of the slave Sol are multiplexed onto the PID lines in four groups. One group consists of the eight DIO lines. Address lines A0 to A7 and A8 to A15 form two other groups. The following status signals form the last group:

PID Line	Status Signal
PID0	SM1
PID1	SINP
PID2	SOUT
PID3	SSTACK
PID4	MEMRITE
PID5	*
PID6	PDBIN
PID7	PWAIT

*PID5 is not connected to the S-100 bus. Optionally, it could be connected to a test probe or auxilliary input.

Outputs from the Debugger to the S-100 bus of the slave Sol consist of seven lines from the control register, and the eight outputs of the DIO register. Bit 0 of the control register pulls the S-100 signal XRDY low (not ready) when it is zero. If XRDY has been pulled low by bit 0, then when Bit 1 of the control register makes a transition from zero to one, XRDY will go high (ready) for a period of time greater than 500 nanoseconds and less than 1 microsecond. Thus, a positive transition on bit 1 of the control register single steps the slave Sol. The S-100 signal $\overline{\text{DIGI}}$ (low active) is controlled by bit 2 of the control register. When bit 2 is low, $\overline{\text{DIGI}}$ forces PDBIN inactive and switches the data input multiplexers of the Sol to the DIO bus, also the tri-state outputs of the DIO register on the Debugger are enabled. This is how the DIO register contents are jammed into the 8080 of the slave Sol.

Bit 3 of the control register controls the S-100 signal PRESET (low active). When bit 3 is low the 8080 of the slave Sol is reset. When bit 4 of the control register is low the S-100 signal FRDY (low active) disables the MWRITE tri-state driver in the slave Sol, and a tri-state driver on the Debugger board forces MWRITE low (inactive). In this state the memory of the slave Sol is write-protected. The S-100 signal $\overline{\text{DO DSBL}}$ (low active) which disables the tri-state drivers between the 8080 of the slave Sol and the DIO bus is controlled by bit 5 of the control register. Bit 6 of the control register is connected to the S-100 signal $\overline{\text{PHANTOM}}$ (low active) in the slave Sol. This signal causes the personality module of the Sol to be addressed at location zero. Finally, bit 7 of the control register activates the signal KDR (low active) at the twenty pin header on the Debugger board (J2) used to connect to the keyboard input of the slave Sol. When Bit 7 makes a positive transition, the KDR flip-flop in the slave Sol is set to indicate that keyboard data is ready.

The S-100 lines designated DIO0 to DIO7 are connected to the tri-state outputs of the DIO register. When these outputs are enabled by bit 2 of the control register the contents of the DIO register is placed on the DIO bus of the slave Sol.

The outputs of the keyboard register are connected to the appropriate pins of the 20-pin header on the Debugger board (J2) used to connect to the keyboard input of the slave Sol via a flat cable.

3.5 CONTROL SEQUENCES

The ParaSol Debugger is controlled by input and output instruction sequences of the program in the master Sol. A sequence normally begins with an output to port FA (hex). Bits 4 and 5 of this port control the parallel interface signals PIE and PUS which are used by the debugger to select the source or destination for data. Table 3.1 below summarizes the PIE and PUS selection codes.

Table 3-1, PIE and PUS SELECTION CODES

PUS	PIE	SOURCE	DESTINATION
0	0	A8 to A15	Keyboard Register
0	1	A0 to A7	Keyboard Register
1	0	DIO Bus	Control Register
1	1	Status	DIO Register

For example, if the value 1Ø(hex) is output to port FA(hex), then PUS will be a one and PIE will be one also since it is connected to the Q output of a flip-flip. The status lines will become the source for data and the DIO register will be the destination. The next input or output instruction of the sequence references port FD(hex) to either load the accumulator with the data on the status lines of the slave Sol, or transfer the contents of the accumulator to the DIO register on the Debugger board.

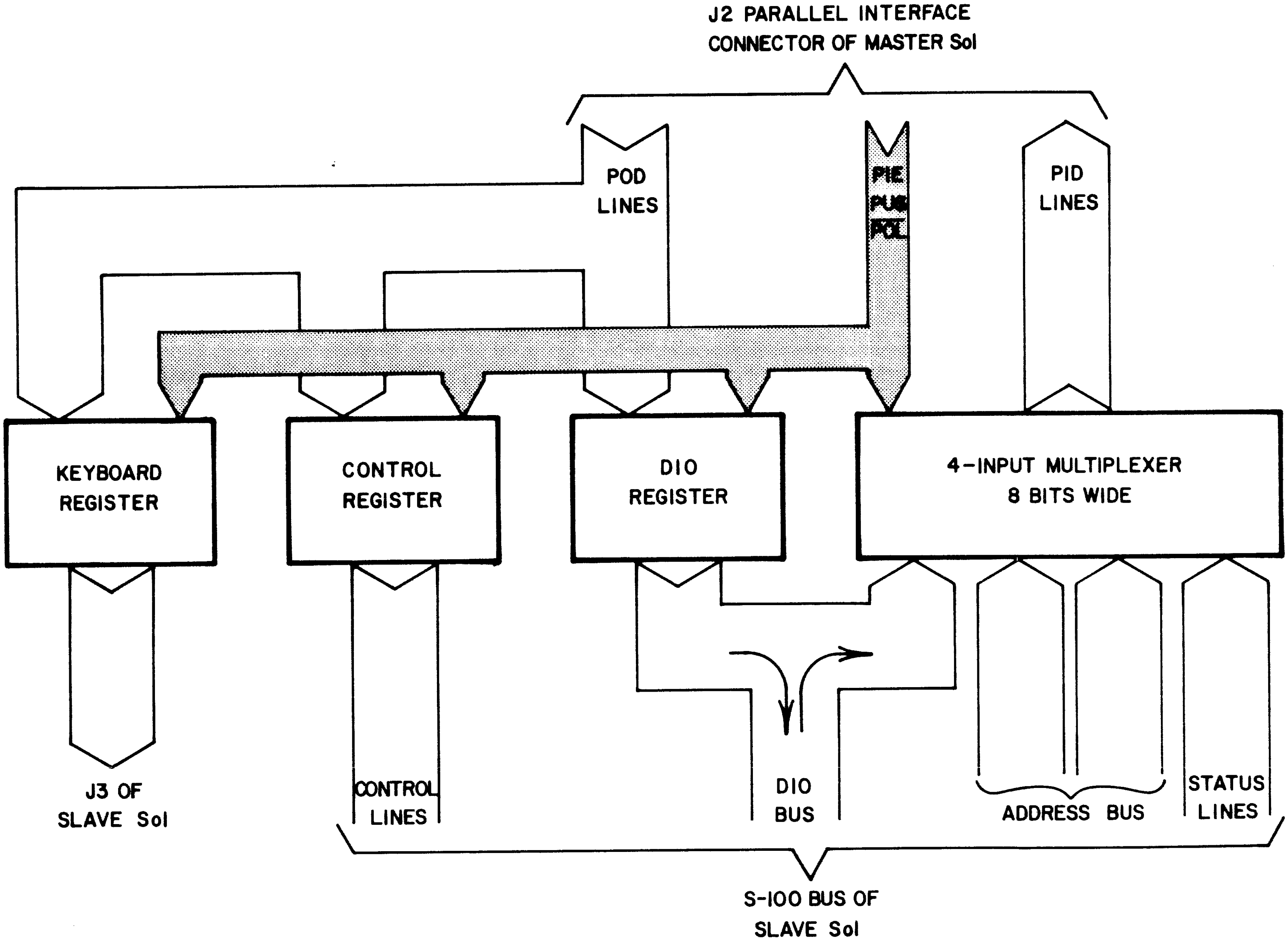


Fig. 3-1. ParaSol Functional Block Diagram

SECTION 4

SOFTWARE THEORY OF OPERATION

4.Ø INTRODUCTION

This section includes instructions for loading the source files of the Debugger program, descriptions of the subroutines used in the program, flow charts of test routines, and a source listing of the debugger program.

4.1 TRANSFERRING SOURCE FILES TO DISK

Follow this procedure to load the source files for the Debugger Program and transfer them to PTDOS disk files. Note that these files are not directly compatible with the ALS-8.

1. Insert a PTDOS System diskette in the disk drive. Enter the command: BOOT

2. Connect a tape recorder to the Sol as described in Section 7 of the Sol Systems Manual. This Sol must also have a Helios II Disk System installed.

3. Place the cassette tape in the recorder; depress the play lever, and enter this command: GET PSRC1

4. When the file is loaded, the Sol will display the first address, and the load count like this: PSRC1 XXXX YYYY

The first address is represented here by XXXX; the load count is represented by YYYY. Write down these two numbers since they will be erased from the screen shortly.

5. TYPE: EX BCBØ.

6. When the PTDOS prompt (*) appears, use the two addresses you wrote down when the cassette file was loaded in this PTDOS command: WRITE PSRC1,XXXX,>YYYY

7. Repeat this procedure from step 4 for the two remaining files, PSRC2 and PSRC3.

8. To return to SOLOS from PTDOS, use the command EXEC CØØ4. To reenter PTDOS from SOLOS use the command EXEC BCBØ.

4.2 BASIC PARASol SUBROUTINES

4.2.1 SETCTL

This routine is called to change the contents of the control register on the debugger board. A RAM buffer named CTLSAV is updated by SETCTL. CTLSAV contains a copy of the control register contents and is used when individual bits are to be changed. No registers or flags are altered by SETCTL.

Example 1: Set PRESET low.

```
LDA CTLSAV       GET CURRENT CONTENTS
ANI FF-RESET     AND WITH F7 hex
CALL SETCTL      CHANGE CONTROL REG.
```

Example 2: Set PRESET high.

```
LDA CTLSAV       GET CURRENT CONTENTS
ORI RESET        OR WITH Ø8 hex
CALL SETCTL      CHANGE CONTROL REG.
```

4.2.2 SETDIO

This subroutine is called to load the DIO register. When called, the value in the accumulator is transferred to the DIO register. No registers or flags are altered.

Example: Load DIO register with C3 hex.

```
MVI A,ØC3H       ACCUMULATOR=C3
CALL SETDIO      DIO REGISTER=C3
```

4.2.3 SETKBD

This routine is called to simulate the depression of a key on the keyboard of the slave Sol. The value in the accumulator is sent to the slave Sol. The accumulator and flags are altered.

Example: Simulate depression of the A key:

```
MVI A,'A'        ACCUMULATOR=A
CALL SETKBD      SEND TO SLAVE
```

4.2.4 GETDIO

Call this subroutine to move the contents of the DIO bus of the slave Sol to the accumulator, and to the RAM buffer DIOBUF.

4.2.5 GETSTA

Call this subroutine to move the contents of the status lines of the slave Sol to the accumulator and the RAM buffer SBUF.

4.2.6 GETADR

Call this subroutine to move the contents of the address bus of the slave Sol to registers H and L and to the RAM buffer ADBUF. The accumulator is altered by this subroutine.

4.2.7 SSTEP

Call this subroutine to allow the slave Sol to complete the current machine cycle and begin the next cycle. The accumulator and flags are altered.

4.2.8 M1TST

This subroutine returns with the zero flag set if the current machine cycle is an instruction fetch cycle. The accumulator and flags are altered.

4.2.9 FINDM1

If the current cycle is not a fetch cycle, this subroutine will call SSTEP until a fetch cycle is reached. If the current cycle is an instruction fetch cycle, FINDM1, will return without single stepping. The accumulator and flags are altered.

4.2.10 TILM1

This subroutine first calls SSTEP then checks for an instruction fetch cycle. If necessary the process is repeated until an instruction fetch cycle is reached. TILM1 differs from FINDM1 in that if the current cycle is an instruction fetch cycle, TILM1 will single step to the next instruction fetch cycle; FINDM1 will not.

4.2.11 AJAM

This subroutine is used to jam the contents of the accumulator into the 8080 of the slave Sol. To accomplish this, the subroutine loads the DIO register, calls SETCTL to activate the $\overline{\text{DIGI}}$ signal, checks that the DIO bus matches the data, and finally calls SSTEP. If the DIO bus does not match the data, a jump to the error routine DIBAB takes place. The B-register and flags are altered.

4.2.12 POKE

Call this subroutine to move the contents of the accumulator to the memory locaton pointed to by register-pair HL. This is like a MOV M,A instruction except that the memory referenced is in the slave Sol. No registers or flags are altered.

4.2.13 PEEK

Call this subroutine to move the contents of the memory location pointed to by register-pair HL to the accumulator. This is like a MOV A,M instruction except that the memory referenced is in the slave Sol. The accumulator is altered.

Example: Move location 1000 in the slave Sol to the accumulator.

```
        LXI H,1000H        POINT HL
        CALL PEEK          GET THE DATA
```

4.2.14 PORTI

Call this routine to move the data at an input port to the accumulator. When called, register A should contain the port number to be accessed. The accumulator and the flags are altered.

Example: Load the accumulator from port FF hex of the slave Sol.

```
MVI A,0FFH          PORT TO BE READ
CALL PORTI          GET THE DATA
```

4.2.15 JUMP1

Call this routine to cause the slave Sol to jump to the address in register-pair HL. The accumulator and flags are altered.

Example: Jump slave Sol to address C000 hex:

```
LXI H,0C000H        DESIRED ADDRESS
CALL JUMP1          JUMP THERE
```

4.3 I/O ROUTINES

4.3.1 OSOUT

Call this subroutine with the character to be output in register-B.

4.3.2 INA

This subroutine waits for a character from the keyboard. If the ESCAPE key is pressed this routine exits to SOLOS. If any other key is pressed it is returned in the accumulator and register-B.

4.3.3 INCHR

This subroutine checks the keyboard once. On return, if a key has been pressed and it is not the ESCAPE key, the character is in the accumulator and B-register, and the zero flag is reset. If no key has been pressed, the zero flag is set. If the ESCAPE key has been pressed this subroutine returns control to SOLOS.

4.3.4 ADOUT

This subroutine prints the contents of register-pair HL as a 4 digit hex number.

4.3.5 HEXA

This subroutine prints the contents of the accumulator as a 2 digit hex number.

4.3.6 BINOT

This subroutine prints the contents of the accumulator as a binary number.

4.3.7 BINOX

This Subroutine is similar to BINOT except that instead of printing a binary number with ones and zeros, the character in register L is substituted for zeros, and the character in register H is substituted for ones. This subroutine is used to print the error report for the RAM tests.

4.3.8 AOUTB

Call this routine to print the contents of register pair HL as a binary number.

4.3.9 TEXTO

This subroutine is called to print strings of text. Register-pair HL should be loaded with the address of the beginning of the string. When the up-arrow character (^) is encountered in the string, all the characters until the next up-arrow character are converted to control characters. The slash character (/) terminates the string.

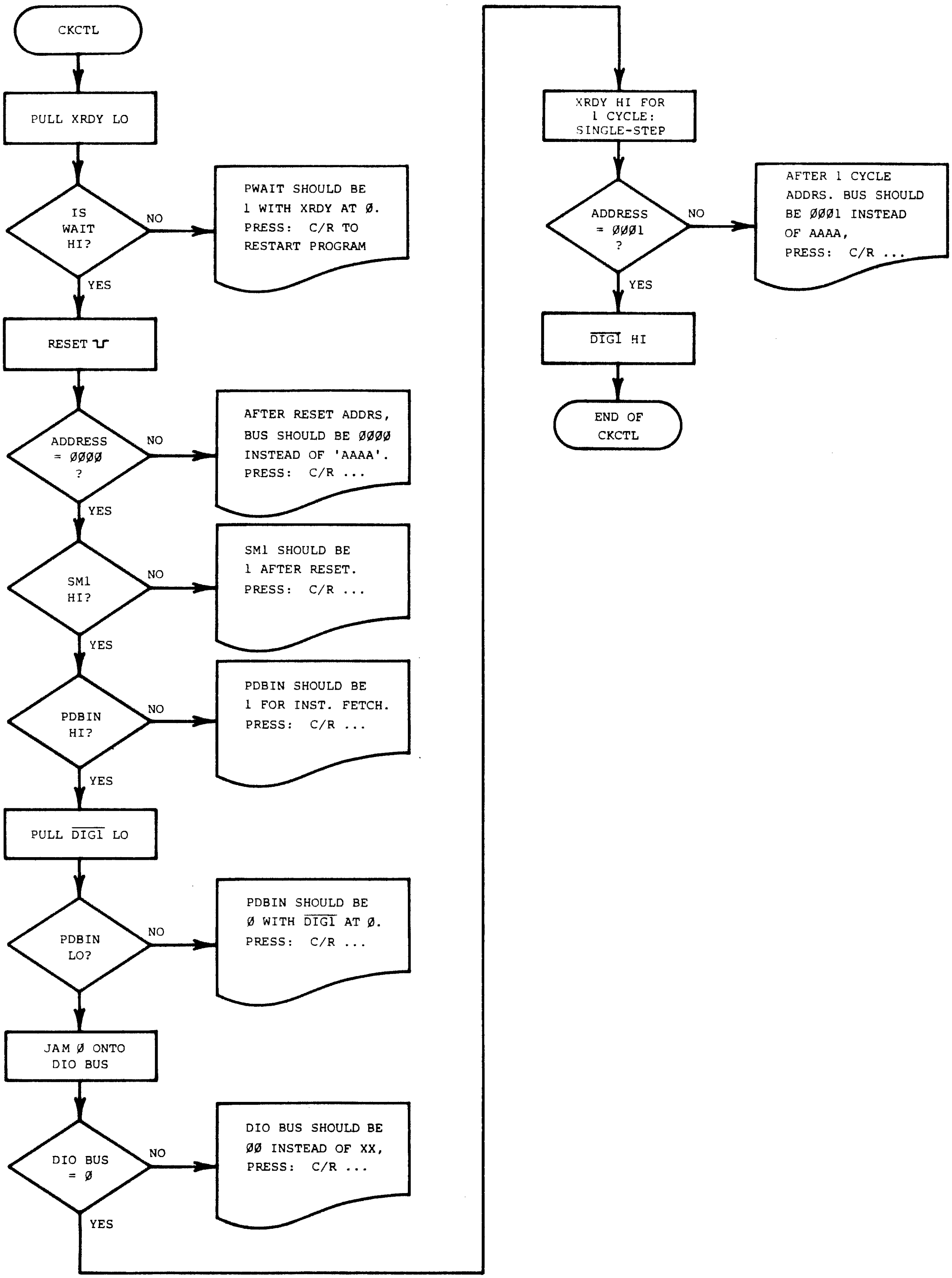
Example:

```
        LXI H,MSG1        POINT TO STRING
        CALL TEXTO        PRINT IT

      MSG1 ASC "LINE 1^MJ^LINE 2^MJ^LINE 3/"

        Prints as:

                    LINE 1
                    LINE 2
                    LINE 3
```

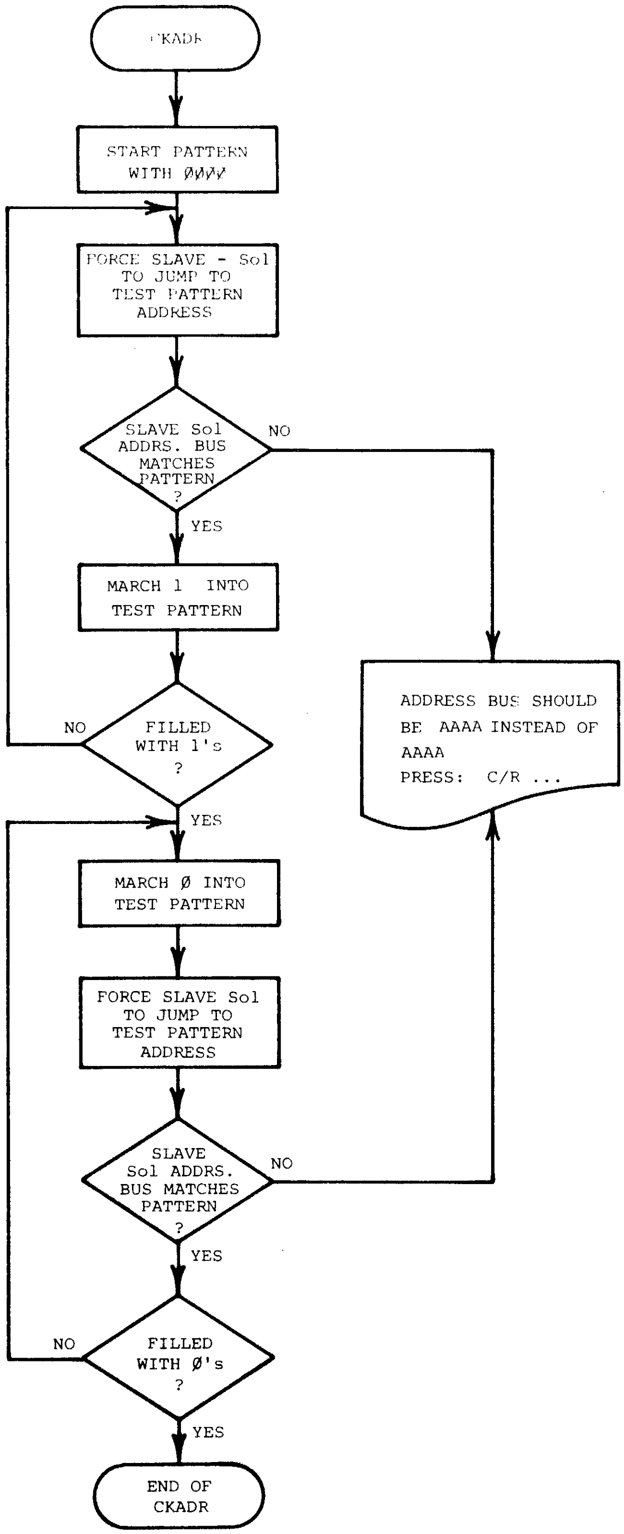
4.4 FLOW CHARTS

The following flowcharts depict the test routines run during the Test Mode by the Debugger Program. They are presented here to provide information concerning what the debugger program was doing when an error was encountered.

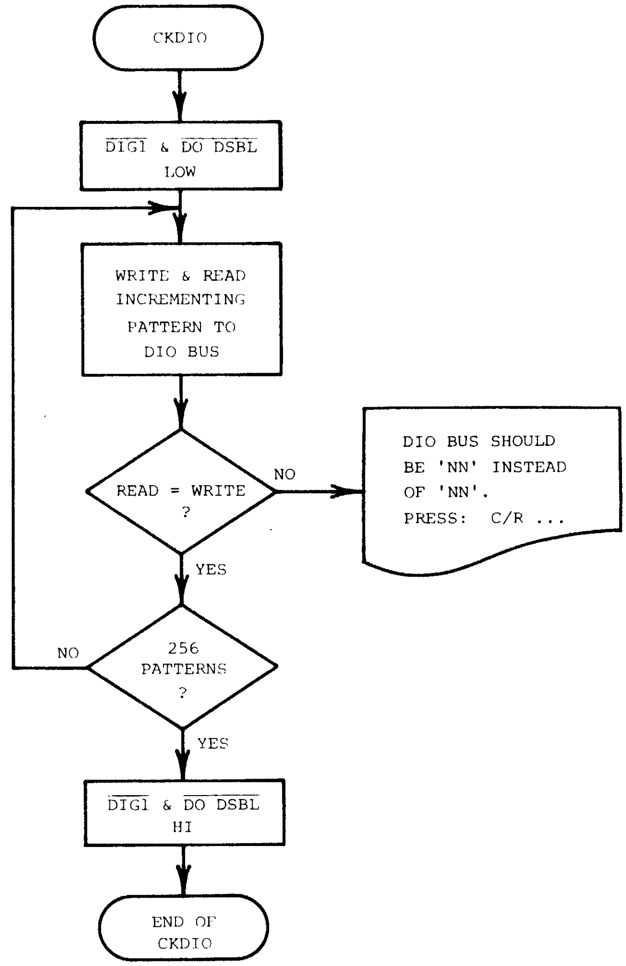
TEST CONTROLS



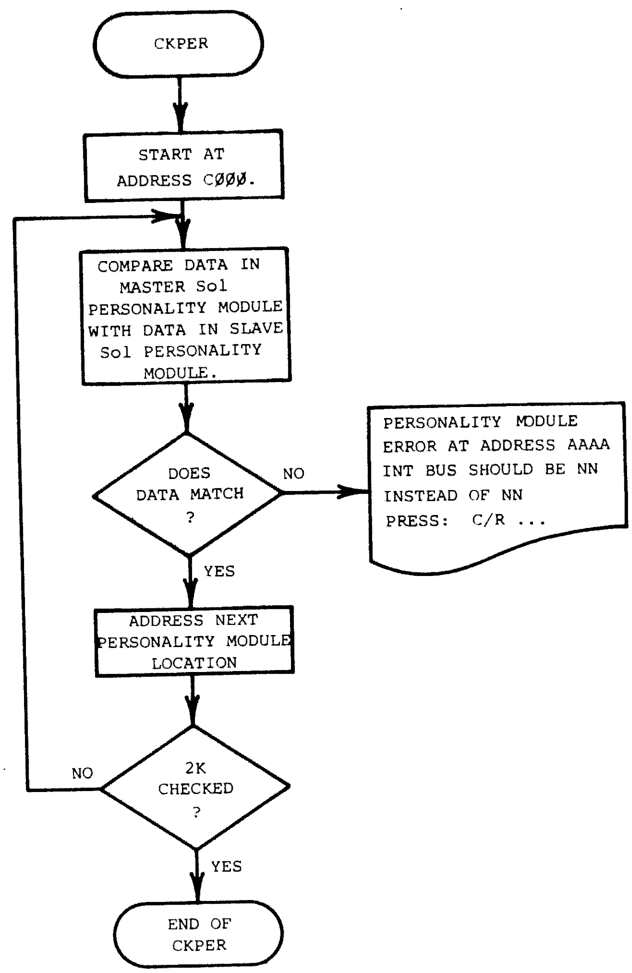
TEST ADDRESS BUS



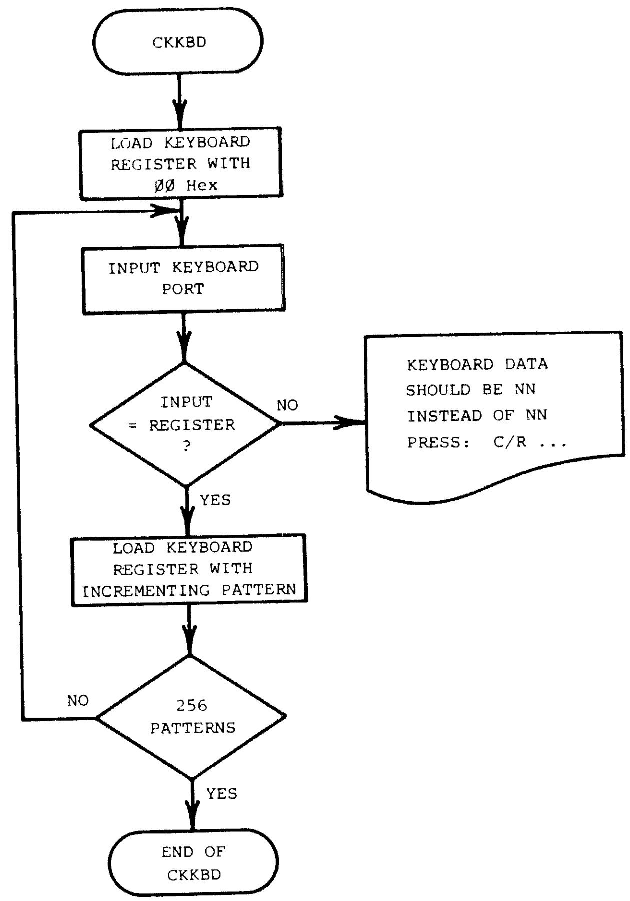
TEST DIO BUS



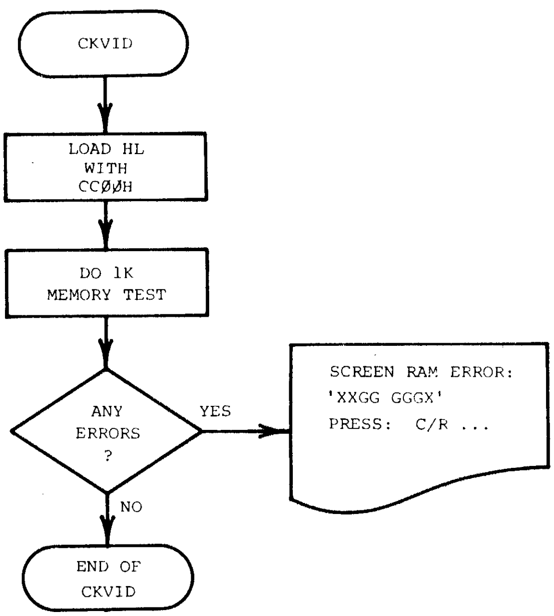
TEST INTERNAL BUS & PERSONALITY MODULE



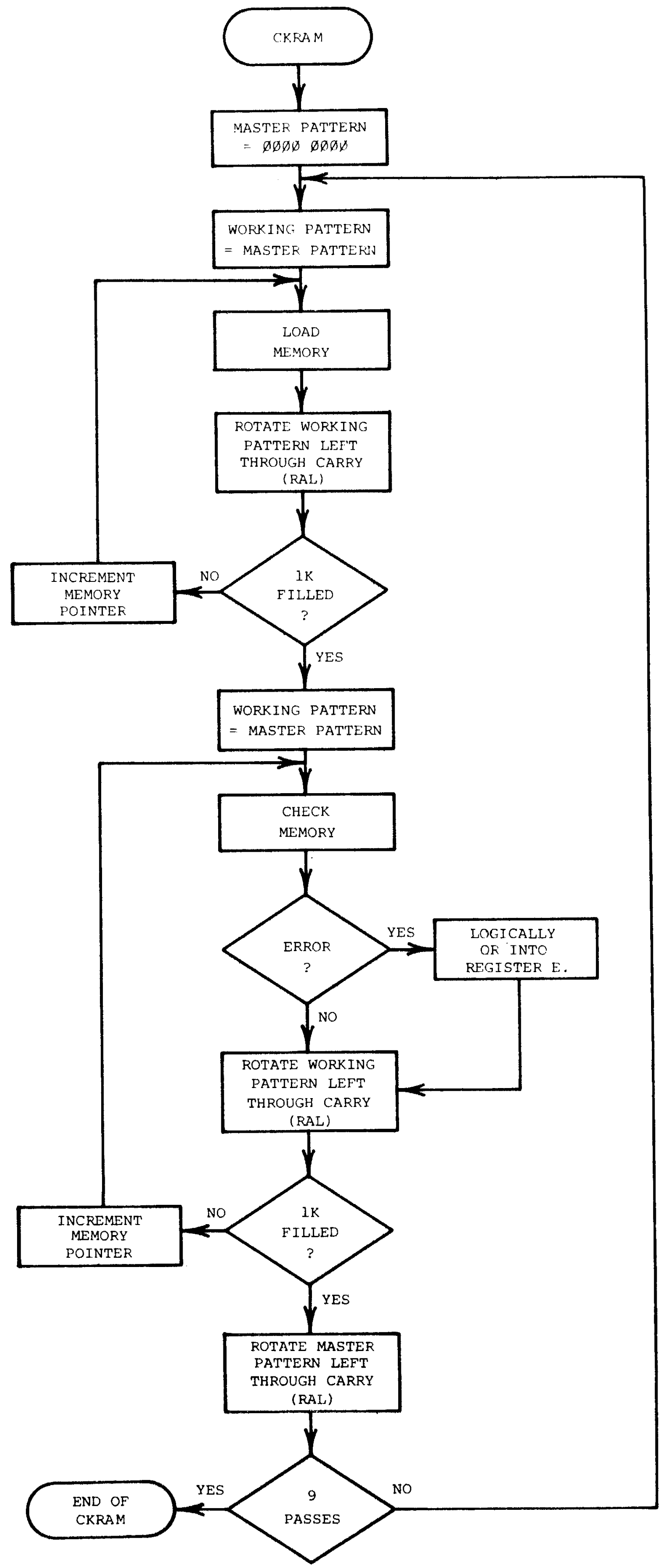
KEYBOARD INTERFACE TEST



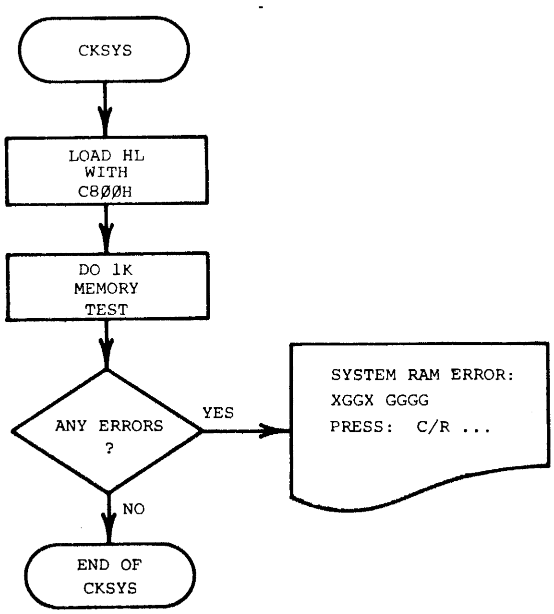
TEST VIDEO RAM



1K MEMORY TEST



TEST SYSTEM RAM



4.5 SOURCE LISTING

The Source listing which follows is included in this manual to encourage and assist you in expanding the usefulness of the Debugger Program. It may also be of assistance when a question arises as to how the program has identified an error.

```
                    *******************************
                    ** ParaSol Debugger Program **
                    **                           **
                    **         Rev. A            **
                    **                           **
                    **         4/1/78            **
                    **                           **
                    **  Copyright (C) 1978, by   **
                    **  Processor Technology     **
                    **       Corporation         **
                    **   All rights reserved.    **
                    *******************************
                    *
                    *
0000                         ORG     0
                             XEQ     0
                    *
                    **  PIE & PUS SELECTION EQUATES
                    *
                    **  DATA DESTINATIONS
                    *
     0018           OUTDIO   EQU     18H        DIO REGISTER
     0010           OUTCTL   EQU     10H        CONTROL REGISTER
     0000           OUTKBD   EQU     0          KEYBOARD REGISTER
                    *
                    **  DATA SOURCES
                    *
     0010           INDIO    EQU     10H        DIO BUS
     0018           INSTAT   EQU     18H        STATUS
     0000           INHIAD   EQU     0          A8 TO A15
     0008           INLOAD   EQU     8          A0 TO A7
                    *
                    **  PORT EQUATES
                    *
     00FA           SELECT   EQU     0FAH       PIE PUS PORT
     00FD           DATA     EQU     0FDH       DATA PORT
     00FC           KPORT    EQU     0FCH       KEYBOARD PORT
                    *
                    **  CONTROL REGISTER BIT EQUATES
                    *
     0001           STOP     EQU     1          STOP/RUN BIT
     0002           STEP     EQU     2          SINGLE STEP BIT
     0004           JAM      EQU     4          DIG1 CONTROL BIT
     0008           RESET    EQU     8          RESET BIT
     0010           FRDY     EQU     10H        FRDY BIT
     0020           DODSB    EQU     20H        DO DSBL BIT
     0040           PHNTM    EQU     40H        PHANTOM BIT
     0080           KDR      EQU     80H        KBD. STROBE BIT
                    *
     00FF           FF       EQU     0FFH
                    *
                    *
                    ******************************************
                    ** MAIN ROUTINE. EXECUTION BEGINS HERE **
                    ******************************************
                    *
                    *
0000 C3 0C 00                JMP     MAIN       STARTUP FROM 0000 IS NORMAL
                    *
0003 31 FF CB                LXI     SP,0CBFFH  STARTUP FROM 0003 SKIPS TESTS
0006 CD 73 02                CALL    TITLE
0009 C3 E1 04                JMP     DISP
                    *
```

```
    000C                MAIN     EQU     $
000C 31 FF CB                    LXI     SP,0CBFFH  STACK AT TOP OF SOL RAM
                        *
000F CD 73 02                    CALL    TITLE      ANNOUNCE PROGRAM
0012 CD 3C 00                    CALL    CKCTL      CHECK CONTROLS
0015 CD 26 0B                    CALL    BOUT       SPACE AFTER EACH TEST
0018 CD A0 00                    CALL    CKDIO      CHECK DIO BUS
001B CD 26 0B                    CALL    BOUT
001E CD C3 00                    CALL    CKADR      CHECK ADRS BUS
0021 CD 26 0B                    CALL    BOUT
0024 CD F2 00                    CALL    CKPER      CHECK PERS. MODULE
0027 CD 26 0B                    CALL    BOUT
002A CD 29 01                    CALL    CKKBD      CHECK KEYBOARD INPUT
002D CD 26 0B                    CALL    BOUT
0030 CD 3C 01                    CALL    CKVID      CHECK SCREEN MEMORY
0033 CD 26 0B                    CALL    BOUT
0036 CD 48 01                    CALL    CKSYS      CHECK SYSTEM MEMORY
0039 C3 E1 04                    JMP     DISP
                        *
                        *
                        **  THESE ROUTINES PERFORM TESTS. FAILURES ABORT NORMAL
                        **  RETURN AND RESULT IN JUMP BACK TO MAIN.
                        *
                        **  CHECK CONTROL SIGNALS
                        *
    003C                CKCTL    EQU     $
003C 3E FF                       MVI     A,FF
003E CD 34 08                    CALL    SETCTL     FREE RUN
                        *
0041 AF                          XRA     A
0042 E3                 FDLAY    XTHL
0043 E3                          XTHL
0044 3D                          DCR     A          WASTE TIME
0045 C2 42 00                    JNZ     FDLAY      FOR A WHILE
                        *
0048 3E F7                       MVI     A,FF-RESET
004A CD 34 08                    CALL    SETCTL     RESET LO
004D E6 FE                       ANI     FF-STOP
004F CD 34 08                    CALL    SETCTL     NOW XRDY LO
0052 F6 08                       ORI     RESET
0054 CD 34 08                    CALL    SETCTL     THEN RESET HI
                        *
0057 CD 6A 08                    CALL    GETSTA     GET STATUS
005A E6 80                       ANI     80H        WAIT HI ?
005C CA CD 01                    JZ      ERR01      NO --->
                        *
005F CD 73 08                    CALL    GETADR     GET ADRS.
0062 B5                          ORA     L          ADRS=0 ?
0063 C2 ED 01                    JNZ     ERR02      NO --->
                        *
0066 3A 2F 0C                    LDA     SBUF       GET STATUS
0069 47                          MOV     B,A
006A E6 01                       ANI     1          M1 HI ?
006C CA 4C 02                    JZ      ERR03      NO --->
                        *
006F 78                          MOV     A,B
0070 E6 40                       ANI     40H        PDBIN HI ?
0072 CA 55 02                    JZ      ERR04      NO --->
                        *
0075 3A 2D 0C                    LDA     CTLSAV     GET CTL. BITS
0078 E6 FB                       ANI     FF-JAM
007A CD 34 08                    CALL    SETCTL     DIG1 LOW
007D CD 6A 08                    CALL    GETSTA
```

```
0080 E6 40                   ANI      40H         PDBIN LO ?
0082 C2 5E 02                JNZ      ERR05       NO --->
                   *
0085 AF                      XRA      A
0086 CD A8 08                CALL     AJAM        NOP TO TEST Sol
0089 CD 73 08                CALL     GETADR
008C AF                      XRA      A           CHECK ADRS
008D B4                      ORA      H           H=0
008E C2 67 02                JNZ      ERR07       NO --->
0091 7D                      MOV      A,L
0092 3D                      DCR      A           L=1 ?
0093 C2 67 02                JNZ      ERR07       NO --->
0096 3A 2D 0C                LDA      CTLSAV
0099 F6 04                   ORI      JAM
009B CD 34 08                CALL     SETCTL      DIG1 HI
009E AF                      XRA      A
009F C9                      RET      .           CONTROLS OK
                   *
                   **  CHECK DIO BUS
                   *
00A0 3A 2D 0C      CKDIO     LDA      CTLSAV
00A3 E6 DB                   ANI      FF-JAM-DODSB
00A5 CD 34 08                CALL     SETCTL      DIG1&DO DSBL LOW
00A8 06 00                   MVI      B,0         INITIALIZE PATTERN
                   *
00AA 78            CKDI1     MOV      A,B
00AB CD 40 08                CALL     SETDIO      WRITE PATTERN
00AE CD 61 08                CALL     GETDIO      READ PATTERN
00B1 B8                      CMP      B           SAME ?
00B2 C2 A7 01                JNZ      DIBAD       NO --->
00B5 04                      INR      B           NEXT PATTERN
00B6 C2 AA 00                JNZ      CKDI1       TILL B=0
                   *
00B9 3A 2D 0C                LDA      CTLSAV
00BC F6 24                   ORI      JAM+DODSB
00BE CD 34 08                CALL     SETCTL      DIG1&DO DSBL HI
00C1 AF                      XRA      A
00C2 C9                      RET      . Z SET=DIO BUS OK
                   *
                   ** CHECK ADDRESS BUS
                   *
00C3 21 00 00      CKADR     LXI      H,0         INITIALIZE PATTERN
00C6 0E 11                   MVI      C,17        PASS COUNT=17
                   *
00C8 CD 42 09      CKA1      CALL     JUMP1       LOAD TEST PATTERN
00CB CD E2 00                CALL     ADCHK       CHECK PATTERN
00CE 29                      DAD      H
00CF 23                      INX      H           MARCH 1's
00D0 0D                      DCR      C
00D1 C2 C8 00                JNZ      CKA1        TILL ALL 1's
                   *
00D4 0E 10                   MVI      C,16        PASS COUNT
00D6 29            CKA2      DAD      H           MARCH 0's
00D7 CD 42 09                CALL     JUMP1       LOAD TEST PATTERN
00DA CD E2 00                CALL     ADCHK       CHECK PATTERN
00DD 0D                      DCR      C
00DE C2 D6 00                JNZ      CKA2        TILL PATTERN=1
00E1 C9                      RET      .           ADDRESS BUS OK
                   *
00E2 EB            ADCHK     XCHG     DE=TEST PATTERN
00E3 CD 73 08                CALL     GETADR      HL=ADRS BUS
00E6 7A                      MOV      A,D
00E7 BC                      CMP      H           HI BYTE OK ?
```

```
00E8 C2 F6 01                 JNZ     ADBAD        NO --->
00EB 7B                       MOV     A,E
00EC BD                       CMP     L            LO BYTE OK ?
00ED C2 F6 01                 JNZ     ADBAD        NO --->
00F0 EB                       XCHG    .            HL=TEST PATTERN
00F1 C9                       RET     .            ADRS OK
                    *
                    **  CHECK PERSONALITY MODULE
                    *
     00F2           CKPER     EQU     $
00F2 21 00 C0                 LXI     H,0C000H     P.M. STARTS HERE
                    *
00F5 CD 0A 09       CKPE1     CALL    PEEK         GET TEST VALUE
00F8 BE                       CMP     M            COMPARE WITH MASTER
00F9 C2 04 01                 JNZ     PMBAD        IT´S BAD
                    *
00FC 23                       INX     H            BUMP POINTER
00FD 7C                       MOV     A,H
00FE FE C8                    CPI     0C8H         DONE ?
0100 C2 F5 00                 JNZ     CKPE1        NOT YET
0103 C9                       RET
                    *
0104 F5             PMBAD     PUSH    PSW          SAVE DATA
0105 E5                       PUSH    H            SAVE ADDRESS
0106 21 C4 03                 LXI     H,ERM30
0109 CD 60 0B                 CALL    TEXTO        "AT P.M. ADDRESS "
                    *
010C E1                       POP     H
010D E5                       PUSH    H            DISPLAY ERROR ADDRESS
010E CD 05 0B                 CALL    ADOUT
                    *
0111 21 ED 03                 LXI     H,ERM31
0114 CD 60 0B                 CALL    TEXTO        "INT BUS S.B."
0117 E1                       POP     H
0118 7E                       MOV     A,M
0119 CD 0E 0B                 CALL    HEXA         DISP. CORRECT DATA
                    *
011C 21 AC 03                 LXI     H,ERM18
011F CD 60 0B                 CALL    TEXTO        "INSTEAD OF "
0122 F1                       POP     PSW
0123 CD 0E 0B                 CALL    HEXA         DISPLAY BAD DATA
0126 C3 D3 01                 JMP     ERS
                    *
                    ** TEST KBD INTERFACE
                    *
0129 06 00          CKKBD     MVI     B,0          INITIALIZE PATTERN
                    *
012B 78             CKBD1     MOV     A,B
012C CD 46 08                 CALL    SETKBD       PRESS A KEY
012F 3E FC                    MVI     A,KPORT
0131 CD 53 09                 CALL    IPORT        INPUT KBD PORT
0134 B8                       CMP     B            DATA OK ?
0135 C2 12 02                 JNZ     KBBAD        NO. PROCESS ERROR
0138 04                       INR     B            YES
0139 C2 2B 01                 JNZ     CKBD1 TRY NEXT PATTERN
                    *
                     TEST VIDEO RAM
                    *
013C 21 00 CC       CKVID     LXI     H,0CC00H     SCREEN MEMORY
013F CD 54 01                 CALL    CKRAM        IS TESTED
0142 7B                       MOV     A,E
0143 B7                       ORA     A            ANY ERRORS ?
0144 C2 30 02                 JNZ     VIBAD        YES
```

```
0147 C9                        RET      .                 NO
                        *
                        **   TEST SYSTEM RAM
                        *
0148 21 00 C8           CKSYS  LXI      H,0C800H          SYSTEM SEMORY
014B CD 54 01                  CALL     CKRAM
014E 7B                        MOV      A,E
014F B7                        ORA      A                 ANY ERRORS ?
0150 C2 42 02                  JNZ      SYBAD             YES
0153 C9                        RET      .                 NO
                        *
                        **   THIS ROUTINE TESTS 1K OF MEMORY BEGINNING AT THE
                        **   ADDRESS POINTED TO BY HL.  ON RETURN E REGISTER
                        **   BITS WHICH ARE 1 CORRESPOND TO BAD MEMORY BITS.
                        *
0154 2E 00              CKRAM  MVI      L,0
0156 7C                        MOV      A,H
0157 E6 FC                     ANI      0FCH              FORCE A 1K BOUNDRY
0159 67                        MOV      H,A
                        *
015A C6 04                     ADI      4
015C 57                        MOV      D,A               D = H + 1K
                        *
015D 1E 00                     MVI      E,0               START WITH NO ERRORS
015F 43                        MOV      B,E               MASTER PATTERN = 0
                        *
0160 48                 MPASS  MOV      C,B               C=WORKING PATTERN
                        *
0161 79                 MPOKE  MOV      A,C               A=WORKING PATTERN
0162 CD C8 08                  CALL     POKE              PATTERN TO MEMORY
                        *
0165 B7                        ORA      A                 CARRY=0. PATTERN=0 ?
0166 CA 6D 01                  JZ       SET11             YES. GO INCREMENT
0169 17                        RAL      .                 ELSE SHIFT IT
016A C3 6E 01                  JMP      MTES3             AND UPDATE ADDRESS
                        *
016D 3C                 SET11  INR      A                 INC. PATTERN
                        *
016E 4F                 MTES3  MOV      C,A               C=NEW PATTERN
016F 23                        INX      H
0170 7C                        MOV      A,H
0171 BA                        CMP      D                 1K FILLED ?
0172 C2 61 01                  JNZ      MPOKE             NO. KEEP POKIN ALONG
                        *
0175 7C                        MOV      A,H               YES IT'S FULL
0176 DE 04                     SBI      4                 REWIND ADDRESS
0178 67                        MOV      H,A
                        *
0179 48                        MOV      C,B               C=WORKING PATTERN
017A CD 0A 09           MPEEK  CALL     PEEK              CHECK MEMORY
017D A9                        XRA      C                 ANY ERRORS ?
017E CA 83 01                  JZ       NOERR             NOPE
                        *
0181 B3                        ORA      E                 OPPS !
0182 5F                        MOV      E,A               ACCUMULATE ERRORS
                        *
0183 79                 NOERR  MOV      A,C               A=WORKING PATTERN
0184 B7                        ORA      A                 CARRY=0. PATTERN=0 ?
0185 CA 8C 01                  JZ       SET12             YES. INCREMENT IT
                        *
0188 17                        RAL      .                 ELSE SHIFT PATTERN
0189 C3 8D 01                  JMP      MTES4             AND UPDATE PATTERN
                        *
```

```
018C 3C                 SET12    INR    A             INC. PATTERN
                        *
018D 4F                 MTES4    MOV    C,A           C=NEW PATTERN
018E 23                          INX    H
018F 7C                          MOV    A,H
0190 BA                          CMP    D             1K CHECKED ?
0191 C2 7A 01                    JNZ    MPEEK         NOPE

0194 7C                          MOV    A,H           YES
0195 DE 04                       SBI    4             REWIND ADDRESS
0197 67                          MOV    H,A
                        *
0198 78                          MOV    A,B
0199 B7                          ORA    A             CARRY=0. MASTER PATTERN=0
019A C2 A1 01                    JNZ    MTES5         NO. JUST ROTATE
                        *
019D 04                          INR    B             ELSE UPDATE TO 1
019E C3 60 01                    JMP    MPASS         NEW PASS
                        *
01A1 17                 MTES5    RAL    .             NEW MASTER
01A2 D8                          RC     .             DONE IF 9 PASSES
01A3 47                          MOV    B,A           ELSE UPDATE PATTERN
01A4 C3 60 01                    JMP    MPASS         NEW PASS
                        *
                        **  THESE ROUTINES PROCESS ERRORS, THEN CONTROL GOES
                        **  BACK TO MAIN THROUGH THE ROUTINE ERS.
                        *
                        ** DISP. DIO BUS ERRORS
                        *
     01A7               DIBAD    EQU    $
01A7 78                          MOV    A,B
01A8 F5                          PUSH   PSW           SAVE WRITE PATTERN
01A9 21 94 03                    LXI    H,ERM08
01AC CD 60 0B                    CALL   TEXTO         " DIO S.B."
01AF F1                          POP    PSW           WRITE PAT.
01B0 CD 2B 0B                    CALL   BINOT
01B3 21 AC 03                    LXI    H,ERM18
01B6 CD 60 0B                    CALL   TEXTO         "INSTEAD OF"
01B9 3A 2E 0C                    LDA    DIOBUF        READ PAT.
01BC CD 2B 0B                    CALL   BINOT
01BF CD 84 0B                    CALL   CRLF
01C2 3A 2D 0C                    LDA    CTLSAV
01C5 F6 24                       ORI    JAM+DODSB
01C7 CD 34 08                    CALL   SETCTL        DIG1&DO DSBL HI
01CA C3 D3 01                    JMP    ERS           ERROR RETURN
                        *
01CD 21 7A 02           ERR01    LXI    H,ERM01
01D0 CD 60 0B                    CALL   TEXTO         "NO WAIT"
                        *
                        ** ERRORS END UP HERE
                        *
01D3 21 A4 02           ERS      LXI    H,ERM11
01D6 CD 60 0B                    CALL   TEXTO         "CR TO RESTART"
01D9 CD EF 0A           ERS1     CALL   INA           WAIT FOR A KEY
01DC FE 0D                       CPI    0DH           CR ?
01DE C2 D9 01                    JNZ    ERS1          ---> NO
01E1 3A 34 08                    LDA    SETCTL
01E4 F6 04                       ORI    JAM
01E6 CD 40 08                    CALL   SETDIO        DIG1 HI
01E9 B7                          ORA    A
01EA C3 0C 00                    JMP    MAIN          START FRESH
                        *
01ED 21 C5 02           ERR02    LXI    H,ERM02
```

```
01F0 CD 60 0B              CALL    TEXTO      "ADRS. <> 0000"
01F3 11 00 00              LXI     D,0
                      *
                      ** DISP. ADRS BUS ERRORS
                      *
01F6 21 D5 02         ADBAD    LXI     H,ERM12
01F9 CD 60 0B              CALL    TEXTO      "ADRS S.B."
01FC EB                    XCHG    .          GOOD ADRS.
01FD CD 51 0B              CALL    AOUTB      IN BINARY
0200 21 EF 02              LXI     H,ERM22
0203 CD 60 0B              CALL    TEXTO      "INSTEAD OF"
0206 2A 30 0C              LHLD    ADBUF      BAD ADRS.
0209 CD 51 0B              CALL    AOUTB      IN BINARY
020C CD 84 0B              CALL    CRLF
020F C3 D3 01              JMP     ERS
                      *
                      ** PROCESS KEYBOARD ERRORS **
                      *
0212 78               KBBAD    MOV     A,B
0213 F5                    PUSH    PSW        SAVE CORRECT VALUE
0214 21 35 04              LXI     H,ERM60
0217 CD 60 0B              CALL    TEXTO "KEYBOARD PORT S.B."
021A F1                    POP     PSW
021B CD 2B 0B              CALL    BINOT      DISPLAY CORRECT DATA
                      *
021E 21 EF 02              LXI     H,ERM22
0221 CD 60 0B              CALL    TEXTO "INSTEAD OF"
0224 3A 2E 0C              LDA     DIOBUF
0227 CD 2B 0B              CALL    BINOT      DISPLAY BAD DATA
022A CD 84 0B              CALL    CRLF
022D C3 D3 01              JMP     ERS
                      *
                      ** PROCESS VIDEO RAM ERRORS **
                      *
0230 F5               VIBAD    PUSH    PSW
0231 21 05 04              LXI     H,ERM40
0234 CD 60 0B              CALL    TEXTO      "SCREEN RAM ERROR: "
                      *
0237 2E 47            MAPX     MVI     L,'G'
0239 26 58                 MVI     H,'X'
023B F1                    POP     PSW
023C CD 2F 0B              CALL    BINOX
023F C3 D3 01              JMP     ERS
                      *
                      ** PROCESS SYSTEM RAM ERRORS **
                      *
0242 F5               SYBAD    PUSH    PSW
0243 21 1D 04              LXI     H,ERM50
0246 CD 60 0B              CALL    TEXTO      "SYSTEM RAM ERROR: "
0249 C3 37 02              JMP     MAPX
                      *
024C 21 09 03         ERR03    LXI     H,ERM03
024F CD 60 0B              CALL    TEXTO      "SM1 S.B. 1"
0252 C3 D3 01              JMP     ERS
                      *
0255 21 2C 03         ERR04    LXI     H,ERM04
0258 CD 60 0B              CALL    TEXTO      "PDBIN S.B. 1"
025B C3 D3 01              JMP     ERS
                      *
025E 21 55 03         ERR05    LXI     H,ERM05
0261 CD 60 0B              CALL    TEXTO      "PDBIN S.B. 0"
0264 C3 D3 01              JMP     ERS
                      *
```

```
0267 21 7E 03         0441 ERR07    LXI     H,ERM07
026A CD 60 0B         0442          CALL    TEXTO       "AFTER S.S."
026D 11 01 00         0443          LXI     D,0001
0270 C3 F6 01         0444          JMP     ADBAD       DISP. ADRS´S.
                      0445
                      0446 **  ANNOUNCE PROGRAM
                      0447 *
0273 21 4F 04         0448 TITLE    LXI     H,TIMSG
0276 CD 60 0B         0449          CALL    TEXTO
0279 C9               0450          RET
                      0451
                      0452 **  ERROR MESSAGES
                      0453 *
027A 5E 4D 4A 5E      0454 ERM01    ASC     "^MJ^´PWAIT´ SHOULD BE 1 WITH ´XRDY´ AT 0./
     27 50 57 41
     49 54 27 20
     53 48 4F 55
     4C 44 20 42
     45 20 31 20
     57 49 54 48
     20 27 58 52
     44 59 27 20
     41 54 20 30
     2E 2F
02A4 5E 4D 4A 5E      0455 ERM11    ASC     "^MJ^TYPE ´CR´ TO RESTART PROGRAM/"
     54 59 50 45
     20 27 43 52
     27 20 54 4F
     20 52 45 53
     54 41 52 54
     20 50 52 4F
     47 52 41 4D
     2F
                      0456 *
02C5 5E 4D 4A 5E      0457 ERM02    ASC     "^MJ^AFTER RESET/
     41 46 54 45
     52 20 52 45
     53 45 54 2F
02D5 5E 4D 4A 5E      0458 ERM12    ASC     "^MJ^ADRS. BUS SHOULD BE: /
     41 44 52 53
     2E 20 42 55
     53 20 53 48
     4F 55 4C 44
     20 42 45 3A
     20 2F
02EF 5E 4D 4A 5E      0459 ERM22    ASC     "^MJ^         INSTEAD OF: /
     20 20 20 20
     20 20 20 20
     20 49 4E 53
     54 45 41 44
     20 4F 46 3A
     20 2F
                      0460 *
0309 5E 4D 4A 5E      0461 ERM03    ASC     "^MJ^´SM1´ SHOULD BE 1 AFTER RESET./
     27 53 4D 31
     27 20 53 48
     4F 55 4C 44
     20 42 45 20
     31 20 41 46
     54 45 52 20
     52 45 53 45
     54 2E 2F
                      0462 *
```

```
032C 5E 4D 4A 5E      0463 ERM04     ASC      "^MJ^'PDBIN' SHOULD BE 1 FOR INST. FETCH./
     27 50 44 42
     49 4E 27 20
     53 48 4F 55
     4C 44 20 42
     45 20 31 20
     46 4F 52 20
     49 4E 53 54
     2E 20 46 45
     54 43 48 2E
     2F
                      0464 *
0355 5E 4D 4A 5E      0465 ERM05     ASC      "^MJ^PDBIN' SHOULD BE 0 WITH 'DIG1' AT 0./
     50 44 42 49
     4E 27 20 53
     48 4F 55 4C
     44 20 42 45
     20 30 20 57
     49 54 48 20
     27 44 49 47
     31 27 20 41
     54 20 30 2E
     2F
                      0466 *
037E 5E 4D 4A 5E      0467 ERM07     ASC      "^MJ^AFTER SINGLE STEP/
     41 46 54 45
     52 20 53 49
     4E 47 4C 45
     20 53 54 45
     50 2F
                      0468 *
0394 5E 4D 4A 5E      0469 ERM08     ASC      "^MJ^DIO BUS SHOULD BE: /
     44 49 4F 20
     42 55 53 20
     53 48 4F 55
     4C 44 20 42
     45 3A 20 2F
03AC 5E 4D 4A 5E      0470 ERM18     ASC      "^MJ^       INSTEAD OF: /
     20 20 20 20
     20 20 20 49
     4E 53 54 45
     41 44 20 4F
     46 3A 20 2F
                      0471 *
03C4 5E 4D 4A 5E      0472 ERM30     ASC      "^MJ^PERSONALITY MODULE ERROR AT ADDRESS /"
     50 45 52 53
     4F 4E 41 4C
     49 54 59 20
     4D 4F 44 55
     4C 45 20 45
     52 52 4F 52
     20 41 54 20
     41 44 44 52
     45 53 53 20
     2F
03ED 5E 4D 4A 5E      0473 ERM31     ASC      "^MJ^INT BUS SHOULD BE: /"
     49 4E 54 20
     42 55 53 20
     53 48 4F 55
     4C 44 20 42
     45 3A 20 2F
```

```
     53 43 52 45
     45 4E 20 52
     41 4D 20 45
     52 52 4F 52
     3A 20 20 2F
                        0476 *
041D 5E 4D 4A 5E        0477 ERM50     ASC       "^MJ^SYSTEM RAM ERROR:  /"
     53 59 53 54
     45 4D 20 52
     41 4D 20 45
     52 52 4F 52
     3A 20 20 2F
                        0478 *
0435 5E 4D 4A 5E        0479 ERM60     ASC       "^MJ^KBD. PORT SHOULD BE: /"
     4B 42 44 2E
     20 50 4F 52
     54 20 53 48
     4F 55 4C 44
     20 42 45 3A
     20 2F
                        0480 *
044F 5E 4B 4A 4A        0481 TIMSG     ASC       "^KJJ^          Processor Technology Corp.^MJJ^"
     4A 5E 20 20
     20 20 20 20
     20 20 20 20
     50 72 6F 63
     65 73 73 6F
     72 20 54 65
     63 68 6E 6F
     6C 6F 67 79
     20 43 6F 72
     70 2E 5E 4D
     4A 4A 5E
047E 20 20 20 20        0482           ASC       "          ParaSol  Debugger  Program^MJJ^"
     20 20 20 20
     20 20 50 61
     72 61 53 6F
     6C 20 20 44
     65 62 75 67
     67 65 72 20
     20 50 72 6F
     67 72 61 6D
     5E 4D 4A 4A
     5E
04A7 20 20 20 20        0483           ASC       "              Rev. A^MJJJ^TESTS IN PROGRESS...^MJJ^/
     20 20 20 20
     20 20 20 20
     20 20 20 20
     20 20 20 20
     52 65 76 2E
     20 41 5E 4D
     4A 4A 4A 5E
     54 45 53 54
     53 20 49 4E
     20 50 52 4F
     47 52 45 53
     53 2E 2E 2E
     5E 4D 4A 4A
     5E 2F
                        0484 *
                        0485 *
                        0486 ********************************
                        0487 **  TRACE MODE MAIN ROUTINE  **
```

```
                   ***********************************
                   *
                   *
     04E1          DISP     EQU      $
                   *
04E1 3E FE         DISPC    MVI      A,FF-STOP
04E3 CD 34 08               CALL     SETCTL      SET XRDY LO
04E6 21 B6 09               LXI      H,STMSG     DISPLAY
04E9 CD 60 0B               CALL     TEXTO       START-UP MESSAGE
04EC CD 43 07      DISP1    CALL     START       START-UP P.M.
04EF CD 44 06               CALL     BRINI       INITIALIZE BREAK CONDITION
                   *
04F2 3A 37 0C      DISPI    LDA      SFLAG
04F5 F6 01                  ORI      1
04F7 32 37 0C               STA      SFLAG       FORCE SINGLE STEP MODE
                   *
04FA CD 84 0B      DISP2    CALL     CRLF
04FD CD 6A 08               CALL     GETSTA      GET STATUS BITS
0500 CD 99 08               CALL     M1CHK       M1 CYCLE ?
0503 C2 09 05               JNZ      NOTM1       NO--->
0506 CD 84 0B               CALL     CRLF        FOR LF BETWEEN INST'S
0509 CD 2D 07      NOTM1    CALL     DISPA       DISPLAY CYCLE ON SCREEN
                   *
     050C          DISP3    EQU      $
050C CD FF 0A               CALL     INCHR       GOT A KEY ?
050F C2 1D 05               JNZ      DISP4       YES. GO PROCESS
                   *
0512 3A 37 0C               LDA      SFLAG       NO KEY. BUT...
0515 E6 01                  ANI      1           ARE WE SINGLE STEPPING ?
0517 C2 0C 05               JNZ      DISP3       YES. KEEP CHEECKING KBD.
051A C3 72 05               JMP      CYCLE       NO. DO ANOTHER CYCLE
                   *
                   **  IS IT A COMMAND KEY ?
                   *
051D FE 53         DISP4    CPI      'S'         RESTART ?
051F CA E1 04               JZ       DISP        YES --->
0522 FE 20                  CPI      ' '         NEXT CYCLE ?
0524 CA 65 05               JZ       SINGL       YES --->
0527 FE 4A                  CPI      'J'         JUMP ?
0529 CA 5C 06               JZ       GOTO        YES --->
052C FE 42                  CPI      'B'         BREAK ?
052E CA 78 05               JZ       BREAK       YES --->
0531 FE 43                  CPI      'C'         RE-BREAK ?
0533 CA CE 05               JZ       TRAO        YES --->
0536 FE 46                  CPI      'F'         FREE RUN ?
0538 CA 7E 06               JZ       FREE        YES --->
053B FE 52                  CPI      'R'         REGISTER DUMP ?
053D CA A1 06               JZ       REGS        YES --->
                   *
                   **  MAYBE IT'S A SPEED CONTROL KEY
                   *
0540 FE 3A                  CPI      '9'+1       BIGGER THAN 9 ?
0542 D2 72 05               JNC      CYCLE       YES. DO ANOTHER CYCLE
0545 FE 31                  CPI      '1'         GREATER THAN 0 ?
0547 DA 72 05               JC       CYCLE       NO. DO ANOTHER CYCLE
                   *
                   *  IT IS A SPEED CONTROL KEY !
                   *
                   *  1=FASTEST, 9= SLOWEST
                   *
     C80B          SPEED    EQU      0C80BH      SOLOS DISPLAY SPEED PARAMETER
                   *
054A D6 30                  SUI      30H         REMOVE ASCII BIAS
```

```
054C 4F                      MOV     C,A
054D AF                      XRA     A
054E 37                      STC     .           START WITH 1/2
054F 0D            SLOWER    DCR     C           COUNT VALUE DOWN
0550 CA 57 05                JZ      SETSP
0553 17                      RAL     .           MULTIPLY BY 2
0554 C3 4F 05                JMP     SLOWER      AND LOOP
                   *
0557 32 0B C8      SETSP     STA     SPEED       LOAD TRANSLATED VALUE
055A 3A 37 0C                LDA     SFLAG
055D E6 FE                   ANI     0FEH        STOP SINGLE STEPPING
055F 32 37 0C                STA     SFLAG
0562 C3 72 05                JMP     CYCLE       AND DO NEXT CYCLE
                   *
                   **  SP KEY=SINGLE STEP
                   *
0565 3A 37 0C      SINGL     LDA     SFLAG
0568 F6 01                   ORI     1           SET SINGLE STEP FLAG
056A 32 37 0C                STA     SFLAG
056D 3E 04                   MVI     A,4
056F 32 0B C8                STA     SPEED       SET DISP. SPEED
                   *
0572 CD 88 08      CYCLE     CALL    SSTEP       STEP ONCE
0575 C3 FA 04                JMP     DISP2       AND LOOP
                   *
                   ***************************
                   **  TRACE MODE ROUTINES  **
                   ***************************
                   *
                   ** BREAK POINT OPERATION
                   *
0578 21 E5 09      BREAK     LXI     H,BRKMS
057B CD 60 0B                CALL    TEXTO       ASK FOR SPECIFICATION
057E CD DE 0B                CALL    GCLIN       GET IT
0581 CD 0C 0C                CALL    GSAD        FIND ON SCREEN
                   *
0584 CD C2 0B                CALL    SKIP        FIND CYCLE NAME
0587 CA BD 05                JZ      BROP0       THERE ISN'T ONE
058A E5                      PUSH    H           SAVE ITS ADDRESS
058B CD 24 06                CALL    SMOVE       PUT IN BUFFER
                   *
058E E1                      POP     H
058F CD C2 0B                CALL    SKIP        FIND ADDRESS
0592 CA BD 05                JZ      BROP0       THERE ISN'T ONE
0595 EB            TRAP2     XCHG    DE=SCREEN ADRS.
0596 CD 8F 0B                CALL    SHEX1       ASCII TO HEX
0599 CA A5 05                JZ      TRAOK       VALUE OK --->
059C 21 D7 0A                LXI     H,QUMES     MARK BAD VALUE
059F CD 60 0B                CALL    TEXTO
05A2 C3 78 05                JMP     BREAK       TRY AGAIN
                   *
05A5 22 35 0C      TRAOK     SHLD    TADRS       SAVE ADDRESS!
                   *
05A8 3A 38 0C                LDA     CBUFF       GET CYCLE NAME
05AB 21 3A 06                LXI     H,CYTAB     NAME TABLE
05AE 11 3F 06                LXI     D,OPTAB     MASK TABLE
05B1 06 05                   MVI     B,5
                   *
                   * THIS TRANSLATES CYCLE NAME TO STATUS MASK
                   *
05B3 BE            BROP1     CMP     M           TEST CYCLE NAME
05B4 CA C6 05                JZ      BROP2       IT MATCHES !
05B7 23                      INX     H           NEXT NAME
```

```
05B8 13                            INX    D          NEXT STATUS MASK
05B9 05                            DCR    B
05BA C2 B3 05                      JNZ    BROP1      5 TIMES
    05BD             BROP0         EQU    $
05BD 21 D7 0A                      LXI    H,QUMES
05C0 CD 60 0B                      CALL   TEXTO      "???"
05C3 C3 78 05                      JMP    BREAK
                     *
05C6 78              BROP2         MOV    A,B
05C7 32 49 0C                      STA    CYNUM      SAVE FOR LATER
05CA 1A                            LDAX   D
05CB 32 48 0C                      STA    BRKST
                     *
05CE 21 0C 0A        TRAO          LXI    H,BRKM2
05D1 CD 60 0B                      CALL   TEXTO      "BREAK CONDITION="
05D4 21 38 0C                      LXI    H,CBUFF
05D7 CD 60 0B                      CALL   TEXTO      "CY. NAME"
                     *
05DA 2A 35 0C                      LHLD   TADRS      GET TRAP ADRS.
05DD 3A 49 0C                      LDA    CYNUM      IT´S LATER
05E0 FE 03                         CPI    3          IS AN I/O BREAK ?
05E2 DA EB 05                      JC     TRAO1      YES
                     *
05E5 CD 05 0B                      CALL   ADOUT      ELSE DISPLAN AN
05E8 C3 F3 05                      JMP    TRAO2      ADDRESS
                     *
05EB 7D              TRAO1         MOV    A,L        GET PORT #
05EC 67                            MOV    H,A        COPY IN H
05ED 22 35 0C                      SHLD   TADRS      RESTORE TO MEMORY
05F0 CD 0E 0B                      CALL   HEXA       DISPLAY IT
                     *
05F3 21 3A 0A        TRAO2         LXI    H,BRKM3
05F6 CD 60 0B                      CALL   TEXTO      MENTION ABORT
05F9 CD 88 08                      CALL   SSTEP      FOR C. OPTION
                     *
05FC CD 71 07        TRAP3         CALL   TSTEP      BREAK ELSE STEP
05FF CA 1B 06                      JZ     FOUND      SPRING THE TRAP
0602 CD FF 0A                      CALL   INCHR      CHECK KBD
0605 CA FC 05                      JZ     TRAP3      KEEP STEPPING
0608 FE 53                         CPI    ´S´
060A CA E1 04                      JZ     DISP       RESET IF "S"
060D FE 41                         CPI    ´A´        ABORT ?
060F C2 FC 05                      JNZ    TRAP3      NO --->
0612 21 53 0A                      LXI    H,ABTMS    <--- YES
0615 CD 60 0B                      CALL   TEXTO      SAY SO
0618 C3 F2 04                      JMP    DISPI      BACK TO NORMAL
                     *
061B 21 72 0A        FOUND         LXI    H,FMSG1
061E CD 60 0B                      CALL   TEXTO      SAY "FOUND"
0621 C3 F2 04                      JMP    DISPI      SHOW IT
                     *
0624 11 38 0C        SMOVE         LXI    D,CBUFF    COMMAND BUFFER
                     *
0627 7E              SMOV1         MOV    A,M        FROM SCREEN
0628 E6 7F                         ANI    7FH        KILL CURSOR
062A 12                            STAX   D          TO BUFFER
062B FE 20                         CPI    20H        FOUND SPACE ?
062D CA 35 06                      JZ     SMOV2      YES
                     *
0630 23                            INX    H          ELSE CONTINUE
0631 13                            INX    D
0632 C3 27 06                      JMP    SMOV1
                     *
```

```
0635 13                 SMOV2   INX     D
0636 3E 2F                      MVI     A,2FH       '/'
0638 12                         STAX    D           FOR TEXTO
0639 C9                         RET
                        *
     063A               CYTAB   EQU     $           CYCLE NAME TABLE
063A 46 52 57 49                DB      'F','R','W','I','O'
     4F
                        *
     063F               OPTAB   EQU     $           BREAK STATUS MASKS
063F 11                         DB      11H         FETCH
0640 10                         DB      10H         READ
0641 00                         DB      00H         WRITE
0642 02                         DB      02H         INPUT
0643 04                         DB      04H         OUTPUT
                        *
                        ** INITIALIZATION OF BREAK CONDITION
                        *
0644 21 56 06           BRINI   LXI     H,ICY
0647 CD 24 06                   CALL    SMOVE       IS'S A FETCH
064A 3E 11                      MVI     A,11H
064C 32 48 0C                   STA     BRKST       LOAD STATUS MASK
064F 21 00 00                   LXI     H,0         ADRS=0000
0652 22 35 0C                   SHLD    TADRS
0655 C9                         RET
                        *
0656 46 45 54 43        ICY     ASC     "FETCH "
     48 20
                        *
                        *
                        ** JUMP OPERATION
                        *
                        *
065C 21 98 0A           GOTO    LXI     H,GOTMS
065F CD 60 0B                   CALL    TEXTO       ASK FOR ADRS.
0662 CD DE 0B                   CALL    GCLIN       TYPE IN ADRS
0665 CD 0C 0C                   CALL    GSAD        FIND ON SCREEN
0668 EB                         XCHG    .           DE=SCREEN ADRS.
0669 CD 8F 0B                   CALL    SHEX1       ASCII->BINARY
066C CA 78 06                   JZ      GOTO1       VALUE OK
                        *
066F 21 D7 0A                   LXI     H,QUMES
0672 CD 60 0B                   CALL    TEXTO       MARK ERROR
0675 C3 5C 06                   JMP     GOTO        BAD VALUE
                        *
0678 CD 42 09           GOTO1   CALL    JUMP1       GO TO IT
067B C3 FA 04                   JMP     DISP2       AND DISPLAY
                        *
                        *
                        **  FREE RUN OPERATION
                        *
                        *
067E 3E FF              FREE    MVI     A,FF        ALL BITS HI
0680 CD 34 08                   CALL    SETCTL      AWAY WE GO
0683 21 66 09                   LXI     H,FREMS
0686 CD 60 0B                   CALL    TEXTO       SAY "FREE RUNNING"
                        *
0689 CD EF 0A           FREE1   CALL    INA         WAIT FOR A KEY
068C FE 53                      CPI     'S'         RESET ?
068E CA E1 04                   JZ      DISP        YES --->
0691 FE 20                      CPI     ' '         STOP AGAIN ?
0693 C2 89 06                   JNZ     FREE1       NO --->
                        *
```

```
0696 3E FE                    MVI    A,FF-STOP  <--- YES
0698 CD 34 08                 CALL   SETCTL     READY LO
069B CD 9E 08                 CALL   FINDM1     GET A FETCH CYCLE
069E C3 F2 04                 JMP    DISPI      DISPLAY MODE
                   *
                   *
                   **  REGISTER DUMP
                   *
                   *
06A1 CD 96 08      REGS       CALL   M1TST      NEED A FETCH
06A4 CA B6 06                 JZ     REGS1      GOT IT
06A7 CD 88 08                 CALL   SSTEP      ELSE SINGLE STEP
06AA CD 84 0B                 CALL   CRLF
06AD CD 6A 08                 CALL   GETSTA     GET STATUS
06B0 CD 2D 07                 CALL   DISPA      DISPLAY CYCLE
06B3 C3 A1 06                 JMP    REGS       TRY AGAIN
                   *
06B6 21 B4 0A      REGS1      LXI    H,REGMS
06B9 CD 60 0B                 CALL   TEXTO      PRINT LABLES
                   *
06BC 3E 77                    MVI    A,77H      MOV M,A
06BE CD 00 07                 CALL   REGET
06C1 CD 24 07                 CALL   BOUT3
                   *
06C4 3E 70                    MVI    A,70H      MOV M,B
06C6 CD 00 07                 CALL   REGET
06C9 CD 26 0B                 CALL   BOUT
                   *
06CC 3E 71                    MVI    A,71H      MOV M,C
06CE CD 00 07                 CALL   REGET
06D1 CD 24 07                 CALL   BOUT3
                   *
06D4 3E 72                    MVI    A,72H      MOV M,D
06D6 CD 00 07                 CALL   REGET
06D9 CD 26 0B                 CALL   BOUT
                   *
06DC 3E 73                    MVI    A,73H      MOV M,E
06DE CD 00 07                 CALL   REGET
06E1 CD 24 07                 CALL   BOUT3
                   *
06E4 3E 74                    MVI    A,74H      MOV M,H
06E6 CD 00 07                 CALL   REGET
06E9 CD 26 0B                 CALL   BOUT

06EC 3E 75                    MVI    A,75H      MOV M,L
06EE CD 00 07                 CALL   REGET
06F1 CD 24 07                 CALL   BOUT3
                   *
06F4 2A 30 0C                 LHLD   ADBUF
06F7 CD 42 09                 CALL   JUMP1
06FA CD 84 0B                 CALL   CRLF
06FD C3 0C 05                 JMP    DISP3
                   *
0700 F5            REGET      PUSH   PSW
0701 3A 2D 0C                 LDA    CTLSAV
0704 E6 EF                    ANI    FF-FRDY    PROTECT MEMORY
0706 CD 34 08                 CALL   SETCTL
                   *
0709 CD 9E 08                 CALL   FINDM1     CHECK FOR FETCH CYCLE
070C F1                       POP    PSW        GET MOVE INST
070D CD A8 08                 CALL   AJAM       TO SLAVE 8080
                   *
0710 CD 61 08                 CALL   GETDIO     GET REGISTER DATA
```

```
0713 CD 88 08             CALL    SSTEP
0716 3A 2D 0C             LDA     CTLSAV
0719 F6 10                ORI     FRDY      UN-PROT. MEMORY
071B CD 34 08             CALL    SETCTL
071E 3A 2E 0C             LDA     DIOBUF
0721 C3 0E 0B             JMP     HEXA      DISPLAY DATA & RET
                     *
0724 CD 26 0B        BOUT3 CALL   BOUT
0727 CD 26 0B             CALL    BOUT
072A C3 26 0B             JMP     BOUT
                     *
                     *
                     **  DISPLAY A CYCLE
                     *
                     *
072D CD 73 08        DISPA CALL   GETADR    GET ADDRESS
0730 CD 05 0B             CALL    ADOUT     DISPLAY IT
0733 CD 26 0B             CALL    BOUT
0736 CD 26 0B             CALL    BOUT      2 SPACES
0739 CD 61 08             CALL    GETDIO    GET DIO BUS
073C CD 0E 0B             CALL    HEXA      DISPLAY IT
073F CD 91 07             CALL    SDECO     DISPLAY STATUS
0742 C9                   RET
                     *
                     *
                     ** START UP SLAVE PERSONALITY MODULE
                     *
                     *
0743 CD 9E 08        START CALL   FINDM1    GET A FETCH CYCLE
0746 21 00 00             LXI     H,0       ADDRESS 0
0749 CD 42 09             CALL    JUMP1     GO THERE
                     *
074C 3A 2D 0C             LDA     CTLSAV    GET CTL BITS
074F E6 BF                ANI     FF-PHNTM  PHANTOM LO
0751 CD 34 08             CALL    SETCTL
0754 06 04                MVI     B,4       FOUR PHASES
0756 C5              STAR1 PUSH   B
0757 CD 6A 08             CALL    GETSTA    GET STATUS
075A CD 2D 07             CALL    DISPA     DISPLAY ALL
075D CD 84 0B             CALL    CRLF
0760 C1                   POP     B
0761 CD 88 08             CALL    SSTEP
0764 05                   DCR     B         4 CYCLES ?
0765 C2 56 07             JNZ     STAR1     NO --->
                     *
0768 3A 2D 0C             LDA     CTLSAV    GET CTL BITS
076B F6 40                ORI     PHNTM     PHANTOM HI
076D CD 34 08             CALL    SETCTL
0770 C9                   RET     .         THAT'S IT !
                     *
     0771            TSTEP EQU    $         GENERAL BREAK POINT ROUTINE
                     *
0771 CD 73 08             CALL    GETADR    TEST Sol ADRS.
0774 EB                   XCHG    TO DE
0775 2A 35 0C             LHLD    TADRS     TRAP ADRS
0778 7C                   MOV     A,H       COMPARE ADDRESSES
0779 BA                   CMP     D         HI MATCH ?
077A C2 8D 07             JNZ     NOTRP     NO --->
077D 7D                   MOV     A,L       <--- YES
077E BB                   CMP     E         LO MATCH ?
077F C2 8D 07             JNZ     NOTRP NOPE
                     *
0782 CD 6A 08             CALL    GETSTA
```

```
0785 E6 1F                      ANI     1FH         IGNORE  BITS 5,6,7
0787 47                         MOV     B,A         GET STATUS TO B
0788 3A 48 0C                   LDA     BRKST       COMPARE WITH BREAK CONDITION
078B B8                         CMP     B
078C C8                         RZ      .           FOUND IT
                  *
078D CD 88 08         NOTRP     CALL    SSTEP       NEXT CYCLE
0790 C9                         RET     NZ SET=NO MATCH
                  *
                  *
                  ** DECODE STATUS BITS IN SBUF
                  *
                  *
0791 3A 2F 0C         SDECO     LDA     SBUF        GET STATUS
0794 E6 1F                      ANI     1FH         IGNORE BITS 5,6,7
0796 21 BB 07                   LXI     H,STABL     TABLE POINTER
0799 47                         MOV     B,A         PUT STATUS HERE
079A 0E 07                      MVI     C,7         ENTRY COUNT
                  *
079C 7E               SDEC1     MOV     A,M         GET TABLE KEY
079D B8                         CMP     B           STAT=KEY ?
079E CA B2 07                   JZ      SFIND       YES -->
                  *
07A1 0D                         DCR     C           LAST ENTRY ?
07A2 C2 AC 07                   JNZ     SDEC2       NO --->
07A5 21 27 08                   LXI     H,BADS <--  YES
07A8 CD 60 0B                   CALL    TEXTO       PRINT "BAD"
07AB C9                         RET
                  *
07AC 23               SDEC2     INX     H           TRY
07AD 23                         INX     H           NEXT
07AE 23                         INX     H           ENTRY
07AF C3 9C 07                   JMP     SDEC1
                  *
07B2 23               SFIND     INX     H           TO MSG ADRS.
07B3 7E                         MOV     A,M
07B4 23                         INX     H
07B5 66                         MOV     H,M         H=HI BYTE
07B6 6F                         MOV     L,A         L=LO BYTE
07B7 CD 60 0B                   CALL    TEXTO       PRINT SATAUS MSG.
07BA C9                         RET
                  *
                  ** MSG POINTER TABLE
                  *
07BB 11               STABL     DB      11H
07BC D0 07                      DW      INST
07BE 10                         DB      10H
07BF DE 07                      DW      MEMR
07C1 00                         DB      0
07C2 EC 07                      DW      MEMW
07C4 18                         DB      18H
07C5 FB 07                      DW      STKR
07C7 08                         DB      8
07C8 08 08                      DW      STKW
07CA 02                         DB      2
07CB 16 08                      DW      INPT
07CD 04                         DB      4
07CE 1E 08                      DW      OUTP
                  *
                  ** STATUS MESSAGES
                  *
07D0 20 20 49 4E      INST      ASC     "           INST. FETCH/"
     53 54 2E 20
```

```
     46 45 54 43
     48 2F
07DE 20 20 4D 45       0926 MEMR      ASC        "          MEMORY READ/"
     4D 4F 52 59
     20 52 45 41
     44 2F
07EC 20 20 4D 45       0927 MEMW      ASC        "          MEMORY WRITE/"
     4D 4F 52 59
     20 57 52 49
     54 45 2F
07FB 20 20 53 54       0928 STKR      ASC        "          STACK READ/"
     41 43 4B 20
     52 45 41 44
     2F
0808 20 20 53 54       0929 STKW      ASC        "          STACK WRITE/"
     41 43 4B 20
     57 52 49 54
     45 2F
0816 20 20 49 4E       0930 INPT      ASC        "          INPUT/"
     50 55 54 2F
081E 20 20 4F 55       0931 OUTP      ASC        "          OUTPUT/"
     54 50 55 54
     2F
0827 20 20 42 41       0932 BADS      ASC        "          BAD STATUS/"
     44 20 53 54
     41 54 55 53
     2F
                       0933 *
```

```
               *
               *
               *
               *************************************************
               **  THESE ARE THE BASIC PARASOL SUBROUTINES  **
               *************************************************
               *
               *
               **  THESE THREE MOVE DATA FROM THE ACCUMULATOR
               **  TO A REGISTER ON THE DEBUGGER BOARD
               *
               ** MOVE TO CONTROL REGISTER
               *
0834 F5        SETCTL   PUSH     PSW        SAVE ARG.
0835 32 2D 0C           STA      CTLSAV
0838 3E 10              MVI      A,OUTCTL
083A D3 FA     SET1     OUT      SELECT
083C F1                 POP      PSW        GET ARG.
083D D3 FD              OUT      DATA
083F C9                 RET
               *
               ** MOVE TO DIO REGISTER
               *
0840 F5        SETDIO   PUSH     PSW
0841 3E 18              MVI      A,OUTDIO
0843 C3 3A 08           JMP      SET1
               *
               **  MOVE TO KEYBOARD REGISTER
               **  AND STROBE KDR
               *
0846 F5        SETKBD   PUSH     PSW
0847 3E 00              MVI      A,OUTKBD
0849 D3 FA              OUT      SELECT     SET PIE & PUS
084B F1                 POP      PSW
084C D3 FD              OUT      DATA       LOAD DATA
               *
084E 3E 10              MVI      A,OUTCTL
0850 D3 FA              OUT      SELECT     SELECT CTL REG
0852 3A 2D 0C           LDA      CTLSAV     GET CONTROL BYTE
               *
0855 C5                 PUSH     B          MAKE SOME ROOM
0856 06 80              MVI      B,KDR
0858 E6 7F              ANI      FF-KDR
085A D3 FD              OUT      DATA       KDR LOW
085C B0                 ORA      B
085D D3 FD              OUT      DATA       KDR HIGH
085F C1                 POP      B          THANK YOU
0860 C9                 RET
               *
               **  THESE FOUR ROUTINES GET DATA FROM BUS OF SLAVE
               *   AND ALSO PUT THE DATA IN A  RAM BUFFER
               *
               ** FROM DIO BUS
               *
0861 3E 10     GETDIO   MVI      A,INDIO    SEL. DIO
0863 CD 83 08           CALL     GET1       GET DATA
0866 32 2E 0C           STA      DIOBUF     SAVE IT
0869 C9                 RET
               *
               ** FROM STATUS LINES
               *
086A 3E 18     GETSTA   MVI      A,INSTAT   SEL. STATUS
```

```
086C CD 83 08                       CALL     GET1       GET DATA
086F 32 2F 0C                       STA      SBUF       SAVE IT
0872 C9                             RET
                  *
                  ** FROM ADDRESS BUS
                  *
0873 3E 08        GETADR            MVI      A,INLOAD
0875 CD 83 08                       CALL     GET1
0878 6F                             MOV      L,A        L=LO ADRS
0879 3E 00                          MVI      A,INHIAD
087B CD 83 08                       CALL     GET1
087E 67                             MOV      H,A        H=HI ADRS
087F 22 30 0C                       SHLD     ADBUF      SAVE ADRS
0882 C9                             RET
                  *
                  ** THE THREE ABOVE CALL THIS
                  *
0883 D3 FA        GET1              OUT      SELECT     POINT MULTIPLEXER
0885 DB FD                          IN       DATA       GET DATA
0887 C9                             RET
                  *
                  ** CALL HERE TO SINGLE STEP SLAVE
                  *
0888 3A 2D 0C     SSTEP             LDA      CTLSAV
088B E6 FD                          ANI      FF-STEP
088D CD 34 08                       CALL     SETCTL     SET STEP LO
0890 F6 02                          ORI      STEP
0892 CD 34 08                       CALL     SETCTL     NOW HI
0895 C9                             RET
                  *
                  ** SET ZERO IF INSTRUCTION FETCH CYCLE
                  *
0896 CD 6A 08     M1TST             CALL     GETSTA     STATUS TO ACC.
0899 E6 1F        M1CHK             ANI      1FH        IGNORE BITS 5,6,7
089B FE 11                          CPI      11H        M1+MEMR
089D C9                             RET
                  *
                  ** SINGLE STEP TO AN INSTRUCTION FETCH CYCLE
                  ** UNLESS THIS IS ONE.
                  *
089E CD 96 08     FINDM1            CALL     M1TST
08A1 C8                             RZ
                  *
                  ** ENTER HERE TO SINGLE STEP TO NEXT
                  ** INSTRUCTION FETCH CYCLE
                  *
08A2 CD 88 08     TILLM1            CALL     SSTEP
08A5 C3 9E 08                       JMP      FINDM1
                  *
                  ** JAM ACCUMULATOR ONTO DIO BUS
                  ** THEN SINGLE STEP.
                  ** JMP TO DIBAD IF ERROR
                  *
08A8 CD 40 08     AJAM              CALL     SETDIO     ACC. TO DIO BUS
08AB 47                             MOV      B,A        SAVE DATA
08AC 3A 2D 0C                       LDA      CTLSAV     GET CTL BITS
08AF E6 FB                          ANI      FF-JAM     DIG1 LO
08B1 CD 34 08                       CALL     SETCTL     JAM IT
08B4 CD 61 08                       CALL     GETDIO
08B7 B8                             CMP      B          DIO=DATA ?
08B8 C2 A7 01                       JNZ      DIBAD      DIO <> DATA
                  *
08BB CD 88 08                       CALL     SSTEP      NOW STEP
```

```
08BE 3A 2D 0C                    LDA      CTLSAV
08C1 F6 04                       ORI      JAM         DIG1 HI
08C3 CD 34 08                    CALL     SETCTL
08C6 78                          MOV      A,B
08C7 C9                          RET
                      *
                      **  DEPOSIT ACCUMULATOR CONTENTS AT ADDRESS
                      **  POINTED TO BY HL. KEEP REGISTERS CLEAN
                      *
08C8 E5               POKE       PUSH     H
08C9 D5                          PUSH     D
08CA C5                          PUSH     B           STACK UP EVERYTHING
08CB F5                          PUSH     PSW
08CC F5                          PUSH     PSW         EXTRA COPY OF ACC
                      *
08CD CD 9E 08                    CALL     FINDM1
08D0 3A 2D 0C                    LDA      CTLSAV
08D3 E6 FB                       ANI      FF-JAM
08D5 CD 34 08                    CALL     SETCTL      DIG1 ON
                      *
08D8 3E 32                       MVI      A,32H       'STA' OP CODE
08DA CD 40 08                    CALL     SETDIO
08DD CD 88 08                    CALL     SSTEP
08E0 7D                          MOV      A,L         LO ADDRESS
08E1 CD 40 08                    CALL     SETDIO
08E4 CD 88 08                    CALL     SSTEP
08E7 7C                          MOV      A,H         HI ADDRESS
08E8 CD 40 08                    CALL     SETDIO
08EB CD 88 08                    CALL     SSTEP
                      *
08EE 3A 2D 0C                    LDA      CTLSAV
08F1 E6 DF                       ANI      FF-DODSB
08F3 CD 34 08                    CALL     SETCTL      DODSBL ON
                      *
08F6 F1                          POP      PSW         GET DATA
08F7 CD 40 08                    CALL     SETDIO
08FA CD 88 08                    CALL     SSTEP
                      *
08FD 3A 2D 0C                    LDA      CTLSAV
0900 F6 24                       ORI      DODSB+JAM
0902 CD 34 08                    CALL     SETCTL      DODSBL & DIG1 OFF
                      *
0905 F1                          POP      PSW
0906 C1                          POP      B
0907 D1                          POP      D           UNSTACK EVERYTHING
0908 E1                          POP      H
0909 C9                          RET
                      *
                      **  LOAD ACCUMULATOR FROM ADDRESS POINTED TO
                      **  BY HL. KEEP REGISTERS CLEAN
                      *
090A E5               PEEK       PUSH     H
090B D5                          PUSH     D           STACK UP EVERYTHING
090C C5                          PUSH     B
090D F5                          PUSH     PSW
                      *
090E CD 9E 08                    CALL     FINDM1      GET FETCH CYCLE
0911 3A 2D 0C                    LDA      CTLSAV
0914 E6 FB                       ANI      FF-JAM
0916 CD 34 08                    CALL     SETCTL      DIG1 ON
                      *
0919 3E 3A                       MVI      A,3AH       'LDA' OP CODE
091B CD 40 08                    CALL     SETDIO
```

```
091E CD 88 08              CALL      SSTEP       TO CPU
                 *
0921 7D                    MOV       A,L         LO ADDRESS
0922 CD 40 08              CALL      SETDIO
0925 CD 88 08              CALL      SSTEP       TO CPU
0928 7C                    MOV       A,H         HI ADDRESS
0929 CD 40 08              CALL      SETDIO
092C CD 88 08              CALL      SSTEP       TO CPU
                 *
092F 3A 2D 0C              LDA       CTLSAV
0932 F6 04                 ORI       JAM         DIG1 OFF
0934 CD 34 08              CALL      SETCTL
0937 CD 61 08              CALL      GETDIO      GET DATA
                 *
093A F1                    POP       PSW
093B C1                    POP       B           UNSTACK EVERYTHING
093C D1                    POP       D
093D E1                    POP       H
093E 3A 2E 0C              LDA       DIOBUF      DATA TO ACCUMULATOR
0941 C9                    RET
                 *
                 ** FORCE SLAVE Sol TO ADRS IN HL
                 *
0942 CD 9E 08    JUMP1     CALL      FINDM1      FORCE A JMP
0945 3E C3                 MVI       A,0C3H
0947 CD A8 08    JUMP2     CALL      AJAM        DOWN
094A 7D                    MOV       A,L
094B CD A8 08              CALL      AJAM        IT'S
094E 7C                    MOV       A,H
094F CD A8 08              CALL      AJAM        THROAT
0952 C9                    RET
                 *
                 **  INPUT PORT SELECTED BY ACCUMULATOR
                 *
0953 C5          IPORT     PUSH      B           IT'S NOT ALTERED
0954 F5                    PUSH      PSW         SAVE PORT #
0955 CD 9E 08              CALL      FINDM1      GET A FETCH CYCLE
                 *
0958 3E DB                 MVI       A,0DBH      INPUT OP CODE
095A CD A8 08              CALL      AJAM        TO CPU
095D F1                    POP       PSW         PORT #
095E CD A8 08              CALL      AJAM        TO CPU
0961 CD 61 08              CALL      GETDIO      GET THE DATA
0964 C1                    POP       B
0965 C9                    RET
                 *
                 ********************* END OF BASIC ROUTINES ********************
                 *
                 *
0966 5E 4D 4A 4A FREMS     ASC       "^MJJ^SLAVE Sol IS FREE RUNNING.^MJ^"
     5E 53 4C 41
     56 45 20 53
     6F 6C 20 49
     53 20 46 52
     45 45 20 52
     55 4E 4E 49
     4E 47 2E 5E
     4D 4A 5E
0989 53 54 52 49           ASC       "STRIKE SPACE BAR TO STOP, 'S' TO RESTART^MJ^/"
     4B 45 20 53
     50 41 43 45
     20 42 41 52
     20 54 4F 20
```

```
      53 54 4F 50
      2C 20 27 53
      27 20 54 4F
      20 52 45 53
      54 41 52 54
      5E 4D 4A 5E
      2F
                     1174 *
09B6  5E 4B 4A 4A    1175 STMSG    ASC    "^KJJ^SLAVE Sol STARTED FROM ADDRESS 0000.^MJJ^/"
      5E 53 4C 41
      56 45 20 53
      6F 6C 20 53
      54 41 52 54
      45 44 20 46
      52 4F 4D 20
      41 44 44 52
      45 53 53 20
      30 30 30 30
      2E 5E 4D 4A
      4A 5E 2F
                     1176 *
09E5  5E 4D 4A 4A    1177 BRKMS    ASC    "^MJJ^"
      5E
09EA  45 4E 54 45    1178 BRKM1    ASC    "ENTER BREAK POINT CONDITION.^MJ^ /"
      52 20 42 52
      45 41 4B 20
      50 4F 49 4E
      54 20 43 4F
      4E 44 49 54
      49 4F 4E 2E
      5E 4D 4A 5E
      20 2F
0A0C  5E 4D 57 5E    1179 BRKM2    ASC    "^MW^SLAVE Sol RUNNING WITH BREAK CONDITION = /"
      53 4C 41 56
      45 20 53 6F
      6C 20 52 55
      4E 4E 49 4E
      47 20 57 49
      54 48 20 42
      52 45 41 4B
      20 43 4F 4E
      44 49 54 49
      4F 4E 20 3D
      20 2F
0A3A  5E 4A 4D 4D    1180 BRKM3    ASC    "^JMM^TYPE 'A' TO ABORT  /
      5E 54 59 50
      45 20 27 41
      27 20 54 4F
      20 41 42 4F
      52 54 20 20
      2F
                     1181 *
0A53  5E 4D 57 5E    1182 ABTMSG   ASC    "^MW^BREAK SEARCH ABORTED^MJMJ^/
      42 52 45 41
      4B 20 53 45
      41 52 43 48
      20 41 42 4F
      52 54 45 44
      5E 4D 4A 4D
      4A 5E 2F
0A72  5E 57 4D 4D    1183 FMSG1    ASC    "^WMM^BREAK CONDITION ENCOUNTERED^JMM^/
      5E 42 52 45
      41 4B 20 43
```

```
     4F 4E 44 49
     54 49 4F 4E
     20 45 4E 43
     4F 55 4E 54
     45 52 45 44
     5E 4A 4D 4D
     5E 2F
                         *
0A98 5E 4D 4A 4A         GOTMS    ASC      "^MJJ^ENTER JUMP ADDRESS^MJ^/
     5E 45 4E 54
     45 52 20 4A
     55 4D 50 20
     41 44 44 52
     45 53 53 5E
     4D 4A 5E 2F
                         *
0AB4 5E 4D 4A 4A         REGMS    ASC      "^MJJ^A    B  C    D  E    H  L^MJ^/"
     5E 41 20 20
     20 20 42 20
     20 43 20 20
     20 20 44 20
     20 45 20 20
     20 20 48 20
     20 4C 5E 4D
     4A 5E 2F
                         *
0AD7 3F 3F 3F 2F         QUMES    ASC      "???/"
                         *
                                  COPY     TSTIO.S/1
                         *
                         **********************************
                         *** INPUT/OUTPUT INTERFACE FOR ***
                         ***       PARASOL PROGRAM      ***
                         **********************************
                         *
                         *
0ADB E5                  OSOUT    PUSH     H          DON'T ALTER IT
0ADC 2E 19                        MVI      L,SOUT     PAGE OFFSET
0ADE C3 E4 0A                     JMP      OSIO
                         *
0AE1 E5                  OSIN     PUSH     H          DON'T ALTER IT
0AE2 2E 1F                        MVI      L,SINP     PAGE OFFSET
                         *
0AE4 D5                  OSIO     PUSH     D          DON'T ALTER IT
0AE5 26 C0                        MVI      H,0C0H     HL=ENTRY POINT
0AE7 11 EC 0A                     LXI      D,IORTN    RETURN ADDRESS
0AEA D5                           PUSH     D          TO STACK
0AEB E9                           PCHL     .          TO ENTRY POINT
                         *
0AEC D1                  IORTN    POP      D          DONT FORGET !
0AED E1                           POP      H
0AEE C9                           RET      .          I/O DONE
                         *
0AEF CD E1 0A            INA      CALL     OSIN       INPUT READY ?
0AF2 CA EF 0A                     JZ       INA        NO.  KEEP TRYING
0AF5 E6 7F               INB      ANI      7FH        ELSE KILL PARITY
0AF7 FE 1B                        CPI      1BH        ESCAPE ?
0AF9 CA 04 C0                     JZ       0C004H     YES.  BACK TO SOLOS
0AFC 47                           MOV      B,A        ELSE
0AFD B7                           ORA      A          CORRECT STATUS
0AFE C9                           RET      .          WITH A & B = INPUT
                         *
0AFF CD E1 0A            INCHR    CALL     OSIN       INPUT READY ?
```

```
0B02 C3 F5 0A              JMP     INB         TRY JUST ONCE
                     *
                     * PRINT HL AS HEX. NUMBER
                     *
0B05 7C              ADOUT MOV     A,H         HI BYTE FIRST
0B06 CD 0E 0B              CALL    HEXA        PRINTED IN HEX
0B09 7D                    MOV     A,L         LO BYTE NEXT
0B0A CD 0E 0B              CALL    HEXA        PRINTED IN HEX
0B0D C9                    RET
                     *
0B0E 4F              HEXA  MOV     C,A         SAVE VALUE
0B0F 0F                    RRC
0B10 0F                    RRC
0B11 0F                    RRC     .           SWAP  NIBBLES
0B12 0F                    RRC
0B13 CD 17 0B              CALL    HEXA1       CONVERT HI NIBBLE
0B16 79                    MOV     A,C
                     *
0B17 E6 0F           HEXA1 ANI     0FH         CONVERT LO NIBBLE
0B19 C6 30                 ADI     48          ADD ASCII BIAS
0B1B FE 3A                 CPI     58          GREATER THAN 9 ?
0B1D DA 22 0B              JC      HEXA2       NOPE
0B20 C6 07                 ADI     7           YES. CONVERT TO A-F
0B22 47              HEXA2 MOV     B,A
0B23 C3 DB 0A              JMP     OSOUT       PRINT IT & RETURN
                     *
0B26 06 20           BOUT  MVI     B,20H
0B28 C3 DB 0A              JMP     OSOUT       PRINT 1 SPACE
                     *
                     * BINARY OUTPUT
                     *
     0B2B            BINOT EQU     $
0B2B 2E 30                 MVI     L,'0'       MARK 0 BITS
0B2D 26 31                 MVI     H,'1'       MARK 1 BITS
                     *
0B2F 5F              BINOX MOV     E,A         VALUE TO PRINT
0B30 16 08                 MVI     D,8         BIT COUNT
0B32 CD 26 0B              CALL    BOUT        A SPACE
                     *
0B35 7B              BINO1 MOV     A,E
0B36 07                    RLC     .           MSB ON LEFT
0B37 5F                    MOV     E,A
0B38 45                    MOV     B,L         0 MARK
0B39 D2 3D 0B              JNC     BPUT        IF BIT IS 0
0B3C 44                    MOV     B,H         ELSE 1 MARK
0B3D CD DB 0A        BPUT  CALL    OSOUT       PRINT IT
0B40 CD 26 0B              CALL    BOUT        ADD A SPACE
                     *
0B43 15                    DCR     D           8 BITS DONE ?
0B44 C8                    RZ      .           YES. DONE
                     *
0B45 7A                    MOV     A,D
0B46 FE 04                 CPI     4           ONE BYTE DONE ?
0B48 C2 35 0B              JNZ     BINO1       NO. LOOP
0B4B CD 26 0B              CALL    BOUT        YES.  ADD EXTRA SPACE
0B4E C3 35 0B              JMP     BINO1       LOOP
                     *
                     * PRINT HL AS BINARY NUMBER
                     *
0B51 D5              AOUTB PUSH    D           KEEP IT CLEAN
0B52 7C                    MOV     A,H
0B53 E5                    PUSH    H
0B54 CD 2B 0B              CALL    BINOT       PRINT HI BYTE
```

```
0B57 E1                     POP     H
0B58 7D                     MOV     A,L
0B59 E5                     PUSH    H
0B5A CD 2B 0B               CALL    BINOT       PRINT LO BYTE
0B5D E1                     POP     H
0B5E D1                     POP     D           IT'S CLEAN
0B5F C9                     RET
                  *
     001F         SINP      EQU     1FH
     0019         SOUT      EQU     19H
                  *
                  *
                  * TEXT OUTPUT ROUTINE
                  * HL POINT TO STRING
                  * ^ TOGGLES BETWEEN NORM. & CTL. CHARS.
                  * IE.  ^M^  IS EQUIVALENT TO C/R.
                  * / IS TERMINATOR CHAR.
                  *
                  *
     0B60         TEXTO     EQU     $
0B60 0E FF                  MVI     C,0FFH      NORM CHAR MASK
                  *
0B62 7E           TEXL      MOV     A,M         GET TEXT CHAR
0B63 23                     INX     H           BUMP POINTER
0B64 FE 2F                  CPI     2FH         TERMINATOR ?
0B66 C2 6A 0B               JNZ     TEXT1       NO. GO PROCESS
0B69 C9                     RET                 ELSE DONE
                  *
0B6A FE 5E        TEXT1     CPI     5EH         CONTROL TOGGLE ?
0B6C C2 76 0B               JNZ     TEXT2       NO. GO OUTPUT
                  *
0B6F 79                     MOV     A,C         ELSE GET MASK
0B70 EE E0                  XRI     0E0H        TOGGLE BITS 5,6,7
0B72 4F                     MOV     C,A         RESTORE MASK
0B73 C3 62 0B               JMP     TEXL        DO NEXT CHAR.
                  *
0B76 A1           TEXT2     ANA     C           INCLUDE CTL MASK
0B77 CD 7D 0B               CALL    PUT         OUTPUT CHAR
0B7A C3 62 0B               JMP     TEXL        DO NEXT CHAR
                  *
0B7D C5           PUT       PUSH    B           DON'T CRASH IT
0B7E 47                     MOV     B,A
0B7F CD DB 0A               CALL    OSOUT       PRINT CHAR
0B82 C1                     POP     B           GOOD AS NEW
0B83 C9                     RET
                  *
                  *  PRINT CR AND LF
                  *
0B84 3E 0D        CRLF      MVI     A,0DH       C/R
0B86 CD 7D 0B               CALL    PUT
0B89 3E 0A                  MVI     A,0AH       LF
0B8B CD 7D 0B               CALL    PUT
0B8E C9                     RET
                  *
                  ** THE END OF TSTIO.S**
                            COPY    UTIL.S/1
                  *
                  *
                  **             UTILITY ROUTINES
                  *
                  * CONVERT ASCII STRING POINTED TO
                  * BY DE TO BINARY VALUE IN HL.
                  * NZ SET = ERROR.
```

```
                      *
0B8F 21 00 00         SHEX1    LXI     H,0       CLEAR IT
0B92 0E 05                     MVI     C,5
0B94 1A               SHE2     LDAX    D         GET CHAR.
0B95 FE 20                     CPI     20H       SP ?
0B97 C8                        RZ      DONE
0B98 FE A0                     CPI     0A0H      CURSOR ?
0B9A C8                        RZ      DONE
0B9B 0D                        DCR     C         5TH BYTE
0B9C CA BF 0B                  JZ      SHERR     TOO MANY !
0B9F 29                        DAD     H
0BA0 29                        DAD     H         SHIFT
0BA1 29                        DAD     H         HL LEFT
0BA2 29                        DAD     H         4 BITS
0BA3 CD B5 0B                  CALL    HCOV2     ASCII TO HEX
0BA6 D2 BF 0B                  JNC     SHERR
0BA9 85                        ADD     L
0BAA 6F                        MOV     L,A       WITH NEW BYTE
0BAB 13                        INX     D         TO NEXT CHAR
0BAC 7B                        MOV     A,E       PAST END
0BAD E6 3F                     ANI     3FH       OF LINE ?
0BAF CA BF 0B                  JZ      SHERR     ---> YES
0BB2 C3 94 0B                  JMP     SHE2
                      *
0BB5 D6 30            HCOV2    SUI     48        ASCII BIAS
0BB7 FE 0A                     CPI     10
0BB9 D8                        RC      .         < 9
0BBA D6 07                     SUI     7         ALPHA BIAS
0BBC FE 10                     CPI     10H
0BBE C9                        RET
                      *
0BBF AF               SHERR    XRA     A
0BC0 3C                        INR     A
0BC1 C9                        RET     .         NZ SET=ERROR
                      *
                      *
                      **  SKIP TO NEXT ARG. ON RETURN
                      **    Z SET=PAST END OF LINE
                      *
0BC2 06 0F            SKIP     MVI     B,15
0BC4 7E               SKIP1    MOV     A,M       GET CHAR
0BC5 E6 7F                     ANI     7FH       KILL CURSOR
0BC7 FE 20                     CPI     ' '       SPACE ?
0BC9 CA D2 0B                  JZ      SKP2      ---> YES
0BCC 23                        INX     H         NEXT CHAR.
0BCD 05                        DCR     B         TOO FAR ?
0BCE C8                        RZ      .         YES ---> Z SET=ERROR
0BCF C3 C4 0B                  JMP     SKIP1 TILL SPACE FOUND
                      *
0BD2 23               SKP2     INX     H         NEXT CHAR.
0BD3 05                        DCR     B         TOO FAR ?
0BD4 C8                        RZ      .         YES ---> Z SET=ERROR
0BD5 7E                        MOV     A,M       GET CHAR
0BD6 E6 7F                     ANI     7FH       KILL CURSOR
0BD8 FE 20                     CPI     ' '       SPACE ?
0BDA CA D2 0B                  JZ      SKP2      ---> TILL NON-SPACE FOUND
0BDD C9                        RET     'NZ SET=OK
                      *
                      ** GET AN INPUT LINE
                      *
0BDE CD EF 0A         GCLIN    CALL    INA       WAIT FOR CHAR
0BE1 FE 0D                     CPI     0DH       CR ?
0BE3 C8                        RZ      .         YES ---> GOT A LINE
```

```
0BE4 FE 0A                      CPI     0AH      LF ?
0BE6 CA DE 0B                   JZ      GCLIN    IGNORE IT
0BE9 FE 18                      CPI     18H      CANCEL ?
0BEB C2 F4 0B                   JNZ     DELL     NO --->
0BEE CD 01 0C                   CALL    CANCL    ERASE LINE
0BF1 C3 DE 0B                   JMP     GCLIN    TRY AGAIN
0BF4 FE 7F            DELL      CPI     7FH      DEL ?
0BF6 C2 FB 0B                   JNZ     ECHO     NO --->
0BF9 06 5F                      MVI     B,5FH    SUBSTITUTE BS
0BFB CD DB 0A         ECHO      CALL    OSOUT
0BFE C3 DE 0B                   JMP     GCLIN
                      *
0C01 06 0D            CANCL     MVI     B,0DH
0C03 CD DB 0A                   CALL    OSOUT    GOBBLE TO END
0C06 06 0D                      MVI     B,0DH
0C08 CD DB 0A                   CALL    OSOUT    GOBBLE THE REST
0C0B C9                         RET
                      *
                      ** GET LINE ADRS. TO SADRS
                      *
0C0C 3A 09 C8         GSAD      LDA     0C809H   LINE POS.
0C0F 6F                         MOV     L,A
0C10 3A 0A C8                   LDA     0C80AH   TEXT OFFSET
0C13 85                         ADD     L
0C14 0F                         RRC
0C15 0F                         RRC
0C16 47                         MOV     B,A
0C17 E6 C0                      ANI     0C0H     FOR START OF LINE
0C19 6F                         MOV     L,A
0C1A 78                         MOV     A,B
0C1B E6 03                      ANI     3
0C1D C6 CC                      ADI     0CCH
0C1F 67                         MOV     H,A
0C20 22 32 0C                   SHLD    SADRS    SAVE ADRS
0C23 C9                         RET
                      *
                      **  GET CURSOR POS. TO CPOS
                      *
0C24 3A 08 C8         GCPOS     LDA     0C808H   NCHR
0C27 32 34 0C                   STA     CPOS
0C2A C9                         RET
                      *
                      ** END OF UTIL.S **
                      *
                      *
     0C2B             END       EQU     $
0C2B 00               SUM1      DB      0
0C2C A6               SUM2      DB      0A6H     CORRECT CHECKSUM
                      *
                      *
                      **  RAM AREA FOLLOWS  **
                      *
                      *
0C2D                  CTLSAV    DS      1        COPY OF CTL REG.
0C2E                  DIOBUF    DS      1        CURRENT DIO BUS
0C2F                  SBUF      DS      1        CURRENT STATUS
0C30                  ADBUF     DS      2        CURRENT ADRS. BUS
0C32                  SADRS     DS      2        SCREEN LINE ADRS.
0C34                  CPOS      DS      1        CURSOR POSITION
0C35                  TADRS     DS      2        TRAP ADRS
0C37                  SFLAG     DS      1        G.P. FLAGS
0C38                  CBUFF     DS      16       BREAK CYCLE NAME
0C48                  BRKST     DS      1        BREAK STATUS MASK
0C49                  CYNUM     DS      1        BREAK OPREATION #
                      *
```

SECTION 5

OPERATING INSTRUCTIONS

5.1 INTRODUCTION

This section describes how to connect the ParaSol Debugger to the master and slave Sols and how to operate the Debugger Program. Some of the terminology used in this section is explained in the Hardware Theory of Operation section. If you have not read the Hardware Theory of Operation, please do so at this time, then return to this section. Refer to the Software Theory of Operation section for flow charts, source listings, and additional information on the routines that make up the Debugger Program.

5.2 SYSTEM CONFIGURATION

The master Sol should be a unit that you have as much confidence in as possible. If the master Sol has a bug that causes the Debugger Program to incorrectly identify errors or ignore errors, much time could be wasted on a wild goose chase. A Sol which reliably runs the Helios Disk System, or large BASIC programs would make a good master Sol. If the parallel interface has never been checked, the Debugger Board test programs in Section 2, Assembly and Hardware Checkout, can be used to test it.

5.3 OPERATING INSTRUCTIONS

(Refer to Fig. 2-1, ParaSol Debugger System Interconnect Diagram.)

1. Be sure power is OFF in both Sols.

2. Position the Sol to be debugged alongside the master Sol.

3. Plug the 25-pin connector J2 on the cable adapter board into the parallel interface connector of the master Sol. Remember that since there is a difference between the signal connections to the parallel interface between revision D and revision E Sols, the correct configuration plug must be installed in J3 of the cable adapter board or the master Sol must be a revision E unit. For details see Section 2.2.2, Installation of Sol Rev Level Configuration Socket.

4. If the Sol to be debugged does not have the backplane installed, plug the Debugger Board into the 100-pin S-100 Bus connector on the Sol P.C. with the component side of the Debugger toward the power connector of the Sol P.C.

5. Connect the 5-pin molex backplane power connector of the Sol to the J3 auxilliary power connector on the debugger board.

6. If the Sol to be debugged has the backplane installed, plug the Debugger Board into the 100-pin S-100 Bus connector mounted vertically on the top of the backplane. Be sure the component side of the Debugger board is facing toward the keyboard. Also be sure the backplane power cable is connected to the backplane.

7. Disconnect the keyboard cable of the slave Sol from the keyboard and remove the keyboard assembly. The keyboard cable should extend from the connector on the Sol P.C. toward DIP switch S1. Be sure the keyboard cable DOES NOT extend toward the Debugger Board. Plug the free end of the keyboard cable into the J2 20-pin connector on the Debugger board so the flat cable extends upward from the J2 connector. Be sure the cable DOES NOT extend downward from the connector. See Figure 5-1 for an illustration of the proper keyboard cable orientation.

WARNING. INCORRECT INSTALLATION OF THE KEYBOARD CABLE CAN DAMAGE ICs!

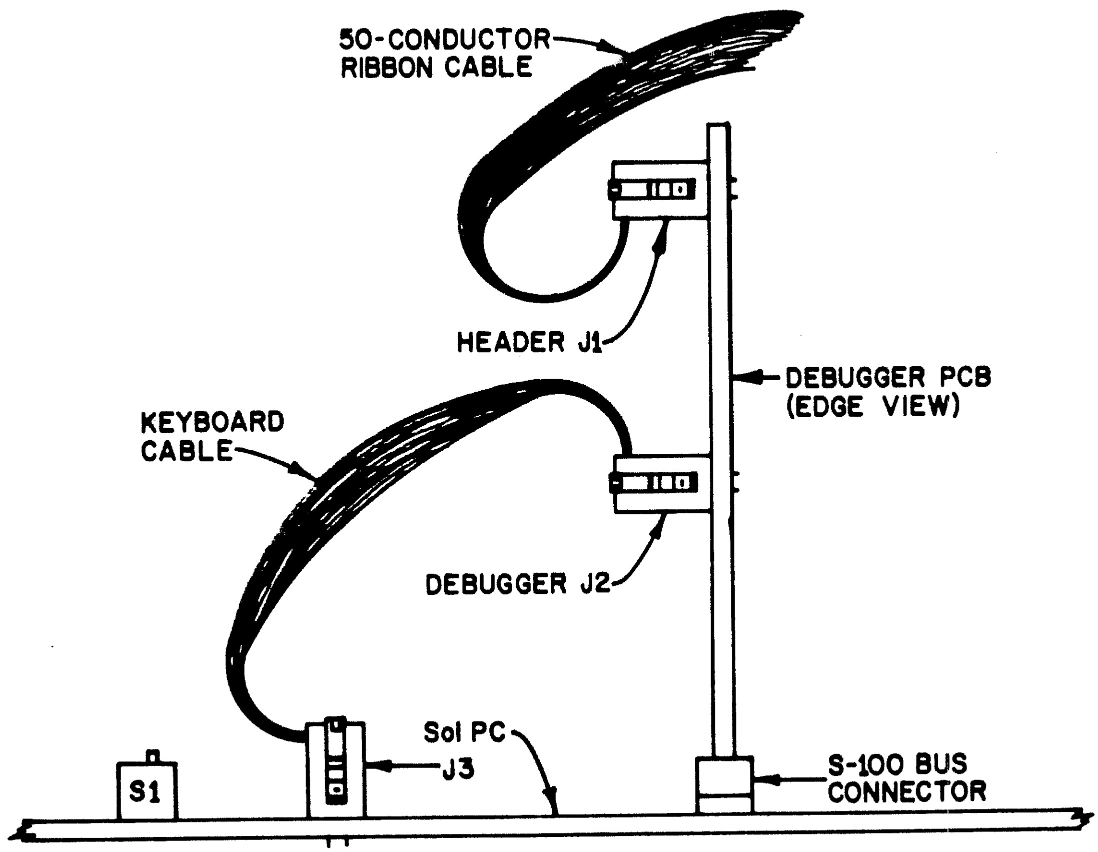


Fig. 5-1. Orientation of Debugger Cable Connections.

8. Remove the phantom jumper from the slave Sol. It is located between U64 and U77 on the Sol PC, and connects between points F and G. During the tests the Debugger Board controls the PHANTOM signal. Do not forget to reinstall the phantom jumper when the Debugger is removed.

9. Check that the same personality module version is installed in the master and slave Sols. The Debugger Program makes a byte for byte comparison of the two personality modules. If different modules are installed an error message will be generated when no problem exists.

10. Apply power to the master and slave Sol.

5.4 THE DEBUGGER PROGRAM

5.4.1 PROGRAM LOADING INSTRUCTIONS

The tape casette provided with the ParaSol Debugger contains the Debugger Program in both source code and machine code. The machine code version loads into memory at address Ø. The source code is split up into 3 files compatible with the PTDOS EDIT and ASSM commands. See the Software Theory of Operation, Section 4, for the procedure used to load the source files and convert them into PTDOS files.

1. To load the machine-code version of the Debugger Program, set up the master Sol with a casette recorder as described in the Sol Systems Manual, Section 7.

2. Insert the cassette into the recorder with the label up. Be sure the tape is rewound.

3. Set the volume control to the middle of its range, and set the tone control for maximum treble.

4. Depress the PLAY lever.

5. Type the command: XEQ PDBUG

The program will automatically load and begin execution at address Ø.

6. To restart the program from the beginning, execute address Ø. Before the actual Debugger Program starts, a checksum routine verifies that the program is correctly loaded into memory. If the program is incorrectly loaded, it attempts to display an error message and return control to SOLOS. If the program is correctly loaded it will display this message:

Processor Technology Corp.

ParaSol Debugger Program

Rev. A

TESTS IN PROGRESS...

5.4.2 THE TEST MODE (DESCRIPTION OF TEST ROUTINES)

The debugger program is now performing a series of tests on the slave Sol. If an error is detected the program will stop and display a message describing the error, otherwise the program will enter the trace mode and start up the personality module program in the slave Sol.

The test routines which the Debugger Program performs will now be described. Flow charts in Section 4, Software Theory of Operation, will aid in understanding the routines and diagnosing the problem when an error is detected. Reference to the source listing in the same section can be of help.

1. CONTROL SIGNAL TEST ROUTINE

This routine tests the signals which the Debugger Board uses to control the slave Sol. The XRDY signal is exercised, and a reset of the slave Sol is performed. Then the Address and Status signals are checked for the correct levels following a reset. $\overline{\text{DIGI}}$ (low active) is exercised and the Debugger attempts to jam all DIO lines low. Following a single step sequence, the address lines are checked for the address 0001 (hex). If an error occurs, the program will stop with the buses in a static state so that signals can be easily traced.

2. DIO BUS TEST ROUTINE

This routine activates the $\overline{\text{DIGI}}$ and $\overline{\text{DO DSBL}}$ (low active) signals, then jams a pattern onto the DIO bus. The pattern starts at 0 and increments to FF (hex). For each pattern the DIO bus is checked. If an error occurs, the program will stop and display the error, and the buses will be in a static state. If the DIO bus is good, $\overline{\text{DIGI}}$ and $\overline{\text{DO DSBL}}$ (low active) will go inactive at the conclusion of the test.

3. ADDRESS BUS TEST ROUTINE

This test routine utilizes a jump instruction to place a marching pattern on the address bus of the slave Sol. The pattern begins with all zeros; then first A0, then A1, then A2, etc., fill with ones. When all address lines are one, A0, then A1, then A2, etc., fill with zeros until all address lines are zero. For each pattern the address bus is checked, and if an error occurs it will be displayed, and the buses will be in a static state.

4. INT BUS AND PERSONALITY MODULE TEST ROUTINE

Using the PEEK subroutine in the Debugger Program, this routine compares each byte of the personality module in the master Sol with the personality module in the slave Sol. If the two bytes do not match, The address and the data in both modules are displayed, and the buses are in a static state. Since the INT bus is in the data path between the personality module and the Debugger Board, failures during this test could be caused by a bad INT bus or a bad personality module.

5. KEYBOARD INTERFACE TEST ROUTINE

This routine uses the keyboard register on the Debugger Board to test the keyboard interface of the slave Sol. An incrementing pattern as in the DIO bus test is used. If an error is encountered it is displayed and the buses are in a static state.

6. VIDEO RAM TEST ROUTINE

This routine uses the PEEK and POKE subroutines to perform a 1K memory test on the RAM at CC00 (hex). When an error is encountered it is accumulated with any other errors until the test is completed. Then if any errors occurred, a row of Xs and Gs is displayed. One X or G for each bit of the memory. Bit 7 is on the left and bit Ø is on the right. Bits marked with a G tested good, bits marked with an X tested bad. Each RAM IC contains 1 bit of the memory; thus bad RAMS are easily identified. For example, if the display is:

G X G G G G G X

Bits 6 and Ø are bad. The bad ICs are U2Ø, and U14.

If the error is not caused by a bad RAM, the problem is more difficult to isolate. Familiarity with the signals on the RAM pins while the test is running will help with these kinds of problems. Running this test on a good Sol for comparison is worthwhile.

7. SYSTEM RAM TEST ROUTINE

The only difference between this routine and the Video RAM Test is that this test is run on system RAM at C8ØØ (hex).

5.4.3 THE TRACE MODE

If the Test Mode does not detect any errors, the Debugger program enters the Trace Mode. This mode can be used to help diagnose problems which occur during the operation of the personality module program. It could also be used as a general program debugging tool. During the trace mode, individual machine cycles are displayed as the Debugger single steps through a program. Each line of the display presents the data on the buses for the cycle currently being executed. For example:

CCØØ 2Ø MEMORY WRITE

The address bus currently contains CCØØ hex, the DIO bus contains 2Ø hex, and the 8Ø8Ø is performing a memory write cycle.

SPEED CONTROL. When the Trace Mode is entered, a new cycle is displayed each time the space bar is depressed. The number keys 1 to 9 can be used to single step at variable rates. When the 1 key is depressed the Debugger will single step at the fastest rate. Depressing the next higher numbered key cuts the display rate in half until 9 sets the slowest rate. Depressing the space bar sets the display speed to one line at a time.

COMMAND KEYS. During the trace mode, six keys are used to control the operation of the program.

[S] Striking the S key causes the program to start up the personality module program in the slave Sol. It is equivalent to a reset operation.

[F] Striking the F key causes the Debugger to release the bus of the slave Sol. The slave free runs at full speed until the space bar is depressed to stop the slave and return to the normal trace mode. Alternately, the S key may be depressed to reset the slave and return to the normal trace mode.

[J] Striking the J key allows you to force the slave Sol to jump to any address. The program will ask for an address; then when the return key is depressed, a jump instruction will be jammed onto the bus of the slave Sol.

[B] Striking the B key allows you enter a break point condition, then the debugger program will rapidly single step the slave Sol without displaying cycles until the break point condition is met.

When the program requests a break point condition, one of 5 different conditions (as follows) may be specified by typing the name of the cycle, and a memory or I/O port address. <address> is a hex number between Ø and FFFF. <port> is a hex number between Ø and FF.

1. FETCH <address> The break occurs when an instruction is fetched from the address specified.

2. READ <address> The break occurs when a memory read cycle accesses the address specified. Fetching an instruction is not considered a memory read cycle.

3. WRITE <address> The break occurs when a memory write cycle accesses the address specified.

4. INPUT <port> The break occurs when an input cycle accesses the port specified.

5. OUTPUT <port> The break occurs when an output cycle accesses the port specified.

For example, if you wanted to let the slave Sol run until the last screen address location is cleared you would type: WRITE CFFF

Then strike the return key. The slave Sol will run until the write-to-address-CFFF causes a break to occur. The slave Sol will then be stopped and the normal trace mode will be restored.

[C] Striking the C key will cause the Debugger program to single step rapidly without displaying cycles until the most recently entered break point condition is met. This command allows you to step through a looping program segment without having to describe the break-point condition each time.

[R] Striking the R key during the trace mode causes the program to display the registers of the 8Ø8Ø in the slave Sol. The display looks like this:

A	B	C	D	E	H	L
12	34	56	78	9A	Ø1	23

SECTION 6

DRAWINGS

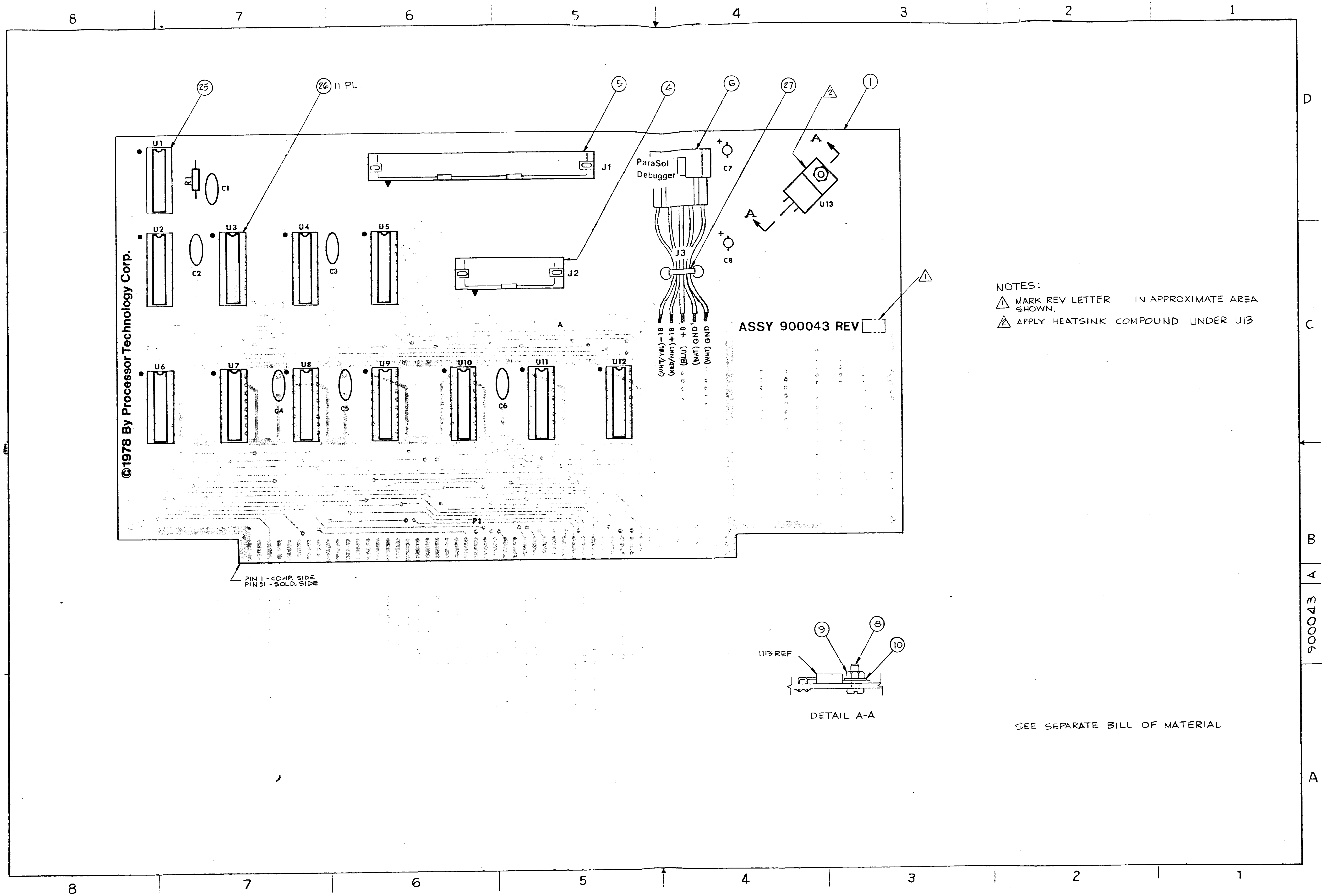


Fig. 6-1 ParaSol Debugger PCB Assembly

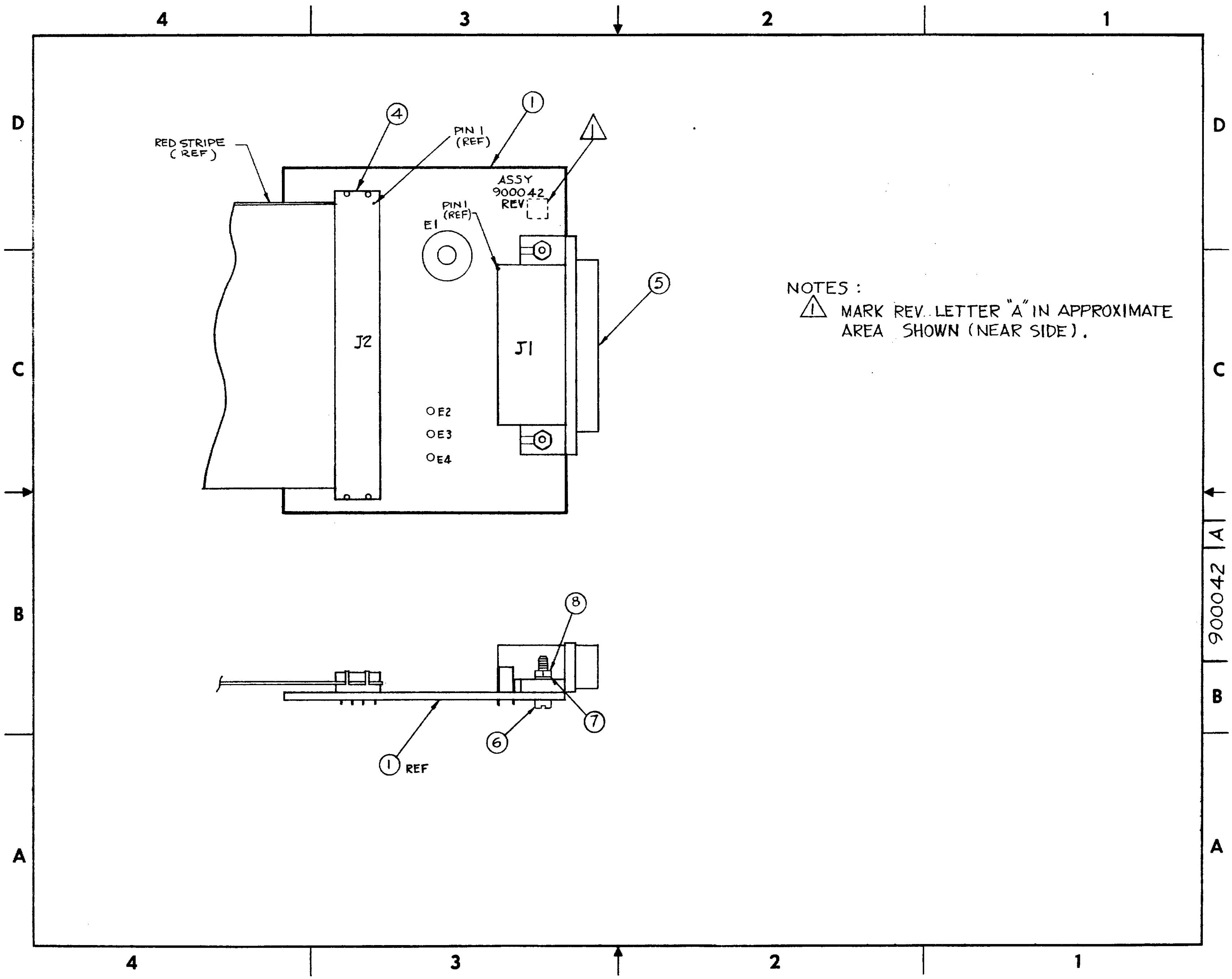


Fig. 6-2 ParaSol Adapter PCB Assembly

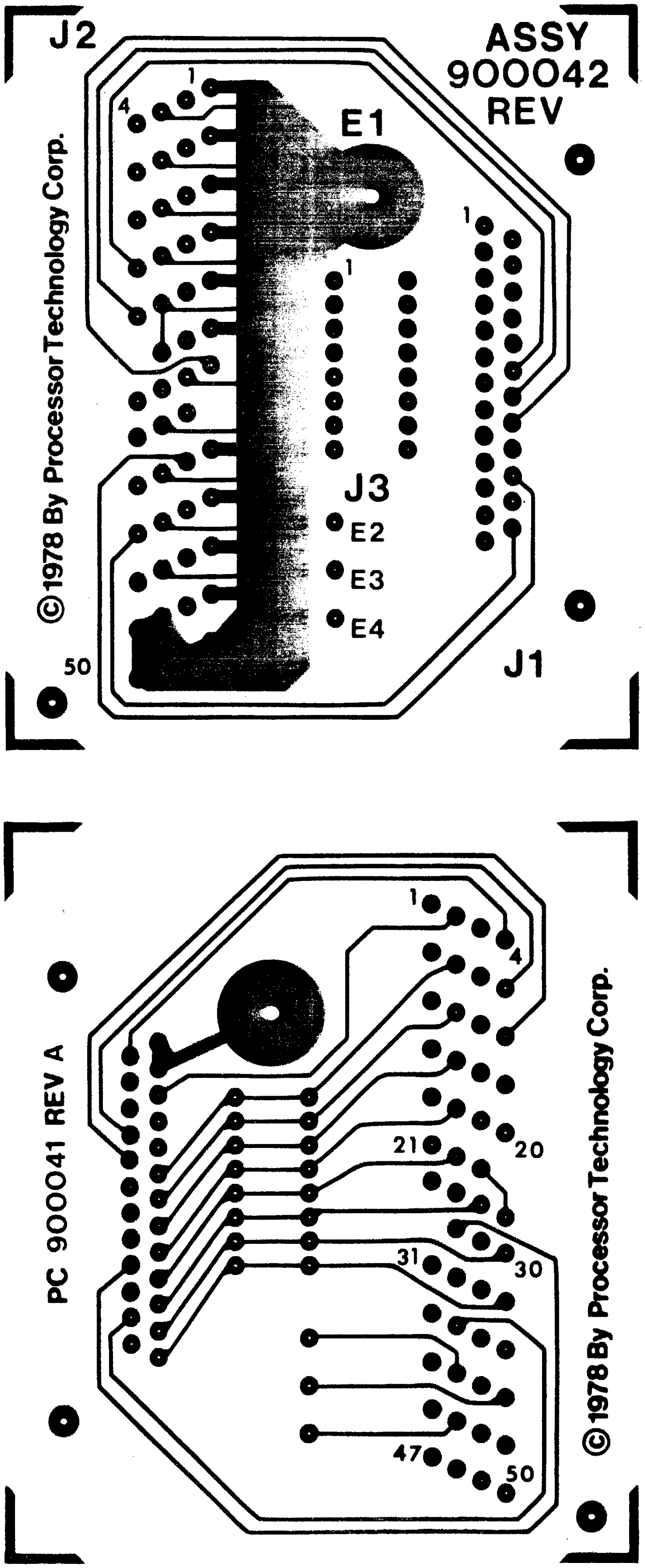


Fig. 6-3 ParaSol Adapter Foil Pattern

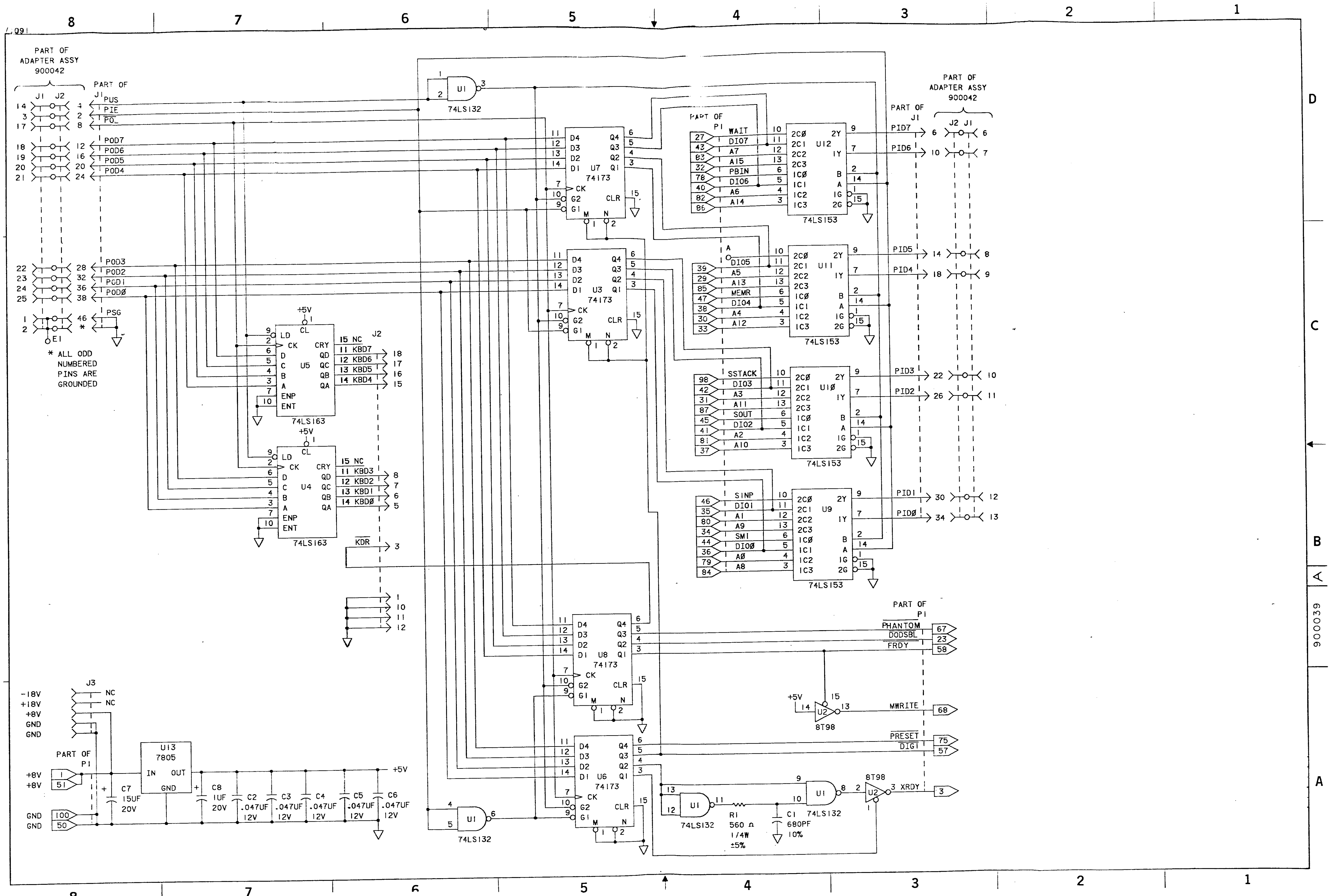


Fig. 6-4 ParaSol Debugger PCB Schematic